Computer Engineering and Communication Networks (CECN)

Prof. Dr. Burkhard Stiller
Communication Systems Group CSG
Department of Informatics IfI, University of Zürich UZH
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

Phone: +41 44 635 6710, FAX: +41 44 635 6809
E-Mail: stiller@ifi.uzh.ch

Universität Zürich[UZH]

Assistents:
Dr. Corinna Schmitt, Sina Rafati
Phone: +41 44 635 [7585¦6702], FAX: +41 44 635 6809
E-Mail: [schmitt¦rafati]@ifi.uzh.ch

CSG Communication Systems Group

© 2017 Burkhard Stiller

ifi

Lecture Goals

- Computer Engineering
  - Competencies in basic systems of underlying hardware and computer architecture
  - To understand that solutions do almost always see a combination of hardware and software with many constraints: costs, temperature, performance ...
  - Basic concepts of information processing in digital forms (data, numbers, chars)
    - Logic
    - Gates, Arithmestic Logical Unit, and Central Processung Unit
    - Basics of chips and computer architecture (performance)
- Communication Networks
  - To understand key concepts of communication networks as ***the*** major pillar of today's society based on ICT (Information and Communications Technology)
  - To understand how the Internet and selected distributed applications work, the Internet being ***the*** prototype of today's as well as future communication and information networks.
  - To be able to extrapolate from today's network and systems examples to future research and development issues and topics.

© 2017 Burkhard Stiller

Lecture Content (1)

- Basics and Introduction
- Arithmetics
  - Number systems, number representation
  - Basic aritrhmetic operations, characters
- Combinatorial Circuits
  - Formal basics of logic (Boolean Algebra), normal- and minimal forms
  - Implementation based on gates, design, and effects of transit times
- Sequential Circuits
  - Formal basics (Finite State Machine)
  - Asynchronous circuits and flipflops, synchronous and dedicated circuits
  - Arithmetic Logical Unit (ALU)
- Computer Architecture and Organization
  - von-Neumann architecture
  - Command formats, data types, addressing
  - Memory and caches
  - Organization in components busses, interfaces, peripherials
  - Technology development

© 2017 Burkhard Stiller

Lecture Content (2)

- History of Communications
- Communication Architectures, Services, and Protocols
  - Communication Characteristics, Layering Examples
- Basic Concepts
  - Building blocks, point-to-point links, reliable service over unreliable channels
- Shared Direct Links (Broadcast)
  - Local Area Networks: Ethernet family, FDDI
- Packet Switching and Internetworking (Network Layer)
  - Internet Protocol and routing
- End-to-end Protocols (Transport Layer)
  - User Datagram Protocol (UDP), Transmission Control Protocol (TCP)
- Security Mechanisms and Protocols
- Applications and Mechanisms (Application Layer)
  - E-mail, Domain Name System (DNS), World Wide Web (WWW)
- Summary and Outlook

© 2017 Burkhard Stiller

Organizational Issues

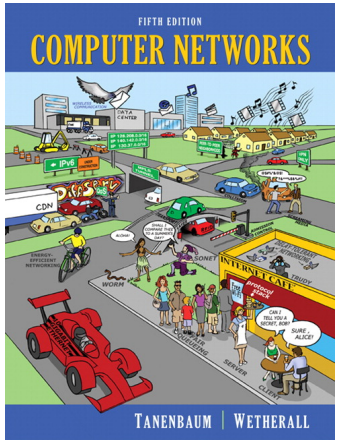
© 2017 Burkhard Stiller

Teaching Material

- Lecture slides
  - http://www.csg.uzh.ch/teaching/fs17/cecn.html

- Reading material
  - Text books (mandatory):
    - H. Herold, B. Lurz, J. Wohlrab: Grundlagen der Informatik, Pearson Studium, München, 2012, 2. Auflage, ISBN-13 978-3-86894-111-1
    - URLs: Karnaugh-Veitch-Diagramme http://ti.itec.uka.de/KVD/ (Universität Karlsruhe)
    - A. S. Tanenbaum, David J. Wetherall: *Computer Networks;* 5th Edition, 2011 Prentice Hall, Indian International Ed, ISBN-13: 978-0-13-212695-3





Contact Information

- Lecture: Thursdays
  8.15am – 9.45am
  15 min break
  and
  10.00am – 11.30am (partially)
- Exercises: Thursdays
  10.00am – 11.30am (partially)
- Access to (additional) lecture information:

  http://www.csg.uzh.ch/teaching/fs17/cecn.html
- Lecturer contact:
  - Contact hours:
    Wednesday morning (11.00 – 12.00)
    Contact Barbara Jost beforehand to reserve a time-slot: jost@ifi.uzh.ch

Many thanks to the following material/slide contributors: Pearson Studium, Prof. T. Braun, Prof. F. Lehmann, Prof. L. Petersen, Prof. B. Plattner, Prof. J. Schiller, Prof. A. Tanenbaum, Dr. Ziegler, Prof. M. Zitterbart

© 2017 Burkhard Stiller

— Computer Engineering —
Module 1: Basics and Introduction



Universität Zürich[UZH]

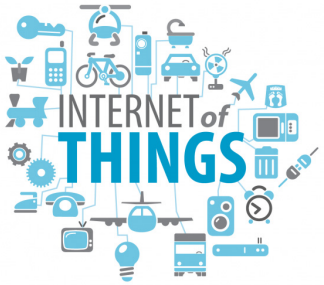
CSG
Communication Systems Group

© 2017 Burkhard Stiller

ifi

Current State of Affairs

- Computer-based devices daily activities:
  - Hand-held/wearable computer
- Computer become “invisible” and become available in vast quantities:
  - Intelligent clothing, Internet-of-Things (IoT), ubiquitous systems, integrated control





- Everything communicates with anything:
  - Radio networks, mobiles, Bluetooth, UMTS, fiber optics, WWW/ …
- Integrated circuits, chips, computer engineering, communication networks shape today’s and tomorrow’s economies and societies:
  - *E.g.,* boom and dead calm of hardware industry impacts politics
  - *E.g.*, Information and Communication Technology (ICT) is today’s society’s pillar

© 2017 Burkhard Stiller

Example: Information Technology in Cars

- Emergency services
- Theft prevention
- Communications (e-mail, WWW, SMS, GSM, ...)
- Information services
- Remote ddiagnosis, remotre maintenance
- Navigation, routing (traffic)
- Engine control, ABS, ESP, ...



...or in a sweing machine!

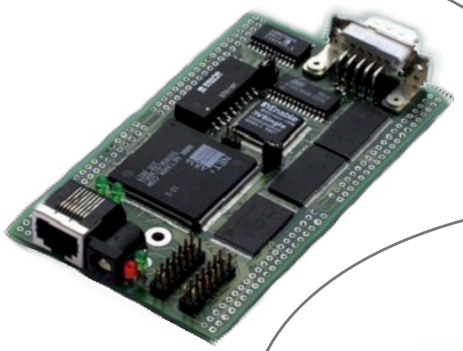


© 2017 Burkhard Stiller

ifi

Example: Communication Technology









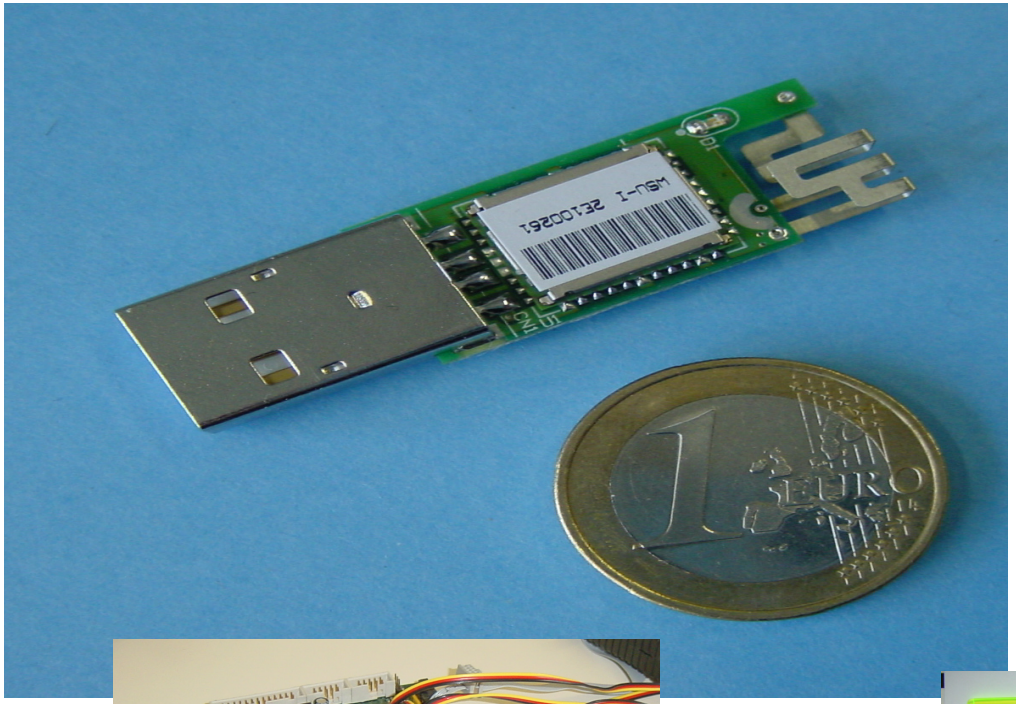


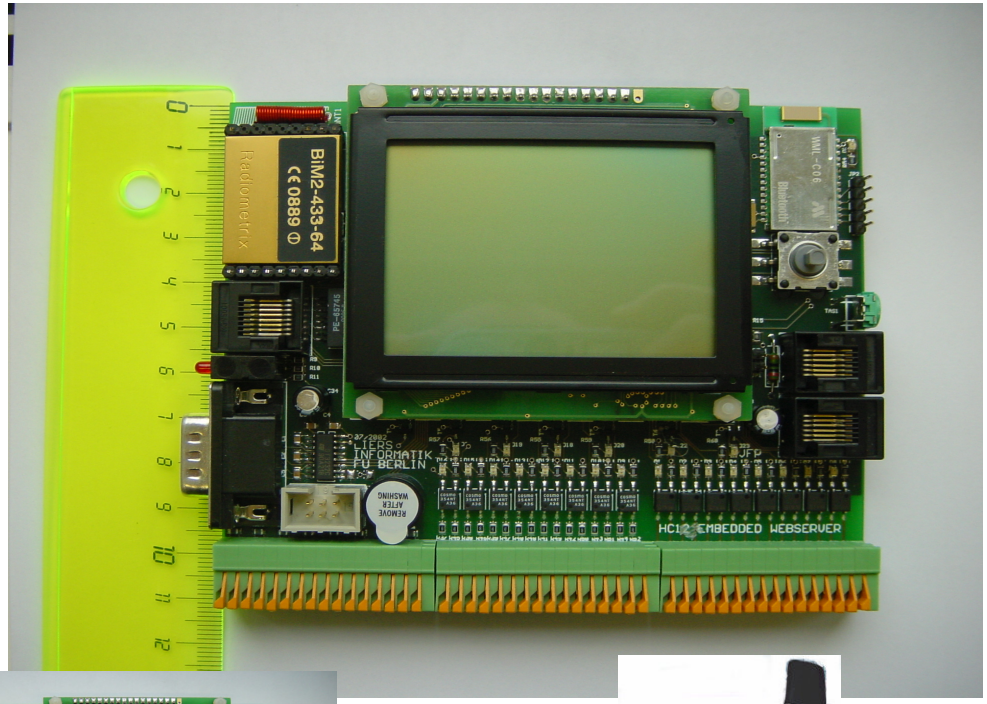


© 2017 Burkhard Stiller

ifi

Examples: Bluetooth Module, Web-Server, ...

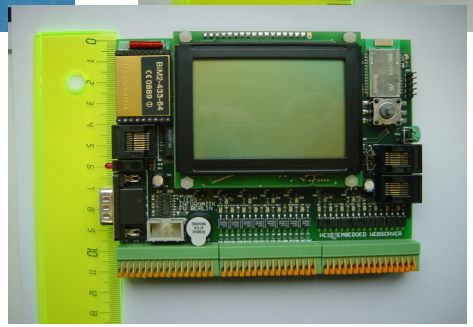






Industrial Controls

RF,
Cable-based



Embedded
Web Server

GPRS,
WLAN,
…



Client

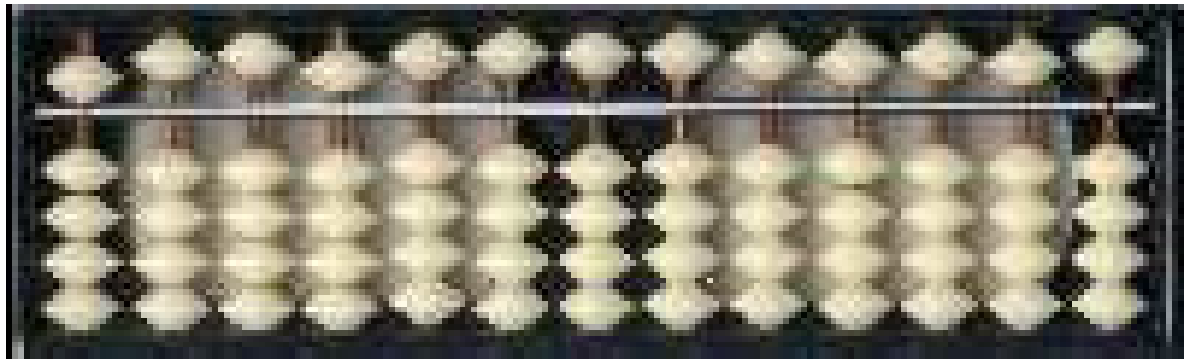
© 2017 Burkhard Stiller

ifi

Developments in Computer Science

- Abakus a calculus board with marbles, typically wood or glas
  - Used more han 3000 years ago in China
  - Addition, subtraction, multiplication, and division
    - As well as extracting of square and cbic roots





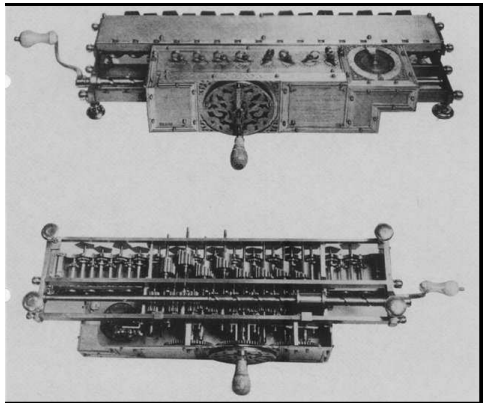
- Algorithms
  - Persian mathematician and astronomer Ibn Musa Al-Chwarismi (9th century)
  - Processing rule, which humans as well as machines can perform
  - Precision needed besides oral languages to define rules uniquely (no interpretation)

© 2017 Burkhard Stiller

Steps from 1500 to 1930 (1)

- Important steps:
  - A. Riese (1492-1559; Staffelstein) – Calculus rules for the decimal system
  - W. Schickard (1592-1635; Tübingen) – First computing device (Rechenmaschine)
  - B. Pascal (1623-1662; Clermont) – Computing device with 6 positions
  - G. Leibniz (1646-1716; Leipzig) – Machine for 4 basic arithmetics
  - P. Hahn (1739-1790; Kornwestheim) – 1st mechanical computing device





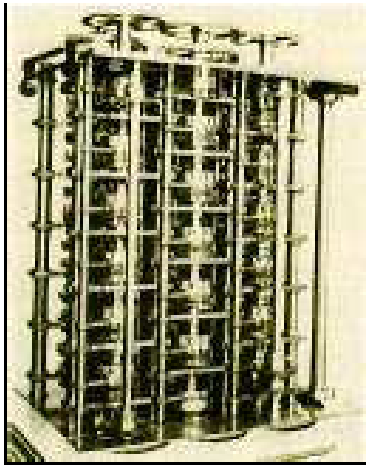


© 2017 Burkhard Stiller

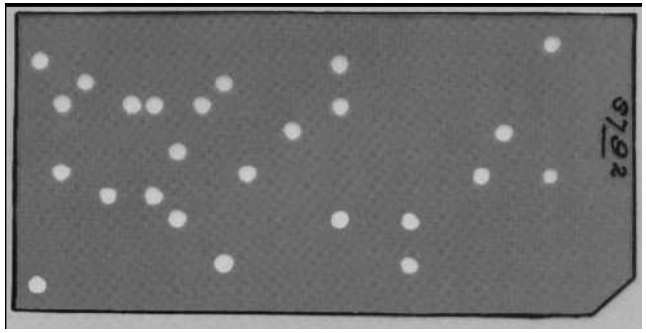
Steps from 1500 to 1930 (2)

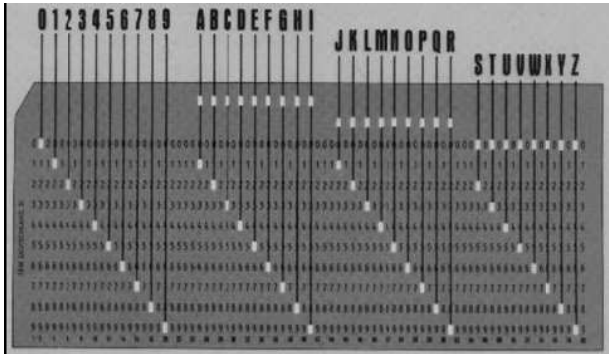
- Charles Babbage (1792-1871) –
  Principle of the ”Analytical Engine“

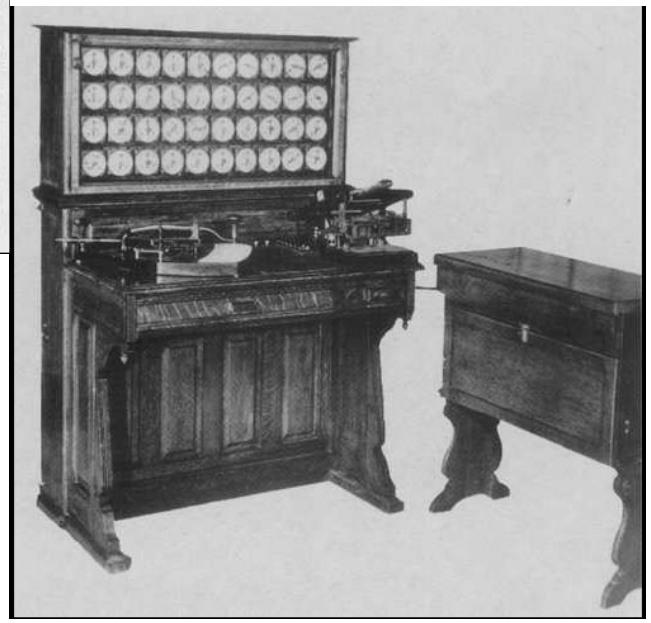
- Hermann Hollerith (1860-1929) –
  Inventor of the punch hole card



Alter in Jahren	bis 5	bis 10	bis 20	bis 30 ●	bis 40	bis 50	bis 60	bis 70	bis 80	über 80
Familien-stand	ledig	verh. ●	gesch.	Zahl der Kinder	1	2 ●	3	4	5	über 5
Beruf	Ind.-Arb.	Land-Arb.	Kfm.-Ang. ●	Leit. Ang.	Staatsdienst	Selbständ.	Sonst.	Bürger-recht	ja ●	nein
Religion	evang.	kath. ●	jüd.	sonst.	monatl. Eink.	bis 100 $	bis 200 $ ●	bis 500 $	über 500 $	





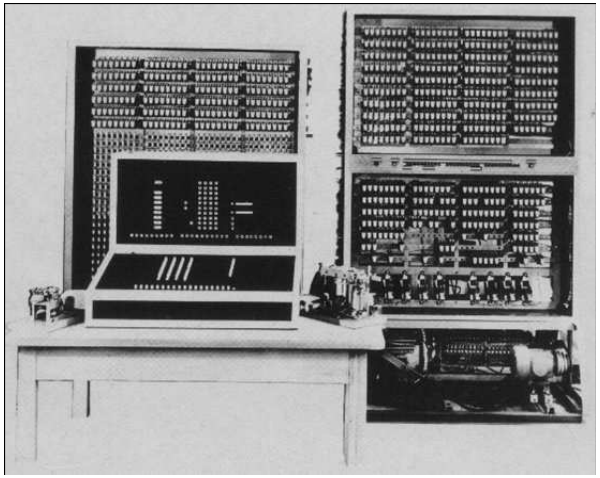


© 2017 Burkhard Stiller

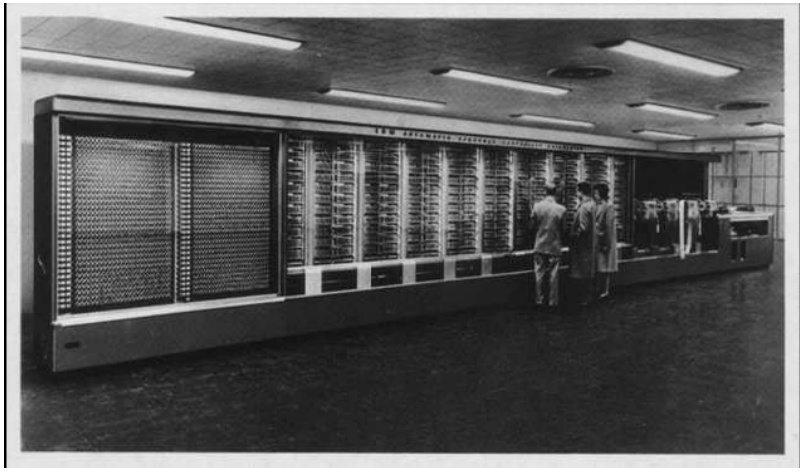
ifi

1st Operational Computer (1940‘s)

- Starting with electrical engineering advances electromechanical components appeared
  - Konrad Zuse built those in Berlin in 1941
  - The electromechanical Z3 used 2,600 air gap switches (Relais) and 64 memory positions with 22 Bit each
  - Multiplications worked in about 3 seconds



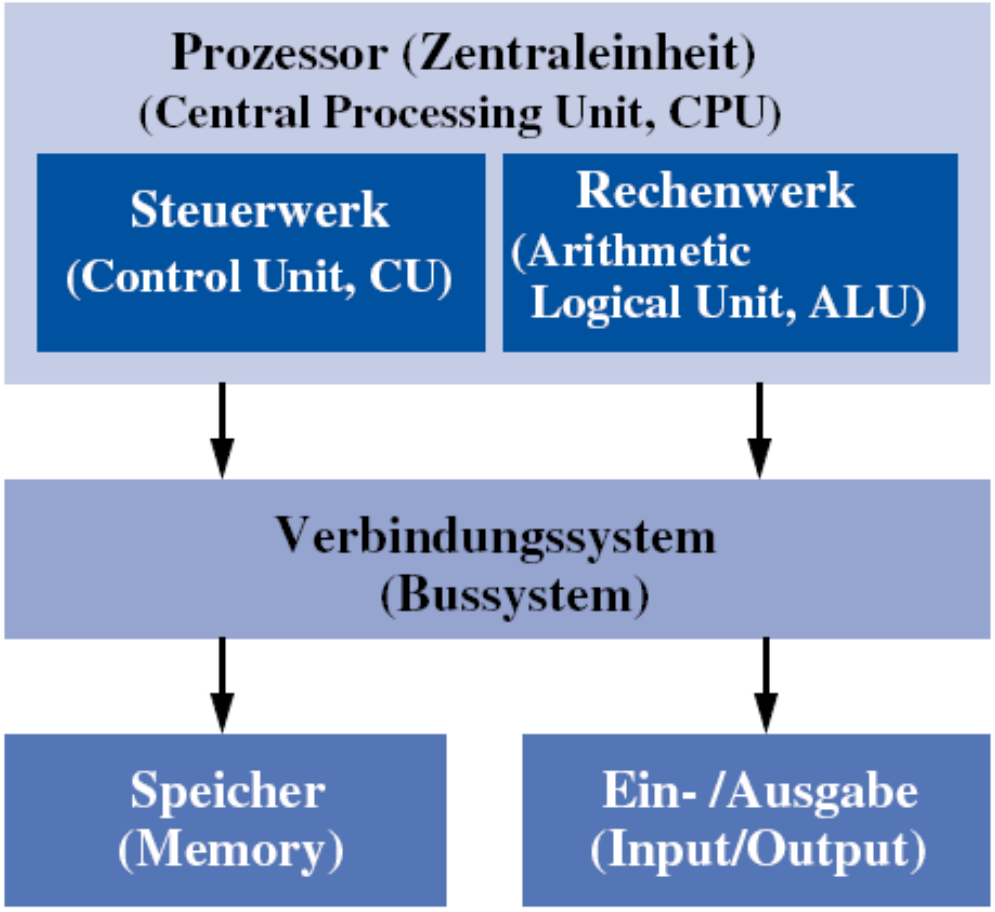
- Howard Aiken built in 1944 – in collaboration with Harvard University and IBM – the partially program-driven computing machinery Mark I.
  - Used about 100,000 parts
  - About 15 m long
  - Addition in about 1/3 second.
  - Multiplication in about 3 seconds.



© 2017 Burkhard Stiller

John von Neumann Architecture

- Mid 1940 John von Neumann developed the fundamental principles of a computer.
  - Central Processing Unit (CPU)
    - Control Unit
    - Arithmetic Logical Unit (ALU)
  - Bus system for interconnections
  - Memory for programs and data
  - Input/ouput interfaces/devices
  - Constrained jumps and branches

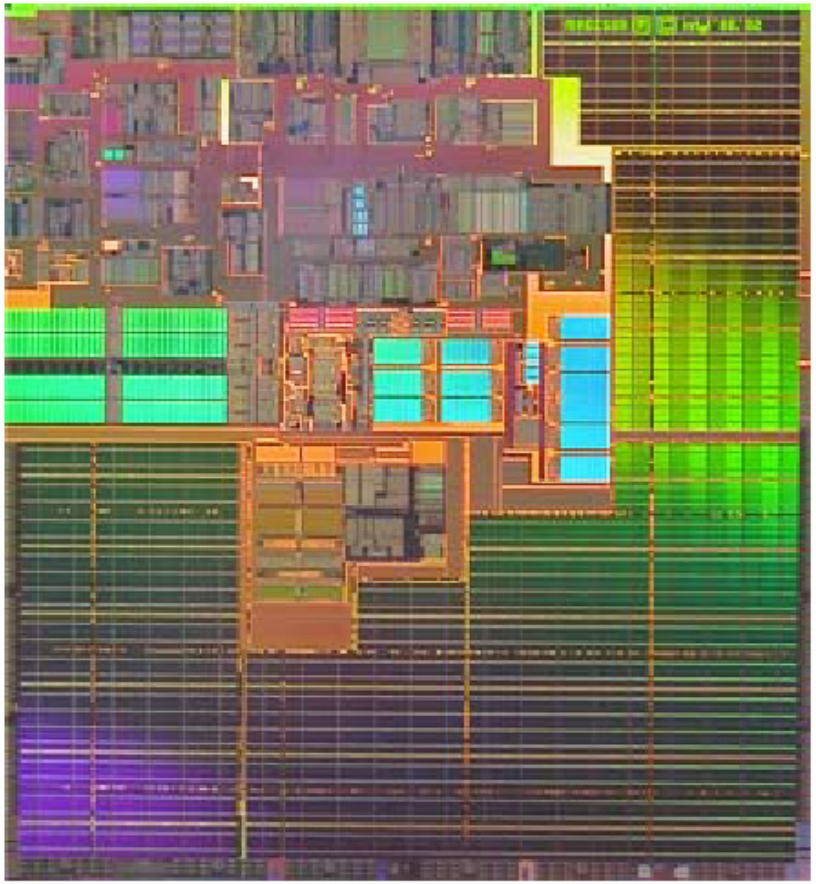


© 2017 Burkhard Stiller

Micro Processors – The Formula 1 in CS

- “Das Wettrennen um die schnellsten Mikroprozessoren könnte man als Formel 1 der Computertechnik bezeichnen.” (Die ZEIT vom 18. März 1994)
  - “The race for the fastest micro processors could be named as the formula 1 of computer science.”

- The **increased performance** of micro processors was reached due to:
  - Increased number of gates per chip
  - Increased clock speed
  - Expansion into three dimensions (started)
  - Advances in hardware design technology (architecture, micro-architecture, tools)



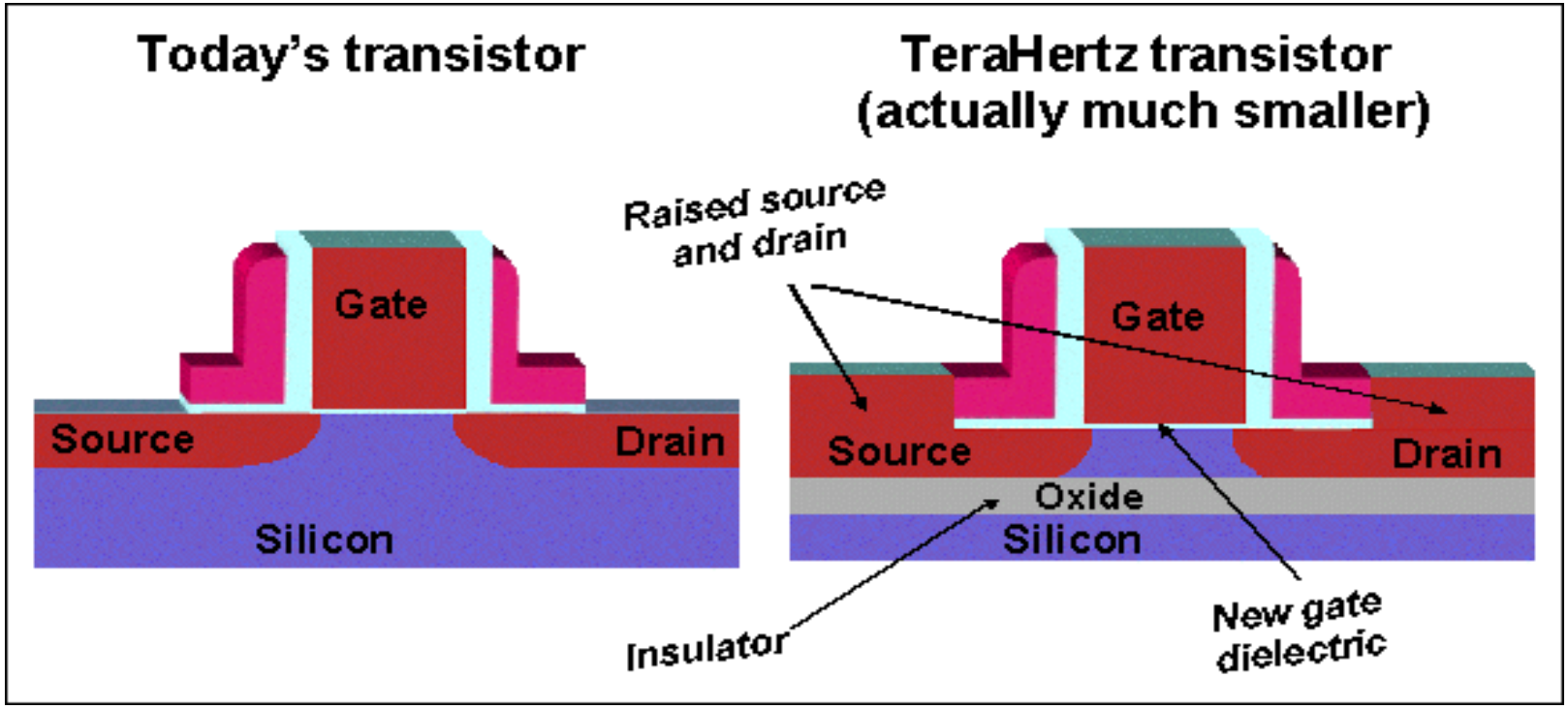
Intel Itanium 2 (Madison)

© 2017 Burkhard Stiller

Example: Terahertz Transistor (Intel)

- Peak Transit Frequency: 2.63 THz, which equals a 0,38 ps gate delay!
  - Only at 0,000 000 000 000 38 s ...
  - Or less than 76 µm path for electromagnetic waves...

Tera: 10^{12}
pico: 10^{-12}
micro: 10^{-6}

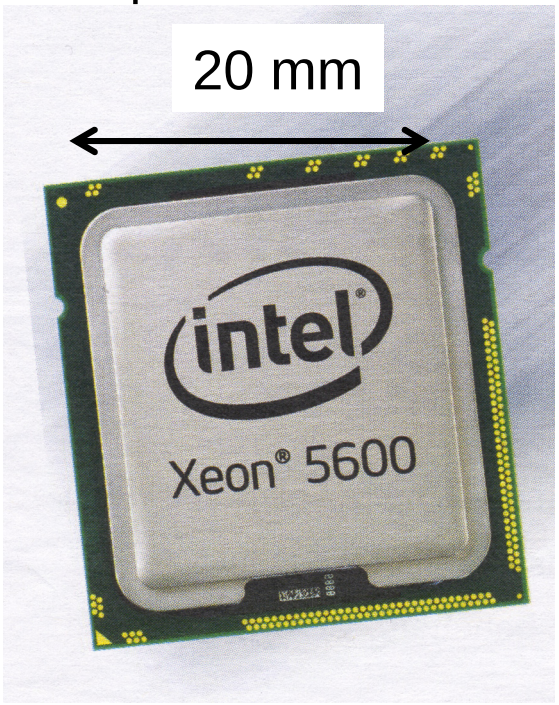


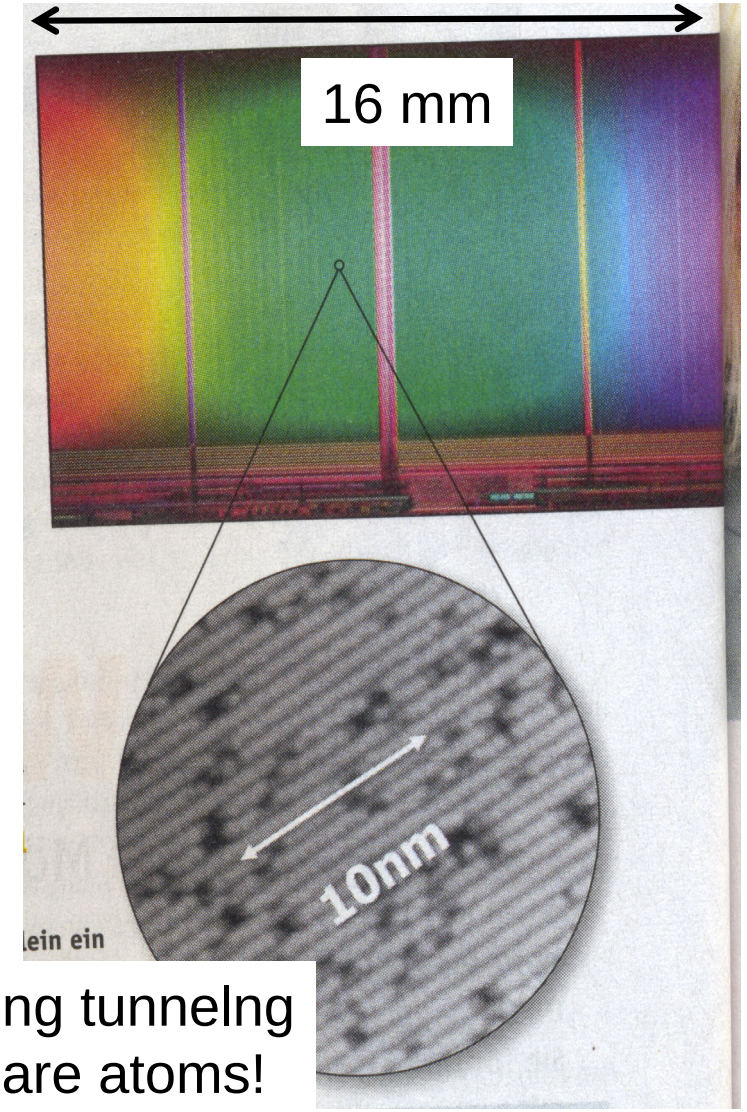
© 2017 Burkhard Stiller

ifi

Circuits, CPUs – At Scale

- 25 nm NAND memory chips
  - 8 GByte memory with 167 mm² surface area
  - Stackable – since 2010 done
- Intels XEON 5600 Series
  - 32 nm technology
  - Up to 6 Cores





Picture of scanning tunnelng microscope: white dots are atoms!

© 2017 Burkhard Stiller

ifi

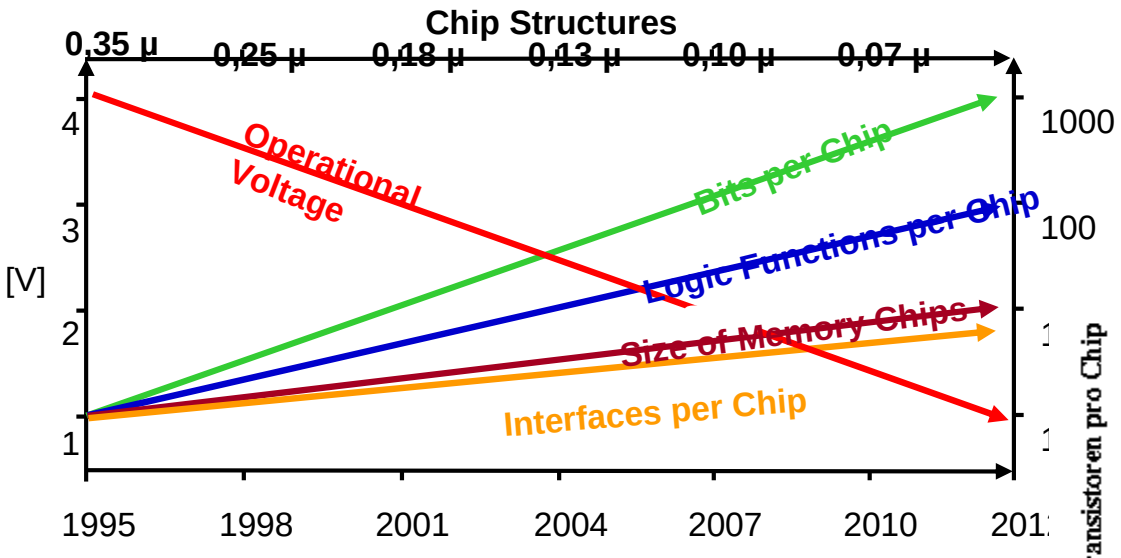
Moore‘s Law

- Number of transistors per (processor) chip doubles every 2 years.
- Computing power of high speed processors doubles every 18 months.
- At the same costs micro electronics delivers double the performance every 2 years.
- A chip manufacturing plant was the most expensinve single investment into a factory in 2002 (10 Billion US-Dollar).
- Only cooperations of larger companies will drive innovation:
  - EUVLLC (extrem ultraviolet limited liability company) from AMD, Motorola, Intel

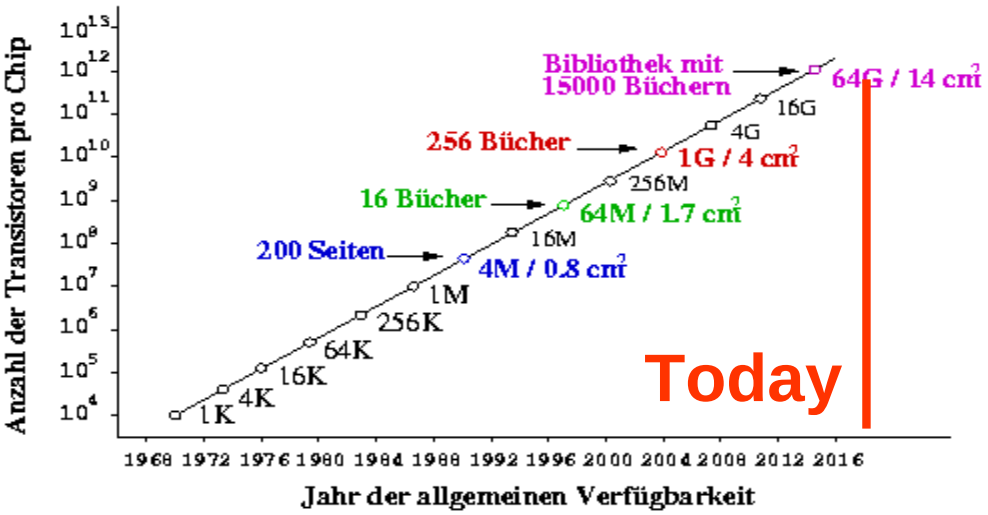
© 2017 Burkhard Stiller

Trends

- More transistors per chip
  - SIA Roadmap for technology trends
    - SIA = American Semiconductor Industry: http://public.itrs.net/
- Enhanced power at lower energy demand



- Memory size of memory chips



© 2017 Burkhard Stiller

ifi

— Computer Engineering —
Module 2: Arithmetics



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2017 Burkhard Stiller

ifi

Module 2: Arithmetics

- Number Systems
- Number Representation
- Basic Arithmetic Operations
- Characterers

© 2017 Burkhard Stiller

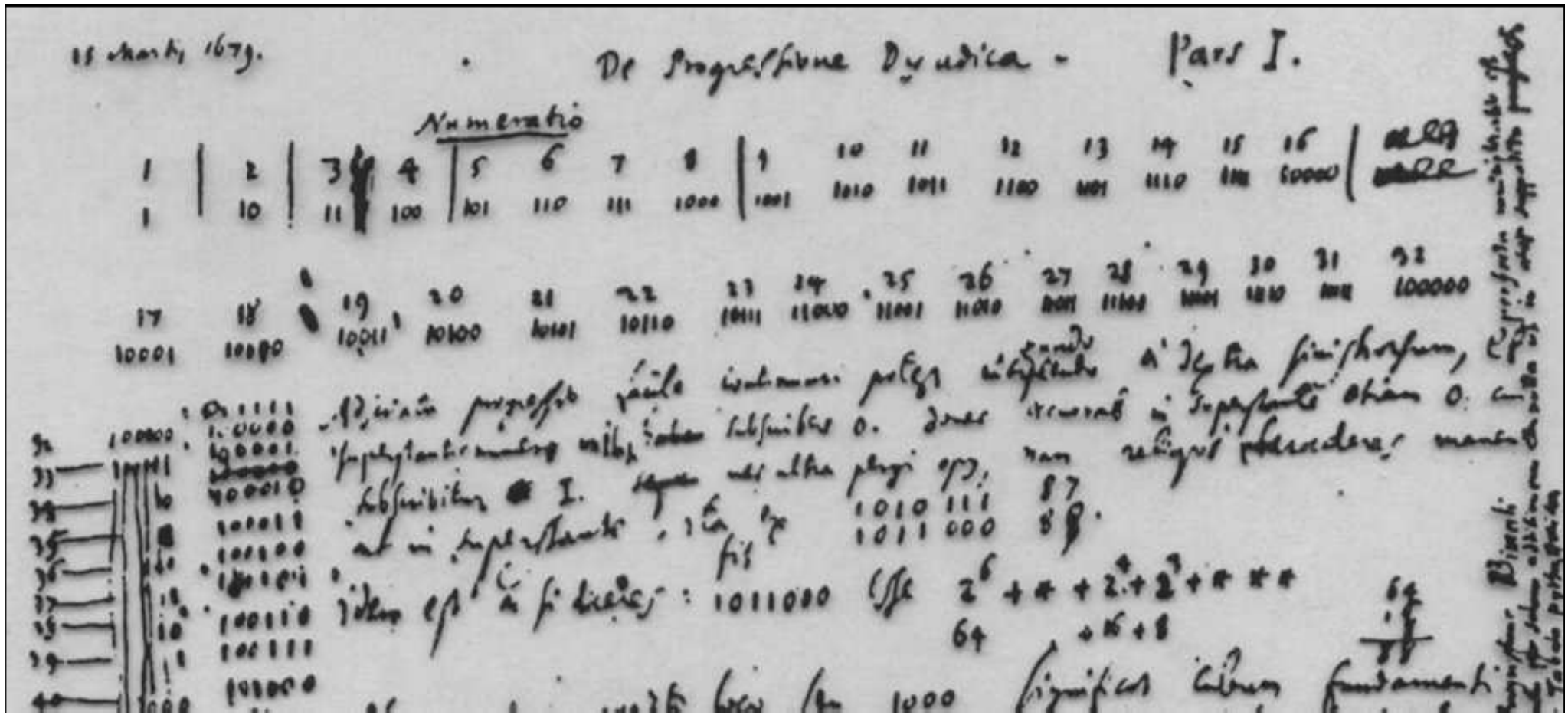
Number Systems – Roman Numbers

I	1	X	10	C	100
II	2	XX	20	CC	200
III	3	XXX	30	CCC	300
IV	4	XL	40	CD	400
V	5	L	50	D	500
VI	6	LX	60	DC	600
VII	7	LXX	70	DCC	700
VIII	8	LXXX	80	DCCC	800
IX	9	XC	90	CM	900
				M	1000

© 2017 Burkhard Stiller

Number Systems – Leibniz‘s Binary Numbers

- Leibniz Traktat from 1679
  - Possibly, basic ideas on binary number system originate from China



Formal Basics

- Arithmetics determine the important basics to handle arbitrary informationen and information units
  - To calculate, convert, save, communicate, ...
- Humans tend to calculate in the decimal system
- Computers typically “calcalate” in the binary system

→ This demands for a conversion

- Different number systems, such as the octal or hexadecimal system (correct: sedecimal), exists to compactify long binary numbers.

→ It is essential to understand the mathematical basics of number systems to be able to “calculate” with machines!

Number Systems (1)

- Typical form: Place Value System (Stellenwertsystem)
- Number representations in form of a series of numerics z_i, for which the decimal separator (“.” or “,”) is placed rigth of z_0:

  $z_n\ z_{n-1}\ \ldots\ldots\ z_1\ z_0\ .\ z_{-1}\ z_{-2}\ \ldots z_{-m}$ *E.g.*, 1234.567

- Each position i of series of numerics determines the place value, which is the power b^i of base b of the number system
- Value X_b of the number is defined by the sum of all values of all separate positions $z_i b^i$:

$$\mathbf{X_b = z_n\, b^n + z_{n-1}\, b^{n-1} + \ldots + z_1\, b + z_0 + z_{-1}\, b^{-1} + \ldots + z_{-m}\, b^{-m}} = \sum_{i=-m}^{n} z_i b^i$$

Number Systems (2)

- Relevant number systems in computer science

b	Number System	Name of System
2	Binary system	Binary number
8	Octal system	Octal number
10	Decimal system	Decimal number
16	Hexadecimal system (Sedecimal system)	Hexadecimal number (Sedecimal number)

- Hexadecimal system: The “numerics” 10 to 15 are represemted by the characters A to F
- Binary system: the number system used for computers today
- Octal and Hexadecimal system can be converted into binary system easily, and they are better remembered for humans than long 0-1 series

© 2017 Burkhard Stiller

Number Systems (3)

- Each separate binary position (0 oder 1) stored in memoryis defined as the **Bit**, an abbreviation for „*BInary digiT*“. This is the smallest unit a binary computer can handle.
- Obviously, the binary number systems is a place value system, too. Each position‘s value equals to the value of a power to 2:

$$10011 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1 \cdot 16 + 0 \cdot 8 + 0 \cdot 4 + 1 \cdot 2 + 1 \cdot 1$$

$$= 19 \text{ (decimal system)}$$

- Examples:

$(10011)_2$	=	$(19)_{10}$	oder:	$10011_{(2)}$	=	$19_{(10)}$
$(1011)_2$	=	$(11)_{10}$	oder:	$1011_{2)}$	=	$11_{(10}$
$(11010110)_2$	=	$(214)_{10}$	oder:	$11010110_{(2)}$	=	$214_{(10)}$

Number Systems (4)

Dual	Oktal	Dezimal	Hexadezimal	Zwölfersystem
0	0	**0**	0	0
1	1	**1**	1	1
10	2	**2**	2	2
11	3	**3**	3	3
100	4	**4**	4	4
101	5	**5**	5	5
110	6	**6**	6	6
111	7	**7**	7	7
1000	10	**8**	8	8
1001	11	**9**	9	9
1010	12	**10**	a	a
1011	13	**11**	b	b
1100	14	**12**	c	10
1101	15	**13**	d	11
1110	16	**14**	e	12
1111	17	**15**	f	13
10000	20	**16**	10	14
10001	21	**17**	11	15

© 2017 Burkhard Stiller

Number Systems (5)

- In case of fractional numbers, a dot separates the number into the integral and fractional part. A series formula determines its value:

$$n = \sum_{i=-M}^{N-1} b_i \cdot B^i \quad \text{with}$$

  - B = base of the number system $(B \in \mathbb{N}, B \geq 2)$
  - b = numericals $(b_i \in \mathbb{N}_0, 0 \leq b_i < B)$
  - N = number of position in front of dot
  - M = number of position after the dot

Conversion: Euclide‘s Algorithm

- Conversion from decimal system into a system with base b
- Representation of number

  $Z = z_n 10^n + z_{n-1} 10^{n-1} + ... + z_1 10 + z_0 + z_{-1} 10^{-1} + ... + z_{-m} 10^{-m}$

  $= y_p b^p + y_{p-1} b^{p-1} + ... + y_1 b + y_0 + y_{-1} b^{-1} + ... + y_{-q} b^{-q}$

  All numericals are calculated step-wise, starting off with the highest valued position.

- Step 1: Calculate **p** according to $\mathbf{b^p \le Z < b^{p+1}}$ (define i = p)
- Step 2: Seek $\mathbf{y_i}$ and the remainder $\mathbf{R_i}$ by dividing $\mathbf{Z_i}$ by $\mathbf{b^i}$: $\mathbf{y_i = Z_i}$ **div** $\mathbf{b^i}$; $\mathbf{R_i = Z_i}$ **mod** $\mathbf{b^i}$;
- Step 3: Repeat step 2 for i = p-1, … and replace after ach step **Z** by $\mathbf{R_i}$, until $\mathbf{R_i = 0}$ or until $\mathbf{b^i}$ (the conversion error acceptable) will be small enough.

© 2017 Burkhard Stiller

Example: Euclide‘s Algorithm

Conversion of 15741.233_{10} i

Step 1: $16^3 \leq 15741.233 < 16^4$ ➔

Step 2: $15741.233 : 16^3 = 3$

Step 3: $3453.233 : 16^2 = D$

Step 4: $125.233 : 16 = 7$

Step 5: $13.233 : 1 = D$

Step 6: $0.233 : 16^{-1} = 3$

Step 7: $0.0455 : 16^{-2} = B$

Step 8: $0.00253 : 16^{-3} = A$

Step 9: $0.000088593 : 16^{-4} = 5$

➔ **$15741.233_{10} \approx 3D7D$**



© 2017 Burkhard Stiller

ifi

Conversion: Horner Scheme

- Conversion from decimal system into a number system of base b
- Adapted Horner Schemas
- Start: separate integral and fractional parts of number to be converted
- Here conversion of integral part:
- An integer $X_b = \sum_{i=0}^{n} z_i b^i$ can be factored out, resulting in the form:

$$X_b = ((...(((y_n\ b + y_{n-1})\ b + y_{n-2})\ b + y_{n-3})\ b\ ...\ )\ b + y_1)\ b + y_0$$

- Finally, convert the fractional part in the same manner and concatenate the result.

Example: Horner Scheme

- Divide the decimal number successively by the value of base **b**.
- Integral remainders define the numericals of $\mathbf{X_b}$ in the order of least significant to highest significant position.

Convert 15741_{10} into the hexadecimal system:

$15741_{10} : 16 = 983$	Remainder 13	(D_{16})
$983_{10} : 16 = 61$	Remainder 7	(7_{16})
$61_{10} : 16 = 3$	Remainder 13	(D_{16})
$3_{10} : 16 = 0$	Remainder 3	(3_{16})

➔ **$15741_{10} = 3D7D_{16}$**

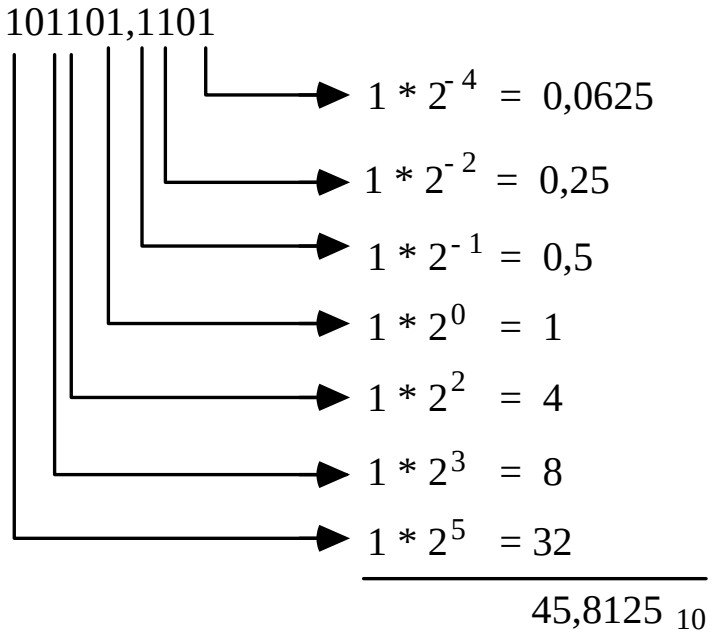
© 2017 Burkhard Stiller

Conversion/Example: Base b → Decimal System

- Values of all separate positions of the number to be converted are represented in the target number system and summed up.
- Value X_b equals the sum of all separate positions $z_i b^i$:

$$X_b = z_n b^n + z_{n-1} b^{n-1} + \ldots + z_1 b + z_0 + z_{-1} b^{-1} + \ldots + z_{-m} b^{-m} = \sum_{i=-m}^{n} z_i b^i$$

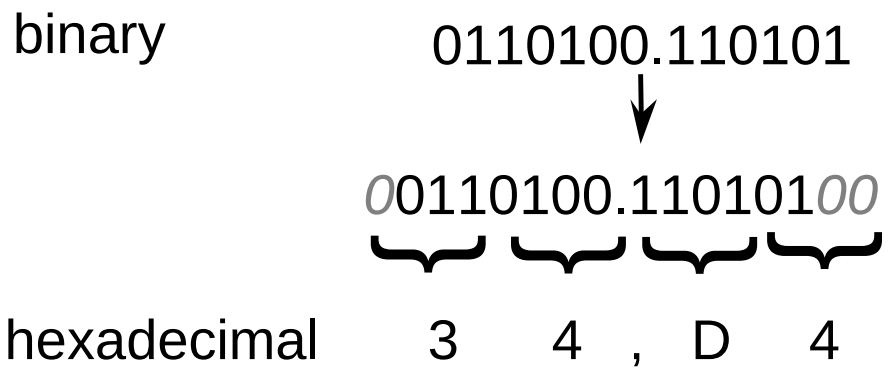
- Example: Conversion of 101101.11012 into the decimal system



© 2017 Burkhard Stiller

Conversion of Arbitrary Place Value Systems

- Convert number into decimal system and apply Euclide‘s algorithm or the Horner scheme to the target system
- Special case:
  - In case of a base b being a power of another base, multiple places can be combined by a single numerical or vice versa.
- Conversion of 0110100.110101_2 into the hexadecimal system
- $2^4 = 16 \Rightarrow$ 4 binary places $\rightarrow$ 1 hexadecimal place



Add zeros to complete groups of 4 characters: before vs. after separator!

© 2017 Burkhard Stiller

Module 2: Arithmetics

- Number Systems
- Number Representation
- Basic Arithmetic Operations
- Characterers

Representation Options for Negative Numbers

1. Representation as absolute value plus algebraic sign
   (Betrag mit Vorzeichen)

2. Representation as ones complement or B-1 complement
   (Stellenkomplement oder Einerkomplement)

3. Representation as two‘s complement
   (Zweierkomplement)

4. Offset Representation
   (Exzeß-Darstellung)

© 2017 Burkhard Stiller

1. Absolute Value Plus Algebraic Sign

- One place used for algebraic sign (Vorzeichen)

- The MSB (Most Significant Bit, the leftmost) determines

  MSB = 0 → positive number
  MSB = 1 → negative number

  Example:

  0001 0010 = +18
  1001 0010 = -18

- Drawbacks:
  - For additions and subtractions signs habe to be handled separately.
  - Two representatios of number zero exist (with positive and with negative sign)

2. Ones Complement or B-1 Complement

- Defined as the complement of the respective positive number
  - To negate a number, each bit is separately negated
- This defined the ones complenents

$$z_{ek} = (2^n - 1) - z$$

Example: $4 = 0100_2$ ➔ $-4 = 1011_{ek}$

$$-4 = 2^4 - 1 - 4 = 11_{10} = 1011_2$$

- Negativ numbers are identified by a bit set on the first place

- Advantages compared to variant 1:
  - First place does not need for addition and subtraction a separate handling
  - But: Still, two representations of the number zero exist.

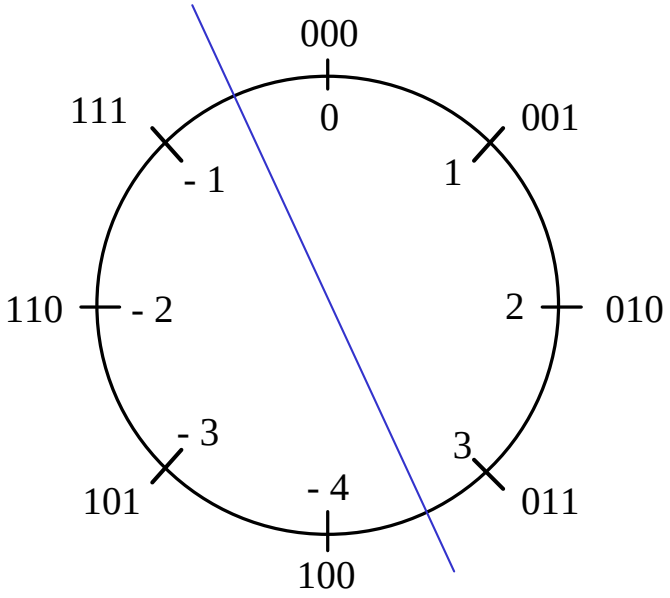
3. Two‘s Complement (1)

- Add to the ones complement the value “1”
- The result is the two’s complement: $z_{zk} = 2^n - z$

0 . . . 0 → Ones Complement 1 . . . 1

→ Two’s complement 0 . . . 0

- Drawback:
  - Asymmetric number space
    The smallest negative number is absolute value-wise larger (by 1) than the largest positive number.



- Example: 3 bit two’s complement numbers:
  - The MSB defines the sign of the number

© 2017 Burkhard Stiller

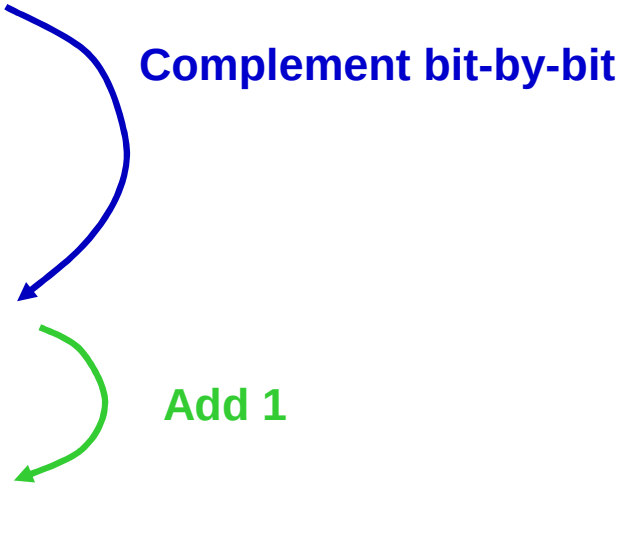
3. Two‘s Complement (2)

- Place value of MSB for a two‘s complement number with ***n+1*** Bit is -2^n:
- $z_n\ z_{n-1}\ \ldots\ z_0$ do have the value: $Z = -z_n \cdot 2^n + z_{n-1} \cdot 2^{n-1} + \ldots + z_0$

Example:

Die Zahl -77_{10} soll mit 8 Bit dargestellt werden

$$77_{10} = 0100\ 1101_2$$



Absolute value/sign: $-77 = 1100\ 1101_2$

Ones complement : $-77 = 1011\ 0010_2$

Two‘s complement : $-77 = 1011\ 0011_2$

4. Offset Representation

- Applied to the representation of the exponent for floating point numbers (see later)
- Representation of such a number is done as a characteristic (bias).
- The full available number space is shifted by an addition of a constant (offset) such that the smallest (negative) number will be represeted as **0…0**.
- For ***n*** places the offset is 2^{n-1}

- The numer space is asymmetric.

Summary and Comparison of All Options

Representation as		
Decimal		Characteristic
-4		0 0 0
-3		0 0 1
-2		0 1 0
-1		0 1 1
0		1 0 0
1		1 0 1
2		1 1 0
3		1 1 1

© 2017 Burkhard Stiller

Fixpoint and Floating Point Numbers

- Number representation in daily life:

Numericals	0 ... 9
Sign	+ -
Decimal seperator	“,” or “.”

- Number representation inside a machine:

Binary numericals	0, 1

➔ Special agreements needed to represent signs and decimal separator besides the value of the number itself

- Representation of sign:
  - See above
- Representation of decimal separator with two options:
  - Fix point
  - Floating point

Fix Point Numbers (1)

- Agreement:
  - The decimal separator is placed inside a machine word such that it will never change its position (“fix point”)
  - Typically, it is placed after the last bit
- Any other numbers can be converted to this agreement by applying scale factors
- Negative numbers:
  - Typically represented as two’s complements
- Today, fix point numbers are typically not used any more from the hardware except for inputs or outputs!

© 2017 Burkhard Stiller

Fix Point Numbers (2)

- Data type "integer" (Ganzzahlen) defines a specific fix point format.
- Different programming languages provide the definition of integers with different lengths.
- Example "C": "short int", "int", "long int", "unsigned"

	DEC-VAX (first generation)		IBM-PC, Apple Macintosh	
Data type	Number of Bits	Number Space	Number of Bits	Number Space
short int	16	-2^{15} ..$2^{15}-1$	16	-2^{15} ..$2^{15}-1$
int	32	-2^{31} .. $2^{31}-1$	16	-2^{15} ..$2^{15}-1$
long int	32	-2^{31} .. $2^{31}-1$	32	-2^{31} .. $2^{31}-1$

© 2017 Burkhard Stiller

Floating Point Numbers (1)

- Numbers with their abolute values being very large or very small floating point numbers are used.
  - These numbers determine a subset of der rational numbers
- Representation done in a semi logarithmic scale

$$X = \pm \text{Mantissa} \cdot b^{\text{Exponent}}$$

- Base b is fixed for a dedciated floating point representation (typically 2 or 16)
  - Thus, the base does not have to be represented in the machine word itself!
- Floating point numbers are represented often with absolute value and sign, but not as a two‘s complement!

© 2017 Burkhard Stiller

Floating Point Numbers (2)

- Mantissa defines the decimal separator at a fxed position, typically agreed upon as left of the MSB.
- Exponent is an integer represented with its characteristic.
- For the mantissa and the characteristic a fixd number of memory bits are defined in the systems architecture.
- Characteristic‘s length y-x defines the size of the number space.
- Mantissa‘s length x defines the precision of the representation,

$$\text{Decimal number} = (-1)^{Vz} * (0,\text{Mantissa}) * b^{\text{Exponent}}$$

$$\text{Exponent} = \text{Characteristic} - b^{(y-1)-x}$$

Vz = sign



© 2017 Burkhard Stiller

Normalization

- The representation of an arbitrary floating point number is not unique.
  - Example: $3.0 * 10^0 = 0.3 * 10^1$
- Enforcing a dedicated representation reachable by a normalized floating point representation rule
  - Simplifies calculations, e.g., representation starts with “0.X” [X<>0]
- Since zero cannot be represented in a normalized format, it will be represented by a dedicated bit sequence. Thus, the first place of the mantissa in normalized form is always 1.
  - Thus, this “1” does not have to be part of the machine word, which saves one bit or increases the granularity by one bit under the assumption of a fixed length for the machine word.
- In case of all arithmetical operations as well as conversions, this “missing” 1 have to be considered!

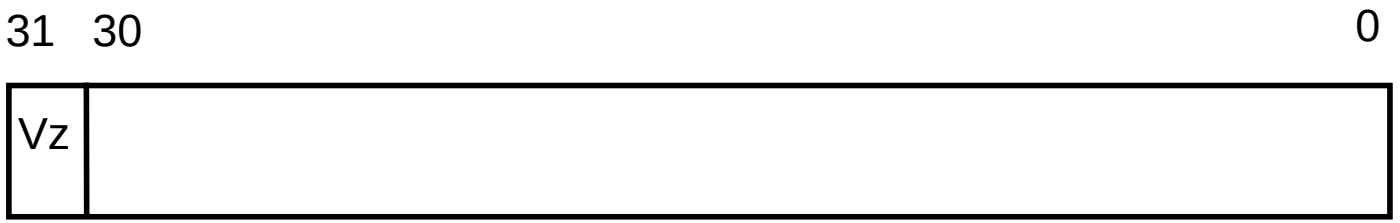
© 2017 Burkhard Stiller

Examples (1)

3 different machine word formats with 32 Bit each and base b = 2.

The number 7135_{10} is represented in all formats below.

a) Fix Point Representation with Two‘s Complement



Vz = sign

0 000 0000 0000 0000 0001 1011 1101 1111_2 = 0000 $1BDF_{16}$

Examples (2)

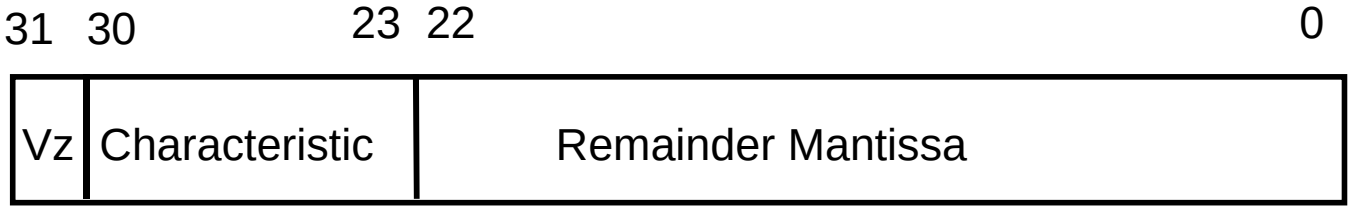
Vz = sign

b) Floating Point Number, normalized:



$0\ 100\ 0101\ 1\ 110\ 1111\ 0111\ 1100\ 0000\ 0000_2 \quad = \quad 46EF\ 7C00_{16}$

c) Floating Point Number, normalized, first "1" hold implicitly:



$0\ 100\ 0101\ 1\ 101\ 1110\ 1111\ 1000\ 0000\ 0000_2 \quad = \quad 46DE\ F800_{16}$

http://www.h-schmidt.net/FloatConverter/IEEE754.html

Representable Number Space (1)

- The amount of numbers to be represented (combinations of all bits) is for all three case identical:

  2^{32}

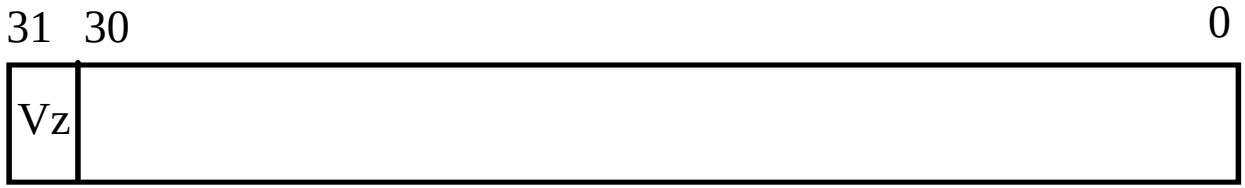
- However, the range and density of those representable numbers on the number line (Zahlenstrahl) differs dramatically!

© 2017 Burkhard Stiller

Representable Number Space (2)

- Format a: Numbers between -2^{31} and $2^{31}-1$

Vz = sign



- Format b:



negative numbers: $-(1-2^{-23}) \cdot 2^{127}$ ... $-0.5 \cdot 2^{-128}$,

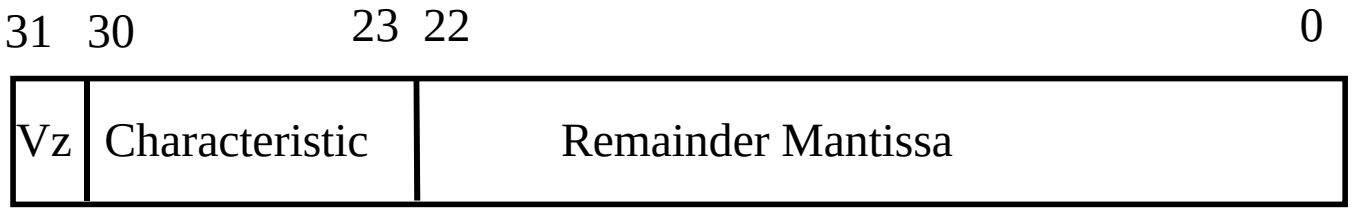
positive numbers : $0.5 \cdot 2^{-128}$ ... $(1-2^{-23}) \cdot 2^{127}$,

and *Zero*

© 2017 Burkhard Stiller

Representable Number Space (3)

- Format c: normalized floating point representation

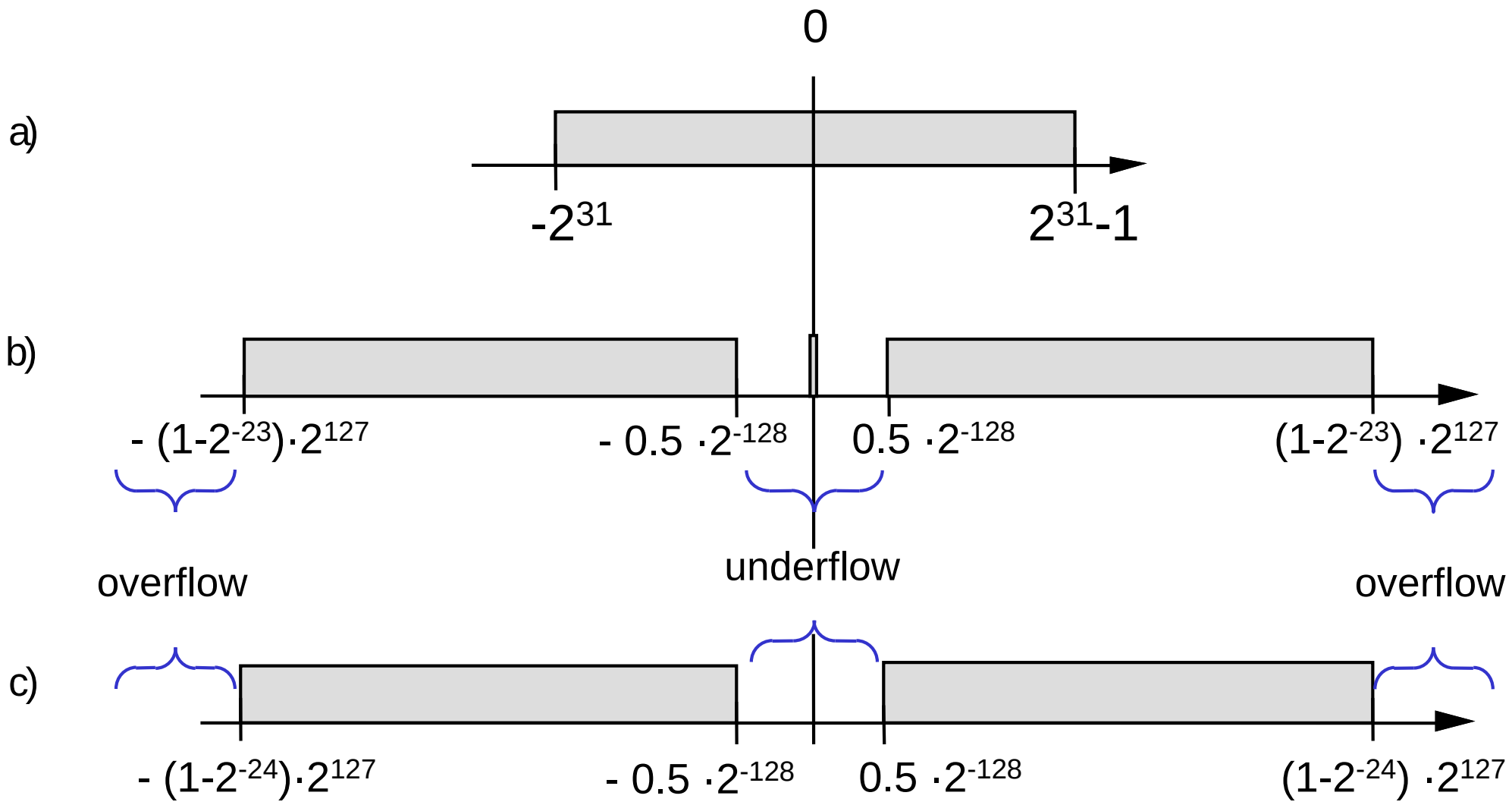


negative numbers: $-(1-2^{-24})\cdot 2^{127}$ ... $-0.5\cdot 2^{-128}$

positive numbers: $0.5\cdot 2^{-128}$ ... $(1-2^{-24})\cdot 2^{127}$

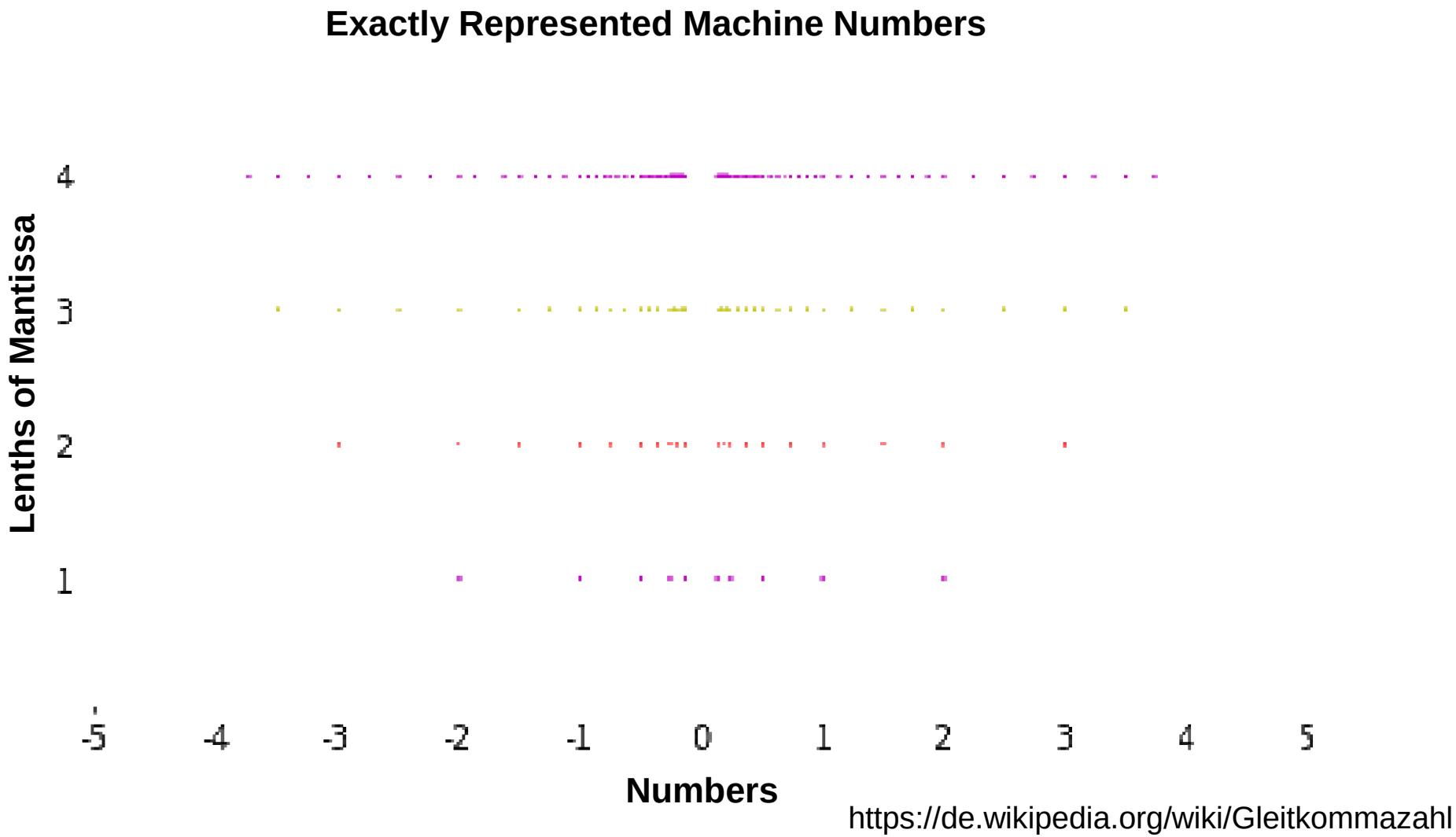
A zero cannot be represented!

Representable Number Space (4)



© 2017 Burkhard Stiller

Representable Number Space (5)



https://de.wikipedia.org/wiki/Gleitkommazahl

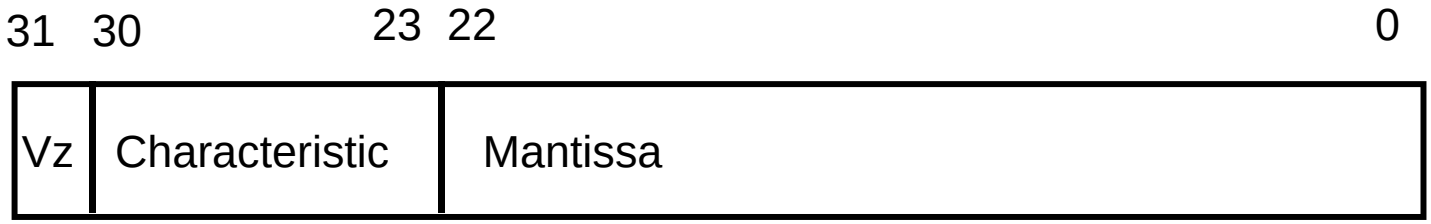
Characteristical Numbers

- To be able to compare different floating point number representations 3 characteristical numbers are defined:
- maxreal the largest possible and representable normalized positive number
- minreal the smallest possible and representable normalized positive number
- smallreal the smallest number, which can be added to 1, to reach a representable number with a value different than 1.

Example

- In format b) from above:

Vz = sign



- maxreal = $(1 - 2^{-23}) \cdot 2^{127}$ minreal = $0.5 \cdot 2^{-128}$

- The number 1 can be normalized and reads as $0.5 \cdot 2^{1}$.

- The next representable number shows within the mantissa in addition to 1 in Bit 22 a 1 in Bit 0.

- smallreal = $0.00000000000000000000001_2 \cdot 2^{1}$,
  thus, smallreal = $2^{-23} \cdot 2^{1} = 2^{-22}$

© 2017 Burkhard Stiller

Inaccuracy

- The difference between two subsequent numbers in floating point representation increases exponentially with the size of these numbers.
  In case of fix point representations the difference remains constant.
  - The representation of large number leads to a large inaccuracy!

- All rules valid mathematically for real numbers do not apply for floating point representations in machine words!

- Explicitly, this is an important statement for the data type of `real` in a wider range of programming languages.

Example

- The associative law (x + y) + z = x + (y + z) may not apply necessarily, even if no overflow or underflow happened:

Assume that x = 1; y = z = smallreal/2

$$
\begin{aligned}
(x + y) + z &= (1 + \text{smallreal}/2) + \text{smallreal}/2 \\
&= 1 + \text{smallreal}/2 \\
&= 1
\end{aligned}
$$

$$
\begin{aligned}
x + (y + z) &= 1 + (\text{smallreal}/2 + \text{smallreal}/2) \\
&= 1 + \text{smallreal} \\
&\neq 1
\end{aligned}
$$

Note: smallreal the smallest number, which can be added to 1, to reach a representable number with a value different than 1.

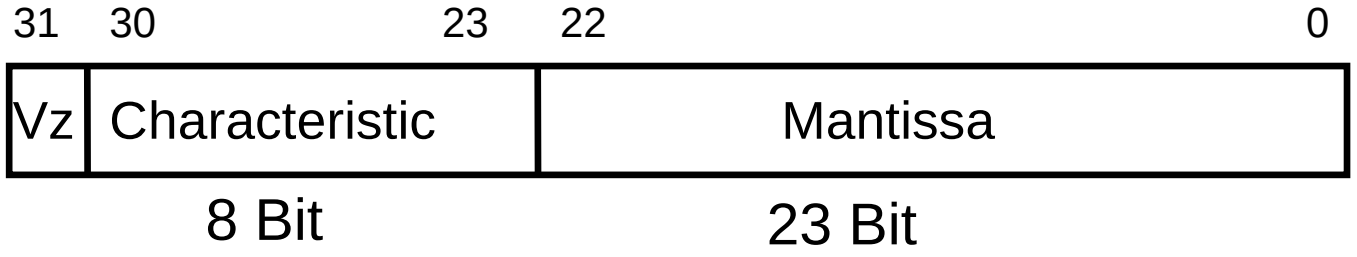
Problems due to Different Definitions

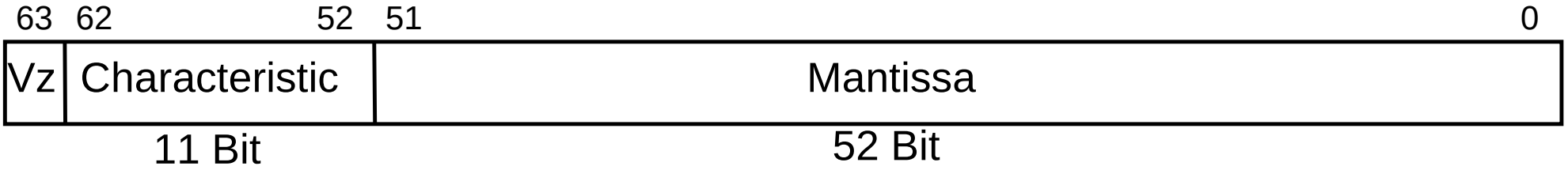
- There is an unlimited range of options to define floating point representations for a fixed number of storage words at hand:
  - Different base, representation of zero, number of places for the characteristic, number of places for the mantissa
- Thus, many vendor-specific formats arose since the mid 80‘s.
- In turn, identical programs compiled onto different hardware resulted in DIFFERENT results
  → An unacceptable situation,
  especially for deterministic calculations needed!
- Standardization became essential!

Standardization: IEEE-Standard

IEEE: Institute of Electrical and Electronics Engineers

- IEEE-P 754-Floating-Point-Standard
- Different programming languages allow for different precisions of floating point numbers:
  - *E.g.*, in C: float, double, long double
- The IEEE Standard defines different valid representations
  - IEEE single (einfach): 32 Bit
  - IEEE double (doppelt): 64 Bit
  - IEEE extended (erweitert): 80 Bit





© 2017 Burkhard Stiller

Module 2: Arithmetics

- Number Systems
- Number Representation
- Basic Arithmetic Operations
- Characterers

Addition of Binary Numbers (1)

- For any binary addition:

$0 + 0 = 0$

$0 + 1 = 1$

$1 + 0 = 1$

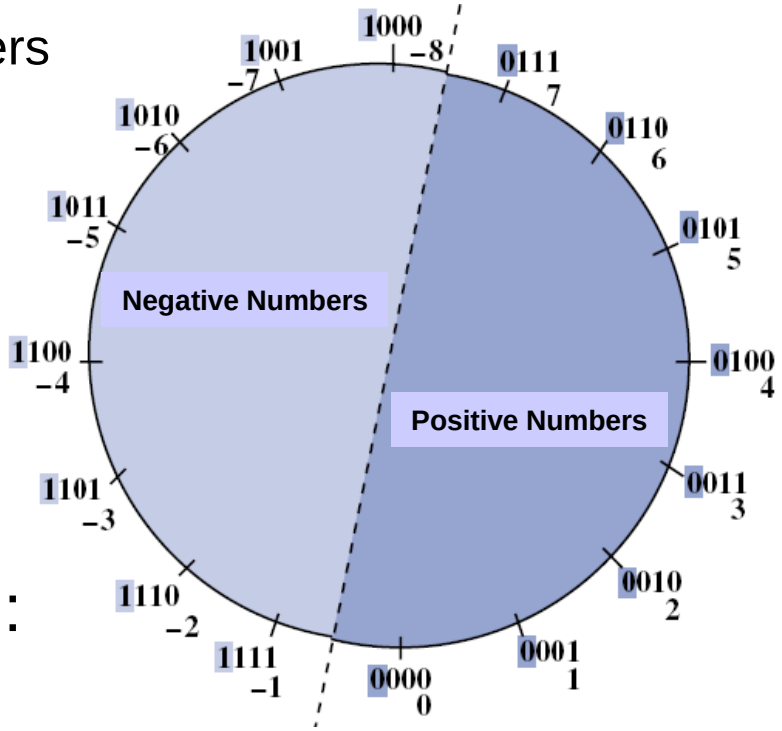
$1 + 1 = 0$ carry 1

$1 + 1 + 1$ (from carry) $= 1$ carry 1

```
 0 1 0 1 1 0 1 =  45
 0 1 1 0 1 1 0 =  54
-----------------------
 1 1 1 1       =  carry
-----------------------
 1 1 0 0 0 1 1 =  99
```

Addition of Binary Numbers (2)

- Negative numbers in absolute values/sign
  - Separate handling of signs possible, but cumbersome
  - Computer-based solution needs addition and substraction
  - Assignment of bits to positive and negative numbers
    - Ring of numbers for vier bits,
      first bit is reserved for the sign bit
  - Complement representation does not force to use any substraction, since every subtraction a - b can be replaced by the addition a + -b



- Detection of overflow wth binary addition:
  - Correct addition: both carry values are identical.
  - Carry: both carry values are not identical.
  - Technical implementation performed by an antivalence gate (XOR) (see later)

Subtraction of Binary Numbers

- Based on addition of fix point binary numbers:
  - That‘s the basis for all arithmetical operations

- Reason:
  - Subtraction ≙ Addition of the negative number
  - X - Y = X + (-Y)

Multiplication and Division (1)

- Numbers in two‘s complement are very suitable for addition and subtraction, since the sign did not have to be considered separately.
- For the multiplication this advantage does not exist.
- Solutions:
  - Two‘s complement numbers are converted to absolute value and sign. Numbers are multiplied. The result will be converted back to the two‘s complement representation.
  - Specific algorithms for multiplication for two‘s complement number (Booth Algorithms).

- Important notes for programmers:
  - Many applications do see an unknown set of additions, multiplications
  - Conversion into different formats will cause delays in overall calculations!

Multiplication and Division (2)

- An integer (ganzzahlige) multiplication and division is performed by a subsequent addition.

- Special cases of the multiplier and divisors (2, 4, …) eases and fastens the multiplication and division due to a right or left shift one or multiple bits:
  With 2 by 1 Bit, with 4 by 2 Bit, with 8 by 3 Bit, etc.

Decimal	: $(20)_{10}$ x $(8)_{10} = 160_{10}$
Binary	: $(10100)_2$ x $(1000)_2 = (10100000)_2$ $[10100 \mid 000]$
Decimal	: $(20)_{10} : (4)_{10} = 5_{10}$
Binary	: $(10100)_2 : (100)_2 = (101)_2$ $[101 \mid 00]$

Multiplication

- Numbers represented as absolute value/sign numbers are just multiplied.
  - Absolute values of those numbers are multiplied as if they were postive numbers.
  - The sign of the results will be calculate as an antivalence (XOR) of the two signs of both multipliers: sign (X•Y) = sign (X) XOR sign (Y)
- A multiplication floating point numbers means to multiply the mantissas and adding their exponents:

$$m_1\, b^{e1} \cdot m_2\, b^{e2} = (m_1 \cdot m_2)\, b^{e1+e2}$$

- If the mantissa is represented in the absolute value/sign representations, the regular multiplication method does apply
  - The result may be required to be normalized!
  - For the addition of the characteristics $\mathbf{c_1 = e_1 + o}$ and $\mathbf{c_2 = e_2 + o}$ the sum needs to be corrected to the right offset **o** to reach the correct result‘scharacteristic $\mathbf{c = (e_1 + e_2) + o}$ zu erhalten.

Division (1)

- Division follows same arguments as with the multiplication
- Paper-based division used for determing suitable algorithms
- Examples:

$$
\begin{array}{rcl}
1224 & : & 12 \\
-\,12\downarrow & & \\
\hline
\;02\downarrow & & \\
\hline
\;024 & & \\
-\;24 & & \\
\hline
\;\;0 & &
\end{array}
\qquad = \quad 102
$$

- Binary division only knows two values as outcomes

  0 $\rightarrow$ Divisor > current Dividend

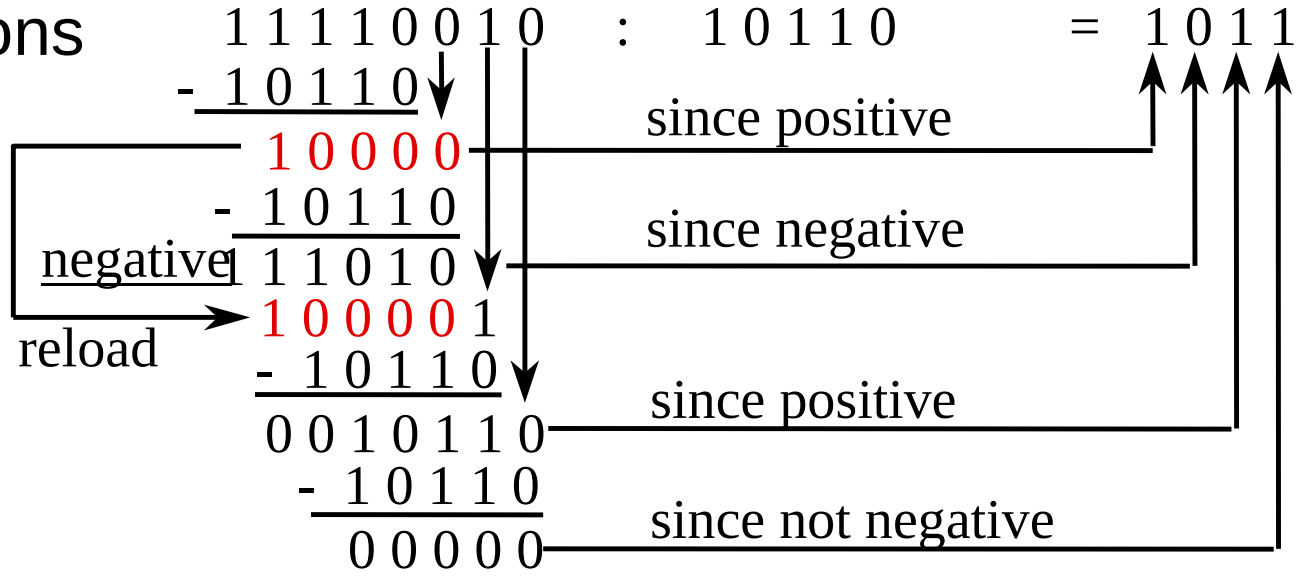
  1 $\rightarrow$ Divisor $\leq$ current Dividend

- In case of “manual” division the human recognizes immediately if the result is a “0” and if “an additional place” is required.

Division (2)

- A computer-based recognition shows 3 alternatives:
  - 1. Comparator to compare divisor with current dividend
  - 2. Subtraction: In case of a negative result the old dividend's value is reloaded
  - 3. Subtraction: In case of a negative result, the divisor is added again (readdition)
- Shortenend division: Readdition and subtraction of a right shifted divisor (by 1 place) combined.

  + Divisor - ½ Divisor = + ½ Divisor
- Division by subtractions
  - Direct subtraction of divisors
  - Addition of the divisor's two's complement



© 2017 Burkhard Stiller

Division (3)

- Important division notes:
  - A division by 0 is not defined mathematically
  - Machine needs to recognize an exception state, which the processor has to consider explicitly (see later)
  - A division needs to be termined in case of an exploitation of the lenght of the results memory/register (periodical binary fractions)

- Circuits for multiplications can be used for divisions if minor modifications are performed:
  - Left-shift of dividend (instead of right-shift in case of multiplication)
  - Substraction of divisor (instead of addition of multiplier)

© 2017 Burkhard Stiller

Module 2: Arithmetics

- Number Systems
- Number Representation
- Basic Arithmetic Operations
- Characterers

ASCII Code to Represent Characters (1)

- ASCII-Code (American Standard for Coded Information Interchange) defines a standardized norm to code binary characters.
  - ASCII code with lower case/upper case characters of the latin alphabet, (arabic) numericals, and a few special characters.
  - Coding is done each in one byte (8 bit), such that the ASCII Code mantains 256 different characters.
  - The first bit of the standard ASCII code remains unused, which leads to 128 useable characters in the standard ASCII code
  - Different, specifically standardized ASCII Code extensions use the first bit, to represent such additional 128 characters.

© 2017 Burkhard Stiller

ASCII Code to Represent Characters (2)

(ASCII Code for Representable Characters (Excerpt)

Zeichen	Dez.	Binär	Hexa	Oktal	Zeichen	Dez.	Binär	Hexa	Oktal
!	33	0010 0001	21	041	P	80	0101 0000	50	120
"	34	0010 0010	22	042	Q	81	0101 0001	51	121
#	35	0010 0011	23	043	R	82	0101 0010	52	122
$	36	0010 0100	24	044	S	83	0101 0011	53	123
%	37	0010 0101	25	045	T	84	0101 0100	54	124
&	38	0010 0110	26	046	U	85	0101 0101	55	125
'	39	0010 0111	27	047	V	86	0101 0110	56	126
(	40	0010 1000	28	050	W	87	0101 0111	57	127
)	41	0010 1001	29	051	X	88	0101 1000	58	130

© 2017 Burkhard Stiller

ASCII Code to Represent Characters (3)

- The storage of texts is performed in separate bytes, which contain a single character each.
  - Those are concatenated to handle strings (Zeichenkette).

- An end of such a string needs a dedicated character, which is not uniquely defined across all languages/systems:
  - The length of a string is part of the first or the first few bytes, just befire the content starts. *E.g.*, used by the programming language PASCAL.

  - The end of a string is defined by a special, not-representable character *E.g.*,. Used by the programming language C/C++. A 0-Byte (byte, with all Bits equalling 0).

© 2017 Burkhard Stiller

Difference of Numericals and Characters

- Numerical in ASCII → Naming of character/numerical in quotes

```
'0':  00110000 ( decimal 48)
'4':  00110100 ( decimal 52)
'5':  00110101 ( decimal 53)
'8':  00111000 ( decimal 56)
```

- Numerical as number → Naming of numerical without quotes

```
0:  00000000 ( decimal 0)
4:  00000100 ( decimal 4)
5:  00000101 ( decimal 5)
8:  00001000 ( decimal 8)
```

© 2017 Burkhard Stiller

UniCode to Represent Characters (1)



- ASCII Code with 256 (128) characters is very limited.
- Unicode defined for characters oder elements of all known written cultures and systems of characters.

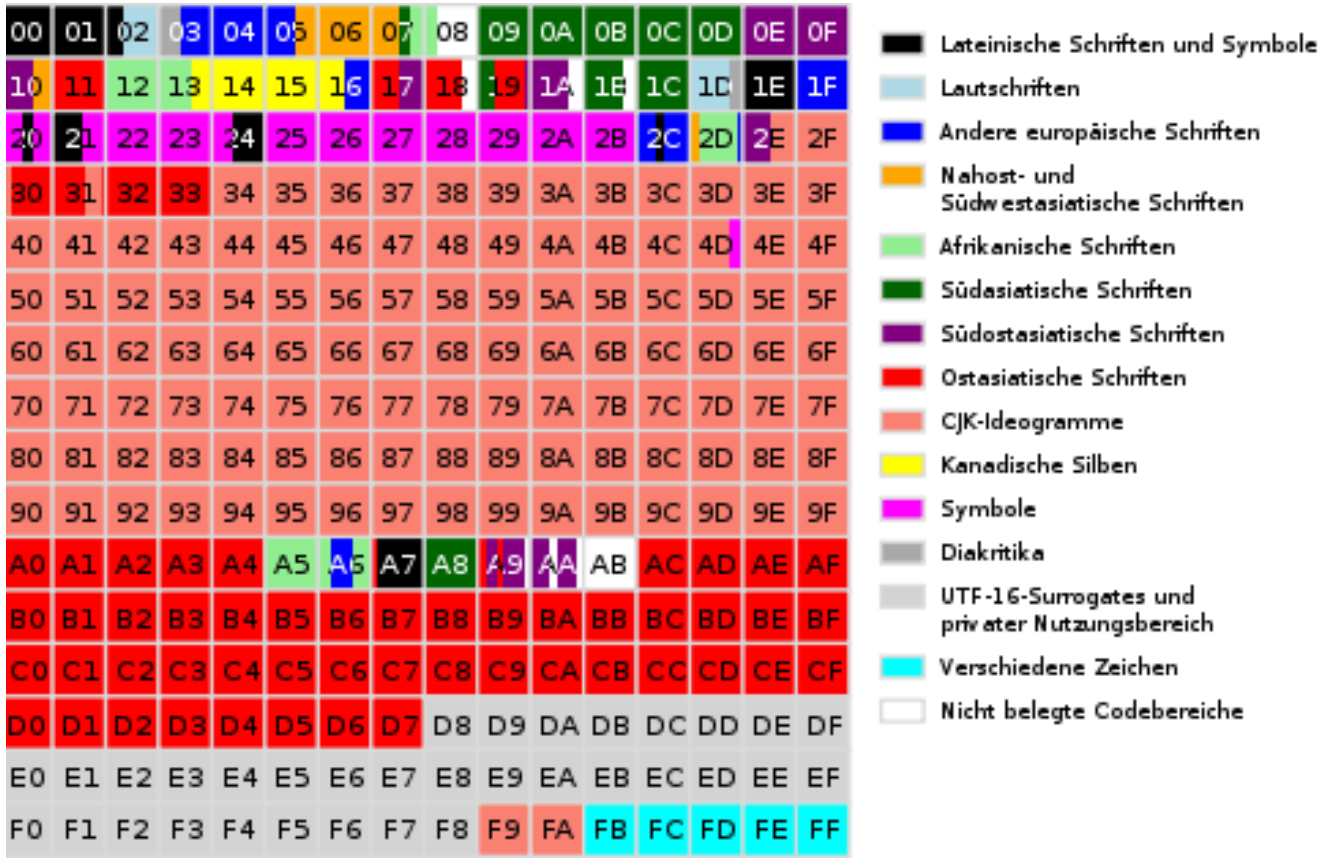
- Character values of Unicode Version 3.0 (September 1999) defined by 2 Byte numbers.
  - 65,536 different characters available
    (2 Byte = 16 Bit = 2^{16} combinations).
  - *2-Byte-Scheme labelled as "Basic Multilingual Plane" (BMP)*

- Version 3.1 (March 2001) cointains 94,140 characters with a more than 2 Byte coding
  - 2 Byte scheme replaced by the 4 Byte scheme

© 2017 Burkhard Stiller

UniCode to Represent Characters (2)

- Version 6.0 (October 2010) conatins 109,449 characters
  - Unicode with 17 planes (Bereiche), each with 65.536 characters
  - Basic Multilingual Plane: main written systems
  - Supplementary Multilingual Plane: historic written systems
  - Supplementary Ideographic Plane: Chinese, Japanese, and Korean writing
  - Supplementary Special-purpose Plane
  - Supplementary Private Use Area-A and -B



© 2017 Burkhard Stiller

UniCode to Represent Characters (3)

- Notation as follows

  &#0000; for decimal notation or

  &#x0000; for hexadecimal notation,

  with 0000 as the Unicode number of the character.
- Formats for storage and transmission different:
  - UTF-8 (Unicode Transformation Format) – Internet and withn some operating systems
  - UTF-16 – Intermediate coding in Java
- Unicode criticized in East-Asian countries.
  - Written signs of different and non-dependent languages have been combined
  - Japan didn‘t accept Unicode in practice
    - With different alternatives to Unicode, such as the Mojiky standard
  - Antique texts in Unicode due to unification of “similar” CJK written language (Chinese, Japanese, and Korean) not historically correct represented.

BCD Code to Represent Characters

- BCD (Binary Coded Decimals) codes binary numericals.
- For each decimal numerical typically four, sometines eight bits.
  - Numerical sequentially define the binary value
  - Storage-wasting approach simplifies especially applications with human inputs
  - Application areas:
    - Claculations in the decimal system
    - Storage of decimal numbers (siuch as phone numbers)
    - Control of LCD display, to light decimal numericals seperately

Decimal Number	Binary Number	BCD Representation
294	100100110	0010.1001.0100 2 9 4
16289	11111110100001	0001.0110.0010.1000.1001 1 6 2 8 9

Binary Scales (1)

Scales and Units for Bytes

Unit		Number of Bytes	KBytes	MBytes
Byte		1		
Kilobyte (KByte)	2^{10}	1024	1	
Megabyte (MByte)	2^{20}	1.048.576	1024	1
Gigabyte (GByte)	2^{30}	1.073.741.824	1.048.576	1024
Terabyte (TByte)	2^{40}	1.099.511.627.776	1.073.741.824	1.048.576
Petabyte (PByte)	2^{50}	1.125.899.906.842.624	1.099.511.627.776	1.073.741.824
Exabyte (EByte)	2^{60}	1.152.921.504.606.846.976	1.125.899.906.842.624	1.099.511.627.776

© 2017 Burkhard Stiller

Binary Scales (2)

- Kilo equals the factor $2^{10} = 1024$
- Mega equals the factor $2^{10} * 2^{10} = 1024 * 1024 = 1.048.576$
- Giga equals the faktor $2^{10} * 2^{10} * 2^{10} = 2^{30}$

- Example: Hard Disk (HD) Vendors and HD capacity

- MB MByte
  - Factor $10^3 = 1000$

- Impacts:
  - 1 GB 1.073.741.824 Byte but 1.000.000.000 Byte are only available
    - This leads to a “missing” number of real 73 MegaByte!
  - 200 GB only equal 186 GByte!

A Short Summary on Binary Prefixes (1)

- The SI prefixes (k, M, G, P, E, Z, ...) are historically ambiguous within the field of computer science as to whether they refer to the powers of 1000 or 1024.

- Formally the base-1000 definitions officially have been standardized by an 1998 amendment to IEC 60027-2 (http://physics.nist.gov/cuu/Units/binary.html). Even further, they have been endorsed by IEEE, CIPM, EU, ISO, and NIST.

- Thus, from a formal point of view, the base-1024 usage is obsolete and on its way out. However, the kilobinary (Ki = 2^10 = 1024^1), megabinary (Mi = 2^20 = 1024^2), gigabinary (Gi = 2^30 = 1024^3), ... units have not made it into the widespread public either.

© 2017 Burkhard Stiller

IEC 60027-2 Units (1)

Factor	Name	Symbol	Origin	Derivation
2^{10}	kibi	Ki	kilobinary: $(2^{10})^1$	kilo: $(10^3)^1$
2^{20}	mebi	Mi	megabinary: $(2^{10})^2$	mega: $(10^3)^2$
2^{30}	gibi	Gi	gigabinary: $(2^{10})^3$	giga: $(10^3)^3$
2^{40}	tebi	Ti	terabinary: $(2^{10})^4$	tera: $(10^3)^4$
2^{50}	pebi	Pi	petabinary: $(2^{10})^5$	peta: $(10^3)^5$
2^{60}	exbi	Ei	exabinary: $(2^{10})^6$	exa: $(10^3)^6$

IEC 60027-2 Units (2)

Examples and comparisons with SI prefixes

one **kibibit** 1 Kibit = 2^{10} bit = **1024 bit**

one **kilobit** 1 kbit = 10^{3} bit = **1000 bit**

one **mebibyte** 1 MiB = 2^{20} B = **1 048 576 B**

one **megabyte** 1 MB = 10^{6} B = **1 000 000 B**

one **gibibyte** 1 GiB = 2^{30} B = **1 073 741 824 B**

one **gigabyte** 1 GB = 10^{9} B = **1 000 000 000 B**

© 2017 Burkhard Stiller

A Short Summary on Binary Prefixes (2)

- However, the only place where we do see the industry referring to powers of 1024 with SI prefixes always is with the size of RAM (or general storage) modules, which for architectural reasons are constrained to being powers of 2.
- Obviously, the old 1024≈1000 approximation was ok, while it lasted in its early days. The error from this approximation grows larger and larger as computers become more and more powerful and the respective storage demands grow. Thus, the error of the powers 1000 vs. 1024 is about as follows, +2.4% for k, +4.9% for M, +7.4% for G, +10.0% for T, ..., which will dangerously tell the "wrong truth" for Exa- and PetaByte numbers being at the horizon these days!
- But, the unit “Bit" and “Byte" remains as the unit - and NOT “Bits" or “Bytes", which defines the plural as used as a noun within a sentence only, in this case as “bits” or “bytes” (lowercase).

— Computer Engineering —
Module 3: Combinatorial Circuits



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2017 Burkhard Stiller

ifi

Modul 3: Combinatorial Circuits

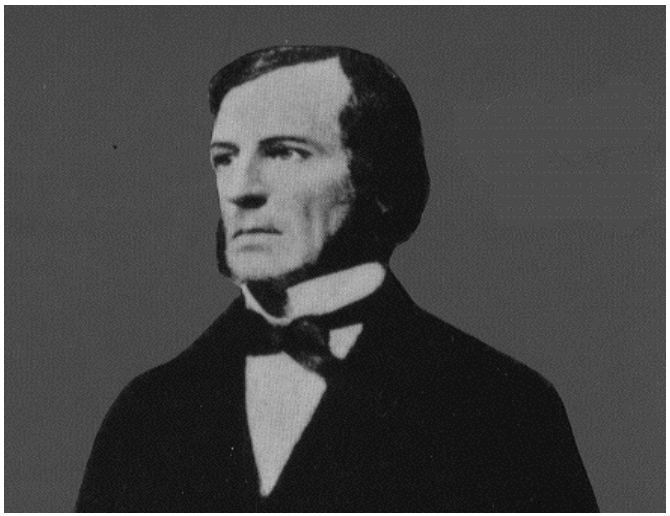
- Formal Basics of Logics
  - Boolean Algebra
- Normalized and Minimized Forms
- Implementation of Combinatorial Circuits by Gates
- Design of Combinatorial Circuits
  - Minimization and KV Diagrams
  - Function Programming
- Delay Effects

© 2017 Burkhard Stiller

Formal Basics of Combinatorial Circuits

- Combinatorial Circuits
  - Purely combinatorial, logic circuits
  - No storage!
  - Logical functios
- Examples
  - Light-off warning in cars
  - “Engine off“ and “Door open“ and “Light on“ $\Rightarrow$ Alarm

- Studying and description of characteristics and behavior of logical functions done by Boolean Algebra
- Designed by mathematician George Boole (1815 –1864) as “Logical Algebra”



© 2017 Burkhard Stiller

Boolean Algebra

- Definition:
  - A Boolean Algebra is defined by the set V = {a,b,c,...}, on which two binary operations $\oplus$ and $\otimes$ can be applied, such that by its application on elements of V, only elements of V are generated (closure property – “Abgeschlossenheit”).

- Closure Property: For all $a, b \in V$:

$$a \otimes b \in V$$

$$a \oplus b \in V$$

- Additionally, the four Huntington Axioms apply.
  - Edward Vermilye Huntington (* April 26, 1874, Clinton, New York, USA, † November 25, 1952, Cambridge, Massachusetts, USA)
  - American mathematician, who put Boolean algebra on a sound axiomatic foundation in 1902 and revisited it in 1933.

© 2017 Burkhard Stiller

Huntington Axioms

- H1 — Commutative Law (“Kommutativgesetz”):
  - $a \otimes b = b \otimes a$
  - $a \oplus b = b \oplus a$
- H2 — Distributive Law (“Distributivgesetz”):
  - $a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$
  - $a \oplus (b \otimes c) = (a \oplus b) \otimes (a \oplus c)$
- H3 — Neutral Element:
  - Two elements exist, especially e, $n \in V$, such that:
    - $a \otimes e = a$ (e is called “one“ element (“Einselement“))
    - $a \oplus n = a$ (n is called “zero” element („Nullelement”))
- H4 — Inverse Element:
  - For all $a \in V$ one element $a \in V$ exists, such that
    - $a \otimes \bar{a} = n$
    - $a \oplus \bar{a} = e$

© 2017 Burkhard Stiller

Boolean Algebra Instantiated

- The instantiated Boolean algebra is defined explicitly by the following table:

Boolean Algebra	Instantiated
V	{0,1}
$\oplus$	$\vee$
$\otimes$	$\wedge$
n	0
e	1
$\bar{a}$	$\bar{a}$

Alternative representations:

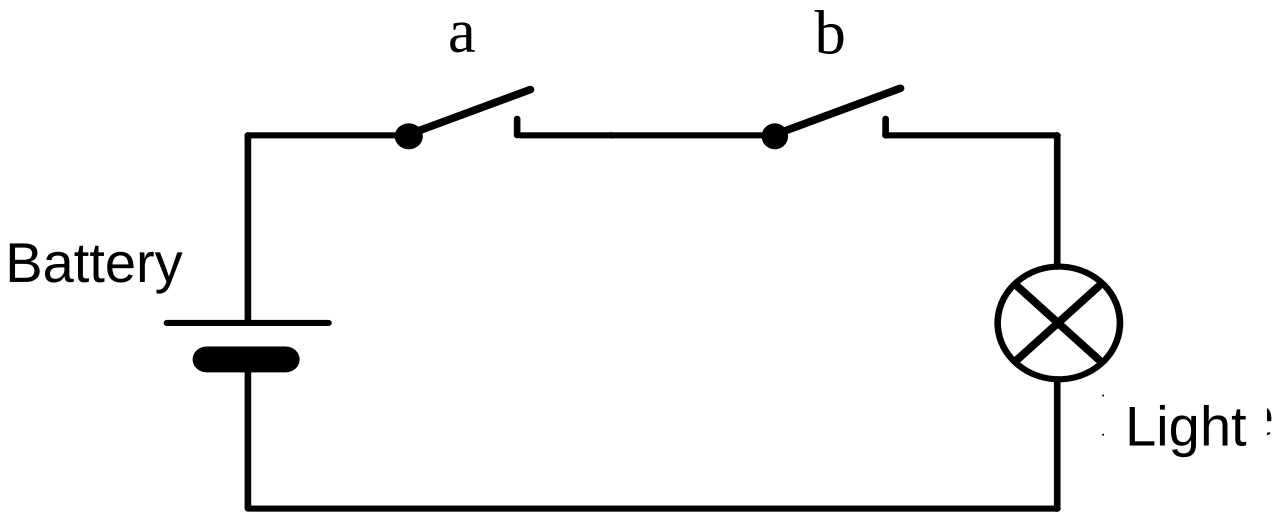
a + b for a $\vee$ b or naming as “OR/ODER”

a & b, ab for a $\wedge$ b or naming as “AND/UND”

© 2017 Burkhard Stiller

Implementation of Logical Conjunctions (1)

- A technical implementation of logical conjunctions uses simple switching models instead of logical elements:
- AND:

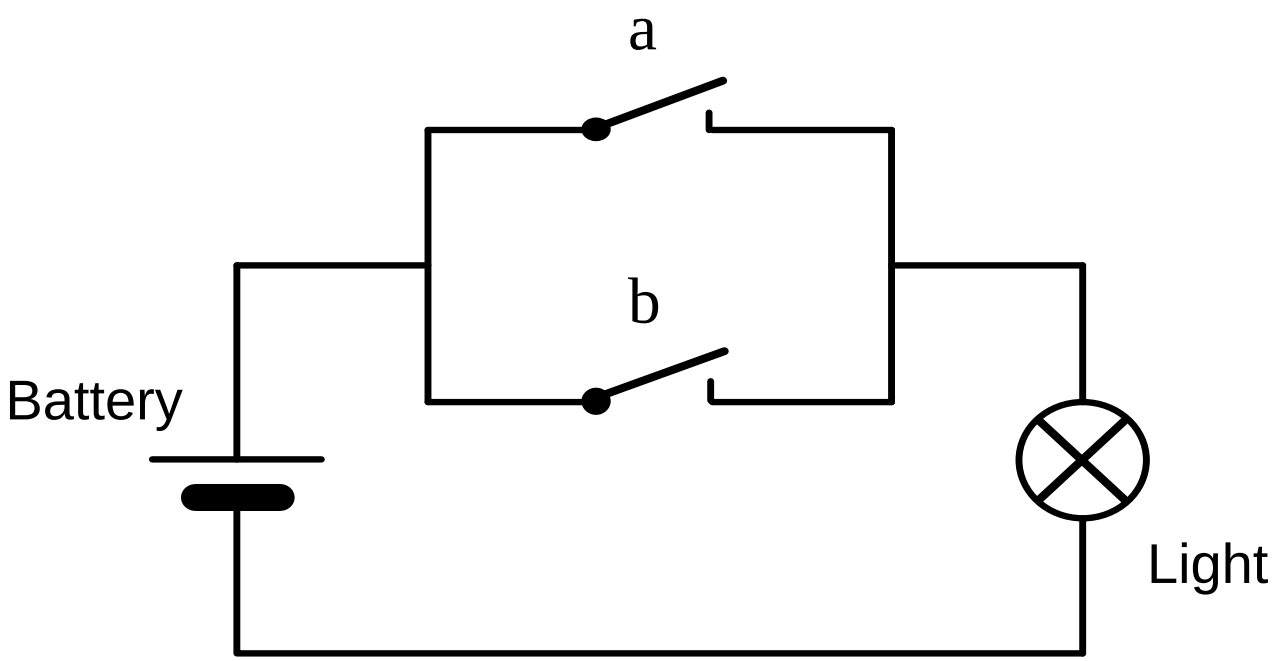


- The light is only switched “on” (function’s value equals 1), if the two switches are shut (a AND b equal 1), otherwise the light is switched “off” (function’s value equals 0).

© 2017 Burkhard Stiller

Implementation of Logical Conjunctions (2)

- OR:
  The light is switched “on”, if the two switches are shut.



© 2017 Burkhard Stiller

Boolean Algebra Instantiated (1)

- All conjunctions are represented in tables:

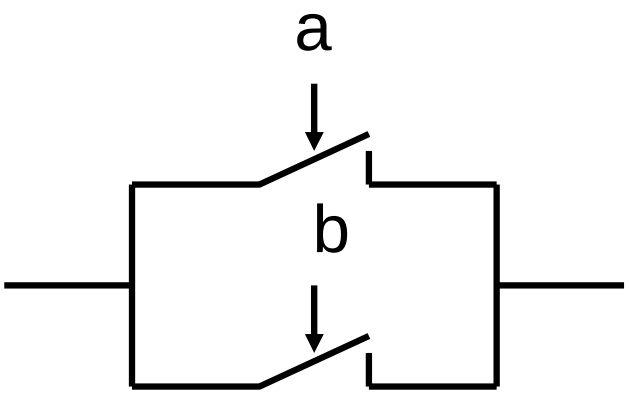
a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

a	$\bar{a}$
0	1
1	0



Technical implementation with switches



no synonym available

© 2017 Burkhard Stiller

ifi

Boolean Algebra Instantiated (2)

- Huntington Axioms:

- H1:
  Commutative Law
  $a \vee b = b \vee a$
  $a \wedge b = b \wedge a$

- H2:
  Distributive Law
  $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$
  $a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c)$

- H3:
  One Element
  $a \wedge 1 = a$
  $a \vee \overline{a} = 1$

- H4:
  Zero Element
  $a \wedge \overline{a} = 0$
  $a \vee 0 = a$

© 2017 Burkhard Stiller

Boolean Algebra Instantiated (3)

- From those 4 Huntington‘s Axioms additional clauses are derived:

- G1: Assoziative Law
  $(a \wedge b) \wedge c = a \wedge (b \wedge c)$
  $(a \vee b) \vee c = a \vee (b \vee c)$

- G2: Idempotence Law
  $a \wedge a = a$
  $a \vee a = a$

- G3: Law of Absorption
  $a \wedge (a \vee b) = a$
  $a \vee (a \wedge b) = a$

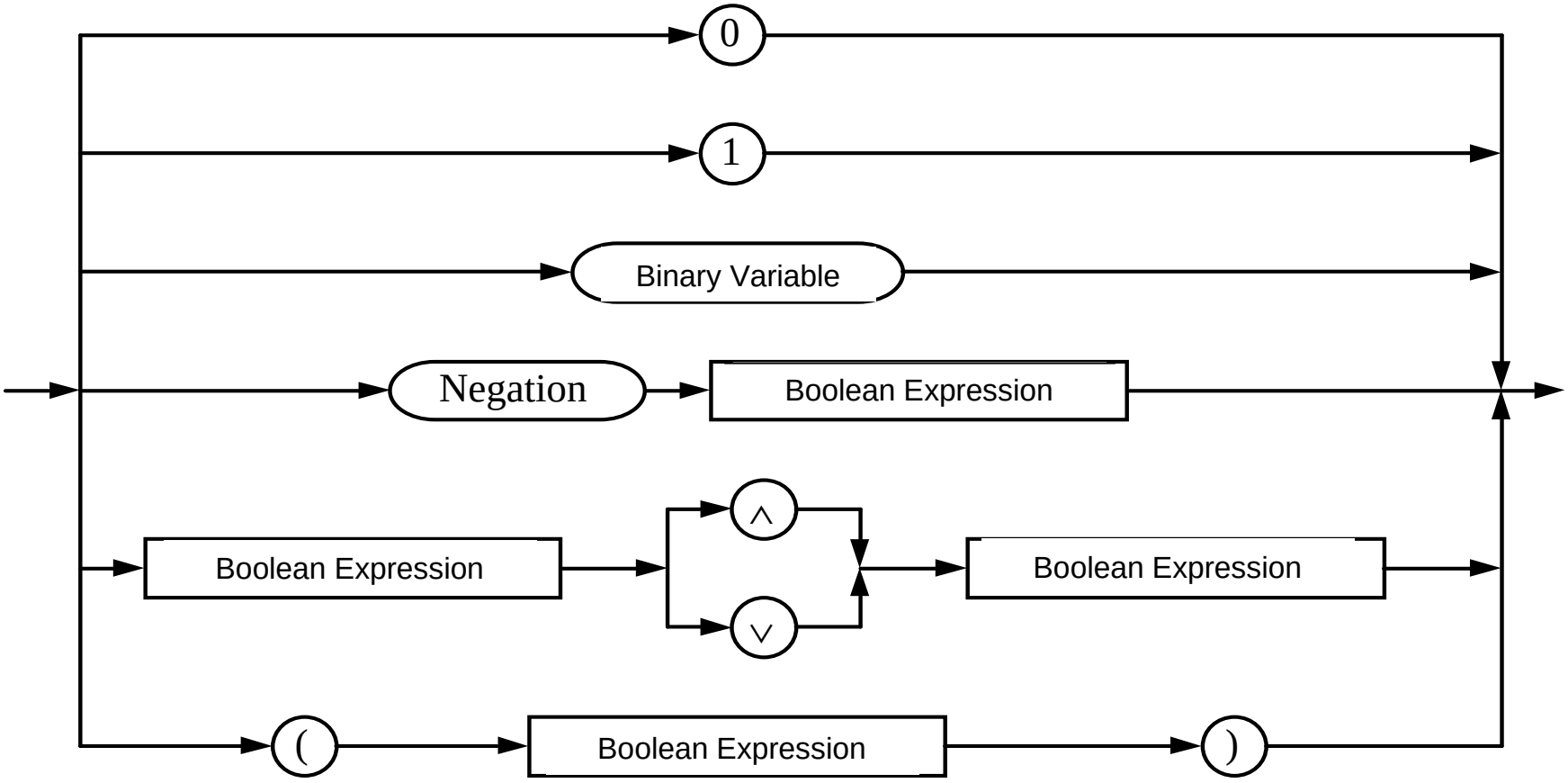
- G4: DeMorgan Law
  $\overline{a \wedge b} = \overline{a} \vee \overline{b}$
  $\overline{a \vee b} = \overline{a} \wedge \overline{b}$

Also named as: NAND, NOR

© 2017 Burkhard Stiller

Boolean Expressions (1)

- A Boolean expression is defined as a sequence of characters, which contains binary variables, the operators $\wedge$ and $\vee$, and brackets, and which follows the syntactical rules, defined by the syntax diagramm:



© 2017 Burkhard Stiller

Boolean Expressions (2)

- Examples:
  - Syntactical correct expressions: $a \vee b$, 0, $(a \wedge b) \vee \overline{c}$
- No Boolean expressions, due to their syntactical incorrectness:
  - $a \vee \vee a$, 1 0, $( ) \vee c$
- For the Boolean Algebra instantiated the constants 0 and 1 are termed sometimes – closely related to statements – as Boolean Truth Values:
  - 0 : false 1 : true
- A Boolean expression by itself does not represent a Boolean value, since it may contain Boolean variables!
  - Only an assignment of a binary value with a Boolean value allows for the calculation of this Boolean expression‘s Boolean truth value

© 2017 Burkhard Stiller

Various Definitions

- An assignment (“Belegung”) of binary variables of a Boolean expression with Boolean values is termed interpretation.
- An interpretation of a Boolean expression leads to a declaration (“Aussage”), which is either true or false.
- Different interpretations of a Boolean expression may lead to the same Boolean value.
- A Boolean expression, for which all possible interpretations lead to the Boolean value “true” is called tautology.
  - Example: $a \vee \overline{a}$ defines a tautologie (often marked as T).

Boolean Functions

- Assume a tuple of binary variables $(x_1,x_2,...,x_n)$
- Definition: A (n-ary) Boolean Function assigns to every possible Boolean value assignment of this Boolean variable exactly one Boolean value:

$$f : \{0,1\}^n \rightarrow \{0,1\}$$

- How many assignments and how many n-ary functions exist?

2^n assignments $2^{(2^n)}$ functions

(for each argument of a Boolean function two different function values exist)

- Examples
  - Negation: A 1-ary Boolean function

$$f : \{0,1\} \rightarrow \{0,1\}$$

(assigns to each operand from {0,1} a function's value from {0,1})

  - $\wedge$ and $\vee$: Two 2-ary Boolean functions:

$$f : \{0,1\} \times \{0,1\} \rightarrow \{0,1\}$$

© 2017 Burkhard Stiller

16 Possible 2-ary Boolean Functions

x_1 / x_0	0011 / 0101	Verbal Form	Symbolic Representation	Name
f_0	0000	constant 0	0	Contradiction, symbol: $\bot$ (unfulfillable)
f_1	0001	x_1 und x_0	$x_1 \wedge x_0$	Conjunction
f_2	0010	not x_0, but x_1	$x_1 \wedge \overline{x_0}$	Inhibition
f_3	0011	identical x_1	x_1	Identity
f_4	0100	not x_1, but x_0	$\overline{x_1} \wedge x_0$	Inhibition
f_5	0101	identical x_0	x_0	Identity
f_6	0110	x_1 unequal x_0	$x_1 \not\leftrightarrow x_0$	Antivalence, XOR
f_7	0111	x_1 or x_0	$x_1 \vee x_0$	Disjunction
f_8	1000	not (x_1 or x_0)	$x_1 \overline{\vee} x_0$	NOR function, Peircean Arrow
f_9	1001	x_1 equal x_0	$x_1 \leftrightarrow x_0$	Equivalence
f_{10}	1010	not x_0	$\overline{x_0}$	Negation
f_{11}	1011	if x_0, then x_1	$x_0 \rightarrow x_1$	Implication
f_{12}	1100	not x_1	$\overline{x_1}$	Negation
f_{13}	1101	if x_1, then x_0	$x_1 \rightarrow x_0$	Implication
f_{14}	1110	not (x_1 und x_0)	$x_1 \overline{\wedge} x_0$	NAND function, Sheffer‘s Line
f_{15}	1111	constant 1	1	Tautology, symbol: T (generally valid)

© 2017 Burkhard Stiller

Representation of Boolean Functions

1. Function table, truth table (“Funktionstabelle”)
2. Algebraic expression (symbolic form)
3. Graph

Function Table

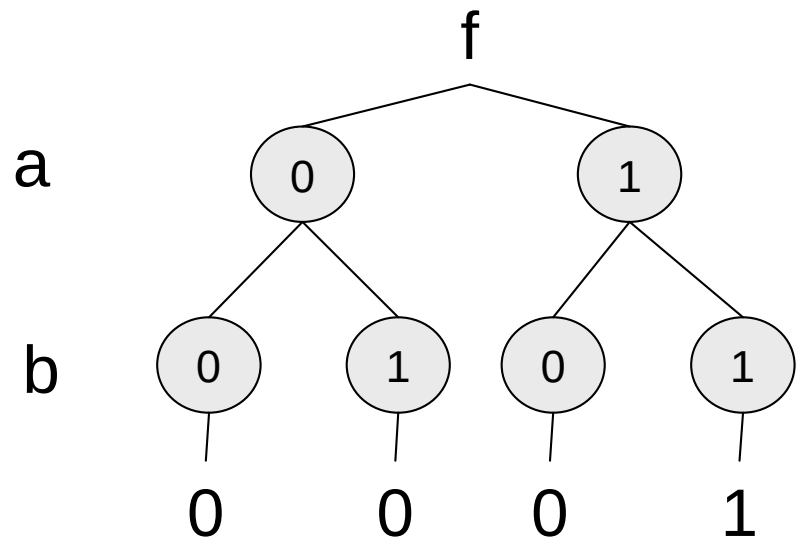
a	b	f
0	0	0
0	1	0
1	0	0
1	1	1

Symbolic form

$f = a \wedge b$

Graph

(*e.g.*, Shannon Tree)



© 2017 Burkhard Stiller

Full Operators System (1)

- Definition:
  A system of operators, with which all Boolean functions can be represented, is termed full operators system.

- The operators ($\wedge$, $\vee$, $\overline{\ }$) do form such a full operators system.

- Example:
  $a \leftrightarrow b$ equivalence with the same result as $(a \wedge b) \vee (\overline{a} \wedge \overline{b})$.

  $\leftrightarrow$ thus, replaced by basic operations $\wedge$, $\vee$, and $\overline{\ }$.

© 2017 Burkhard Stiller

Full Operators System (2)

Operators System	Representation of ...		
	Negation	Conjunction	Disjunction
$(\wedge, \vee, \neg)$	$\bar{a}$	$a \wedge b$	$a \vee b$
$(\wedge, \neg)$	$\bar{a}$	$a \wedge b$	$\overline{\bar{a} \wedge \bar{b}}$
$(\vee, \neg)$	$\bar{a}$	$\overline{\bar{a} \vee \bar{b}}$	$a \vee b$
$(\barwedge)$	$a \barwedge a$	$(a \barwedge b) \barwedge (a \barwedge b)$	$(a \barwedge a) \barwedge (b \barwedge b)$
$(\barvee)$	$a \barvee a$	$(a \barvee a) \barvee (b \barvee b)$	$(a \barvee b) \barvee (a \barvee b)$
$(\wedge, \not\leftrightarrow)$	$a \not\leftrightarrow 1$	$a \wedge b$	$a \not\leftrightarrow b \not\leftrightarrow (a \wedge b)$

Hint: The operand "$\wedge$" may not be required in writing: $a \wedge b \leftrightarrow ab$

Tautology (1)

- In which cases two expressions A and B define the exact same Boolean function?
- Similarly noted as: Is $A \leftrightarrow B$ a tautology?
- Two Boolean functions are defined as follows:

  $f_1(a,b) = (a \wedge b) \vee (\overline{a} \wedge \overline{b})$

  $f_2(a,b) = (a \vee \overline{b}) \wedge (\overline{a} \vee b)$

- Is f_1 identical to f_2

  or

  is $(a \wedge b) \vee (\overline{a} \wedge \overline{b}) \leftrightarrow (a \vee \overline{b}) \wedge (\overline{a} \vee b)$ a tautology?

Tautology (2)

- Proof by utilizing (a) function tables or
  (b) logical transformations by applying algebraic laws.
  - (b) See next slide
  - (a) Two expressions are equivalent, if their results for all interpretations and, thus, all possible combinations of assignments for variables are identical.

a b	$(a \wedge b) \vee (\overline{a} \wedge \overline{b})$	$(a \vee \overline{b}) \wedge (\overline{a} \vee b)$	$A \leftrightarrow B$
0 0	1	1	1
0 1	0	0	1
1 0	0	0	1
1 1	1	1	1

Tautology (3)

- Transformations by applying algebraic laws:

$$(a \wedge b) \vee (\overline{a} \wedge \overline{b}) = [(a \wedge b) \vee \overline{a}] \wedge [(a \wedge b) \vee \overline{b}]$$

(Distributive Law)

$$= [(a \vee \overline{a}) \wedge (b \vee \overline{a})] \wedge [(a \vee \overline{b}) \wedge (b \vee \overline{b})]$$

(Distributive Law)

$$= [1 \wedge (b \vee \overline{a})] \wedge [(a \vee \overline{b}) \wedge 1]$$

(Inverse Element)

$$= (b \vee \overline{a}) \wedge (a \vee \overline{b})$$

(Neutral Element)

$$= (\overline{a} \vee b) \wedge (a \vee \overline{b})$$

(Commutative Law)

Modul 3: Combinatorial Circuits

- Formal Basics of Logics
  - Boolean Algebra
- Normalized and Minimized Forms
- Implementation of Combinatorial Circuits by Gates
- Design of Combinatorial Circuits
  - Minimization and KV Diagrams
  - Function Programming
- Delay Effects

© 2017 Burkhard Stiller

Normal Forms

- A Boolean function may be represented by different Boolean expressions.

- A standardized representation of Boolean functions within a full operator‘s system such as ($\wedge$, $\vee$, $\overline{\ }$) is defined by the conjunctive normal form (CNF) – sometimes called “clausal normal form” – and the disjunctive normal form (DNF).

- Definition:

  A literal L_i is defined as either a positive variable x_i or its negation (negative variable) $\overline{x_i}$, thus, $L_i \in \{x_i, \overline{x_i}\}$

  – Only positive or negative literals exist!

Clauses (Product Terms)

- Definition:

  A conjunctive clause (or product term) $K(x_1,\ldots,x_m)$ is defined as the conjunction of literals

  $$\bigwedge_{i\in\{1,\ldots,m\}} L_i = L_1 \wedge \ldots \wedge L_m$$

  or the constants "0" or "1 "

- Examples: $a \wedge a \wedge b$ $\quad\quad$ $\overline{x_i} \wedge x_i$

- Every conjunctive clause $K(x_1,\ldots,x_m) = \bigwedge_{i\in\{1,\ldots,m\}} L_i$ may be represented in a form such that a single variable x only appears at most once in one literal.

© 2017 Burkhard Stiller

Literals and Clauses

- If $L_h = x$, $L_j = x$, $h \neq j$ (multiple "positive" appearances)

  $\Rightarrow \quad L_h \wedge L_j = x$

  $\Rightarrow \quad K(x_1,...,x_m) = \bigwedge_{i=1}^{m} L_i$

- Multiple appearances of x can be reduced due to the Idempotence Law ($x \wedge x = x$).

- If $L_h = x$ und $L_j = \overline{x}$ (mixed "positive" and "negated" appearances)

  $\Rightarrow L_h \wedge L_j = 0$

  $\Rightarrow K(x_1,...,x_m) = 0$ (clause equals 0)

© 2017 Burkhard Stiller

Implicant and Minterm

- Definition:

  A conjunctive clause $K(x_1,...,x_n)$ is termed implicant of a Boolean function $y(x_1,...,x_n)$, if $K \rightarrow y$
  - K implies y (K is an implicant of y), if y also takes the value 1 whenever K equals 1
  - *I.e.*, for every assignment $B \in \{0,1\}^n$ the following holds true:
    If K(B) = 1, y(B) = 1 holds as well.
- Definition:

  An implicant of a Boolean function $y(x_1,...,x_n)$ is termed minterm, if a literal of each variable x_i within the function y of implicants exists exactly once.

- Examples
  - Minterms of a Boolean function $y(x_1,...,x_4)$: $x_1 \wedge x_2 \wedge x_3 \wedge x_4$ or $x_1 \wedge \overline{x}_2 \wedge x_3 \wedge x_4$
  - No minterms of the Boolean function $y(x_1,...,x_4)$: $x_1 \wedge x_2$ or $x_1 \wedge \overline{x}_2 \wedge x_3 \wedge x_3 \wedge x_4$

© 2017 Burkhard Stiller

Disjunctive Normal Form

- Definition:

  Assuming a Boolean function $y(x_1,...,x_n)$, a Boolean expression is termed to be disjunctive normal form (DNF) of the function y, if it exists as a disjunction of all minterms K_i.

  $y = K_0 \vee K_1 \vee ..... \vee K_k$, $k \quad 2^n-1$

  - No two conjunctions K_i, K_j exist with i j , such that they are equivalent
  - Any Boolean expression is in its disjunctive normal form, iff it is either a single conjunctive clause or a series of conjunctive clauses joined by "$\vee$".

- Examples
  - $f(a,b,c) = a\,b\,c \vee a\,\overline{b}\,c \vee \overline{a}\,\overline{b}\,c$ determines a valid DNF
  - $f(a,b,c) = a\,b\,c \vee a\,\overline{b} \vee c\,a\,b \vee a(b\,c \vee \overline{b}\,\overline{c})$ does not determine a DNF, since
    - $a\,\overline{b}$ does not contain all variables
    - a b c and c a b are equivalent
    - $a(b\,c \vee \overline{b}\,\overline{c})$ does not define a "pure" conjunction

© 2017 Burkhard Stiller

Implicat and Maxterm

- Definition:
  A disjunctive clause $D(x_1,...,x_n)$ is termed implicat of a Boolean function $y(x_1,...,x_n)$, if $D \rightarrow \overline{y}$
  - D implies y (D is an implicant of y), if $\overline{y}$ also takes the value 0 whenever D equals 0
  - *I.e.*, for every assignment $B \in \{0,1\}^n$ the following holds true:
    If D(B) = 0, y(B) = 0 holds as well.
- Definition:
  An implicat of a Boolean function $y(x_1,...,x_m)$ is termed maxterm, if a literal of each variable x_i within the function y of implicants exists exactly once.

- Examples
  - Maxterms of the Boolean function $\overline{y}(x_1,...,x_3)$:
    $x_1 \vee x_2 \vee x_3$ or $x_1 \vee \overline{x}_2 \vee x_3$

Conjunctive Normal Form

- Definition:

  Assuming a Boolean function $y(x_1,..., x_m)$, a Boolean expression is termed to be conjunctive normal form (CNF) of the function y, if it exists as a conjunction of all maxterms D_i.

  $y = D_0 \wedge D_1 \wedge ..... \wedge D_k$, k 2^m-1

  – No two disjunctions D_i, D_j exist with i j , such that they are equivalent
  – Any Boolean expression is in its conjunctive normal form, iff it is either a single disjunctive clause or a series of disjunctive clauses joined by "$\wedge$".

- Examples
  – f (a,b,c) = $(a \vee b \vee c) \wedge (a \vee b \vee c) \wedge (a \vee b \vee c)$ determines a valid CNF
  – f (a,b,c) = $(a \vee b \vee c) \wedge (a \vee \overline{b}) \wedge (c \vee a \vee b)$ does not determine a CNF, since
    - $a \vee \overline{b}$ does not contain all variables
    - $a \vee b \vee c$ and $c \vee a \vee b$ are equivalent

© 2017 Burkhard Stiller

DNF and CNF

- Any function with n variables *may* show up to 2^n minterms and maxterms, respectively.
- Example: For n = 3 those read in full and complete:

	Minterm	Maxterm
0	$\overline{a}\ \overline{b}\ \overline{c}$	$a \vee b \vee c$
1	$a\ \overline{b}\ \overline{c}$	$\overline{a} \vee b \vee c$
2	$\overline{a}\ b\ \overline{c}$	$a \vee \overline{b} \vee c$
3	$a\ b\ \overline{c}$	$\overline{a} \vee \overline{b} \vee c$
4	$\overline{a}\ \overline{b}\ c$	$a \vee b \vee \overline{c}$
5	$a\ \overline{b}\ c$	$\overline{a} \vee b \vee \overline{c}$
6	$\overline{a}\ b\ c$	$a \vee \overline{b} \vee \overline{c}$
7	$a\ b\ c$	$\overline{a} \vee \overline{b} \vee \overline{c}$

© 2017 Burkhard Stiller

“Reading” and Utilizing Normal Forms

- Every single minterm of a DNF equals one dedicated row in the functions table, which maintains always the function‘s value 1.
- Every single maxterm of a CNF equals one dedicated row in the functions table, which maintains always the function‘s value 0.
- Disjunctive and conjunctive normal forms determine unique representations!
  - Except for permutations, such as abc, acb, bac, bca, cab, or cba which are all equivalent.
- Functions represented by a single minterm only, determine for a single assignment only the function‘s value 1
  - Aside from the trivial null function, those functions show a minimal number of 1s!
- Functions represented by a single maxterm only, determine for a single assignment only the function‘s value 0
  - Aside from the trivial one function, those functions show a maximal number of 1s!

© 2017 Burkhard Stiller

Example: DNF and CNF

A Boolean function is uniquely specified by listing all minterms and maxterms, respectively.

	c b a	y	Minterms	Maxterms
	0 0 0	1	$\overline{a}\ \overline{b}\ \overline{c}$	
	0 0 1	0		$\overline{a} \vee b \vee c$
	0 1 0	0		$a \vee \overline{b} \vee c$
	0 1 1	1	$a\ b\ \overline{c}$	
	1 0 0	1	$\overline{a}\ \overline{b}\ c$	
	1 0 1	0		$\overline{a} \vee b \vee \overline{c}$
	1 1 0	0		$a \vee \overline{b} \vee \overline{c}$
	1 1 1	1	$a\ b\ c$	

DNF: $y = (\ \overline{a}\ \overline{b}\ \overline{c}\ ) \vee (a\ b\ \overline{c}\ ) \vee (\ \overline{a}\ \overline{b}\ c\ ) \vee (\ a\ b\ c\ )$

CNF: $y = (\ \overline{a} \vee b \vee c\ )(a \vee \overline{b} \vee c\ )(\ \overline{a} \vee b \vee \overline{c}\ )(a \vee \overline{b} \vee \overline{c}\ )$

© 2017 Burkhard Stiller

Deriving DNF or CNF from Function Tables

- DNF:

  The functions table contains all minterms of that function: For all rows, which show a function‘s value of 1, all input variables are concatenated with $\wedge$, while all of them with the value 0 are negated.
  - The disjunction of all minterms leads to a Boolean expression in DNF.

- CNF:

  The functions table contains all maxterms of that function: For all rows, which show a function‘s value of 0, all input variables are concatenated with $\vee$, while all of them with the value 1 are negated.
  - The disjunction of all minterms leads to a Boolean expression in CNF.

Example: see previous slide!

© 2017 Burkhard Stiller

Deriving DNF or CNF From any Form

- To derive out of any disjunctive or conjunctive form of a Boolean expression the DNF and CNF respectively Shannon‘s Expansion can be applied.

- Expansion to the variable x_i for a given function f:
  - This variable‘s value within function f will be set to 1 and
  - the resulting term will be concatenated in a conjunctiv manner with x_i

  and all will be $\vee$ -concatenated with
  - the same variable‘s value being set to 0 and
  - the resulting term being concatenated in a conjunctiv manner with $\overline{x_i}$

$$y = f(x_1, \ldots, x_n)$$
$$= [x_i \wedge f(x_1, \ldots, x_{i-1}, \mathbf{1}, x_{i+1}, \ldots, x_n)] \vee [\overline{x_i} \wedge f(x_1, \ldots, x_{i-1}, \mathbf{0}, x_{i+1}, \ldots, x_n)]$$

Example: Expansion

$$y = a\ \overline{b}\ c \vee \overline{a}\ \overline{b} \vee b\ c$$

Shannon's Expansion applied to a and $\overline{a}$

$$= a\,[1\ \overline{b}\ c \vee 1\ \overline{b} \vee b\ c\,] \vee \overline{a}\,[0\ \overline{b}\ c \vee 0\ \overline{b} \vee b\ c\,]$$

H4: $1\ \overline{b} = 0$ and H3: $x \vee 0 = x$

$$= a\,[\ \overline{b}\ c \vee b\ c\,] \vee \overline{a}\,[\ \overline{b} \vee b\ c\,]$$

Syntactic adaptation of terms, sorting of b according to positive and negative literals and applying H3: $x \wedge 1 = x$

$$= a\,[\,b\,(\,c\,) \vee \overline{b}\,(\,c\,)\,] \vee \overline{a}\,[\,b\,(\,c\,) \vee \overline{b}\,(\,1\,)\,]$$

Extension of c within the last term by applying H4: $c \vee \overline{c} = 1$

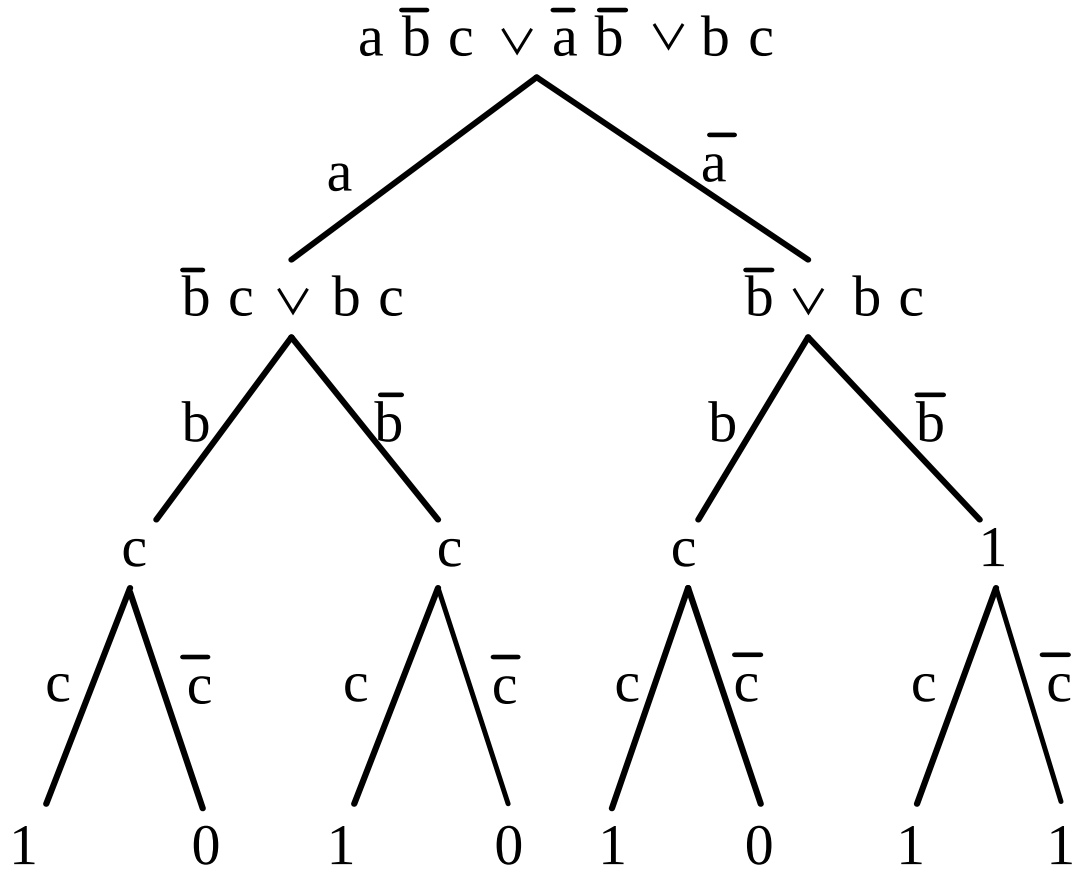
$$= a\,[\,b\,(\,c\,) \vee \overline{b}\,(\,c\,)\,] \vee \overline{a}\,[\,b\,c \vee \overline{b}\,(\,c \vee \overline{c}\,)\,]$$

Application of Distributive Law ("Ausmultiplizieren")

$$= a\ b\ c \vee a\ \overline{b}\ c \vee \overline{a}\ b\ c \vee \overline{a}\ \overline{b}\ c \vee \overline{a}\ b\ \overline{c}$$

© 2017 Burkhard Stiller

Example: Alternative Shannon Tree



After the function had been expanded to all variables, minterms are identified by those leaves of the tree and their paths from the root, which show a value of 1.

Reminder of DNF and CNF

- Disjunctive and conjunctive normal forms determine unique representations!
  - Except for permutations, such as abc, acb, bac, bca, cab, or cba which are all equivalent.

Example: $y = a\ \overline{b} \vee c$

- DNF: $y = \overline{a}\ \overline{b}\ c \vee \overline{a}\ b\ c \vee a\ \overline{b}\ \overline{c} \vee a\ \overline{b}\ c \vee a\ b\ c$
- CNF: $y = (a \vee b \vee c) \wedge (a \vee \overline{b} \vee c) \wedge (\overline{a} \vee \overline{b} \vee c)$

© 2017 Burkhard Stiller

Minimized Forms

- Goals
  - Deriving shortest possible expressions for a given Boolean function
  - Implementing such a function at the lowest possible costs!
- Similarly to the disjunctive and conjunctive normal forms, so-called minimal disjunctive normal forms (“Disjunktive Minimalform”) and minimal conjunctive normal forms (“Konjunktive Minimalform”) exist.

- Multiple minimal disjunctive and conjunctive normal forms may exist for the same Boolean function
  - Example:
    - $y = a\,\overline{b} \vee b\,\overline{c} \vee \overline{a}\,c$ and
    - $y = a\,\overline{c} \vee \overline{b}\,c \vee \overline{a}\,b$

    define an identical function and both are minimal disjunctive normal forms

© 2017 Burkhard Stiller

Deriving Minimized Forms

- In search of such a minimal disjunctive/conjunctive normal form the approach may become complex for functions with many variables
  - No “unique” or even optimal algorithm exists to find “the” normal form
  - Thus, sub-optimal solutions may be determined only by applying heuristics
- The minimization of forms operates in two steps
  - Step 1: Derive the set of implicants and implicats, respectively, for a given Boolean function f, while ensuring that the number of literals is kept as small as possible.
  - Step 2: Select from these sets of implicants and implicats, respectively, the smallest possible number of implicants and implicats, respectively, such that their disjunction and conjunction, respectively, determine this function f.
- Prerequisites for practical minimizations
  - Application of full operators systems with one operator only!

© 2017 Burkhard Stiller

NAND/NOR Conversion

- The ($\overline{\wedge}$) system (NAND System) and ($\overline{\vee}$) system (NOR System) are full operators systems (cf. before)

  ⇨ Any disjunctive and conjunctive expression can be represented within the NAND and NOR System

- Conversion options in four different cases

  Case 1: Function in disjunctive form $\Rightarrow$ ($\overline{\wedge}$) system

  Case 2: Function in disjunctive form $\Rightarrow$ ($\overline{\vee}$) system

  Case 3: Function in conjunctive form $\Rightarrow$ ($\overline{\vee}$) system

  Case 4: Function in conjunctive form $\Rightarrow$ ($\overline{\wedge}$) system

- Why are exactly those conversions relevant?

  ⇨ Much simpler implementation in hardware!

  – NANDs or NORs are very simple to be implemented as combinatorial circuits.

© 2017 Burkhard Stiller

NAND Conversion (Example for Case 1)

- Assume a function f provided in disjunctive form
  - To be converted into $\Rightarrow$ ($\overline{\wedge}$) system

- Conversion in two steps:
  Step 1: Double negation of entire f
  Step 2: Application of DeMorgan law

  - Resulting in an expression, which only contains NAND operators

© 2017 Burkhard Stiller

Example: NAND Conversion

$$y = \overline{a}\,b\,c \vee a\,\overline{b}\,c \vee a\,b\,\overline{c} \vee \overline{a}\,\overline{b}\,\overline{c}$$

$$= \overline{\overline{a}\,b\,c \vee a\,\overline{b}\,c \vee a\,b\,\overline{c} \vee \overline{a}\,\overline{b}\,\overline{c}}$$

$$= \overline{\overline{a}\,b\,c} \wedge \overline{a\,\overline{b}\,c} \wedge \overline{a\,b\,\overline{c}} \wedge \overline{\overline{a}\,\overline{b}\,\overline{c}}$$

$$= \text{NAND}_4(\text{NAND}_3(\overline{a}, b, c), \text{NAND}_3(a, \overline{b}, c), \text{NAND}_3(a, b, \overline{c}), \text{NAND}_3(\overline{a}, \overline{b}, \overline{c}))$$

$\text{NAND}_k(x_1,...,x_k)$ is defined as a k-ary function, for which:

$$\text{NAND}_k(x_1,...,x_k) = \begin{cases} 0 \text{ for } x_1 = ... = x_k = 1 \\ 1 \text{ otherwise} \end{cases}$$

© 2017 Burkhard Stiller

Example: $NAND_2$ Function

Representation of the $NAND_2$ function by the $\overline{\wedge}$ operator:

Problem:

Operators $\overline{\wedge}$ and $\overline{\vee}$ do not show an associative characteristic.

$$(x_1 \mathbin{\overline{\wedge}} x_2) \mathbin{\overline{\wedge}} x_3 \neq x_1 \mathbin{\overline{\wedge}} (x_2 \mathbin{\overline{\wedge}} x_3)$$

$$(x_1 \mathbin{\overline{\vee}} x_2) \mathbin{\overline{\vee}} x_3 \neq x_1 \mathbin{\overline{\vee}} (x_2 \mathbin{\overline{\vee}} x_3)$$

$$NAND_3(x_1, x_2, x_3) = \overline{x_1 \wedge x_2 \wedge x_3} = \overline{(x_1 \mathbin{\overline{\wedge}} x_2)} \mathbin{\overline{\wedge}} x_3$$

$$(x_1 \mathbin{\overline{\wedge}} x_2) \mathbin{\overline{\wedge}} x_3 = \overline{\overline{x_1 \wedge x_2} \wedge x_3} \neq$$

$$x_1 \mathbin{\overline{\wedge}} (x_2 \mathbin{\overline{\wedge}} x_3) = \overline{x_1 \wedge \overline{x_2 \wedge x_3}}$$

© 2017 Burkhard Stiller

Modul 3: Combinatorial Circuits

- Formal Basics of Logics
  - Boolean Algebra
- Normalized and Minimized Forms
- Implementation of Combinatorial Circuits by Gates
- Design of Combinatorial Circuits
  - Minimization and KV Diagrams
  - Function Programming
- Delay Effects

© 2017 Burkhard Stiller

Implementation of Combinatorial Circuits

General hierarchy

- Register transfer level: logic circuits as elements
  - Multiplexer, shift registers, algebraic adder, ... **Examples to come**
- Gate level: logic gates
  - AND, OR, NAND, ...

Just discussed in M3 theoretically, now briefly transferred into gates

- Electronic switch level: Transistors
  - Electrical engineering basics

Example to come

- Layout level: Transistor technologies
  - Electrical engineering basics

Example mentioned in M1



Bottom-up

© 2017 Burkhard Stiller

Gate Level

- Full abstraction from any internal technical details of electrical engineering (almost)
- Full focus on logical behavior
  - Mapping of logic, Boolean functions onto circuits without knowledge of electrical engineering

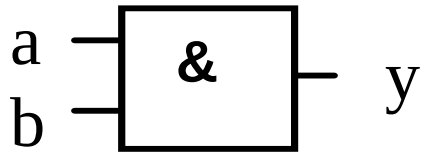
- All circuits are represented in terms of symbols

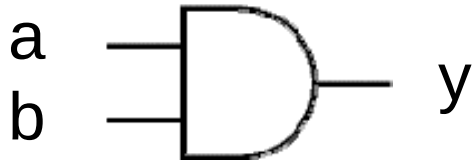
- Algebraic and logic operators differ world-wide:
  - & different symbol for AND “$\wedge$”
  - 1 the output value equals 1, if at all 1-inputs a 1 is applied, thus, OR “$\vee$”
  - ° different symbol for the negation “$\overline{\ }$”

- Further different symbols and operators available on the Web.

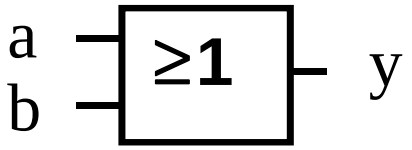
© 2017 Burkhard Stiller

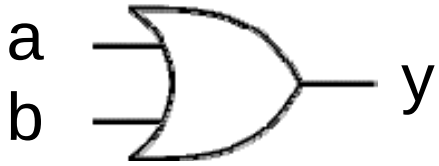
Symboles (DIN 40900 Teil 12, ANSI/IEEE-Standard 91-1984, IEC-Standard & US-Symbole)



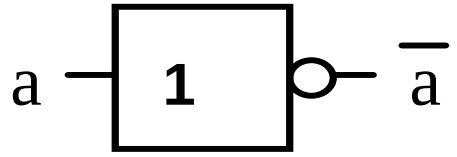


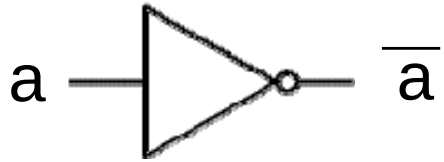
AND Conjunction



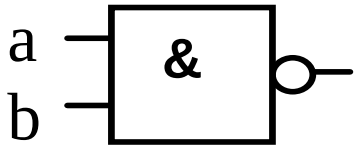


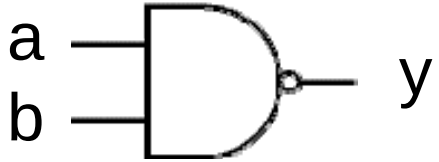
OR Disjunction



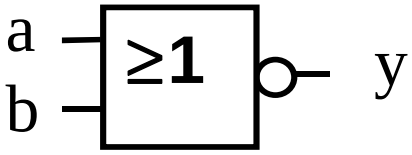


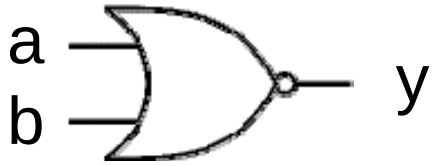
Negation





NAND





NOR

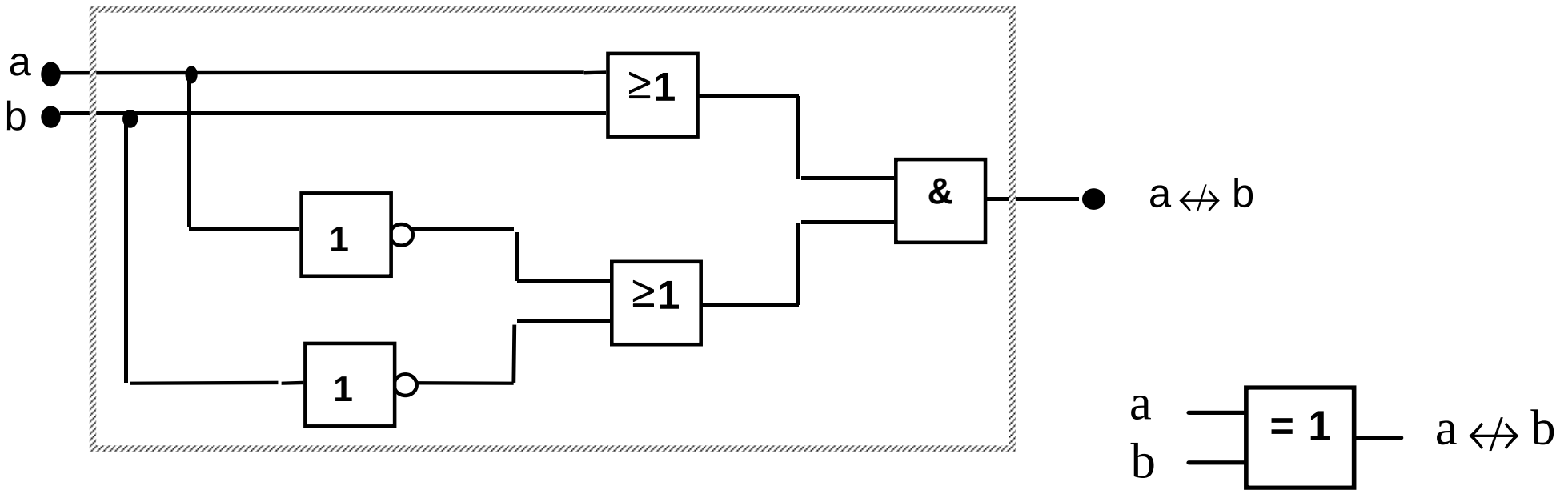
All circuits of these types are termed **gate(s)**.

© 2017 Burkhard Stiller

ifi

Exclusive OR Gate

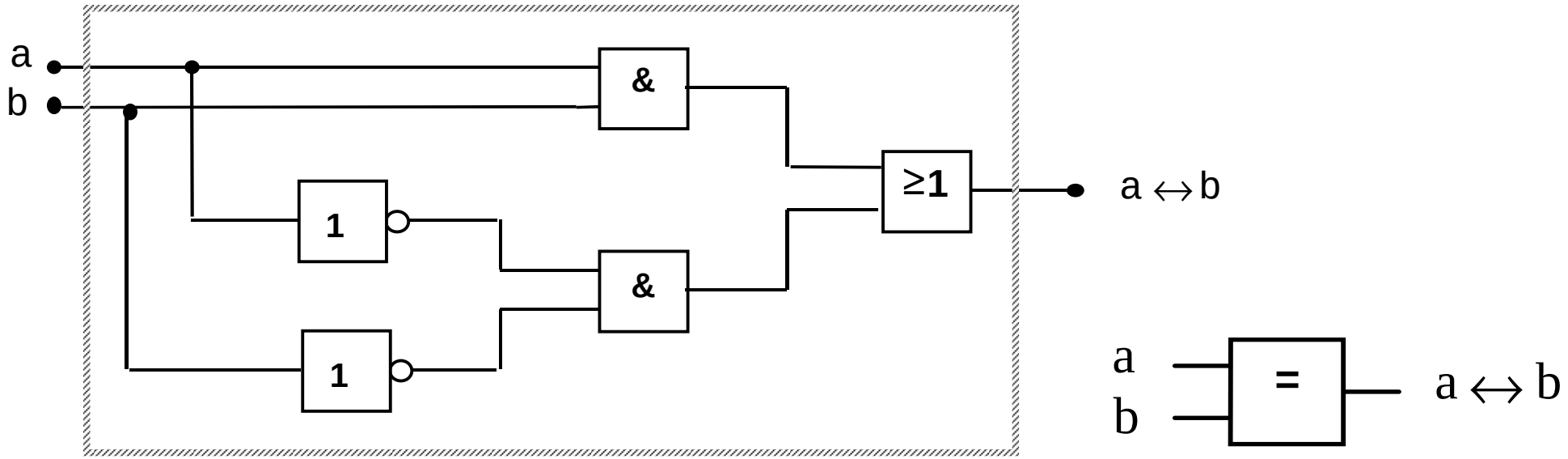
Simple gates are used to set-up hierarchically complex gates, which by themselves use (partially) dedicated symbols.



Exclusive OR (Antivalenz) composed out of five simpler gates. Applied to detect and determine the carry bit (Überlaufbit) in additions.

© 2017 Burkhard Stiller

Equivalence Gate

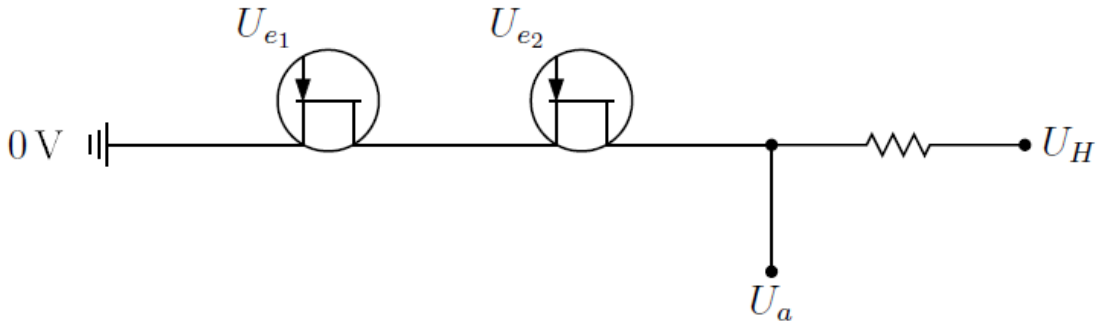


Equivalence (Äquivalenz) composed out of five simpler gates.

© 2017 Burkhard Stiller

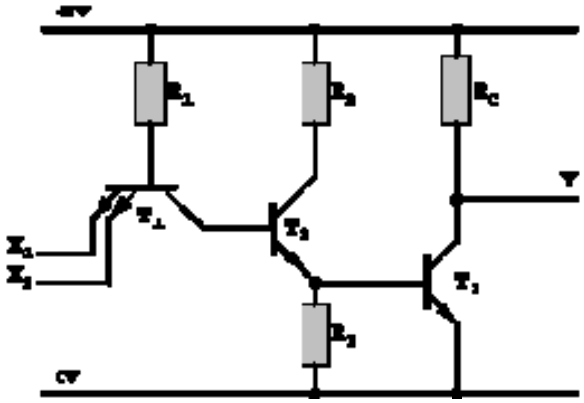
Relation of Gates and Transistors (1)

- U_a with operating voltage (Betriebsspannung) only iff U_{e1} and U_{e2} are not simultaneously powered on: NAND!



Ideal: 2 transistors

- TTL (Transistor Transistor Logic) NAND

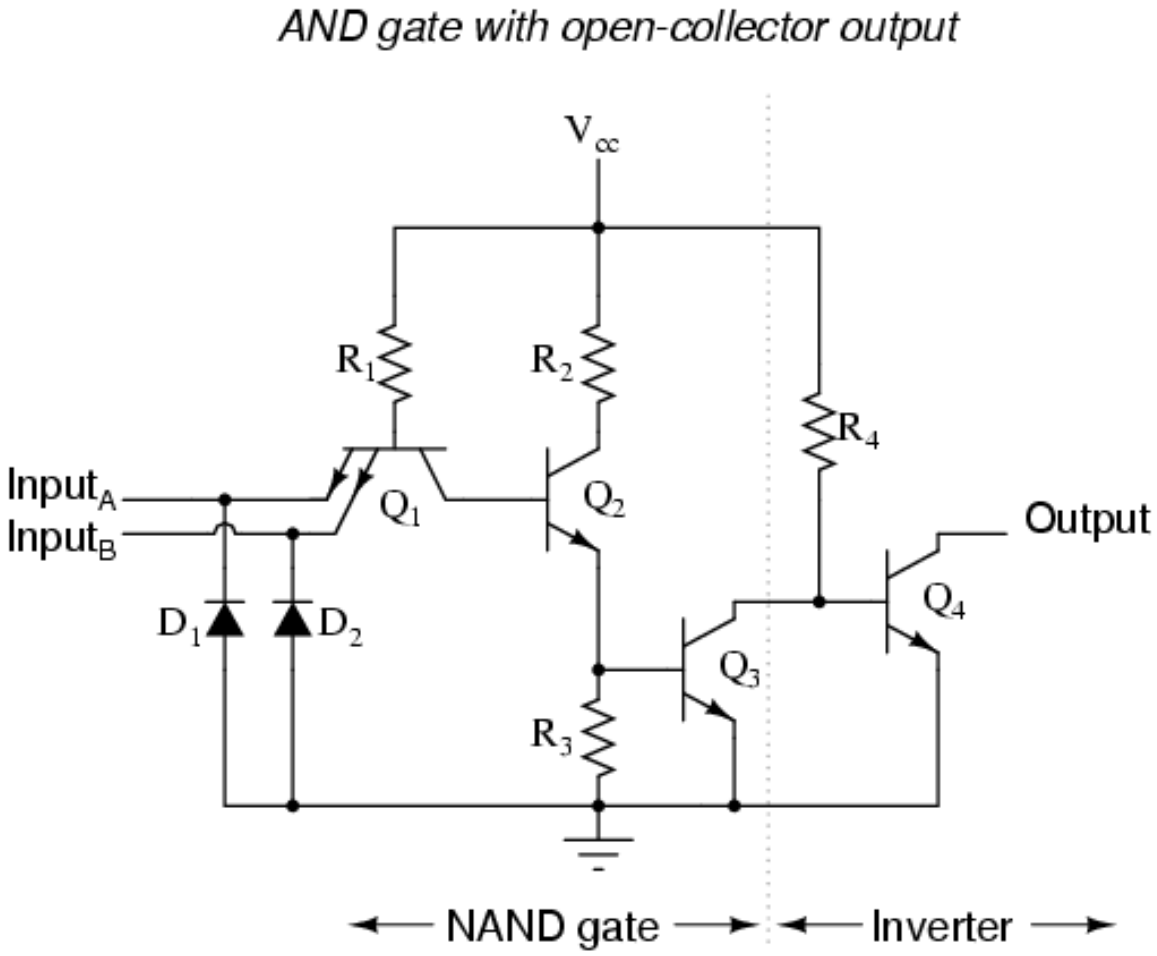


Real: 3 transistors

© 2017 Burkhard Stiller

Relation of Gates and Transistors (2)

- TTL AND (consisting out of NAND and NOT)



Real: 4 transistors

Consequence:

Full operators systems based on NANDs much less space-consuming than the human readable "AND, OR, NOT" system!

© 2017 Burkhard Stiller

Modul 3: Combinatorial Circuits

- Formal Basics of Logics
  - Boolean Algebra
- Normalized and Minimized Forms
- Implementation of Combinatorial Circuits by Gates
- Design of Combinatorial Circuits
  - Minimization and KV Diagrams
  - Function Programming
- Delay Effects

© 2017 Burkhard Stiller

Design of Combinatorial Circuits (1)

- Any practical design of circuits need to address
  - Gates are “ideal”, they generate heat to be diverted
  - Gates need space in terms of square micrometer [μm^2]
  - Gates show delay effects (see end of M3)

- Technical criteria
  - Power consumption, switching time, space, material, durability
  - Correct implementation with respect to static, dynamics (hazards see end of M3)
  - Constraints: #input lines, #output lines, maximal power of inputs/output

- Economic criteria
  - Effort for the design (salaries, simulations, data bases)
  - Effort of implementation (costs per gate, costs per square micrometer [μm^2])
  - Effort of operation (costs for tests, installation, maintenance at operation)

© 2017 Burkhard Stiller

Design of Combinatorial Circuits (2)

- Border line between technical and economic criteria not always clear
  - Selected criteria may be contradictory
    - Enhancing reliability $\Rightarrow$ Increased costs (redundancy, space, …)
    - Lower design costs $\Rightarrow$ Increased design costs
    - Higher density of functionality (less chips) $\Rightarrow$ Increased design costs

➔ The design process has to determine alternatives and for a given problem the best possible compromise on functionality and costs has to be found and selected.

- Major goal: By respecting mandatory technical requirements to reach an economically favorable compromise such that overall full costs are minimized.

© 2017 Burkhard Stiller

Minimization

- Basic assumption
  - Implementation of circuits in two stages
- Representation of function to be found, which minimizes the costs of its implementation, termed in its result as
  - Minimal form
- The approach to generate this minimal form is termed
  - Minimization
- Minimization works in three different ways
  - Algebraic approach
  - Graphical approach
  - Table-based approach
- Algebraic and graphical approaches only work up to 5 or 6 variables, otherwise humans loose the comprehensive view, thus, table-based (maps) approaches are applied thereafter

© 2017 Burkhard Stiller

Reality Check

Minimization Task	Prime Terms	Selection
Number of variables ≤ 6	KV Diagram	KV Diagram Coverage Maps
Assuming DF(CF), in search of DMF(CMF)	Consensus Quine-McCluskey	Coverage Maps (“Überdeckungstabelle”)
Assuming DF(CF) in search of DMF/CMF	Nelson	Coverage Maps

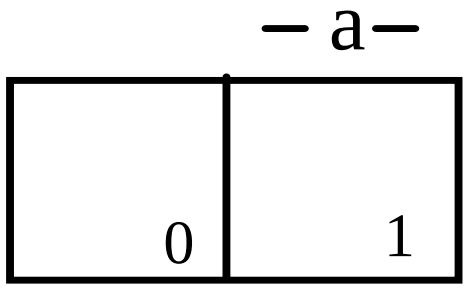
Extended and powerful methods are accessible in selected literature (dedicated field)

© 2017 Burkhard Stiller

Graphical Approach

- KV Diagram (Karnaugh and Veitch)
  - Also termed Karnaugh map, Karnaugh diagram

- Rectangle, for which the right half is assigned to the variable a and the left half is assigned to $\overline{a}$:



KV diagram for a single variable

© 2017 Burkhard Stiller

KV Diagram (1)

- Numbered fields into determine the index of this variable‘s assignment
  - Index equals minterms, with the value 1
- By adding the truth values 0 or 1 in each of the KV diagrams fields, the Boolean function is characterized uniquely
- A KV diagram defines an additional representation of Boolean functions, especially considered as an alternative to function tables

- Examples:

y:

$\bar{a}$	a
0_0	1_1

$y = a$

z:

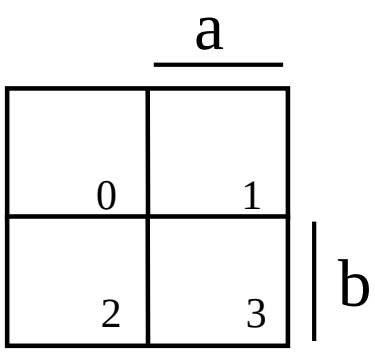
$\bar{a}$	a
1_0	0_1

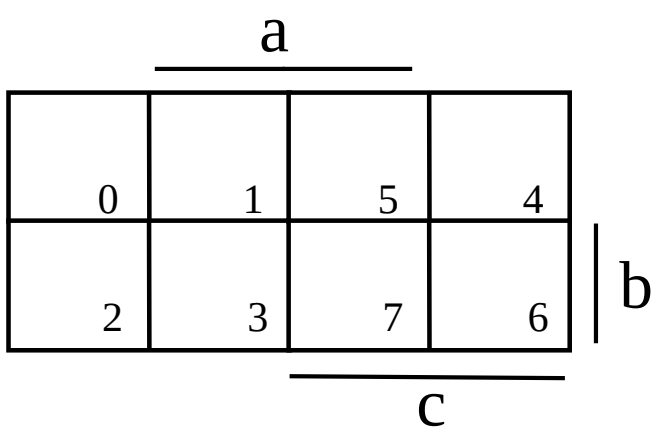
$z = \bar{a}$

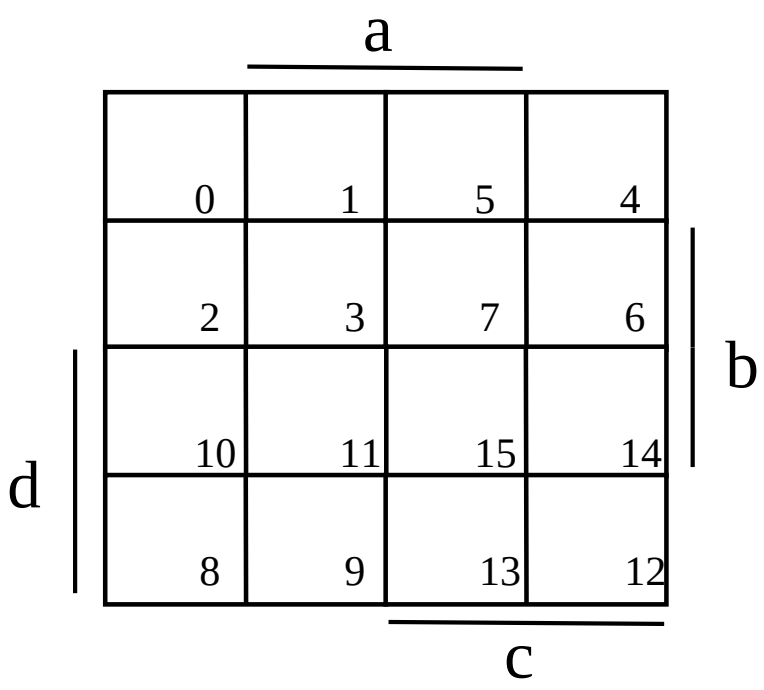
© 2017 Burkhard Stiller

KV Diagrams (2)

- KV diagrams for additional variables are defined via mirroring
  - Each additional variable doubles the number of possible assignments



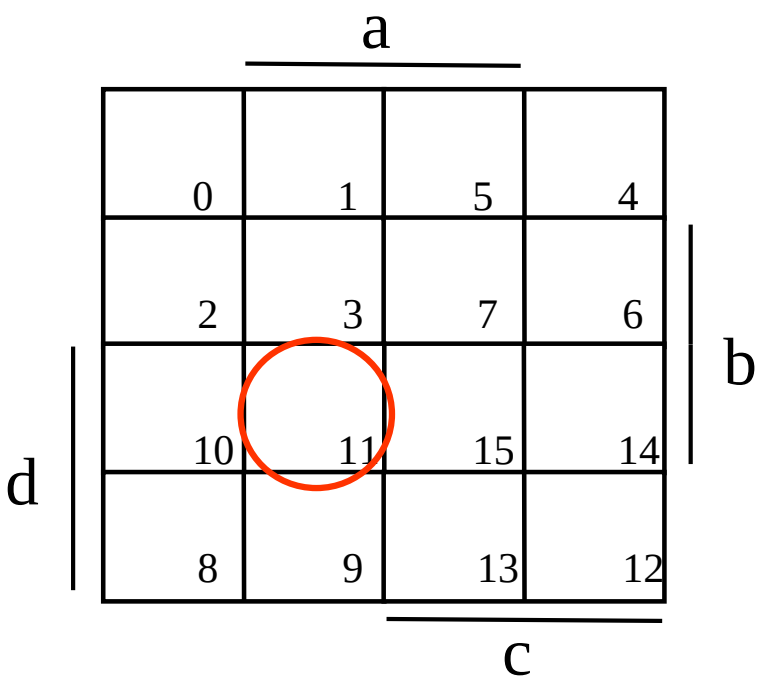




© 2017 Burkhard Stiller

KV Diagrams (3)

- Every field is uniquely identified by reading the edges‘ names
  - Field 11 is assigned to: a b $\overline{c}$ d
- The index with every field defines the index of this field‘s assignment of variables, while all variables are marked in in alphabetically reverse order
  - Field 11 = 1011_2 = d $\overline{c}$ b a



© 2017 Burkhard Stiller

KV Diagrams — Set-up (1)

- Assuming that the function *y* is defined by a function table
  - Each row of that function table corresponds to a single field in the KV diagram
  - ⇒ For each row of the functions table the corresponding indix will be determined and the function‘s value will be inserted

- Hint, to find all fields within the KV diagram:

  - Include all input variables in alphabetically reverse order into the table
  - In turn, the KV diagram can be completed according to their indices, which originate from the table‘s top to the table‘s bottom row

© 2017 Burkhard Stiller

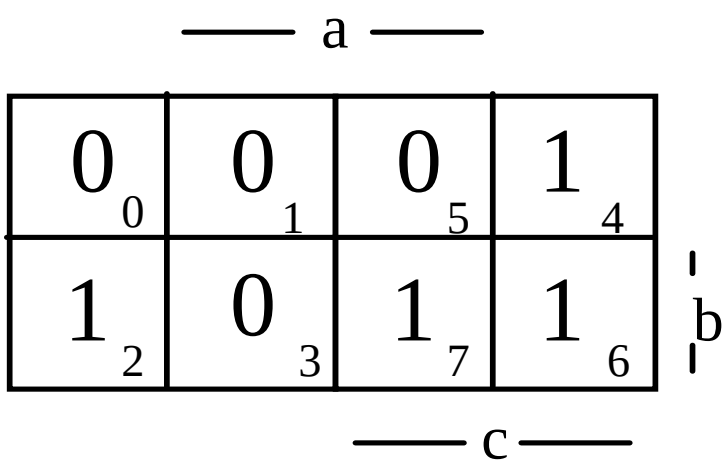
KV Diagrams — Set-up (2)

Example: $y = \overline{a}\, b \vee b\, c \vee \overline{a}\ \overline{b}\, c$

Function table:

Index	c	b	a	y
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	0
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Resulting in KV diagram:

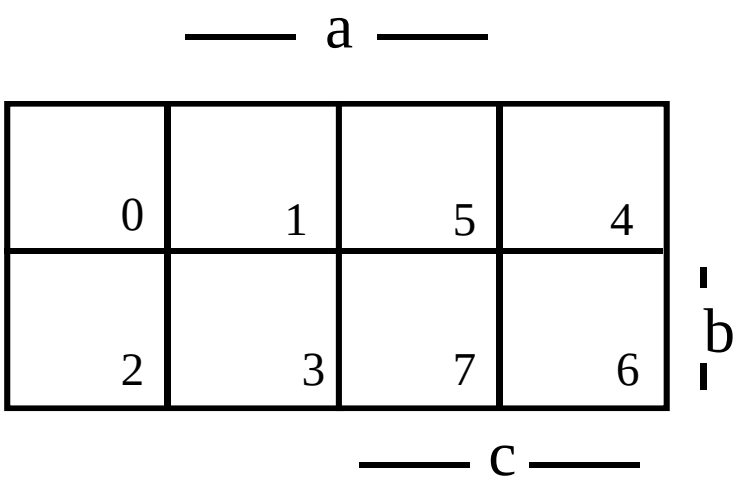


© 2017 Burkhard Stiller

Characteristics of KV Diagrams (1)

- Key characteristic
  - Minterms located symmetrically to any center line differ only in a single variable

- Example:



or

Minterm 0 = $\overline{c}\ \overline{b}\ \overline{a}$
Minterm 1 = $\overline{c}\ b\ \overline{a}$

or

Minterm 0 = $\overline{c}\ \overline{b}\ \overline{a}$
Minterm 4 = $c\ \overline{b}\ \overline{a}$

or

Minterm 1 = $\overline{c}\ \overline{b}\ a$
Minterm 3 = $\overline{c}\ b\ a$

© 2017 Burkhard Stiller

Characteristics of KV Diagrams (2)

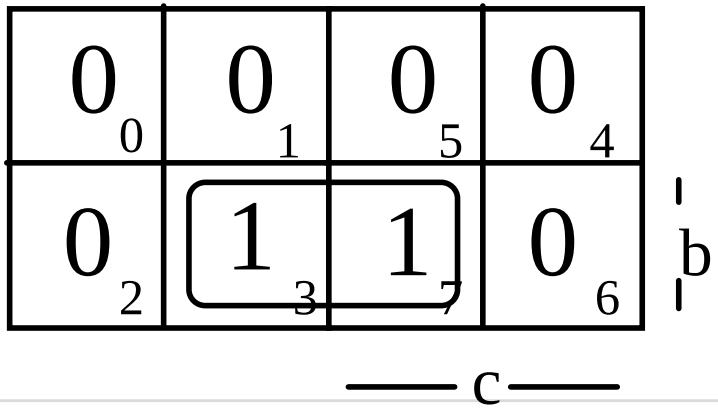
- Boolean algebra tells that terms differing in a single variable only may be minimized!

- Example:

$$a\,b\,c \vee a\,b\,\overline{c} = a\;b\;(c \vee \overline{c})$$
$$= a\;b$$

- Thus, the resulting term does not contain the variable *c* anymore!

- In turn, minterms located symmetrically to any center line can be combined to a simpler term!



© 2017 Burkhard Stiller

Definition: Prime Implicant and Selection

- An implicant p is defined as a prime implicant, if no implicant q ≠ p exists, which is implied by p.

  $\forall q: \quad q \neq p \Leftrightarrow \neg(p \rightarrow q)$

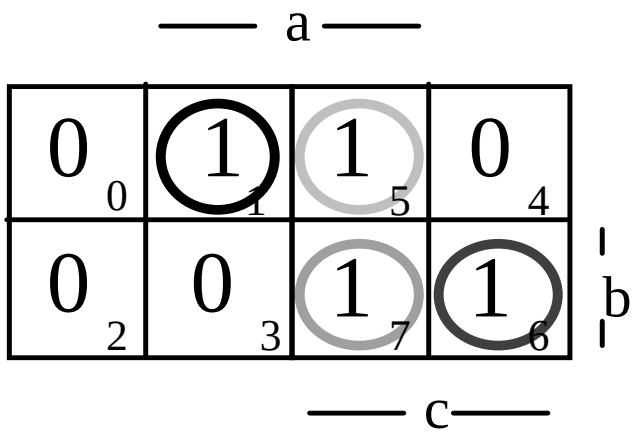
  - *I.e.,* p embraces the largest possible order (von größtmöglicher Ordnung).
- Proposition:
  Every function can be represented as a disjunction of its prime implicants.

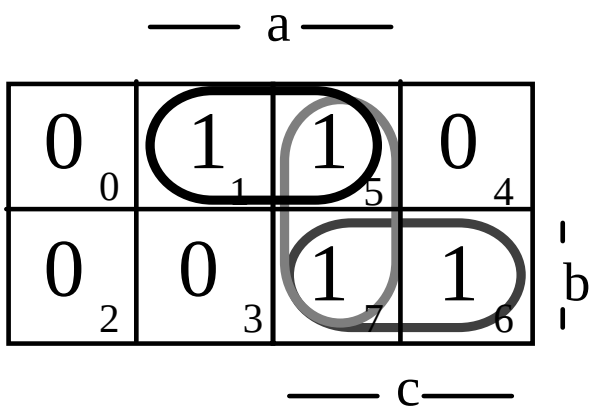
- Prime implicants are accessible from any KV diagram
  - Select “manually” the largest possible blocks of ones, such that each block contains 2^k fields

- Example (continued next slide):

  $f = \overline{a}\ b\ c \vee a\ c \vee a\ \overline{b}\ \overline{c}$

Example (1)





4 minterms:
($a\ \bar{b}\ \bar{c}$, $a\ \bar{b}\ c$, $a\ b\ c$, $\bar{a}\ b\ c$)

3 prime implicants:

first order implicants

($a\ \bar{b}$, $a\ c$, $b\ c$)

But ($a\ \bar{b}$, $b\ c$) would be sufficient!

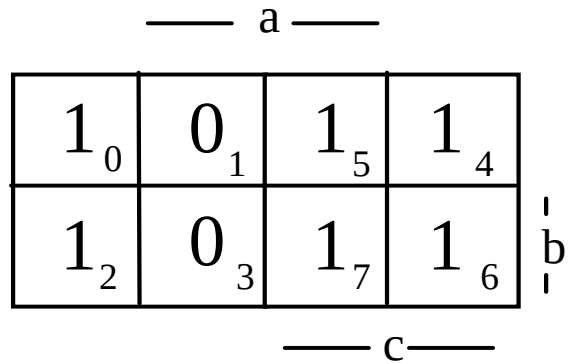
- $f = a\ \bar{b}\ \bar{c} \vee a\ \bar{b}\ c \vee a\ b\ c \vee \bar{a}\ b\ c$
  $= a\ \bar{b} \vee a\ c \vee b\ c = a\ \bar{b} \vee b\ c$

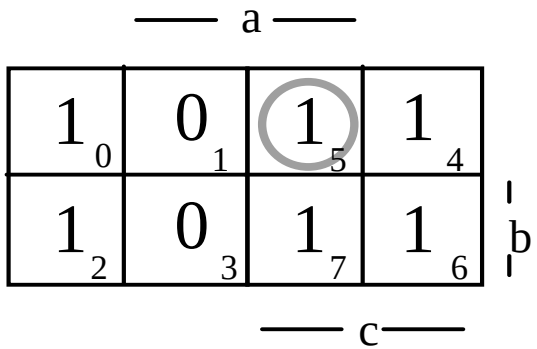
© 2017 Burkhard Stiller

Example (2)

f = minterms (0,2,4,5,6,7)

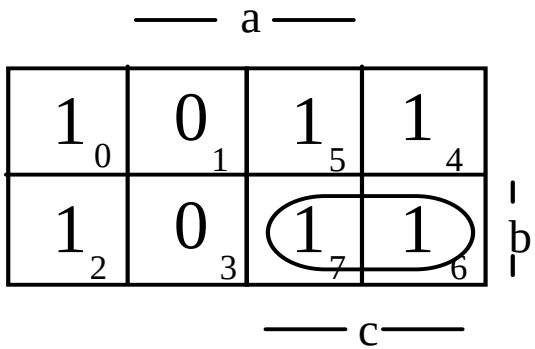
f :





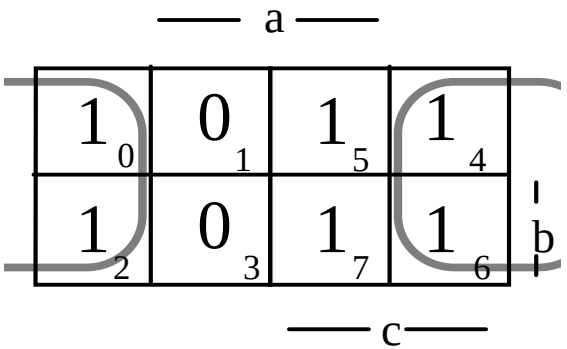
$g = a \bar{b} c$

determines an implicant



$g = b c$

determines an implicant



$g = \bar{a}$

determines an implicant

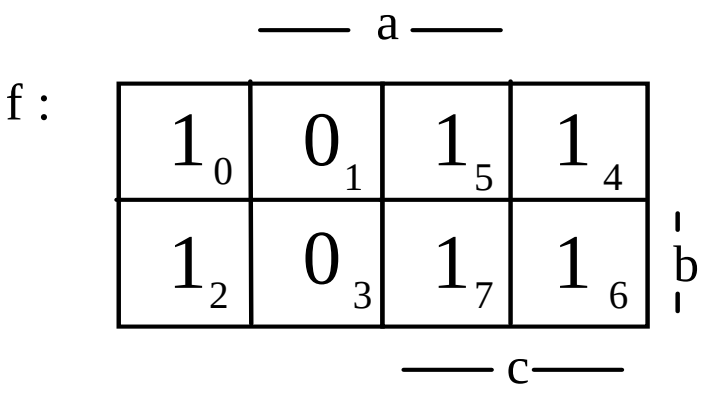
© 2017 Burkhard Stiller

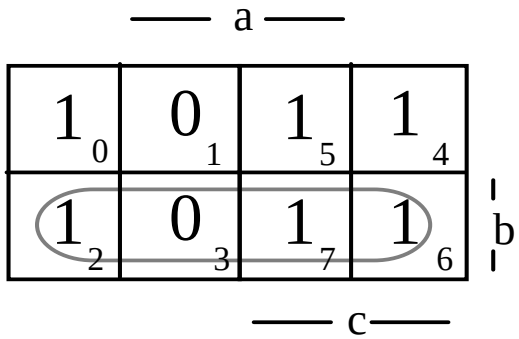
ifi

Example (3)

f = minterms (0,2,4,5,6,7)

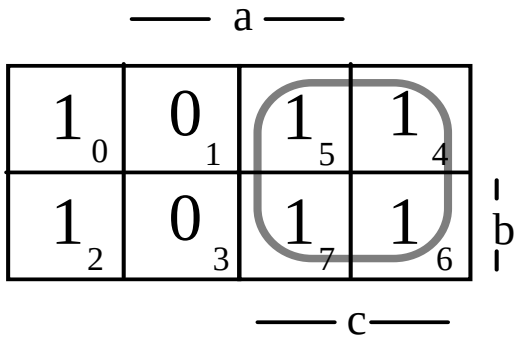
f :





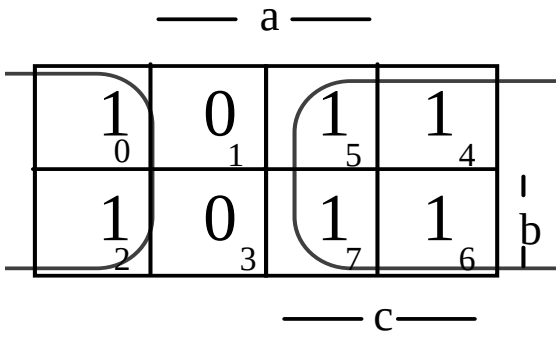
g = b

does not determine an implicant



g = c

determines an implicant



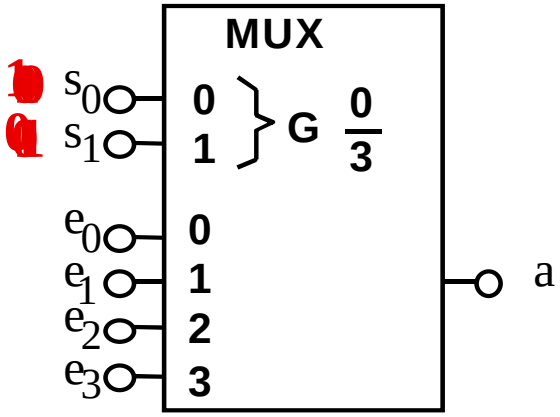
$g = \bar{a} \vee c$

does not determine an implicant

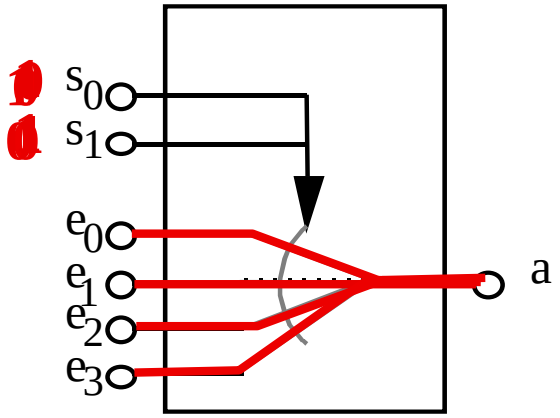
(Impl.)

Alternatives for Functions‘ Implementations

- Implementation of re-appearing functions by two alternatives:
  - By multiplexers/demultiplexers
  - Memory circuits
- A multiplexer (MUX) defines a circuit with multiple inputs and one output, for which by n control lines one of the 2^n inputs is selected and switched to the single output
- Multiplexers are classified according to their size
  - 2^n:1 - Multiplexer (alternatively as a 1-out-of-2^n-multiplexer)
- Example 1-ouf-of-4-multplexer:



s_1	s_0	a
0	0	e_0
0	1	e_1
1	0	e_2
1	1	e_3



© 2017 Burkhard Stiller

Multiplexer-based Functions‘ Implementation

- Traditionally, a multiplexer controls data flows
- Here, it is applied to implement Boolean functions
  - A 2^n: 1 - multiplexer can implement Boolean functions with n+1 variables
- An implementation table defines the behavior
  - 2^n columns for possible assignments of n control lines
  - 2 rows for the negated (negative) and non-negated (positive) (n+1)-st variable
- Content and assignment
  - The table contains all function‘s values depending on the input variables
  - Finally, each column is considered separately and gets assigned a 1-ary function $g \in \{0, 1, a, \overline{a}\}$, with which the input will be applied to according to the combination of all control variables

© 2017 Burkhard Stiller

Example

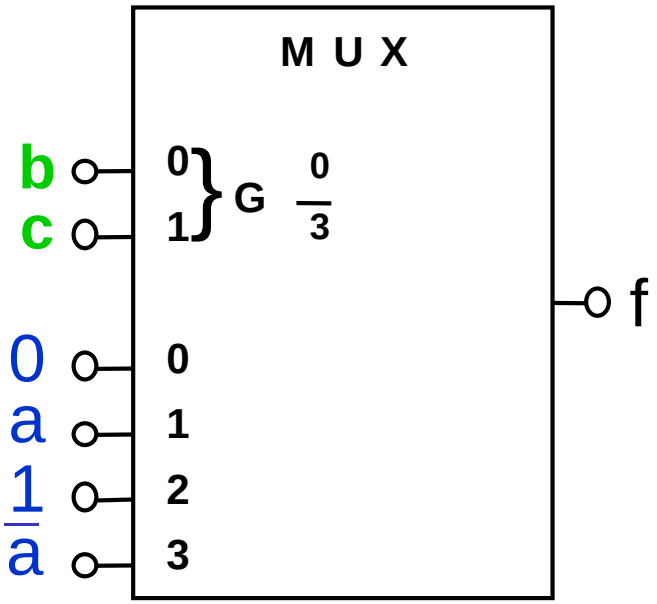
Implementing function f with a multiplexer:

$$f = \bar{a}\, c \vee \bar{b}\, c \vee a\, b\, \bar{c}$$

Implementation table in case of the selection of *b* and *c* as control inputs:

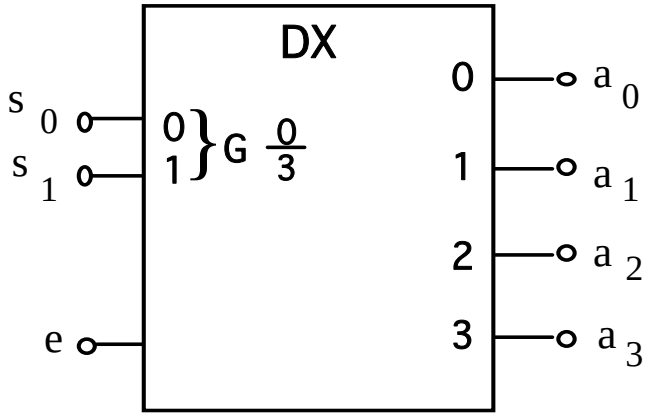
c b	0 0	0 1	1 0	1 1
a = 0	0	0	1	1
a = 1	0	1	1	0
f =	0	a	1	$\bar{a}$

Implementation



Demultiplexers/Decoders

- A circuit, which maps a single input depending on those *n* control lines onto one of the 2^n outputs, is termed demultiplexer
- Example 1-out-of-4 Demultiplexer:



s_1	s_0	a_0	a_1	a_2	a_3
0	0	e	0	0	0
0	1	0	e	0	0
1	0	0	0	e	0
1	1	0	0	0	e

- Remarks
  - A demultiplexer offers an enable input *e* and *n* inputs s_i for a binary number, which is decoded at the 2^n ouptputs a_j
  - If the enable input *e* = 0, all outputs are at 0, otherwise a 2-bit number is decoded
    - Example: By applying the number 2 (s_1= 1, s_0= 0), the ouput at a_2= 1 and all other outputs remain with 0
- Thus, a demultiplexer is termed also decoder

© 2017 Burkhard Stiller

Memory-based Functions‘ Implementation

- All circuits discussed so far, such as gates, multiplexers, or decoders, are defined by the function known in advance

  ➔ hardwired logic

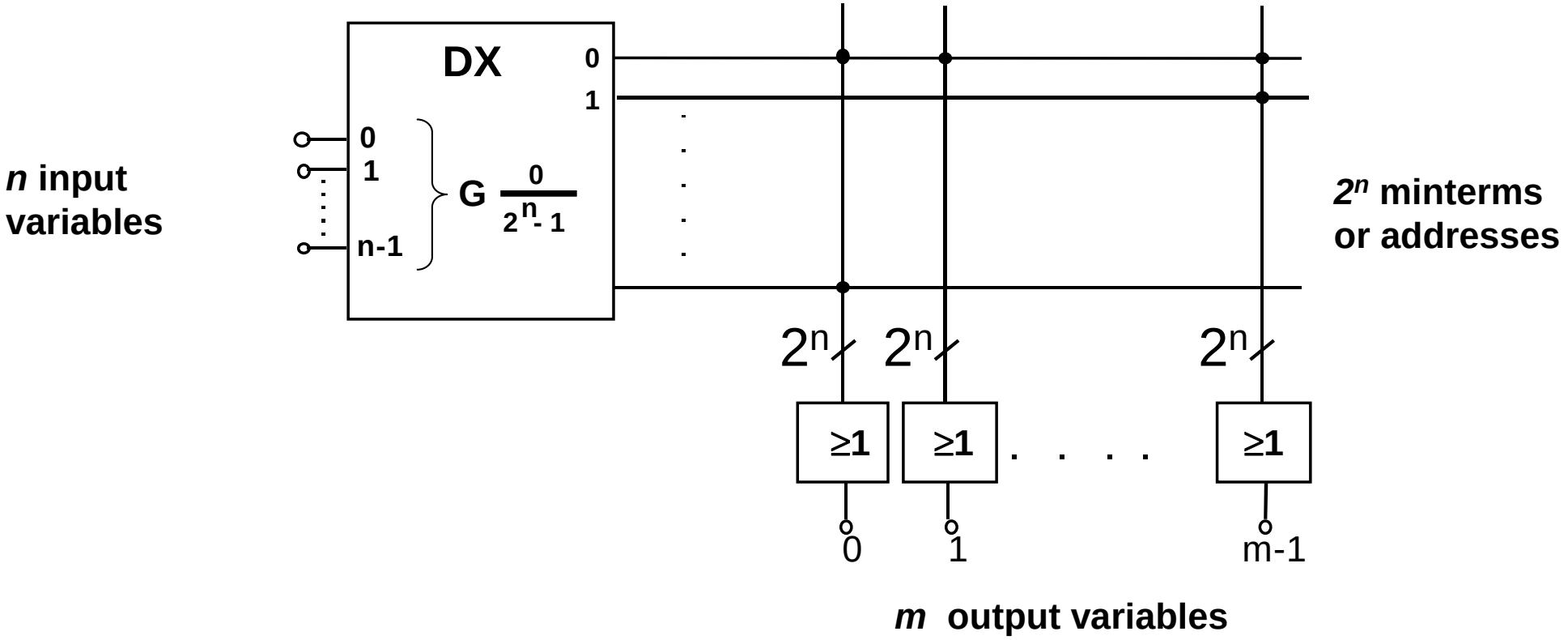
- However, circuits with multiple functions on board have to offer a higher flexibility to be adaptable to many application demands. Such an adaptation is termed personalization or programming

  ➔ micro-programmable logic

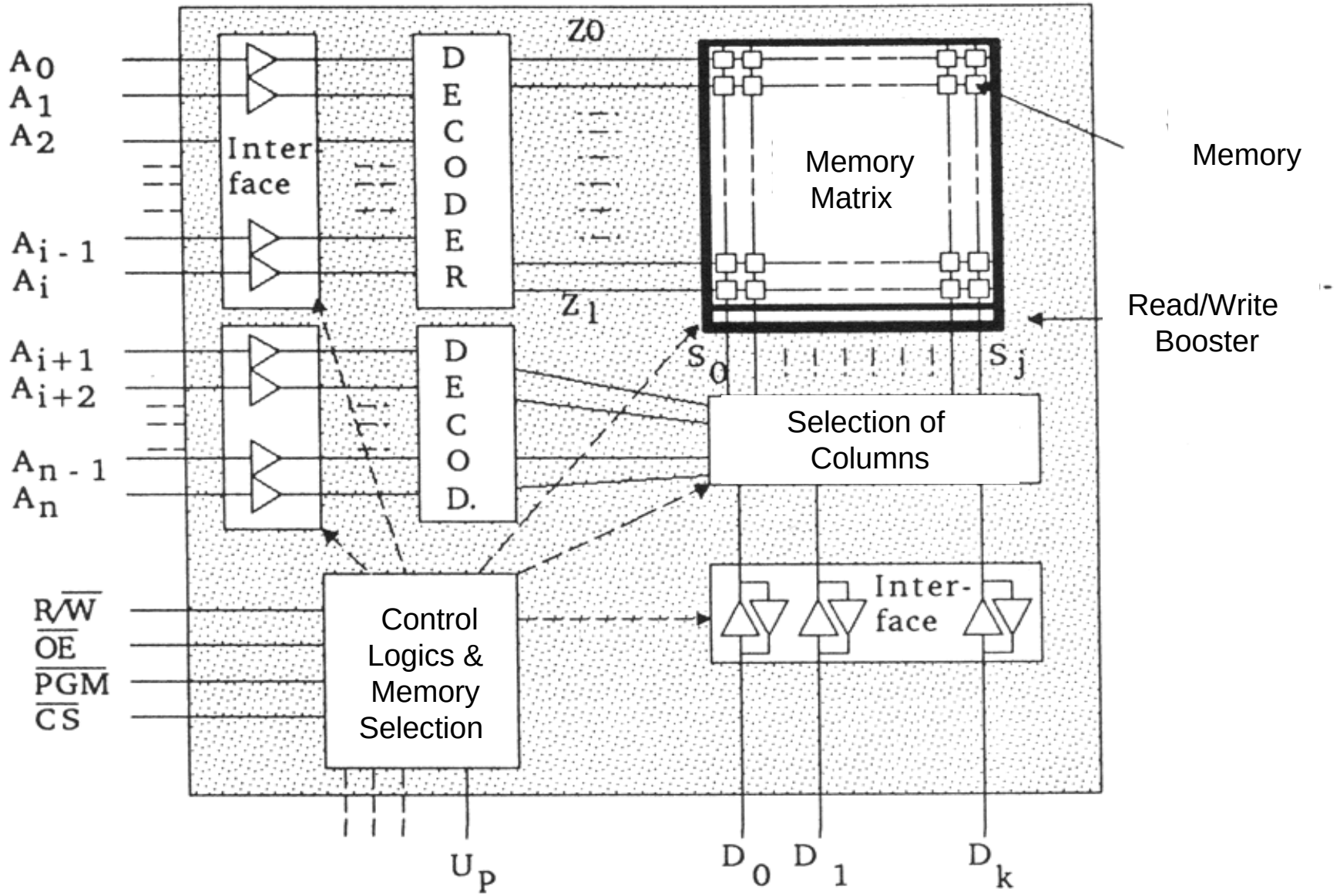
  (cf. M6 for CPUs)

Scheme of a Memory Cells

Scheme of memory cells, in which arbitrary function tables can be stored into:



Organization of Memory Cells (1)



© 2017 Burkhard Stiller

ifi

Organization of Memory Cells (2)

- The input configuration (Eingangssignale) determines the selection of a memory cell (addressing) and the value stored at this address will be offered at the output configuration (Ausgangssignale)
- The output configuration of the decoder determines those min-terms of n input configurations (the rows of the function table)
- Storing the value 1 for a dedicated output configuration *i* means that this minterm is OR-connected to this *i*-th output; the value 0 indicates that this minterm will not be used

© 2017 Burkhard Stiller

Memory Types (1)

- The personalization of memory distuingishes a set of various different memory types with different characteristics:

- ROM (Read Only Memory)

- Memory circuits which can only be read during operation
- Programming was performed only once initially at the vendor‘s side (mask-programmable ROMs)
- Content of storage will remain unchanged for its entire lifetime

© 2017 Burkhard Stiller

Memory Types (2)

- PROM (Programmable Read Only Memory)
- Programmable ROMs, which are programmed by the user
- Programming:
  - Melting of fusible links (microscopic small connectors)
  - Stored charge based on dedicated physical processes, which maintain the charge within the circuit over years

© 2017 Burkhard Stiller

Memory Types (3)

- EPROM (Erasable Programmable Read Only Memory)
- A user-programmable memory circuit, which can be erased by applying UV light and which may be re-programmed again
- Very small quartz windows on top cover of circuit provide access
- Memory circuits with electrical erasing options (higher voltage):
  - EEPROM: Electrically Erasable PROM
  - EAROM: Electrically Alternable ROM
  - Application for memories of digital cameras (Compact Flash, Memory Stick)

© 2017 Burkhard Stiller

Memory Types (4)

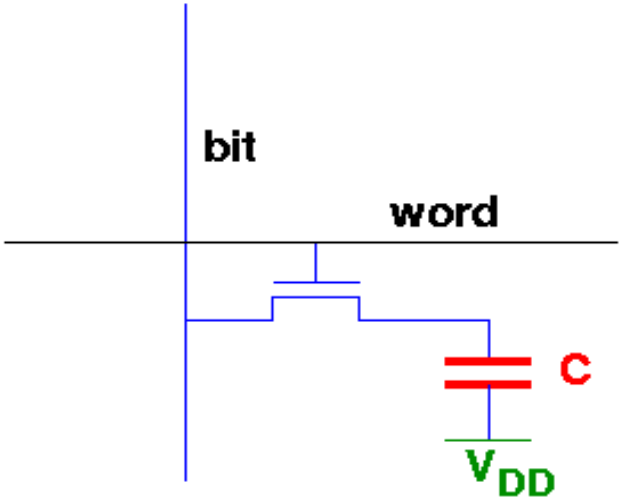
- RAM (Random Access Memory)
- Memory, which enables reads and writes during normal operation
- A RAM will loose its programmed content once it will be decoupled from the power supply!
- Different implementations:
  - Dynamic RAM (DRAM)
  - Static RAM (SRAM)

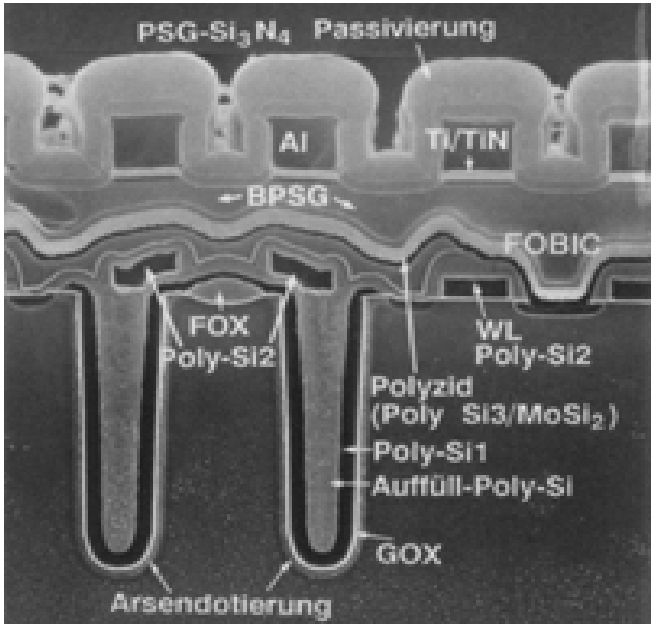
© 2017 Burkhard Stiller

Dynamic RAM (DRAM)

- Capacitor operates as charger, a transistor is used to attach the control

- Memory needs to be refreshed regularly (memory refresh)



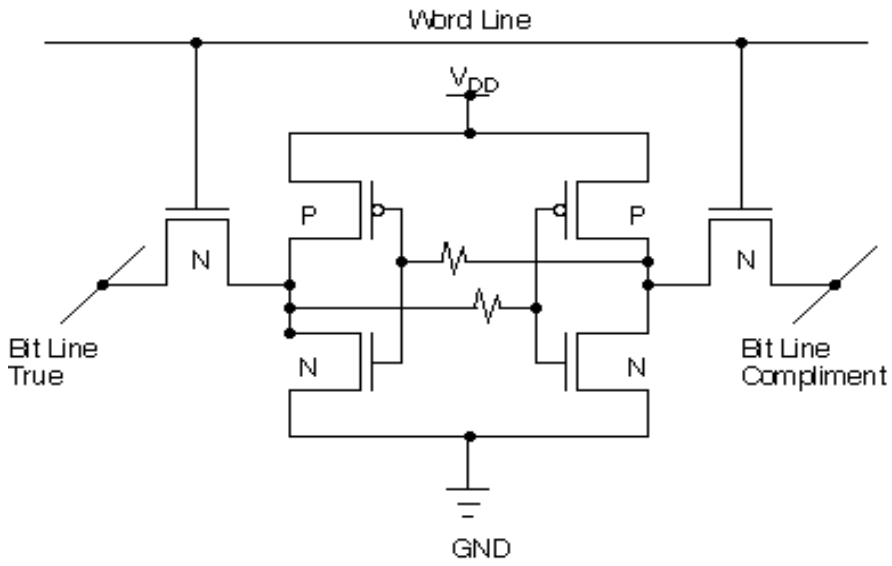


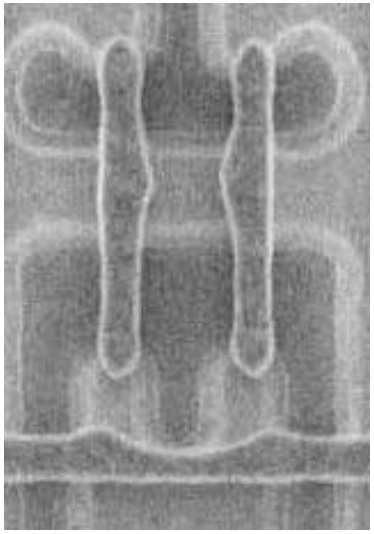
© 2017 Burkhard Stiller

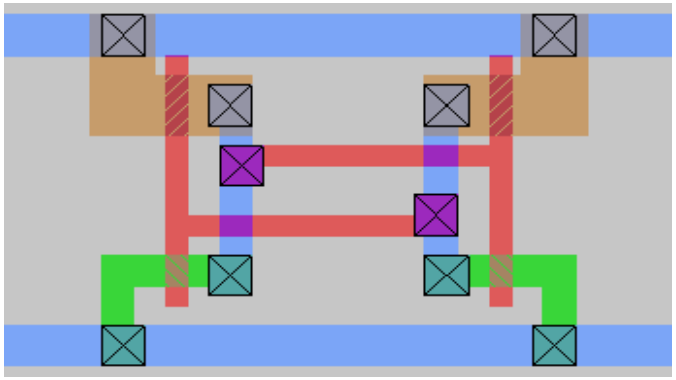
ifi

Static RAM (SRAM)

- The memory cell is implemented by a flipflop (cf. M4)
  - Memory maintains content even without an active power supply
  - Access times to a static RAM are much shorter compared to dynamic RAMs
  - A static memory cell requires about 6-8 transistors, while a dynamic memory cell can easily work with 1 transistor, more reliably with 2 transistors







© 2017 Burkhard Stiller

Comparison of RAM and ROM

- Key differences
  - RW: *read/write*
  - Access Times (AT) for R/W
  - Maintainng memory‘s content (operations OP) without power supply
  - Memory size (MS)

	ROM	PROM	EPROM	Flash-PROM	DRAM	SRAM
RW	R	R	R ((W))	R (W)	RW	RW
AT	+	+	+ / –	+ / –	++	+++
OP	+	+	+	+	–	–
MS	+	+	+	+	+	+

Programmable Logic Array (PLA)

- Up to now the entire function table was stored in memory cells, while the function was implemented by its DNF (disjunctive normal form)
- Instead, by applying the DMF (minimal disjunctive normal form) many functions can be implemented efficiently (fewer memory cells)
- PLA (Programmable Logic Array):
  - In comparison to ROMs PLAs do see as the input configuration prime implicants for covering the function, instead of minterms
  - The decoder used up to now will be replaced by an AND matrix

© 2017 Burkhard Stiller

FPLA and PAL

- PLAs are being personalized during production (similar to ROMs)
- A user-programmable PLA with a fixed number of input configurations *n*, product terms *k*, and output configurations *m* is termed Field Programmable Logic Array (FPLA)

- Technical alternatives include PAL circuits, which did personalize already the AND matrix and the OR matrix at the time of production

- The role of PALs and FPLAs is determined by the demand for “similar” functions to be available in many devices, which may differ in their specific instantiations: reduction of costs, but no need to be fully flexible (programmable), but with fast operation

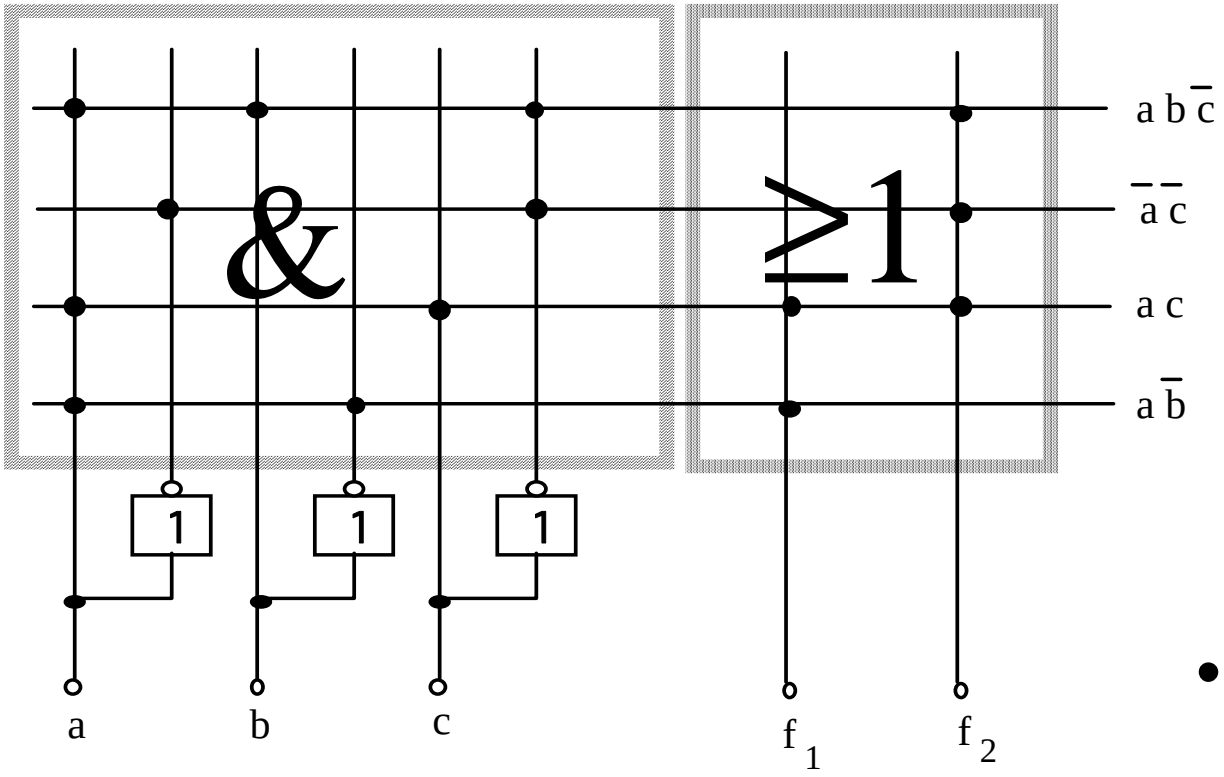
© 2017 Burkhard Stiller

Example: Implementation via PAL

Assume that the functions f_1 and f_2, already have been minimized:

$$f_1 = a\,\bar{b} \vee a\,c \qquad \text{and} \qquad f_2 = \bar{a}\,\bar{c} \vee a\,c \vee a\,b\,\bar{c}$$

Their PAL implementation reads as follows:



- Connector

Modul 3: Combinatorial Circuits

- Formal Basics of Logics
  - Boolean Algebra
- Normalized and Minimized Forms
- Implementation of Combinatorial Circuits by Gates
- Design of Combinatorial Circuits
  - Minimization and KV Diagrams
  - Function Programming
- Delay Effects

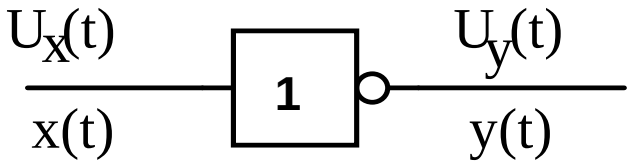
© 2017 Burkhard Stiller

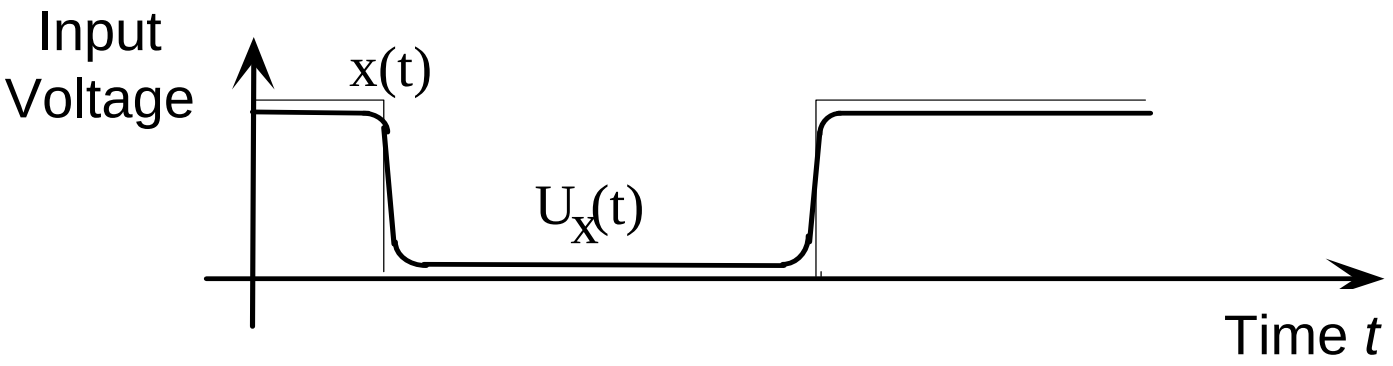
Delay Effects

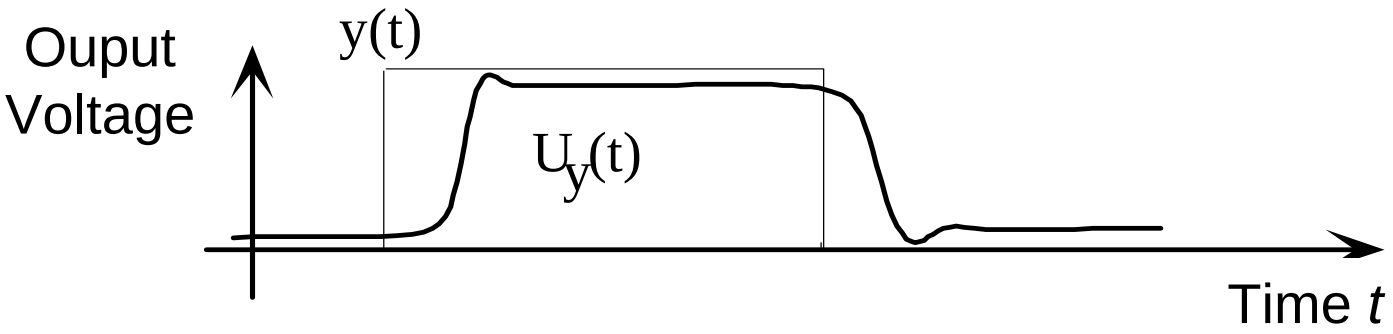
- The discussion of gates assumed so far that any logical operation was performed idealistically!
- In reality, gates are implemented by transistors, resistors (Widerstände), and capacitors, which are visible on the circuit's layout level.
- Thus, a temporal signal curve for an implemented gate differs from its ideal Boolean signal curve!

© 2017 Burkhard Stiller

Realistic and Ideal Signal Curve (Inverter)





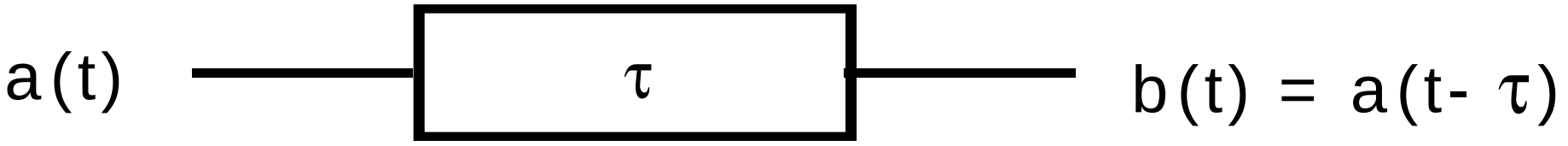


© 2017 Burkhard Stiller

Realistic Characterization of Gates

- Realistic effects on the gate level have been modeled in different models and complexities

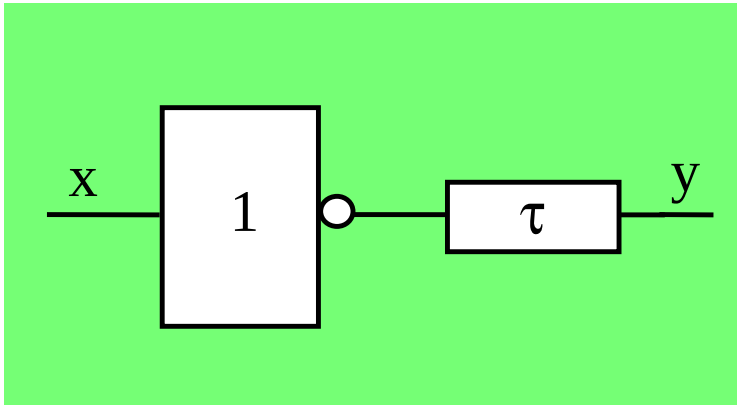
- A simple model is termed Delay Model (Totzeitmodell)
  - Only signal delays are considered, which exist due to gates and links
  - A real gate is modeled by:
    - An ideal gate without any delays and
    - A delay element (Totzeitglied) with a defined delay, which depends on the switching time of the gate's implementation
    - The temporal behavior of a binary variable following the delay element is identical as of before the delay element, however only delayed by the time τ

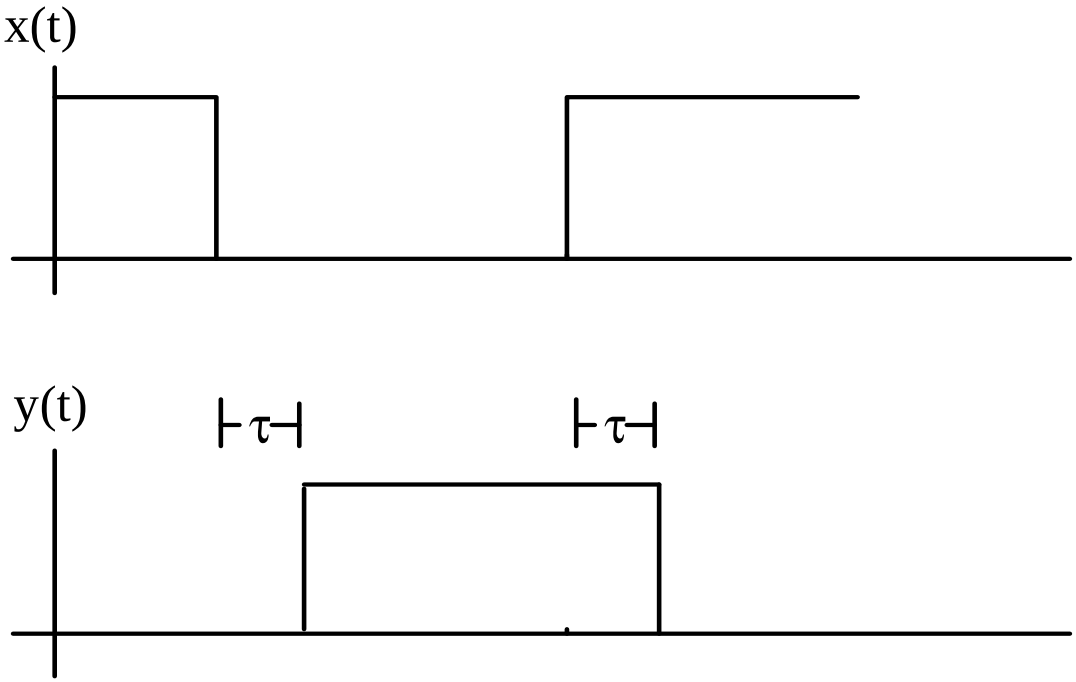


© 2017 Burkhard Stiller

Example: Delay Model of the Inverter/Negator

- Such a simple model allows for an investigation and a principle understanding of delay effects, however, the model is still too idealistic for concrete system designs

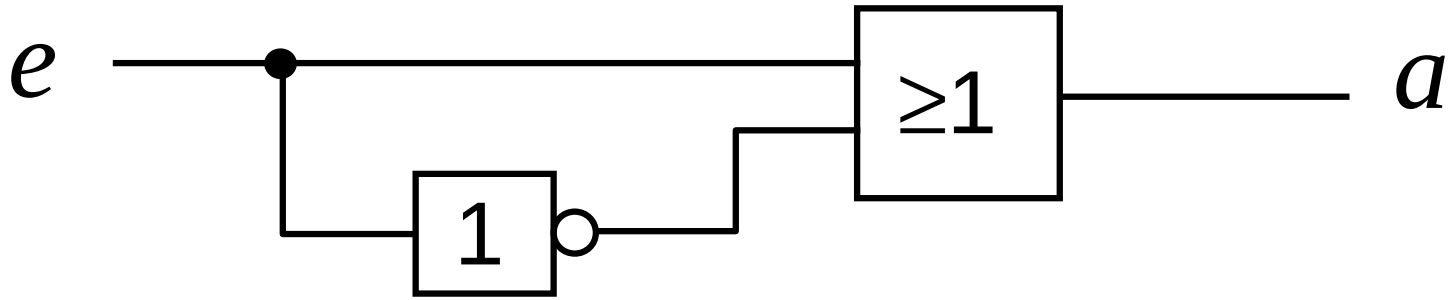




© 2017 Burkhard Stiller

Example: Application of the Inverter

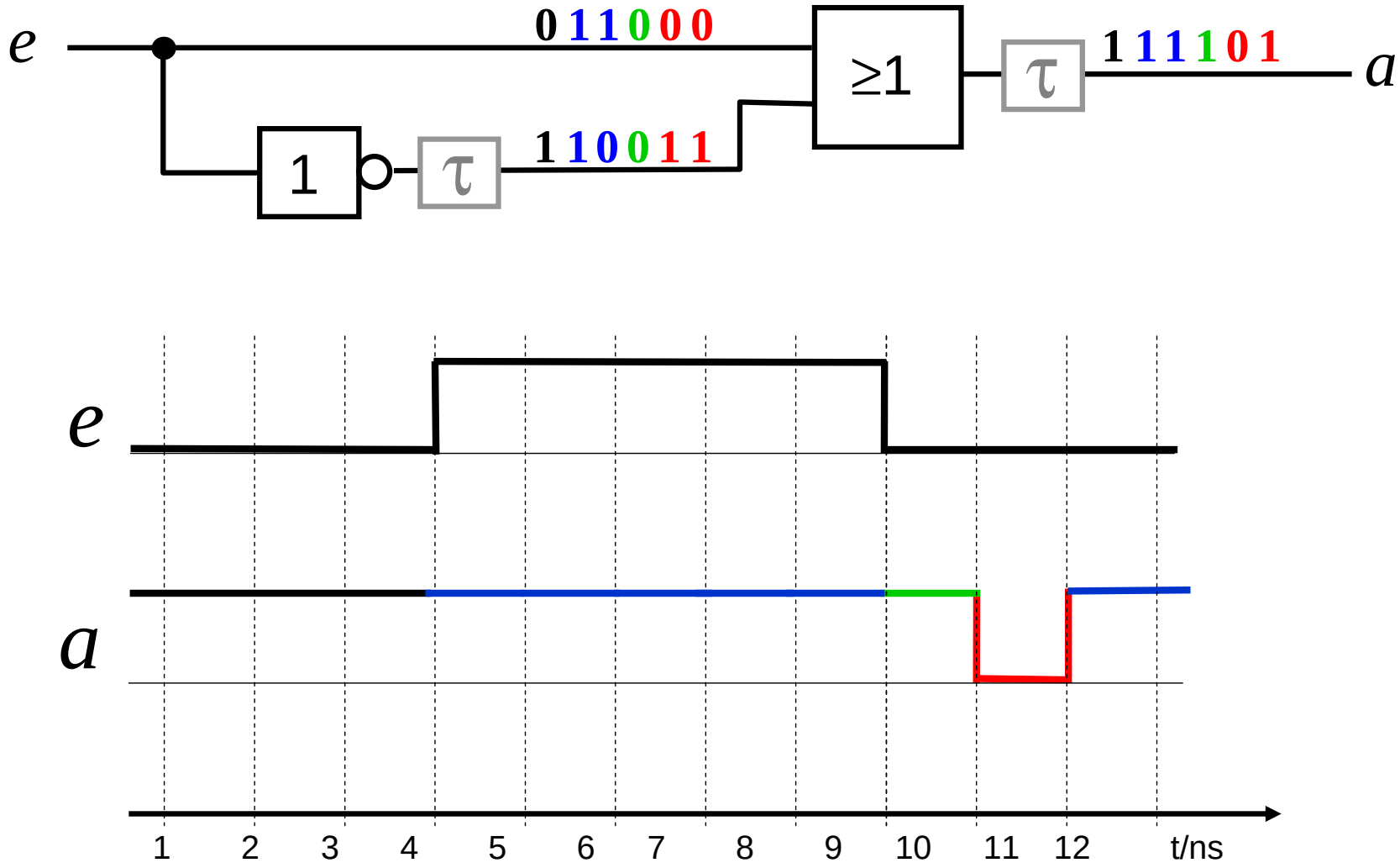
Assume:



$$a = e \vee \overline{e} = 1$$

Both gates shall show a delay of 1 ns

Time Chart



Behavior of Combinatorial Circuits

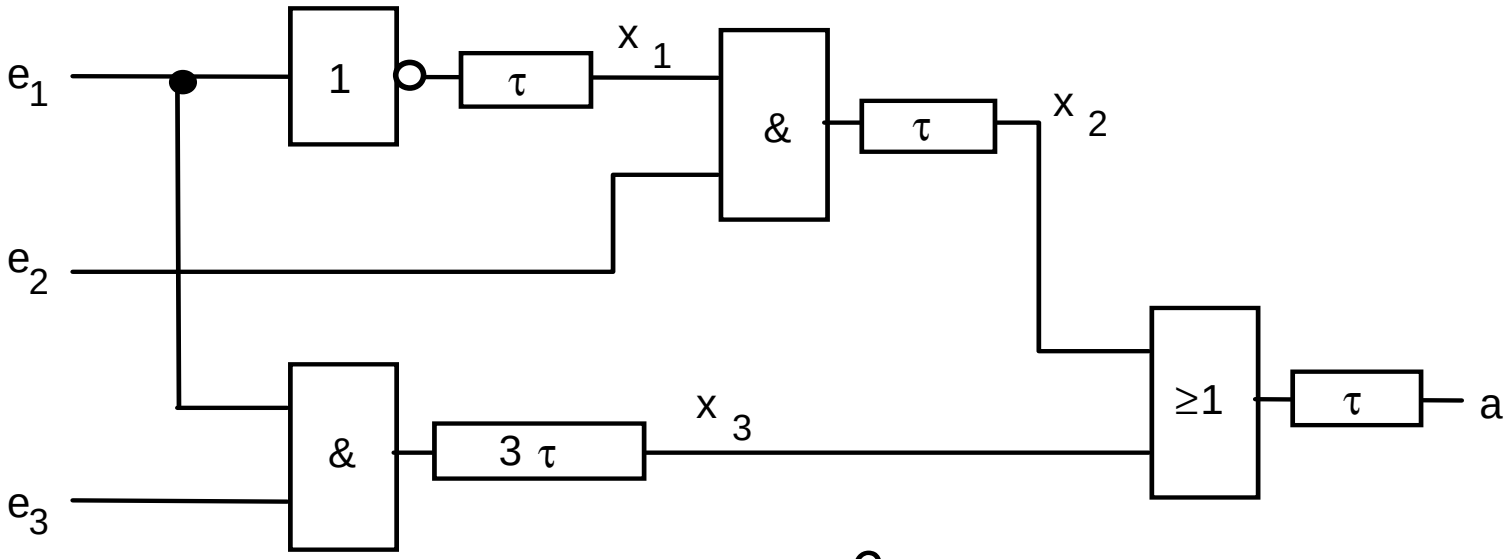
- Ideal combinatorial circuit
  - The output configuration does not change in case of previous and current assignments at the input utilizing the same values
  - The output configuration does change exactly once in case of previous and current assignments at the input utilizing different values
- Realistic combinatorial circuit
  - A change of the input configuration potentially results in different signal delays throughout the circuit
- Multiple changes of the output configuration, and thus the output signal curve, may occur until a stable final value is reached

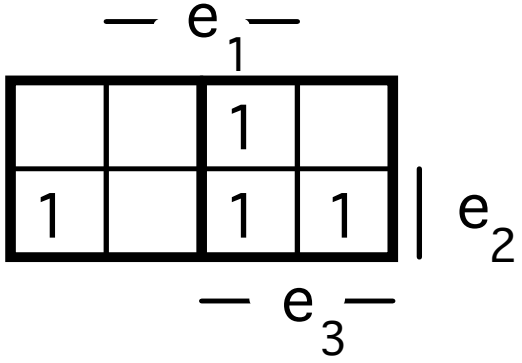
➔ Hazard (Hasardfehler)

© 2017 Burkhard Stiller

Example

Function: $a = \overline{e}_1 e_2 \vee e_1 e_3$





© 2017 Burkhard Stiller

Assumption: Change of Input

- The two following transitions (changes of input values) are considered:

a) Inputs e_2 and e_3 are constant at 1 and the input e_1 changes from 0 to 1

b) The inputs e_2 and e_3 are constant at 1 and the input e_1 changes from 1 to 0

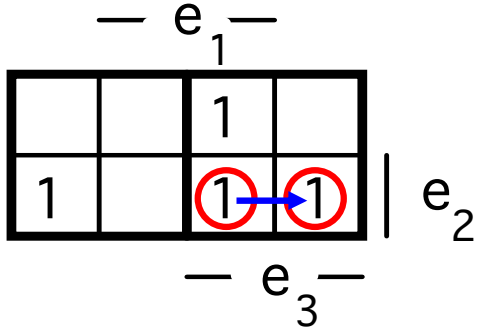
© 2017 Burkhard Stiller

Functional Evaluation

Function‘s values at the 2 transitions:

$(e_3,e_2,e_1) = (1,1,0) \quad \Rightarrow \quad a = 1$

$(e_3,e_2,e_1) = (1,1,1) \quad \Rightarrow \quad a = 1$

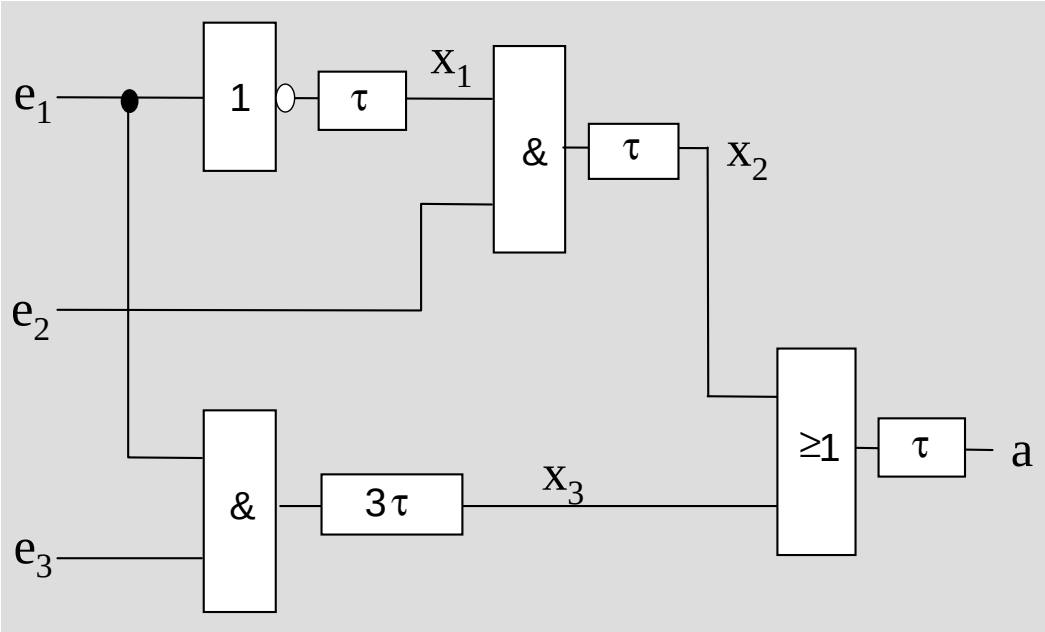


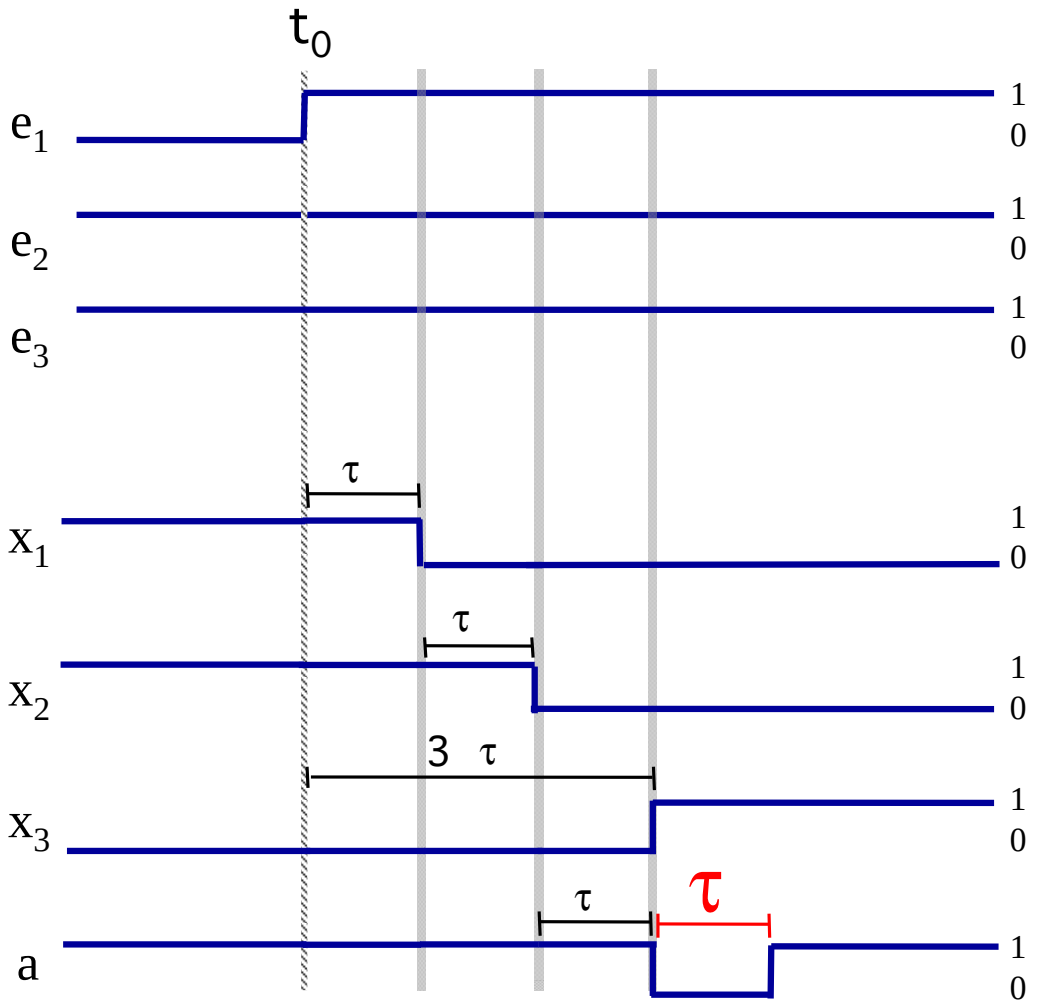
$\Rightarrow$ Correct behavior in these cases

For those transitions the value for *a* does not have to change under any circumstances, it needs to remain **constant 1** at all times

However, this behavior may not be guaranteed!

Behavior Investigation Based on Delay Model

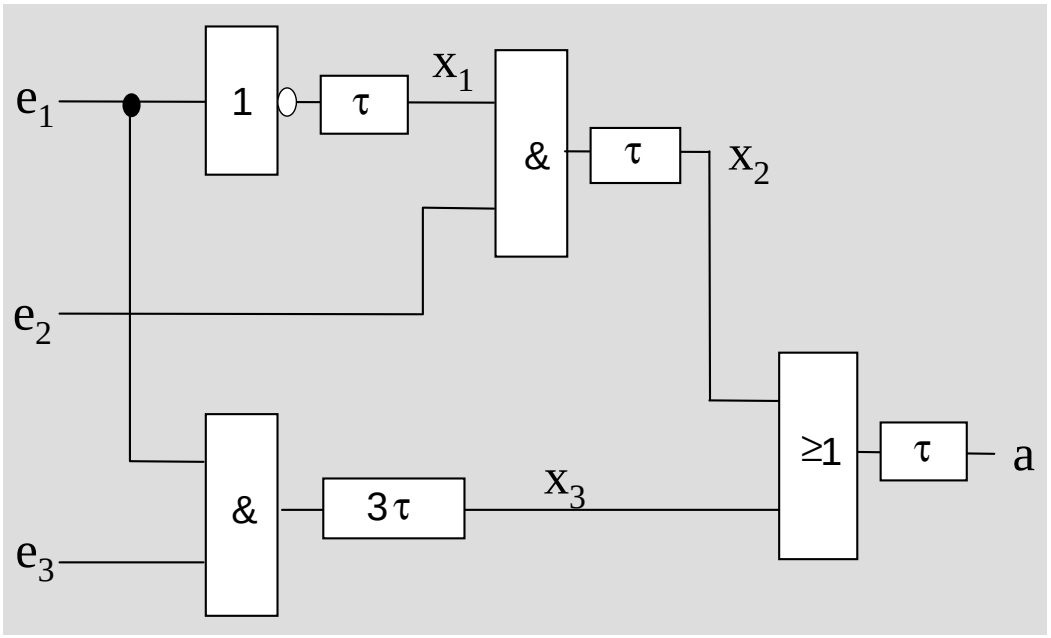


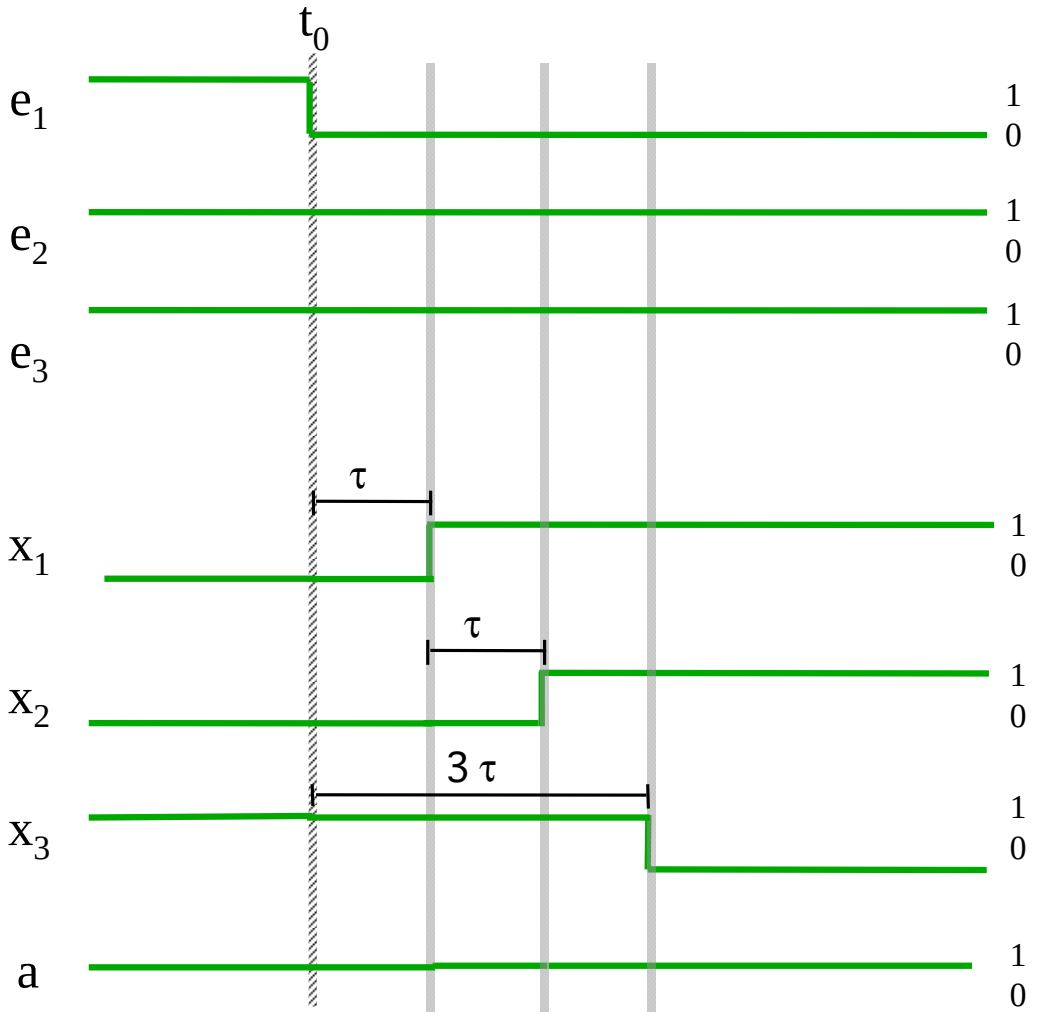


Hazard Error!

© 2017 Burkhard Stiller

Behavior Investigation Based on Delay Model





No Hazard Error

© 2017 Burkhard Stiller

ifi

Result and Interpretation

- For a change of input e_1 from 0 to 1 the output signal curve does not remain constantly at the correct function‘s value
  ➔ Hazard Error

- For a change of input e_1 from 1 to 0 the output signal curve is correct
  ➔ No worries!

- Interpretation of this example’s findings:
  - Functions may inherently carry a hazard (which is instantiated in terms of the hazard error), which may only become visible if dedicated input changes are applied
  - Thus, incorrect behaviors may occur such that output values may change, although ideally that should not happen
  - Does a suitable mechanism exist (a) to find out, if a hazard error may occur and if so (b) to solve this problem in a realistic manner?

© 2017 Burkhard Stiller

Terminology Definitions

- A change of input(s) (Eingabewechsel) is defined as the change of one or multiple input configurations at a defined point in time
  - In case of a required change of multiple input configurations, these have to happen at the same point in time
- A transition (Übergang) determines the behavior of the combinatorial circuit, which is driven by the change of input(s).
  - A transitions starts at the time the input configuration changes and ends at the time the circuit has reached its static state (Ruhezustand)
- A hazard error is defined as multiple changes of an output configuration within a single transition
  - Cf. example in blue above
- A hazard is defined as the logical-structural prerequisite for a hazard error, while no detailed delay values for gates implementing the function are considered

© 2017 Burkhard Stiller

Hazard-afflicted Transitions

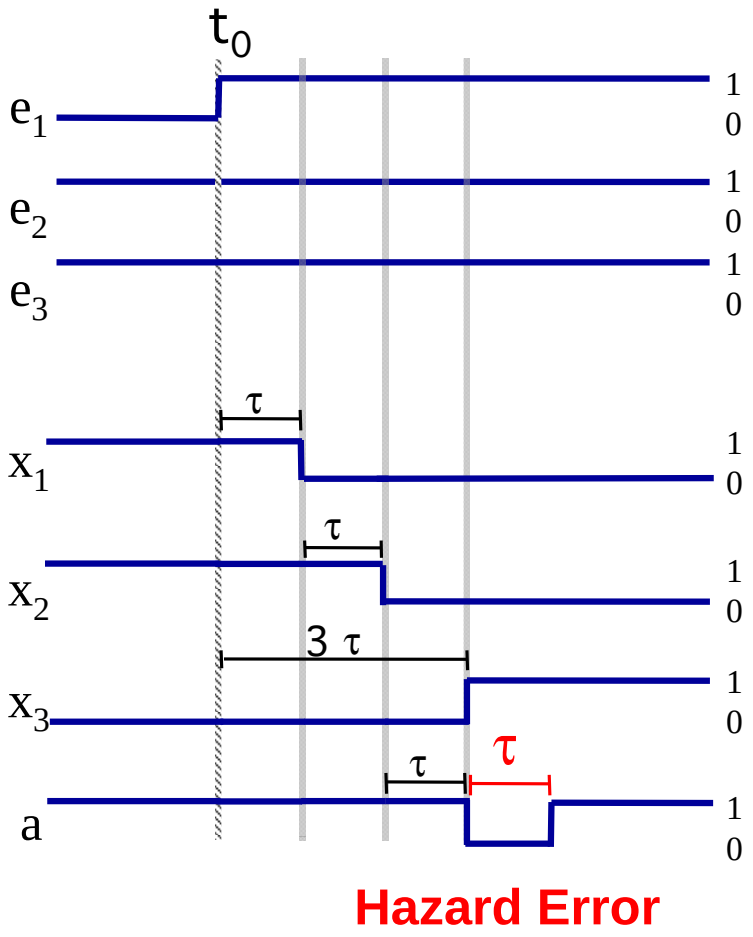
- Each hazard is a characteristics of one transition
- A hazard-afflicted transition is determined by
  - Logical function this combinatorial circuit implements
  - The structure of the same in terms of #gates and order of those (no delays)
- Given a combinatorial circuit and a transition, which ends up to be hazard-afflicted, the circuit contains a hazard

  Hazard error $\rightarrow$ Hazard
- The inverse is not correct: The existence of a hazard-afflicted transition does not lead necessarily to a hazard error

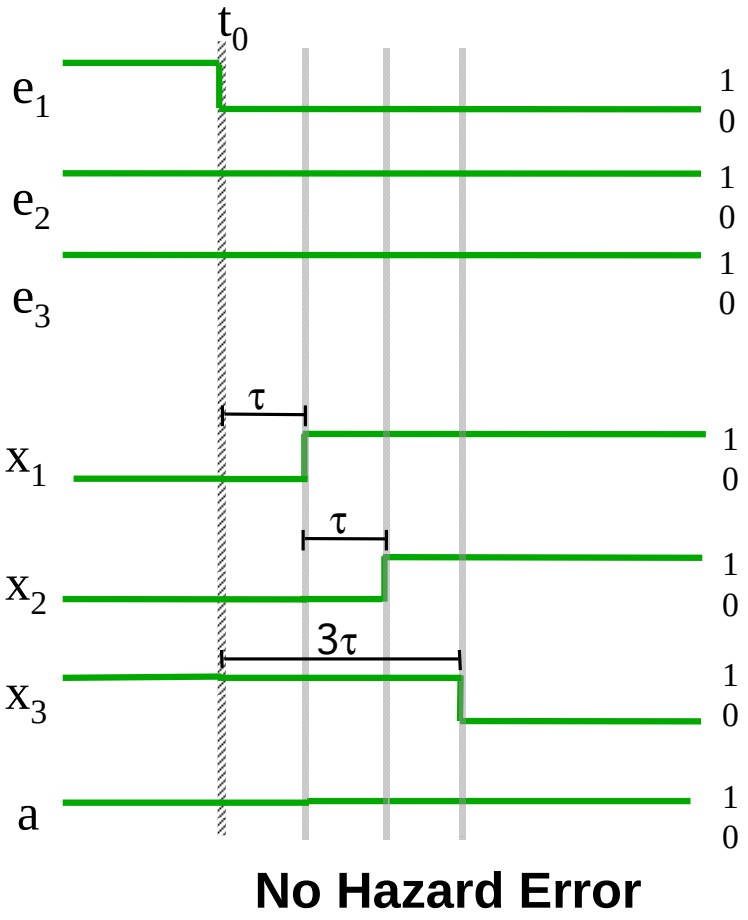
  Hazard $\wedge$ unfavorable delay values $\rightarrow$ Hazard error

© 2017 Burkhard Stiller

Example



Hazard Error



No Hazard Error

Transition (e_3,e_2,e_1) : (1,1,0) → (1,1,1) ist hazard-afflicted, since the potential to show a hazard error exists!

© 2017 Burkhard Stiller

ifi

Functional and Structural Hazards

- A functional hazard is a hazard, which originates from the function itself
  - Functional hazards always appear in any alternative combinatorial circuit implementing this function. It cannot be circumvented as such
  - Given an implementation as a combinatorial circuit, a suitable selection of delay values can prevent the functional hazard error to occur, but not the hazard itself

- A structural hazard is a hazard, which originates from the structure of the implemented circuit
  - Structural hazards can always be circumvented by developing an alternative combinatiorial circuit with a different structure
  - Thus, the selection of a different minimal disjunctive/conjunctive normal form for that function solves the problem in full

© 2017 Burkhard Stiller

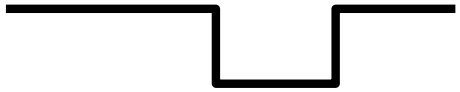
Static 0-Hazard and 1-Hazard

- Compared to transitions, hazards are either dynamic or static
- A hazard within a static 0-transition is termed static 0-hazard
  - Examples





- A hazard within a static 1-transition is termed static 1-hazard
  - Examples





The transition (1,1,0) $\rightarrow$ (1,1,1) within the previous example contains a static 1-hazard

Dynamic 01-Hazard and 10-Hazard

- A hazard within a dynamic 01-transition is termed dynamic 01-hazard
  - Examples

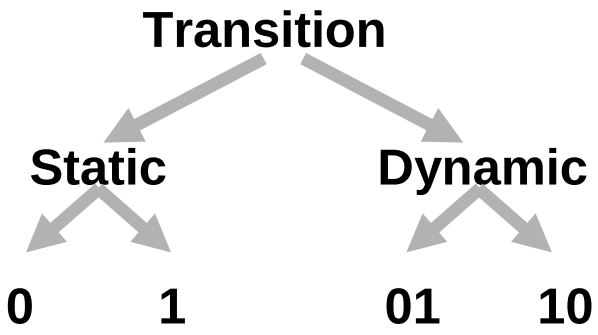


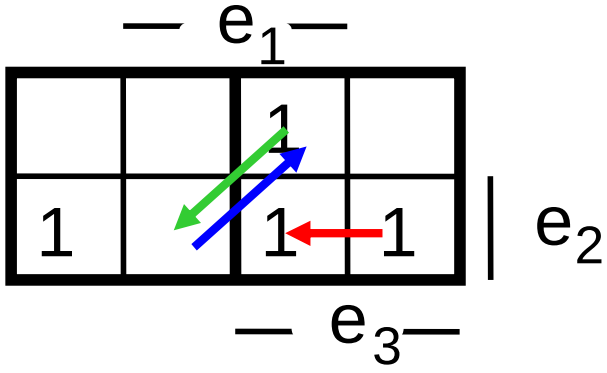
- A hazard within a dynamic 10-transition is termed dynamic 10-hazard
  - Examples



© 2017 Burkhard Stiller

Classifications of Hazards and Examples





Static 1-transition:

Transition (e_3,e_2,e_1): $(1,1,0) \rightarrow (1,1,1)$

Dynamic 01-transition:

Transition (e_3,e_2,e_1): $(0,1,1) \rightarrow (1,0,1)$

Dynamic 10-transition

Transition in inverse direction: $(1,0,1) \rightarrow (0,1,1)$

© 2017 Burkhard Stiller

— Computer Engineering —
Module 4: Sequential Circuits



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2017 Burkhard Stiller

ifi

Module 4: Sequential Circuits

- Basics
  - Finite State Machines (FSM)
- Timing and Flipflops
  - (A)synchronous Circuits
  - Level-triggered, Edge-triggered Flipflop Types
- Design of Synchronous Sequential Circuits
- Dedicated Sequential Circuits
  - Registers, Counters, Shift Registers
  - Half-adder, Full-adder
  - Floating Point Unit
- Arithmetic Logic Unit (ALU)

© 2017 Burkhard Stiller

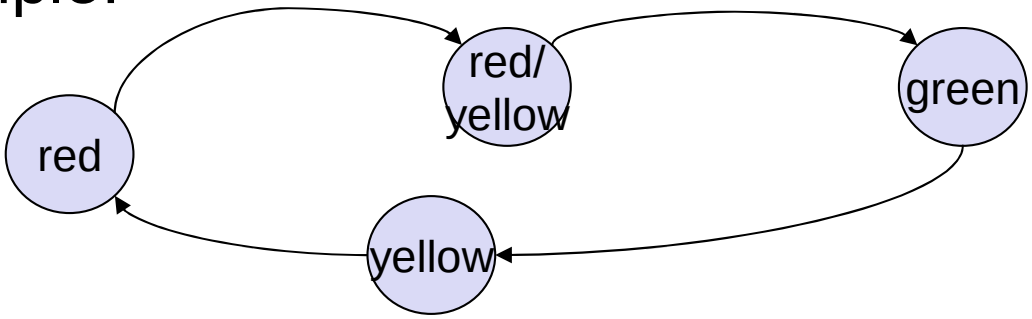
Sequential Circuits

- Reminder: Combinatorial Circuits
  - The output configuration depends only on values of input variables at the same point in time
  - The propagation delay is ignored for this distinction
- Sequential Circuits
  - The output configuration depend also on values of input variables of past points in time!

© 2017 Burkhard Stiller

Sequential Circuits

- All past values' dependencies are summed up into a state
- The sequential circuit generates an output depending on current input configurations and its state
  - Those values influence the following state of the sequential circuit
- Sequential circuits can be interpreted as implementations of deterministic Finite State Machines (Automat).
- Example:



States of a traffic light

© 2017 Burkhard Stiller

Introduction to Finite State Machines

- A 6-tuple M = ($E, A, Z, \delta, \omega, z_0$) is called Finite State Machine (FSM), if E, A, and Z are non-empty sets
  - E defines the set of input configurations (Eingangsbelegungen) e,
  - A defines the set of output configurations (Ausgangsbelegungen) a, and
  - Z defines the set of states (Zustände) z.

- Transition Function (Überführungsfunktion) $\delta: Z \times E \rightarrow Z$
  - δ defines a function based on the set of $Z \times E$, where all values remain within Z

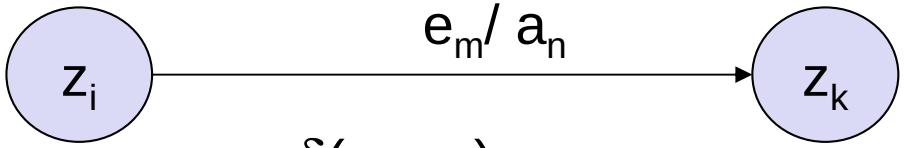
- Output Function (Ausgabefunktion) $\omega: Z \times E \rightarrow A$
  - ω defines a function based on the set $Z \times E$, where all values remain within A

- Initial State (Grundzustand) z_0

© 2017 Burkhard Stiller

Example

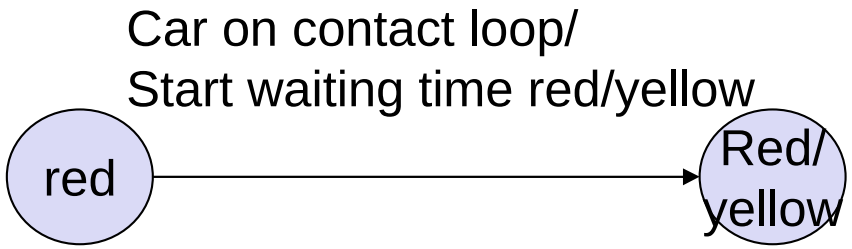
- In general:



$\delta(z_i, e_m) = z_k$
$\omega(z_i, e_m) = a_n$

$e_m \in E$
$a_n \in A$
$z_i, z_k \in Z$

- Traffic light (incomplete):



E = {car on contact loop, pedestrian button pushed, ...}

A = {Start waiting time red/yellow, Start buzzer, ...}

Z = {red, red/yellow, green, yellow}

δ(red, car on contact loop) = red/yellow
ω(red, car on contact loop) = start waiting time red/yellow

© 2017 Burkhard Stiller

Mains of Finite State Machines (FSM)

- The set of states *Z* allows for the storage of “knowledge” (state memory, Zustandsspeicher) originating from input assignments of the past

- The current output assignment is determined by the function ω, such that the new state of the function δ is calculated from current input assignments ***and*** past state

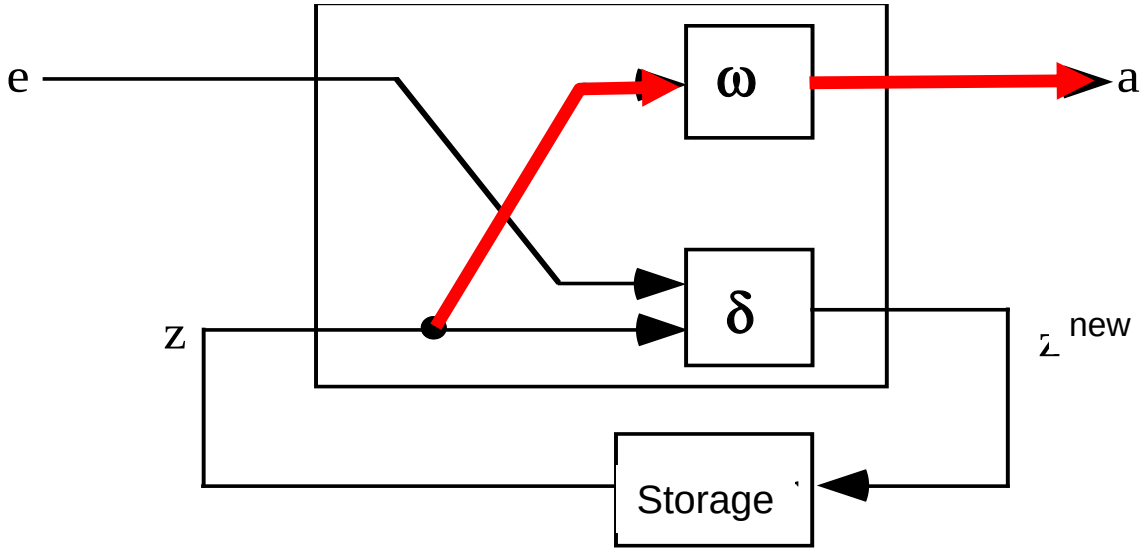
© 2017 Burkhard Stiller

Mealy and Moore FSMs (1)

- In case of the output value being dependent only on the current state, a Moore FSM or a Moore Sequential Circuit exists

$a = \omega(z)$ $\quad z^{new} = \delta(z,e)$

$a \in A;$ $\quad e \in E;$ $\quad z, z^{new} \in Z$

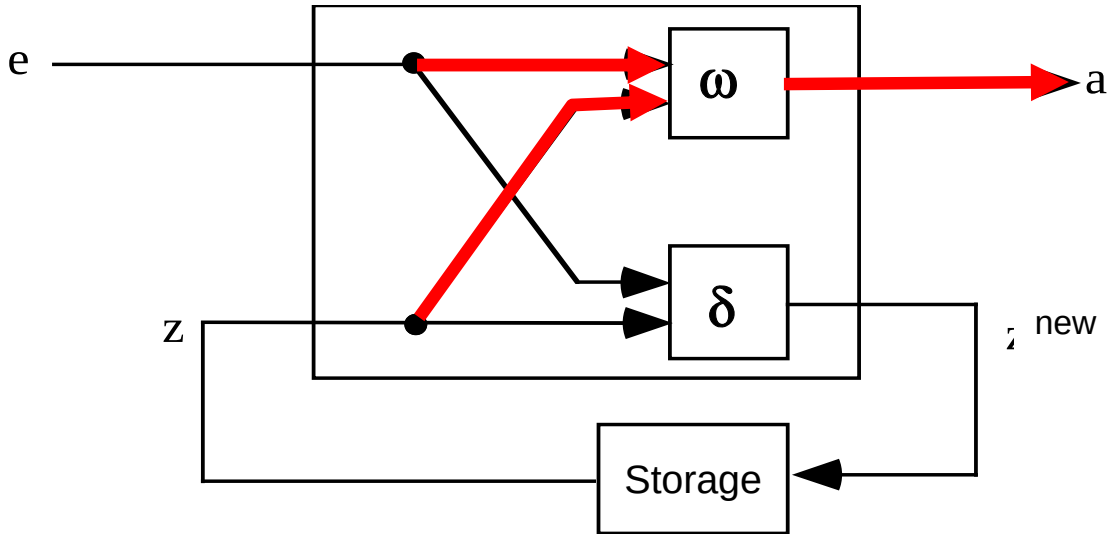


Mealy and Moore FSMs (2)

- In case of the output value being dependent on the current state and the input configuration, a Mealy FSM or a Mealy Sequential Circuit exists

$a = \omega(z,e)$ $\quad z^{new} = \delta(z,e)$

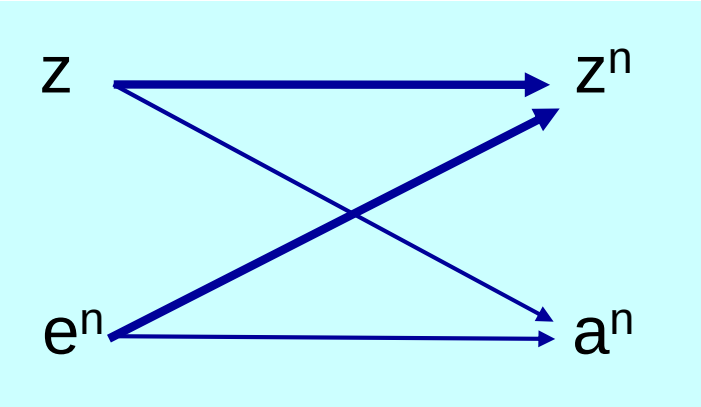
$e \in E;$ $\quad a \in A;$ $\quad z, z^{new} \in Z$



© 2017 Burkhard Stiller

Behavioral Differences of these FSMs (1)

- Mealy FSM
  - Output values can change immediately whenever input variables change
- Mealy FSM Type 1:
  - First based on the new input assignments output assignments are calculated
  - Second, a change into the next state is performed
- Application:
  Synchronous sequential circuits

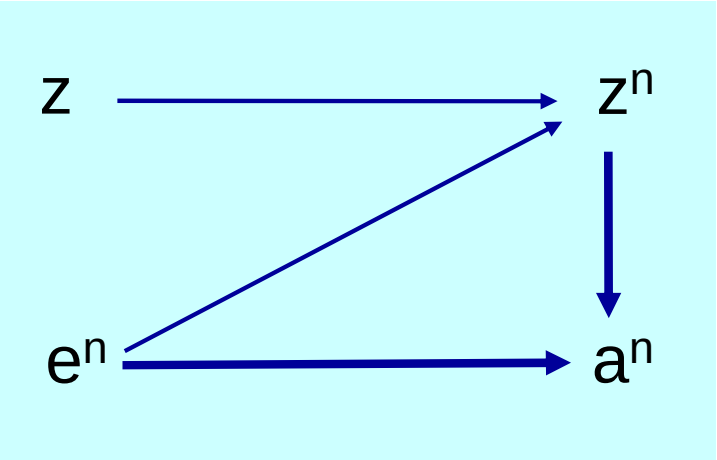


© 2017 Burkhard Stiller

Behavioral Differences of these FSMs (2)

- Mealy FSM Type 2:
  - First the change into the next state is performed based on new input assignments
  - Second, output assignments are calculated, while input assignments remain unchanged and are still valid.

- Application:
  Asynchronous sequential circuits

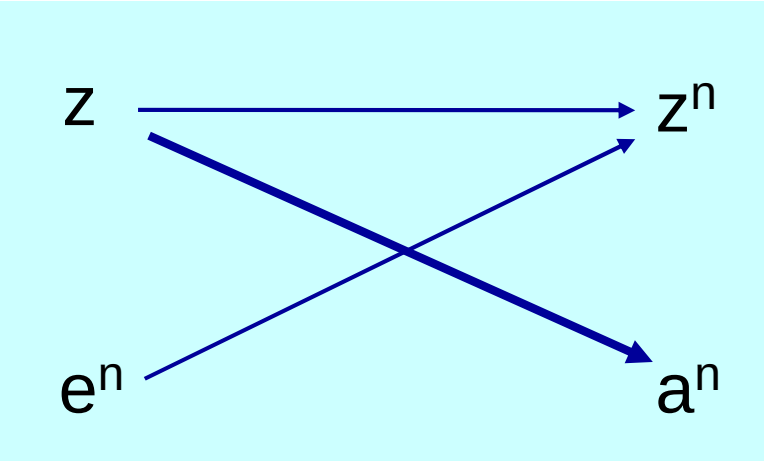


© 2017 Burkhard Stiller

Behavioral Differences of these FSMs (3)

- Moore FSM:
  - The output assignment is independent of the input assignment
  - The output assignment can only change after a state change

- Application:
  Synchronous and
  asynchronous sequential circuits



© 2017 Burkhard Stiller

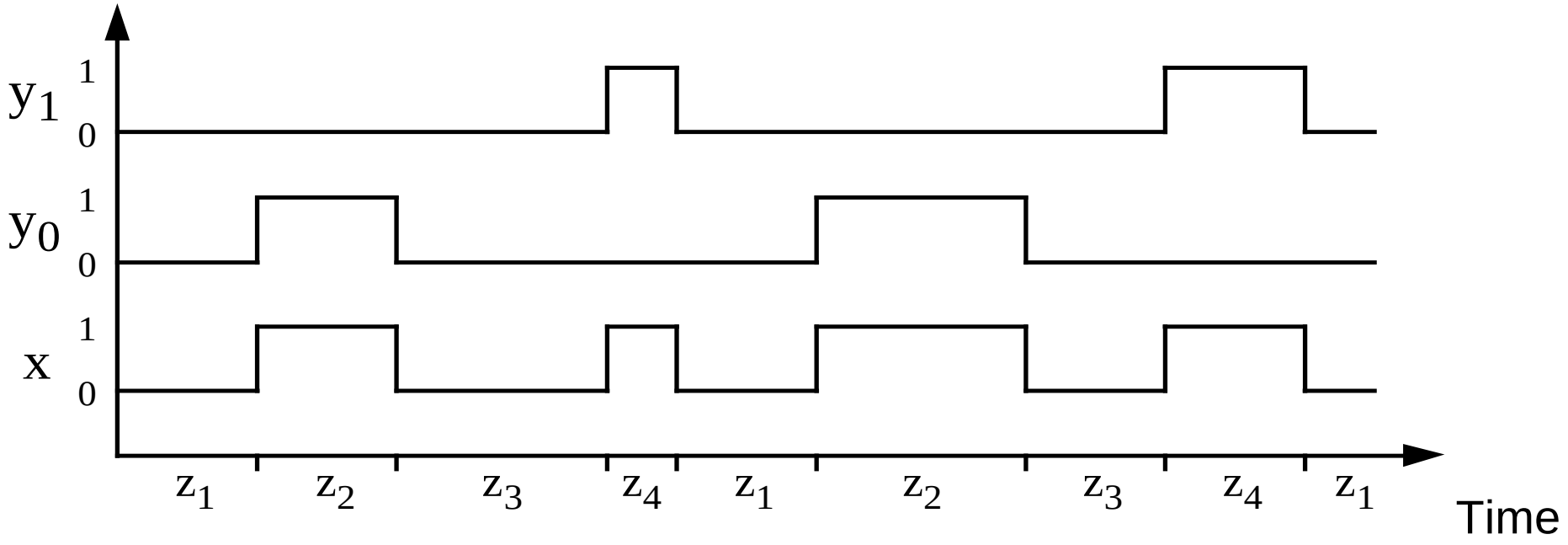
Representation of FSMs

- Initially, at construction times of an FSM, the task description is typically available only in an informal and global outline, *e.g.*:
  - Specification:
    oral description what the planned circuit should handle
  - Other plans:
    Flow charts, technology description, or the like to complete the oral description
- In support of a systematic and possibly computer-based design, transitions to a formalized description are essential, to sufficiently specify the target behavior uniquely
  - 4 different possibilities exist in terms of a formalized description (cf. below)
- Example:
  - A sequential circuit shall handle an incoming signal sequence of an input x in the following manner:
    Input signals x appear in an alternate manner on two outputs y_0 and y_1
- What is the formalized description of this problem?

© 2017 Burkhard Stiller

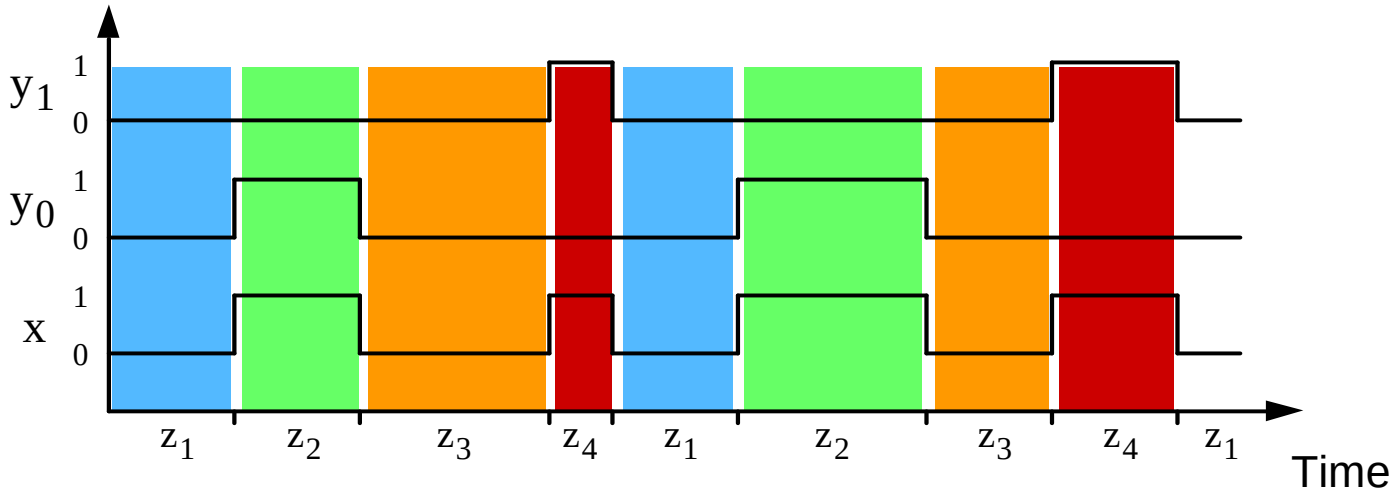
1. Time Chart (1)

- Time charts are used to visualize the problem (human readable)
- An example sequence of input assignments reads as follows:



© 2017 Burkhard Stiller

1. Time Chart (2)



z_1 = Output of $y_0y_1 = 00$ and wait for x = 1 to output at exit y_0

z_2 = Output of $y_0y_1 = 10$ and wait for x = 0

z_3 = Output of $y_0y_1 = 00$ and wait for x = 1 to output at exit y_1

z_4 = Output of $y_0y_1 = 01$ and wait for x =0

© 2017 Burkhard Stiller

2. Sequence Table

If the states are known, a transition function $z_{k+1} = \delta(z_k,x)$ and the output function $(y_0,y_1) = \omega(z_k,x)$ can be represented by a sequence table

z_k	x	z_{k+1}	y_0 y_1
z_1	0	z_1	0 0
z_1	1	z_2	1 0
z_2	1	z_2	1 0
z_2	0	z_3	0 0
z_3	0	z_3	0 0
z_3	1	z_4	0 1
z_4	1	z_4	0 1
z_4	0	z_1	0 0

Initial State z_k | Input Assignment | Next State z_{k+1} | Output Assignment

© 2017 Burkhard Stiller

3. FSM Table

- FSM tables represent identical information to sequence tables:
  - Reading in vertical direction: states
  - Reading in horizontal direction: input assignments
  - Reading the matrix: subsequent states

z_k	z_{k+1}		$y_0\, y_1$
	x=0	x=1	
z_1	z_1	z_2	0 0
z_2	z_3	z_2	1 0
z_3	z_3	z_4	0 0
z_4	z_1	z_4	0 1

Moore FSM

z_k	$z_{k+1}/\, y_0\, y_1$	
	x=0	x=1
z_1	$z_1/00$	$z_2/10$
z_2	$z_3/00$	$z_2/10$
z_3	$z_3/00$	$z_4/01$
z_4	$z_1/00$	$z_4/01$

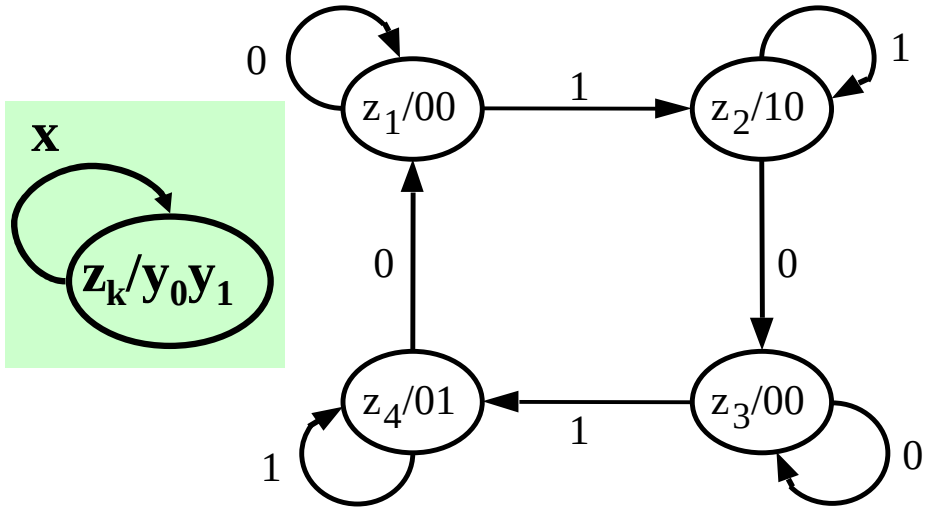
Mealy FSM

© 2017 Burkhard Stiller

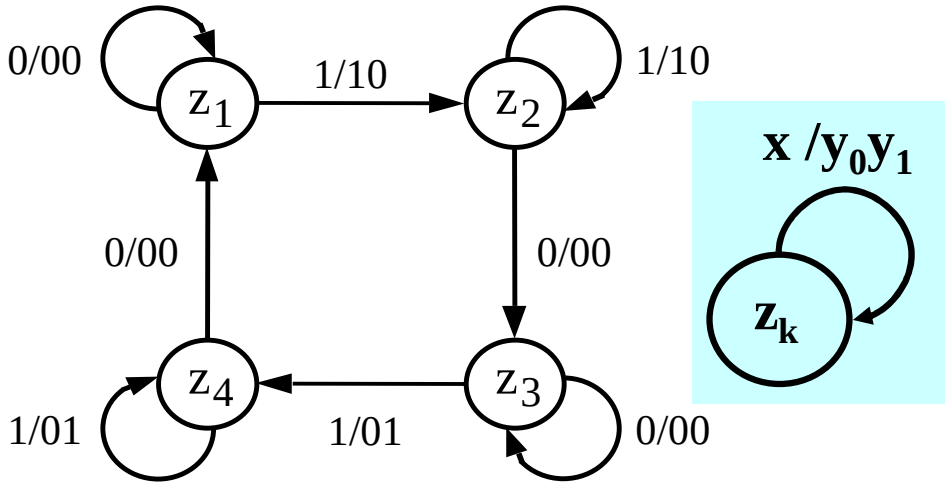
4. State Graph

- The transition function and output function are shown in a directed graph DG = (Z,K), where *Z* equals the set of states *z* and *K* equals the set of transitions *k* between different states

- Input assignments are tagged to the edges of state transitions



Moore FSM



Mealy FSM

© 2017 Burkhard Stiller

ifi

Implementation of FSMs

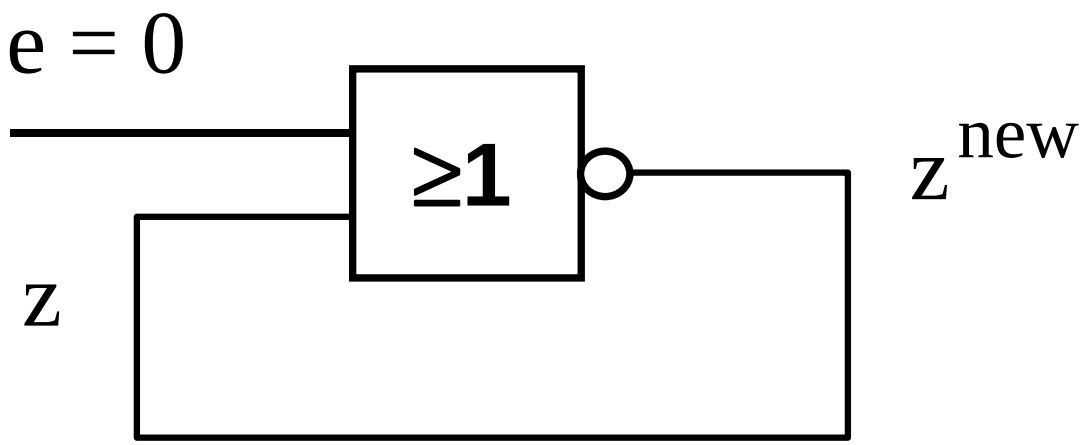
- A state memory is required to store information of the past
- Simplest option for such a state storage:

  Feedback Loop

- A feedback loop allows for the provision of information as input configurations, which would not be available at a later state as input variables anymore

© 2017 Burkhard Stiller

Example: Feedback'ed NOR Gate (1)



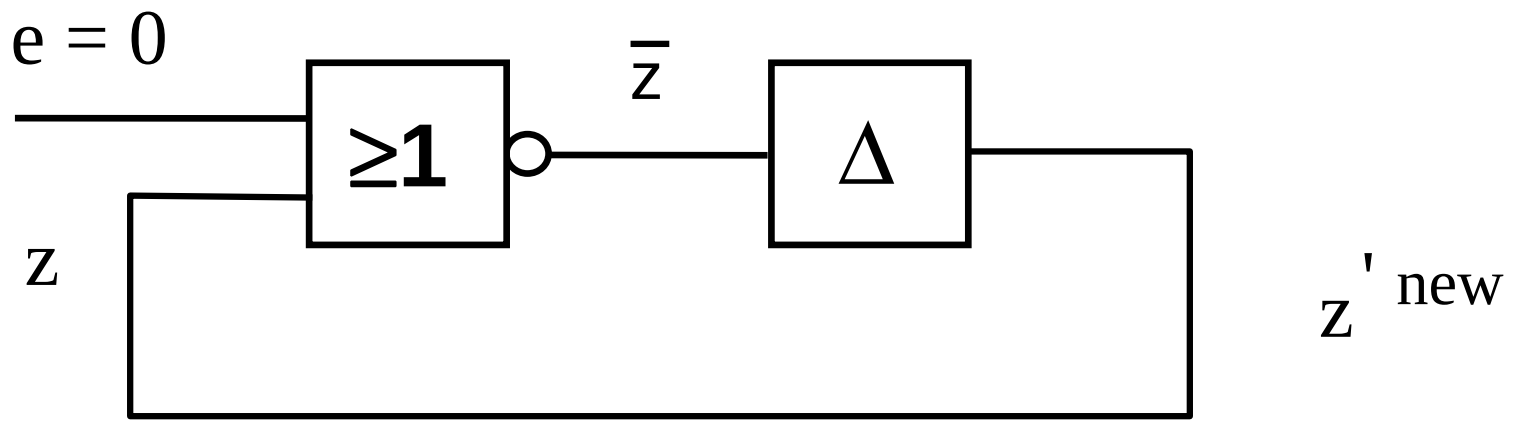
Assuming an ideal gate, this circuit is impossible, because the following equations would have to hold concurrently:

$z^{new} = \overline{z}$ and

$z = z^{new}$

Example: Feedback'ed NOR Gate (2)

However, in reality each circuit shows a delay larger than 0 (cf. delay model in M3):



Assuming this delay, the following holds:

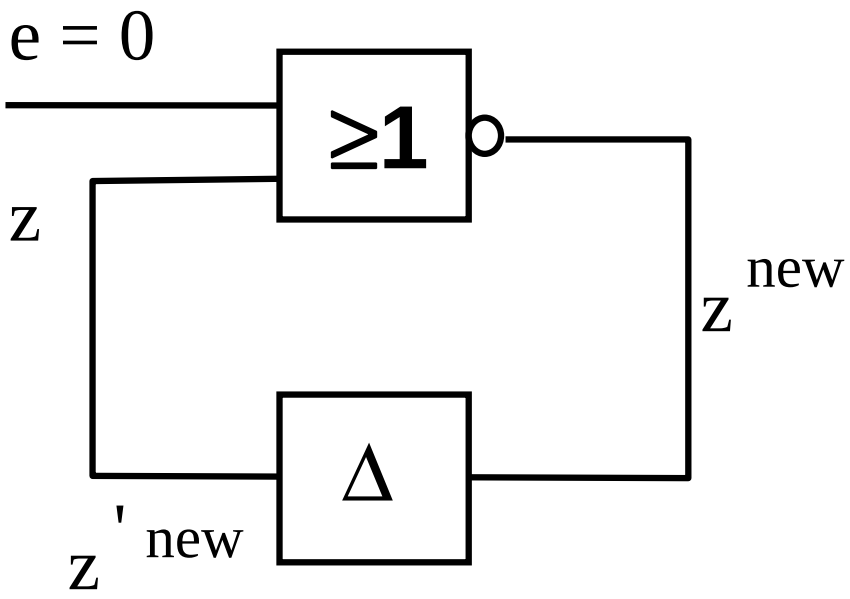
$$z'_{new}(t+\Delta) = \bar{z}(t)$$

$$z(t+\Delta) = z'_{new}(t+\Delta)$$

© 2017 Burkhard Stiller

Example: Feedback'ed NOR Gate (3)

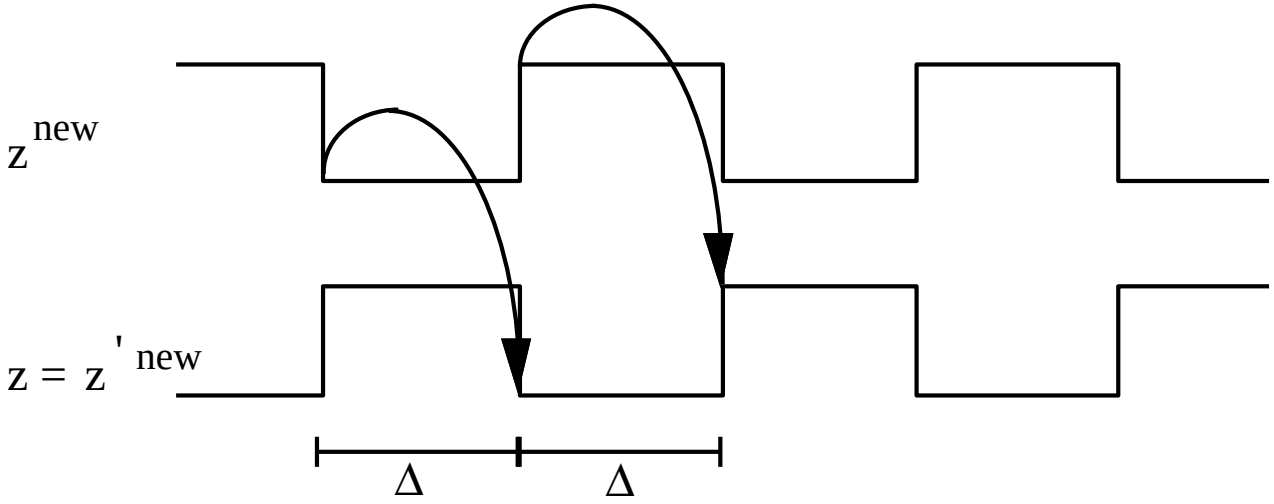
Upon the drawing of the identical circuit slightly different, the Δ delay gate is achieved, which equals a storage for 1 bit

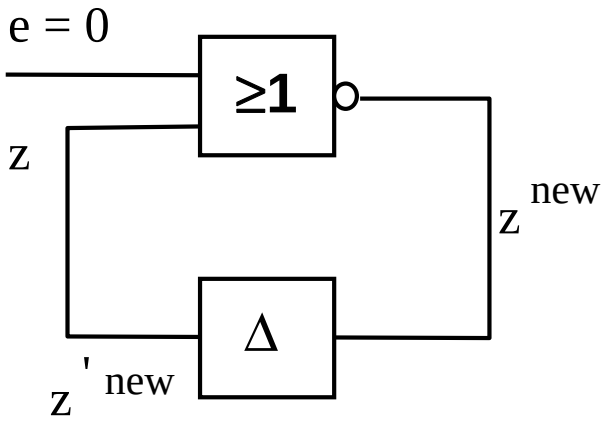


© 2017 Burkhard Stiller

Example: Feedback'ed NOR Gate (4)

The timing of this circuit shown in the time chart:

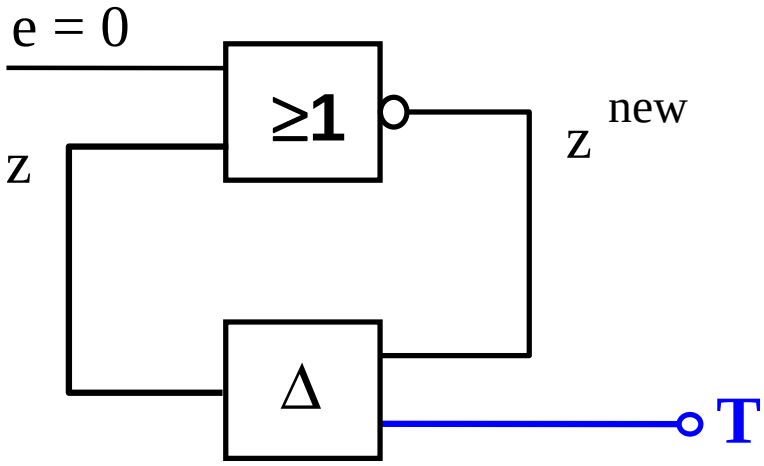




The behavior of this circuit depends strongly on the exact delays

Example: Feedback'ed NOR Gate (5)

- Thus, how can one define a behavior independent of delays?
- To build the memory in the way, that *z* stores its past value as long as *z* is explicitly triggered by a external signal *T* to be set to z^{new}.
- Example:



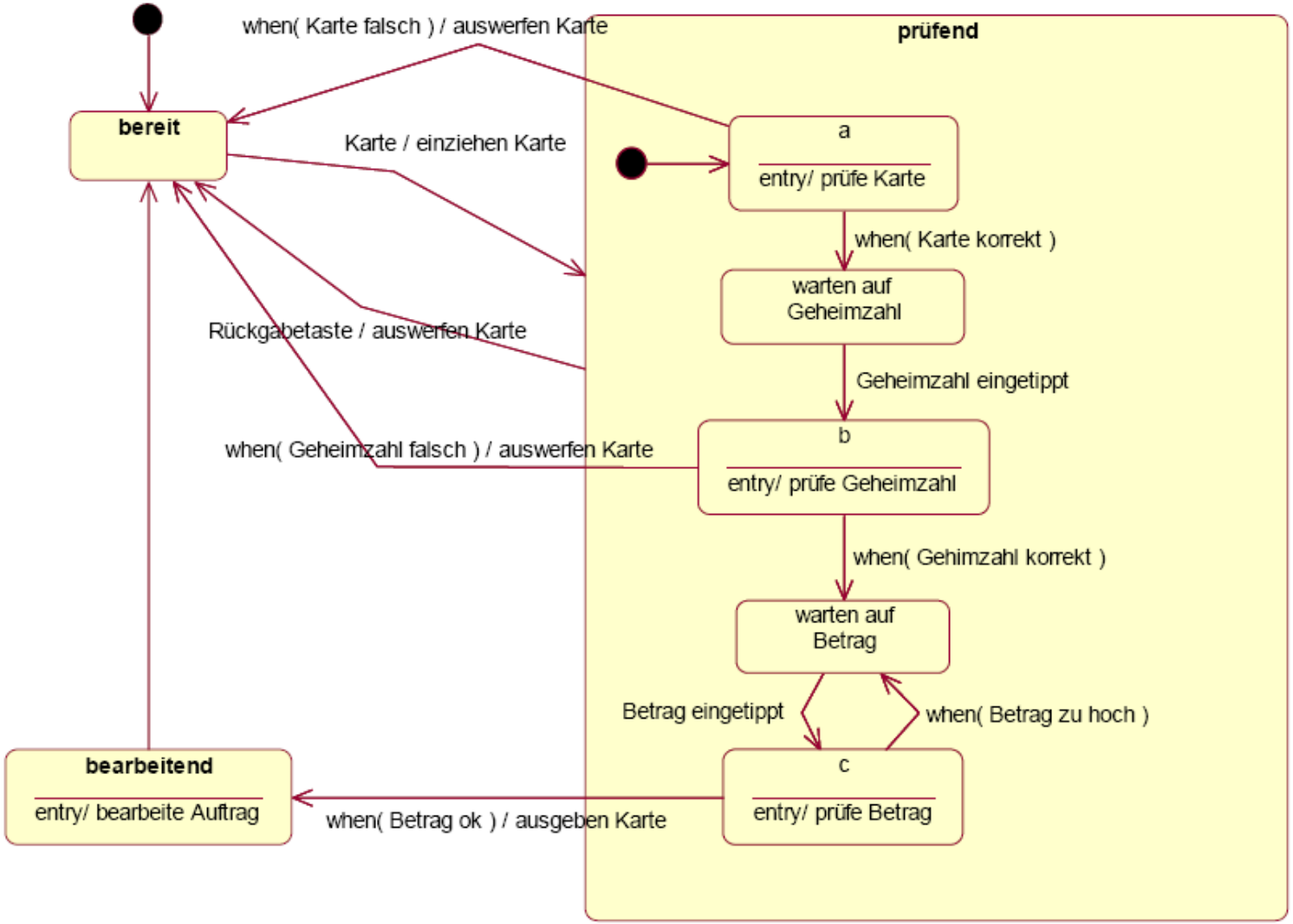
z stores its value until *T* changes from 0 to 1

z is changed to z^{new} and stores this value until the following 01-transition of *T*

© 2017 Burkhard Stiller

FSM Graph (ATM)

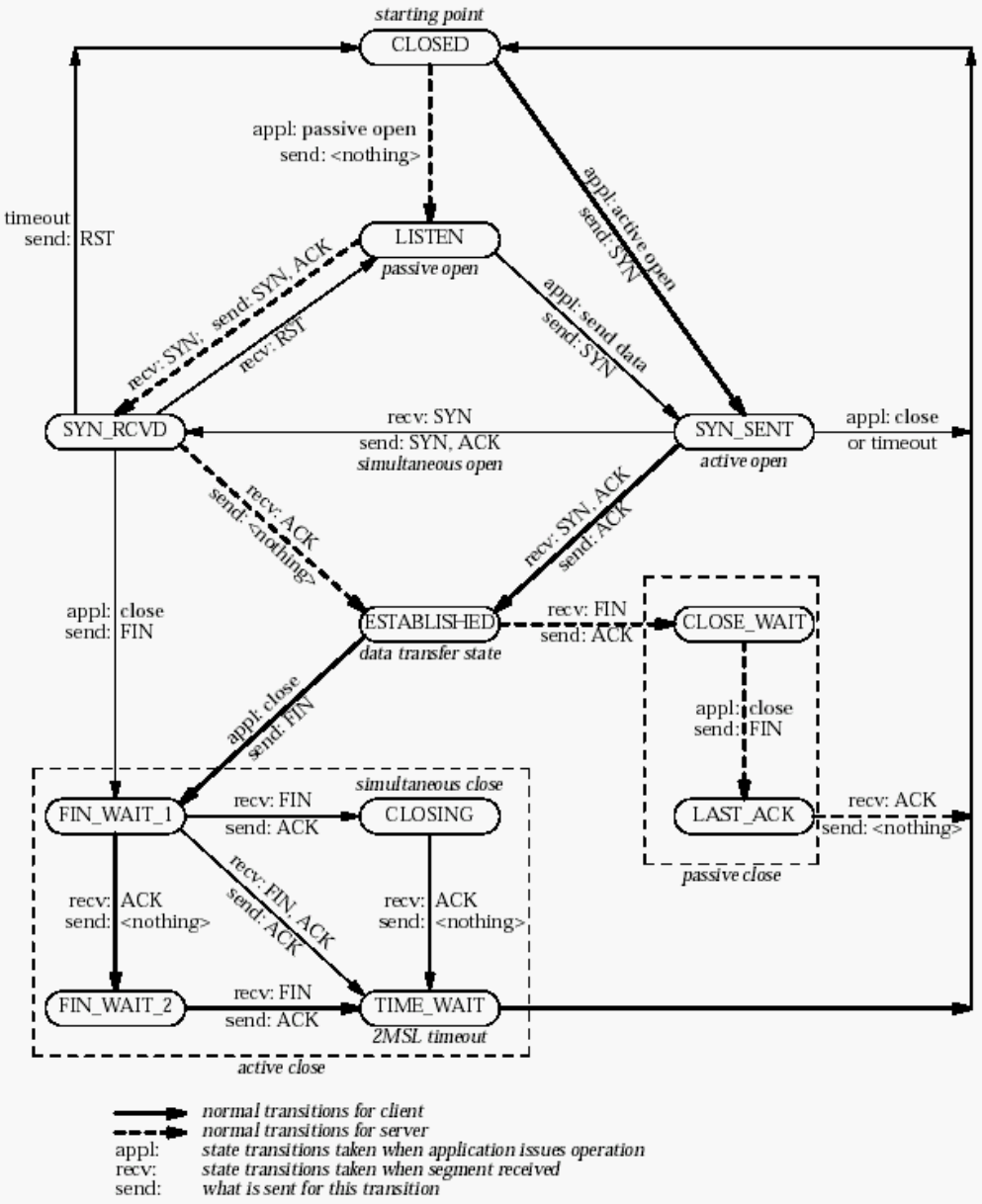
Zustandsdiagramm:
Bankautomat:



© 2017 Burkhard Stiller

FSM (TCP Protocol)

TCP: Transmission Control Protocol



© 2017 Burkhard Stiller

Module 4: Sequential Circuits

- Basics
  - Finite State Machines (FSM)
- Timing and Flipflops
  - (A)synchronous Circuits
  - Level-triggered, Edge-triggered Flipflop Types
- Design of Synchronous Sequential Circuits
- Dedicated Sequential Circuits
  - Registers, Counters, Shift Registers
  - Half-adder, Full-adder
  - Floating Point Unit
- Arithmetic Logic Unit (ALU)

© 2017 Burkhard Stiller

Timing Definitions

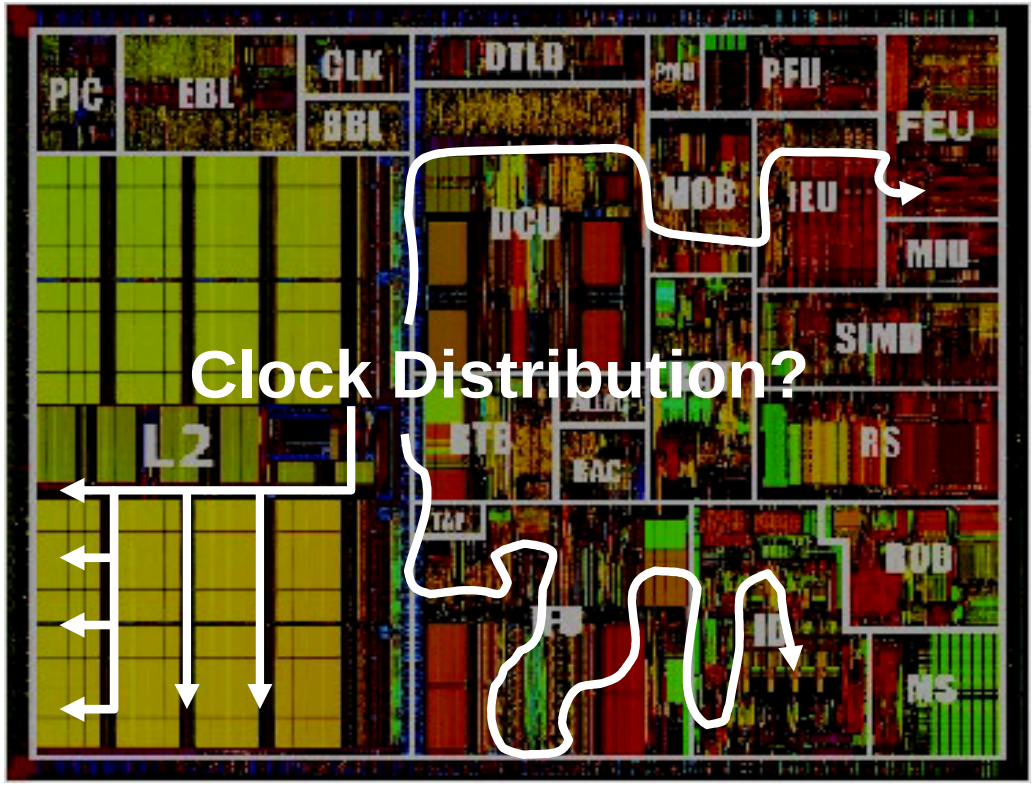
- If all state memory is controlled within a sequential circuit by one or more centrally synchronized signal(s), termed *T* (Clock/Takt), these circuits are termed synchronous sequential circuits
  - Else: asynchronous sequential circuits.
- Synchronous sequential circuits:
  - Middle and larger-sized sequential circuits are usually designed as synchronous sequential circuits
  - Advantage: Simpler to analyze and to design as asynchronous sequential circuits.
    - Reason: Synchronous sequential circuits are independent of (partially deployment-dependent) delays
- Asynchronous sequential circuits:
  - The design of asynchronous sequential circuits is important due to two reasons:
    - The storage elements used in synchronous sequential circuits are smaller than in case of asynchronous sequential circuits
    - Elements with fast growing performance require partly asynchronous sequential circuits design

© 2017 Burkhard Stiller

Reasons for Asynchronous Seq. Circuits

- If the delay of elements used becomes smaller than the signal propagation time on a circuit's backplane or on the chip itself (app. at 20-30 cm/ns) …

  ⇒ … T is no longer synchronous, since T reaches each element depending on its physical distance from the clock!



1 GHz clock = 1 ns clock duration!

© 2017 Burkhard Stiller

Practical Considerations

- In case the duration of *T* is always larger than the maximum delay experienced within a sequential circuit:

  $\Rightarrow$ Outputs of combinatorial circuits δ and ω did stabilize before they do influence z^{new}

  $\Rightarrow$ For the analysis and synthesis of a synchronous sequential circuit only the combinatorial circuits δ and ω must be taken into account

- The control of circuits using *T* can be performed either
  (a) level-triggered or
  (b) edge-triggered

Level-triggered Control

- The level-triggered (pegelgesteuert) memory is transparent in one half of the clock period (Takthälfte) and it stores the value in the other half
  - The input configuration influences only the state, e.g., for T = 1, in case of T = 0 the state is being stored
- Most simple implementation
  - Conjunctive combination of all input variables with T
  - Level-triggered state memory is also called Latch
- Drawback
  - The input configuration may change several times during the active (transparent) clock period

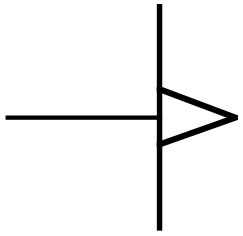
© 2017 Burkhard Stiller

Edge-triggered Control

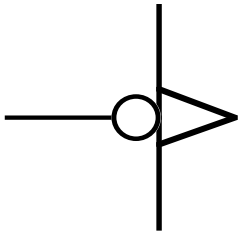
- For an edge-triggered (flankengesteuert) memory values are stored only during a positive (0 → 1) or a negative (1 → 0) clock edge (Taktflanke)
- Advantage
  - Inputs must only be valid for a short time period
  - The do not have to valid for the full half of the clock period's halfas for the level-triggered case

⇒ All "evaluation points" are determined exactly in time

- Circuit symbols for the edge-triggered clock inputs:



01-Clock Edge

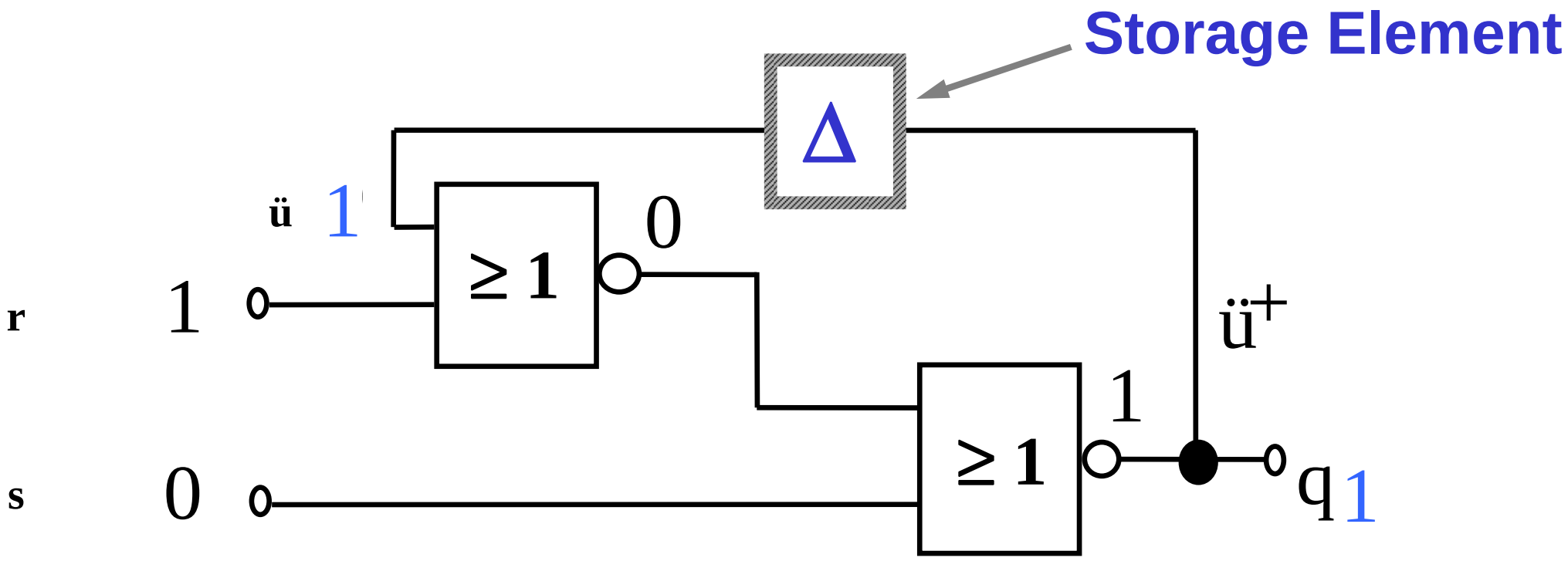


10-Clock Edge

Key Asynchronous Seq. Circuit Example

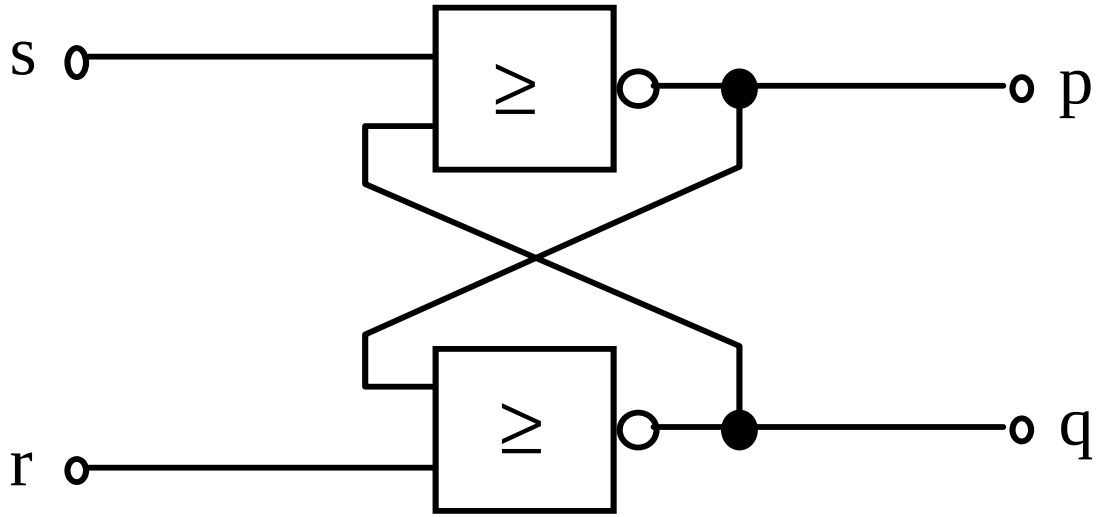
Transition function: $\ddot{u}^+ = \overline{r \vee \overline{\ddot{u} \vee s}} = \bar{r}\,(\ddot{u} \vee s)$

Output function: $q = \ddot{u}$



Asynchronous RS Flipflop

- This memory element defines a standard element
  - Termed asynchronous RS flipflop (FF) (bistabile Kippstufe).
  - Usually it is drawn in a slightly different manner than the memory element, though, semantically identical in major use



  - The additional exit *p* is defined typically as the complement (negation) of *q*, as long as *r* and *s* do not equal 1 at the same time, thus; $p = \overline{q}$
  - Only for impermissible input (*r, s*) = (1, 1), thus not designed, $p = q = 0$ holds

© 2017 Burkhard Stiller

Asynchronous Sequential Circuits

- Asynchronous sequential circuits work without a central clock
  - They react directly on every change of input and state variables
  - They are very sensitive (störungsempfindlich)
    - "Races" of state variables may occur:
      - They cause wrong state transitions
      - Solution: race-free state coding
- Hazard error in the transition of sequential circuits:
  - Asynchronous sequential circuits react on transitions sensitively
  - Hazard errors can also cause incorrect state transitions and oscillations
- Solution:
  - Design of combinatorial circuits with less hazards for transition functions and output functions
  - In order to reduce the risc for hazards asynchronous sequential circuits work within the so called normal fundamental mode
  - In these cases only one input variable is allowed to change at the same time
  - The change of input configurations can only happen if all internal changes died out

© 2017 Burkhard Stiller

Flipflops Used as State Memory

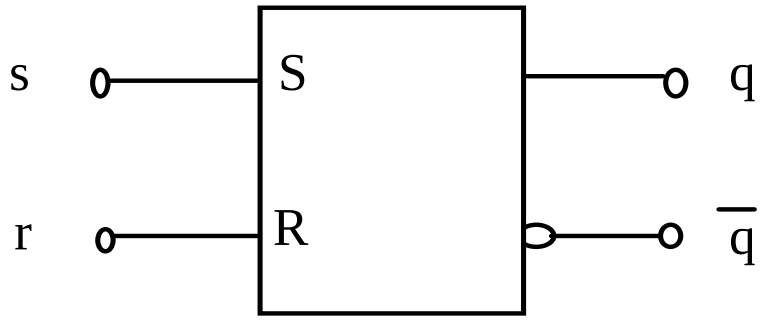
- Problems known about asynchronous sequential circuits do not appear in synchronous sequential circuits
- Due to the fact that all state memory in synchronous sequential circuits is controlled by a central clock, all transitions and state variable races can stabilize before the new state is reached
- Synchronous sequential circuits always require a clock-controlled state memory
  - Therefore, a flipflop is used
  - Different flipflop types exist

© 2017 Burkhard Stiller

RS Flipflop

- Behavior
  - Input *s* sets the storage ($s = 1 \Rightarrow$ output $q = 1$)
  - Input *r* resets the storage ($r = 1) \Rightarrow q = 0$)
  - Store: *r* and *s* both are set to 0 $\Rightarrow$ *q* stores last value available
  - The additional exit *p* is defined typically as the complement (negation) of *q*, as long as *r* and *s* do not equal 1 at the same time, thus; $p = \overline{q}$
  - Impermissible: input (*r, s*) = (1, 1), thus not designed, $p = q = 0$ holds
  - State variables *q* und its negation $\overline{q} = p$ (cf. earlier slide) are available as output

- Symbol of the asynchronous RS flipflop:

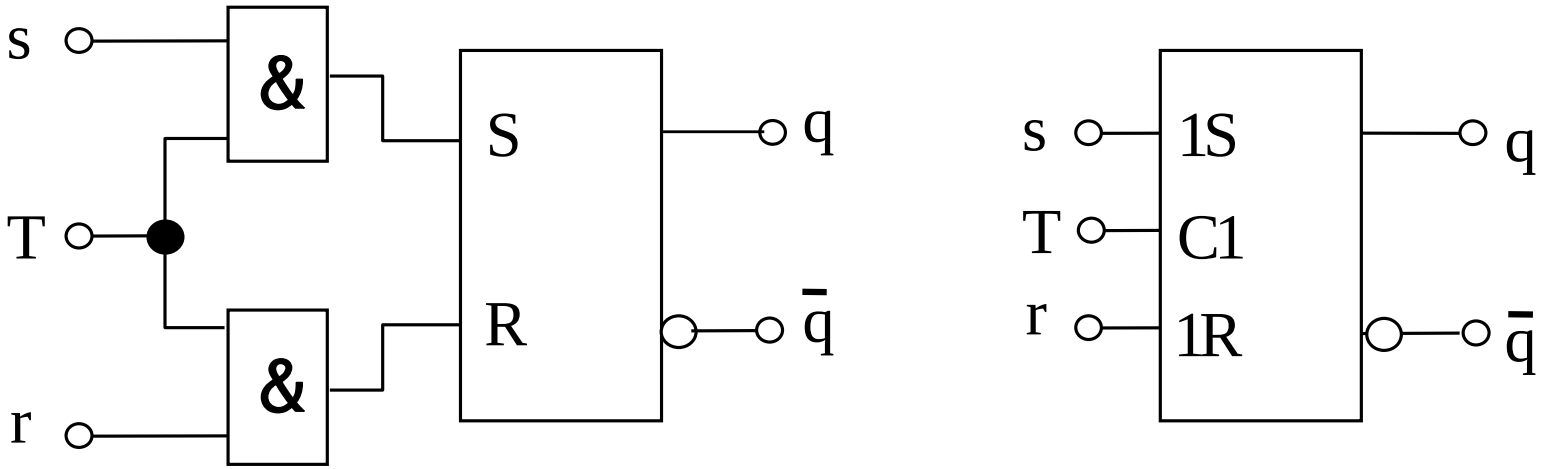


© 2017 Burkhard Stiller

From the RS-Flipflop to the Level-controlled RS Latch

- The use of RS flipflops in synchronous sequential circuits
  - A clock signal is needed to avoid the change of state variables within the inactive clock phase
  - Easy to achieve by connecting the two inputs to the clock by an AND

- Circuit and symbol of the level-controlled RS Latch:

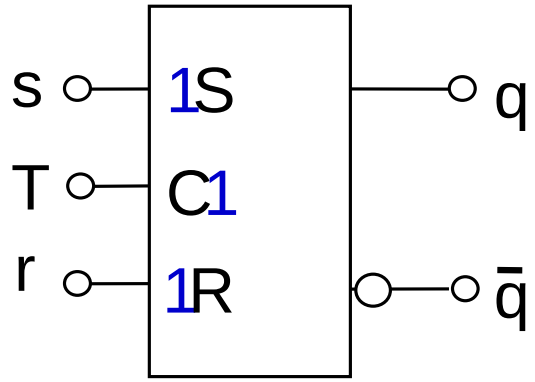


© 2017 Burkhard Stiller

Notation

- A number 1 combined with inputs (1S, 1R) determine that all of those depend on the same clock, likewise marked with 1 and termed clock C1

- If an input causes such a dependency, the number succeeds the input variable, else the number precedes the variable



© 2017 Burkhard Stiller

Control Table (RS Flipflop)

- Upon designing a synchronous sequential circuit the state and the the next state are known.
  - The design is in search of values for the control variables of the flipflop
  - The control table helps as it represents the state transition of a flipflop under different input configurations
    - The control table can be determined from the function table of all output and transition functions
- Control table RS flipflop

q^t	q^{t+1}	r	s	Interpretation
0	0	-	0	hold
0	1	0	1	set (S)
1	0	1	0	reset (R)
1	1	0	-	set

Assumption:
Impermissible input values do not occur!

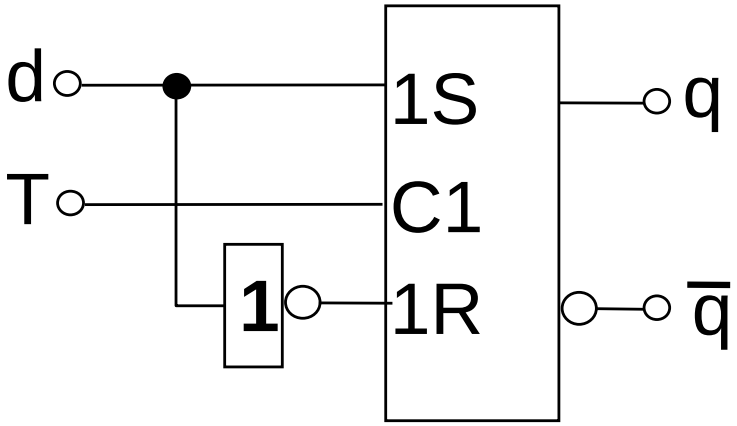
© 2017 Burkhard Stiller

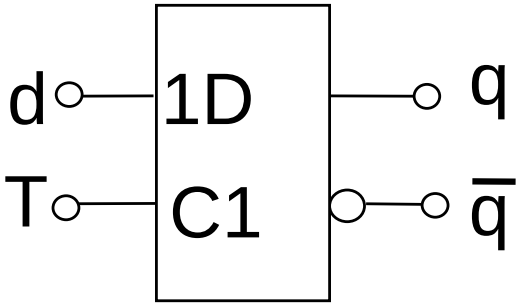
D Flipflop

- For any RS flipflop the condition ($r \barwedge s = 0$) needs to be observed always

- This condition holds true in all cases, when the input variable *d* is added positively (bejahend) to the S-input and negatively (negiert) to the R-input

- Circuit and symbol of the D latch:





© 2017 Burkhard Stiller

Behavior of the D Flipflop

- The input configuration is stored always for a clock period
- The input configuration is delayed by one clock period and accessible afterwards as the output configuration
- Thus the name D (delay) latch (verzögern)
- Function table

d	q^t	q^{t+1}
0	0	0
0	1	0
1	0	1
1	1	1



Control table

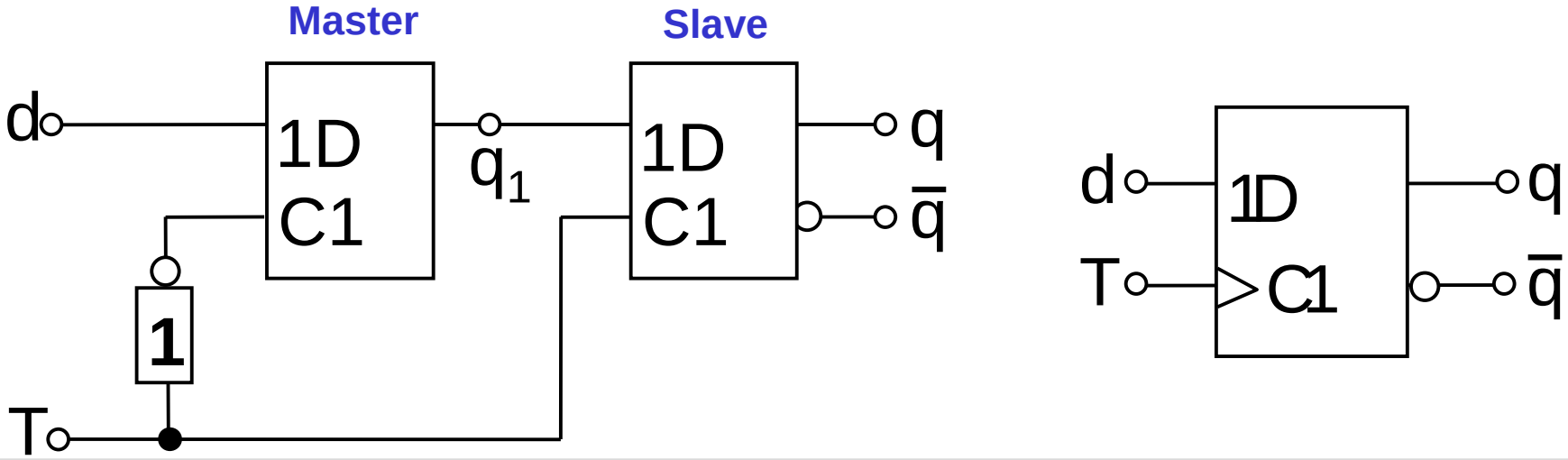
q^t	q^{t+1}	d
0	0	0
0	1	1
1	0	0
1	1	1

  - Control table developed by permutations of columns

© 2017 Burkhard Stiller

Clock-edge Controlled D Flipflop

- A clock-edge controlled D flipflop can be build by combining two D latches that are controlled by complimentary clock levels
  - The first latch is termed master latch, the second slave latch → Master-Slave FF
  - During $T = 0$ the first latch follows changes of the input signal d, whereas the second latch is locked
  - Does T change from 0 to 1 (positive clock edge), the first flipflop is locked
  - Independent of any changes of d, output value q_1 always remains unchanged at value of d, which was formed by the input configuration during the 01-clock edge
  - This value is persisted by the second latch (store), even in case T returns to 0

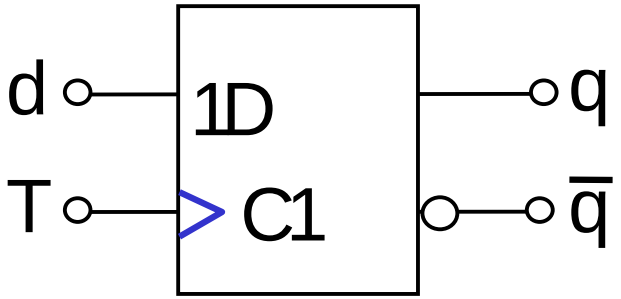


© 2017 Burkhard Stiller

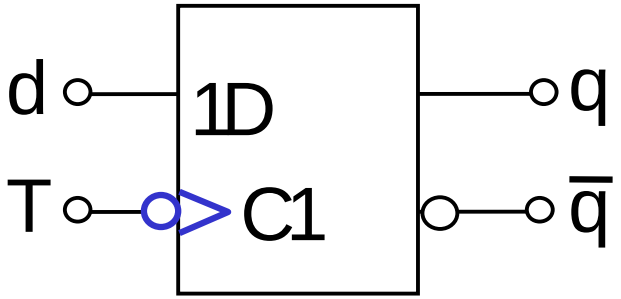
Variants of D Flipflops

- D flipflops can be implemented easily as clock-edge controlled memory elements.
  - Due to their small surface requirements D flipflops are the most used integrated circuits for memory elements

- A clock-edge control is marked by a triangle at the clock input



- At the control of a negative clock-edge the negation sign is combined in front of the triangle



© 2017 Burkhard Stiller

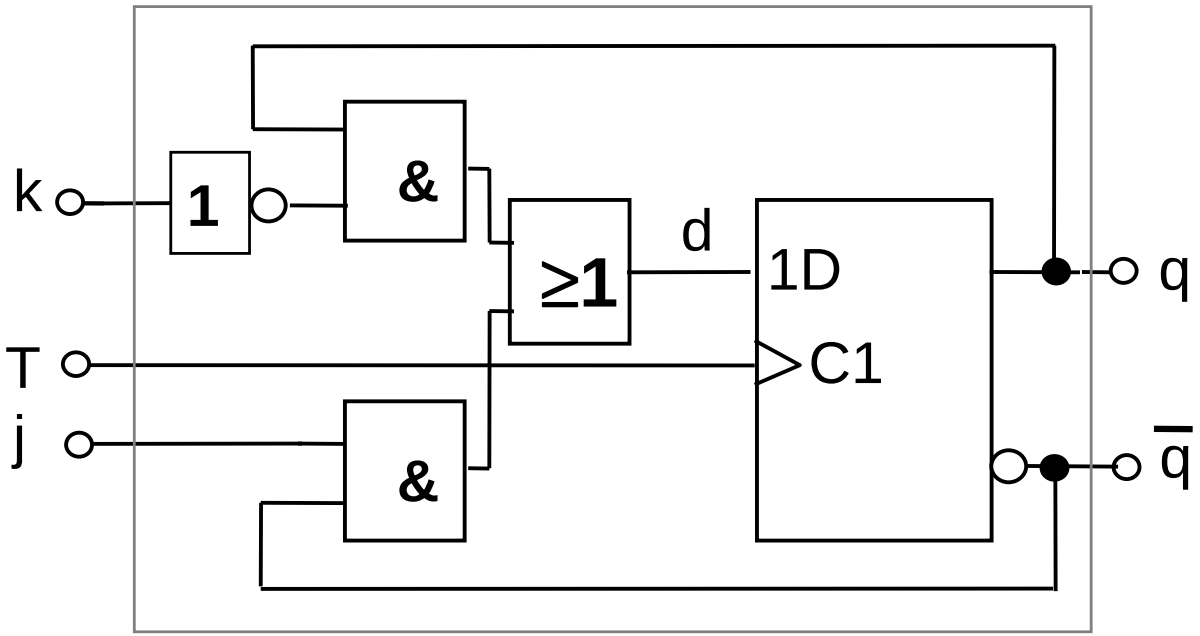
JK Flipflop

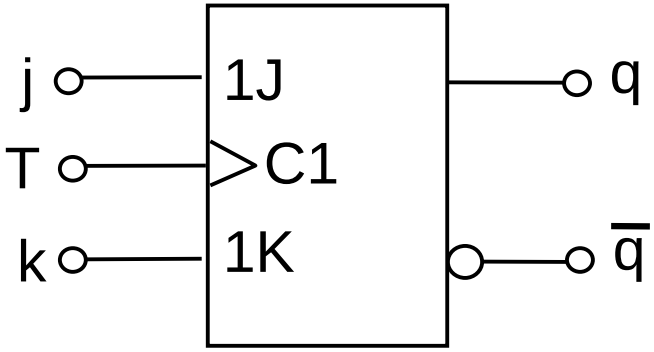
- Reminder: RS flipflops did see the impermissible input combination $r = s = 1$
- Goal: The design of a flipflop using this combination actively
- As a fourth control besides “store”, “set”, and “reset” the content of the flipflop can be complemented at $r = s = 1$
- Notation:
  - j: set
  - k: reset

  $\rightarrow$ JK-Flipflop
- This new behavior can be implemented by adding external circuits to known flipflops

© 2017 Burkhard Stiller

Circuit Diagram of the Synchronous JK Flipflop





$$d = q^t \, \overline{k} \vee \overline{q}^t \, j$$

Function and Control Table of JK Flipflop

Shortend function table:

j	k	q^{t+1}	Function
0	0	q^t	store
0	1	0	reset
1	0	1	set
1	1	$\overline{q^t}$	change

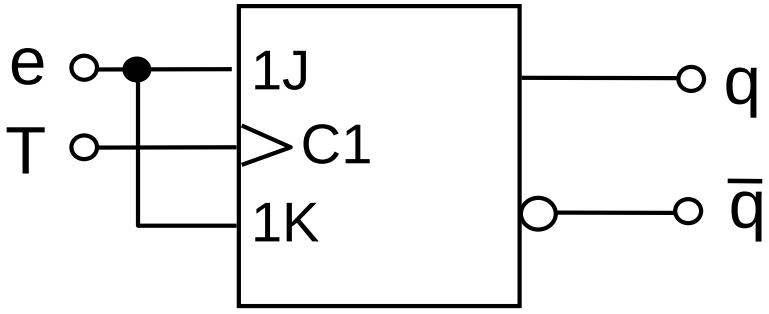
Shortened control table:

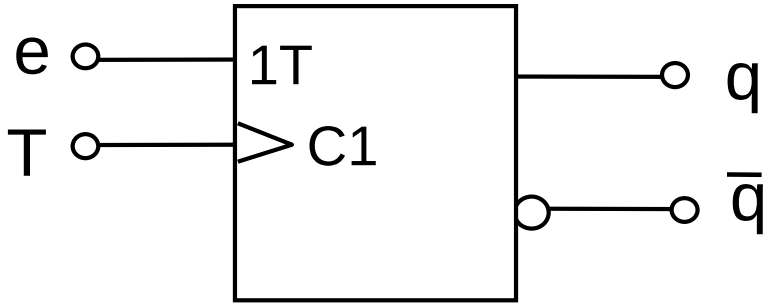
q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

© 2017 Burkhard Stiller

Synchronous T Flipflop

- A T flipflop (toggle, kippen) maintains exactly one input
- Input 1 at variable *e* toggles the flipflop with every clock period into another state; the input 0 keeps the flipflop's past state
- A special input configuration of a JK flipflop ($e = j = k$) leads to the implementation of a T flipflop
- T flipflop's circuit and symbol:





© 2017 Burkhard Stiller

Shortened Function Table of the T Flipflop

e	q^{t+1}	Function
0	q^t	store
1	$\overline{q}^t$	change

- Remarks
  - A synchronous set and reset of T flipflops is not possible with the input e.
  - In order to start the T flipflop within a defined base state, an additional set and reset input is required

© 2017 Burkhard Stiller

Flipflop Control Summary

- RS flipflop (asynchronous): $r = s = 1$ impermissible
  → RS latch (clocked, level-triggered)
- D flipflop, D latch: $r = 0$ and $s = 0$ always observed
- Edge-triggered D flipflop implemented by two D latches
- JK flipflop: $r = s = 1$ defined as “negating” memory
- T-Flipflop: With an input 1 the output is complemented, otherwise the value is stored

© 2017 Burkhard Stiller

Flipflop Control Tables

RS flipflop

q^t	q^{t+1}	r	s
0	0	-	0
0	1	0	1
1	0	1	0
1	1	0	-

D flipflop

q^t	q^{t+1}	d
0	0	0
0	1	1
1	0	0
1	1	1

JK flipflop

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

T flipflop

q^t	q^{t+1}	e
0	0	0
0	1	1
1	0	1
1	1	0

© 2017 Burkhard Stiller

Module 4: Sequential Circuits

- Basics
  - Finite State Machines (FSM)
- Timing and Flipflops
  - (A)synchronous Circuits
  - Level-triggered, Edge-triggered Flipflop Types
- Design of Synchronous Sequential Circuits
- Dedicated Sequential Circuits
  - Registers, Counters, Shift Registers
  - Half-adder, Full-adder
  - Floating Point Unit
- Arithmetic Logic Unit (ALU)

© 2017 Burkhard Stiller

Design of Synchronous Sequential Circuits

- The procedure of designing a synchronous sequential circuit is explained in the following using an example

- Example: Serial adder (Serienaddierer)
  - Two arbitrarily long binary numbers are added place by place
  - The addition starts with the place of lowest value
  - The carry needs to be taken into account in each subsequent step
  - Numbers are entered bit-wise, one place (digit) per clock period
  - The output is generated bit-wise, whereby the output represents at every point in time the sum of all previous input configurations (without considering the carry)

$$\begin{array}{r} 1001101001 \\ +\quad 1010011010 \\ \hline 10100000011 \end{array}$$

617 + 666 = 1283

Design Procedure

Assume a verbal definition of the problem exists (as above)

1. Definition of input and output variables
2. Determination of states
3. Drawing of block diagram
4. Design of finite state machine graph
5. Definition of finite state machine table
6. Determination of state coding
7. Generation of coded sequence table
8. Extension of coded sequence table with flipflop types' control
9. Minimization of output and control networks for flipflops
10. Drawing and implementation of sequential circuit

© 2017 Burkhard Stiller

1. Variables and 2. States

- Example: Serial adder
  - Two input variables, sequences of digits of a number *x* and a number *y*
  - A sequence of one output variable, the sum *s*
- For any given clock period, only the carry from the previous period must be known
- Thus, two states – represented by the state variable set Z – are sufficient for solving this problem:
  - State C (carry, Übertrag):
    In case of a carry from the previous place is to be taken into account
  - State nC (no carry, kein Übertrag):
    Represents the opposite case (no carry)
  - Z = { C, nC }

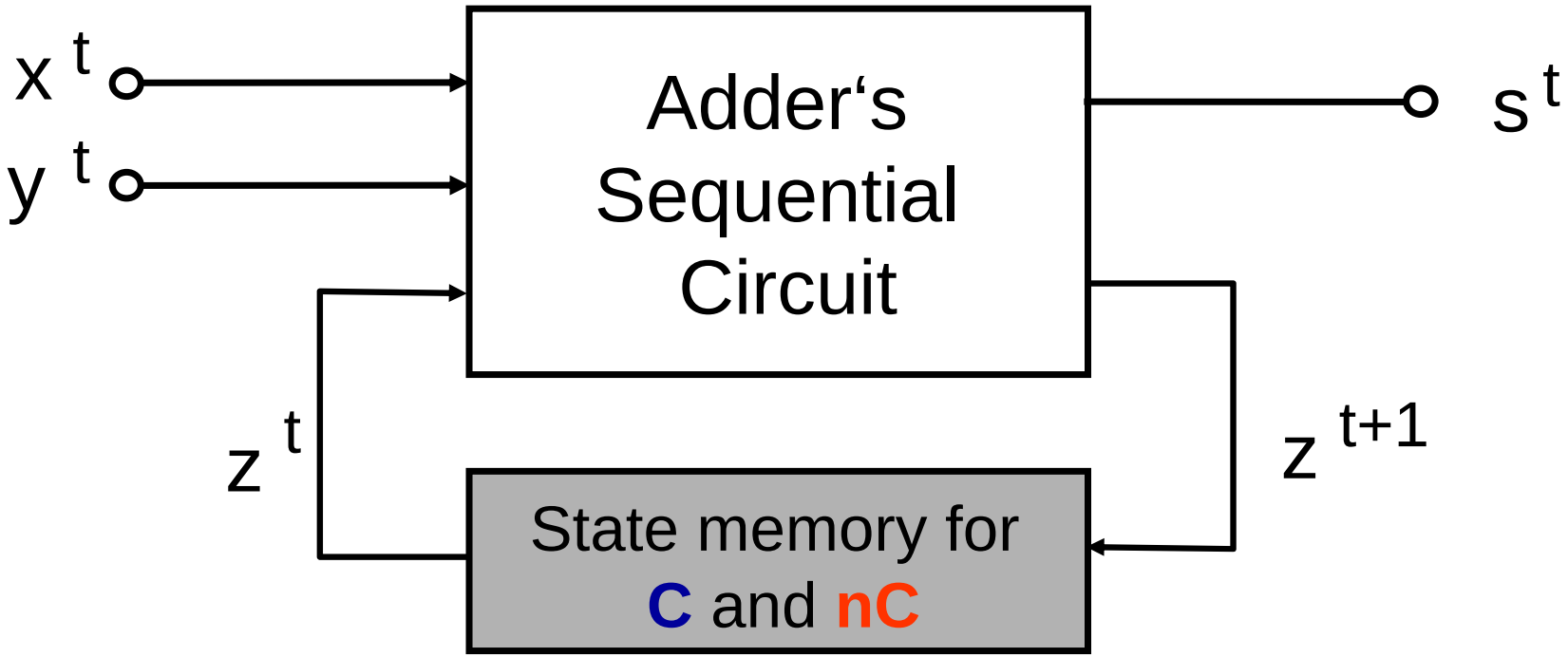
© 2017 Burkhard Stiller

3. Block Diagram

Graphical representation and consistency check:

Input variables: *x* and *y* Output variable: *s*

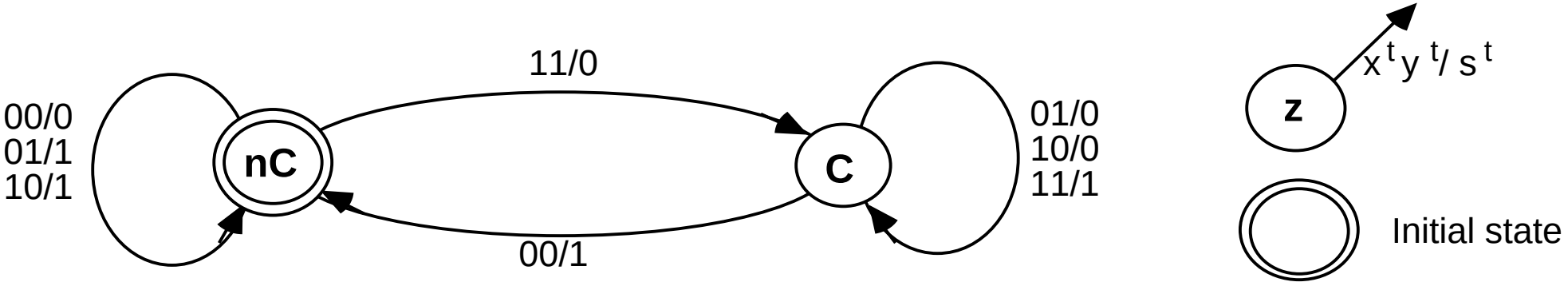
State memory: *C* and *nC*



© 2017 Burkhard Stiller

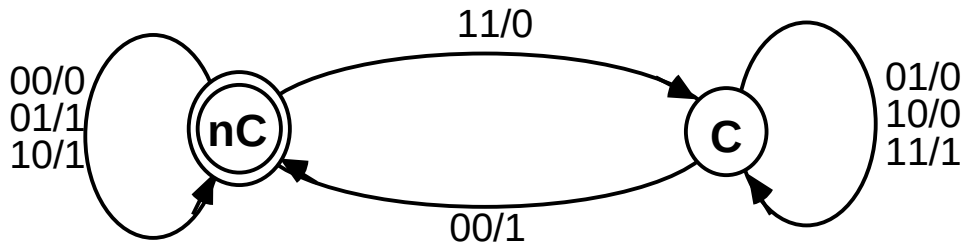
4. Finite State Machine Graph

- Since output configurations depend on the current input configurations a Mealy sequential circuit is required
  - Thus, all output assignments are tagged to edges in the FSM
- FSM graph for serial adder:



5. Finite State Machine Table

- The FSM table can be derived from the FSM graph



- FSM table of a serial adder:

z	z^+ / s			
	x y =			
	00	01	10	11
nC	nC /0	nC /1	nC /1	C /0
C	nC /1	C /0	C /0	C /1

Note:

For synchronous sequential circuits, stable states are not marked separately, since it is assumed that all states are stable until the next clock period occurs.

© 2017 Burkhard Stiller

6. State Coding (1)

- For asynchronous sequential circuits
  - state coding is very critical,
  - since determination of a suitable state coding is crucial for the operation of the sequential circuit

- For synchronous sequential circuits
  - state coding is not critical,
  - since a synchronous sequential circuit basically works with every unique state coding

- However, a "good" state coding can reduce the amount of circuitry required for synchronous sequential circuits
  - Coding with a minimum length for *k* states:

$$\lceil \mathrm{ld}\ k \rceil = \text{minimum number of flipflops}$$

© 2017 Burkhard Stiller

6. State Coding (2)

- State coding trivial, since only two states *C* and *nC* exist
  → One state variable *c*

z	c
nC	0
C	1

- For the generation of the output sequential circuit and transition sequential circuit
  - insert the state coding into the finite state machine table
    → Coded sequence table for sequential circuit

© 2017 Burkhard Stiller

7. Coded Sequence Table

z	c
nC	0
C	1

State coding (6.)

z	z^+ / s			
	x y =			
	00	01	10	11
nC	nC/0	nC/1	nC/1	C/0
C	nC/1	C/0	C/0	C/1

Finite state machine table (5.)

	c^t	x^t	y^t	c^{t+1}	s^t
nC	0	0	0	0	0
	0	0	1	0	1
	0	1	0	0	1
	0	1	1	1	0
C	1	0	0	0	1
	1	0	1	1	0
	1	1	0	1	0
	1	1	1	1	1

Coded sequence table

© 2017 Burkhard Stiller

8. Preparations for Flipflop Type Selections

- Functions and terms of the output sequential circuit can be derived already from the coded sequence table
- In order to determine the transition function's control circuit, the type of flipflop used must be defined first
- Each flipflop type needs a dedicated control circuit
- The selection of a dedicated flipflop type influences the size of the transition function's control circuit
- The quality of state coding decided on can only be assessed based on a certain type of flipflop selected

© 2017 Burkhard Stiller

8. Flipflop Type Selection

- The serial adder shall be implemented by a JK flipflop (type)
- The coded sequence table
  - is extended by control inputs of the flipflop type selected:

q^t	q^{t+1}	j	k
0	0	0	-
0	1	1	-
1	0	-	1
1	1	-	0

JK flipflop control table
"-" reads as value 0 or 1

c^t	x^t	y^t	c^{t+1}	s^t	j^t	k^t
0	0	0	0	0	0	-
0	0	1	0	1	0	-
0	1	0	0	1	0	-
0	1	1	1	0	1	-
1	0	0	0	1	-	1
1	0	1	1	0	-	0
1	1	0	1	0	-	0
1	1	1	1	1	-	0

© 2017 Burkhard Stiller

9. Minimization of Output and Control Circuits

Control variables j^t and k^t:

j^t

	y		
0	0	X	X
0	1	X	X

x

c

k^t

	y		
X	X	0	1
X	X	0	0

x

c

Output variable s^t:

s^t

	y		
0	1	0	1
1	0	1	0

x

c

X in KV diagram:
value may be 0 or 1
(as needed)

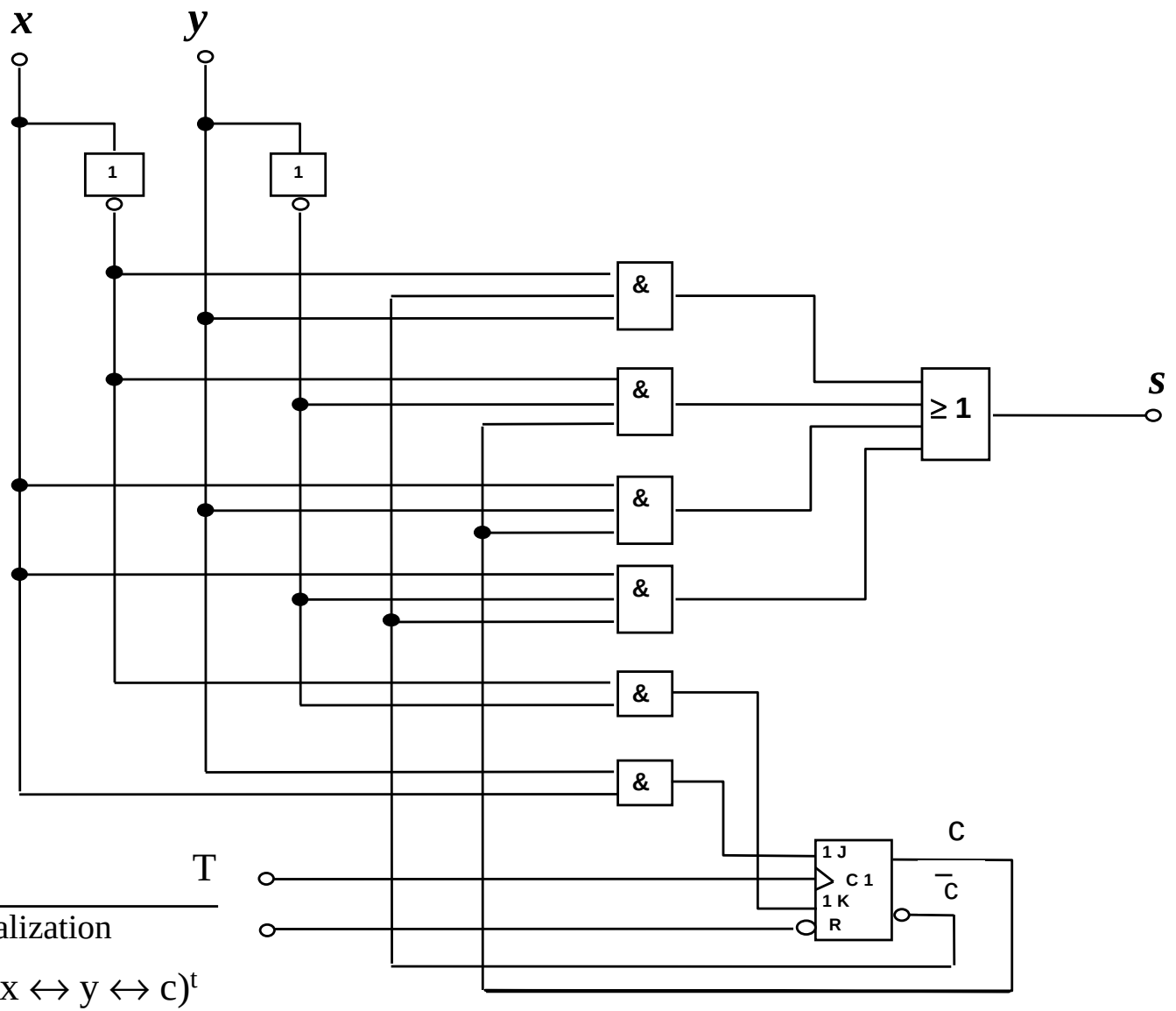
$$j^t = (x\ y)^t$$

$$k^t = (\ \overline{x}\ \overline{y}\ )^t$$

$$s^t = (\ x\ \overline{y}\ \overline{c} \vee \overline{x}\ y\ \overline{c} \vee x\ y\ c \vee \overline{x}\ \overline{y}\ c\ )^t = (x \leftrightarrow y \leftrightarrow c)^t$$

© 2017 Burkhard Stiller

10. Drawing of Sequential Circuit



$j^t = (x\ y)^t$

$k^t = (\overline{x}\ \overline{y})^t$

$s^t = (x\ \overline{y}\ \overline{c} \vee \overline{x}\ y\ \overline{c} \vee x\ y\ c \vee \overline{x}\ \overline{y}\ c)^t = (x \leftrightarrow y \leftrightarrow c)^t$

© 2017 Burkhard Stiller

Module 4: Sequential Circuits

- Basics
  - Finite State Machines (FSM)
- Timing and Flipflops
  - (A)synchronous Circuits
  - Level-triggered, Edge-triggered Flipflop Types
- Design of Synchronous Sequential Circuits
- Dedicated Sequential Circuits
  - Registers, Counters, Shift Registers
  - Half-adder, Full-adder
  - Floating Point Unit
- Arithmetic Logic Unit (ALU)

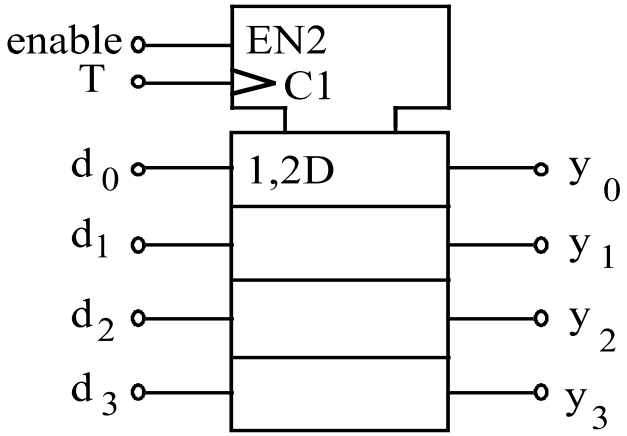
© 2017 Burkhard Stiller

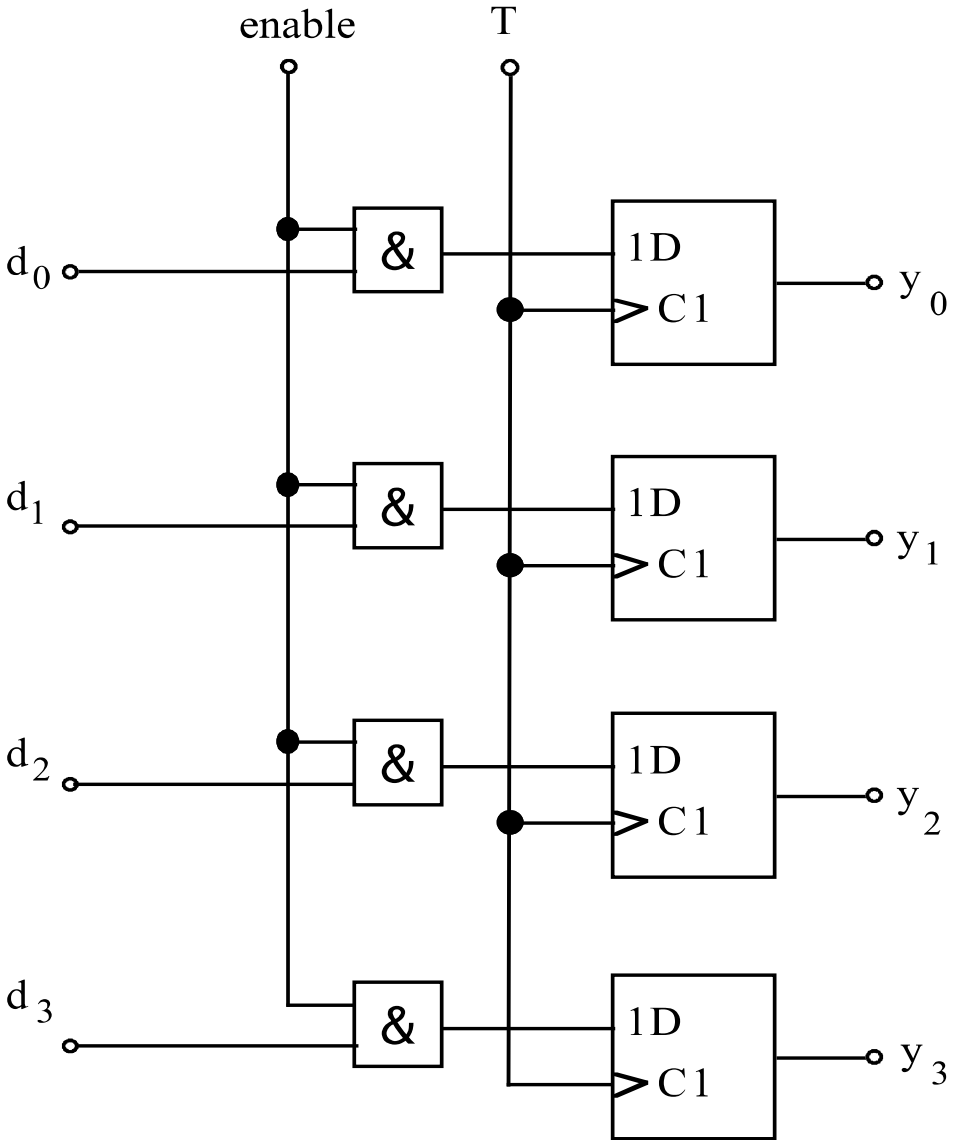
Registers

- Registers are linear arrangement of flipflops for storing several bits, a “bit vector”
- All flipflops for a register are controlled by a single clock
- Simple registers:
  - Un-coupled adjacent D flipflops
- In general, all flipflops are influenced by additional joint controls
  - Store
  - Read
  - Delete
  - …

© 2017 Burkhard Stiller

4-bit Register (D Flipflop with Enable Signal)



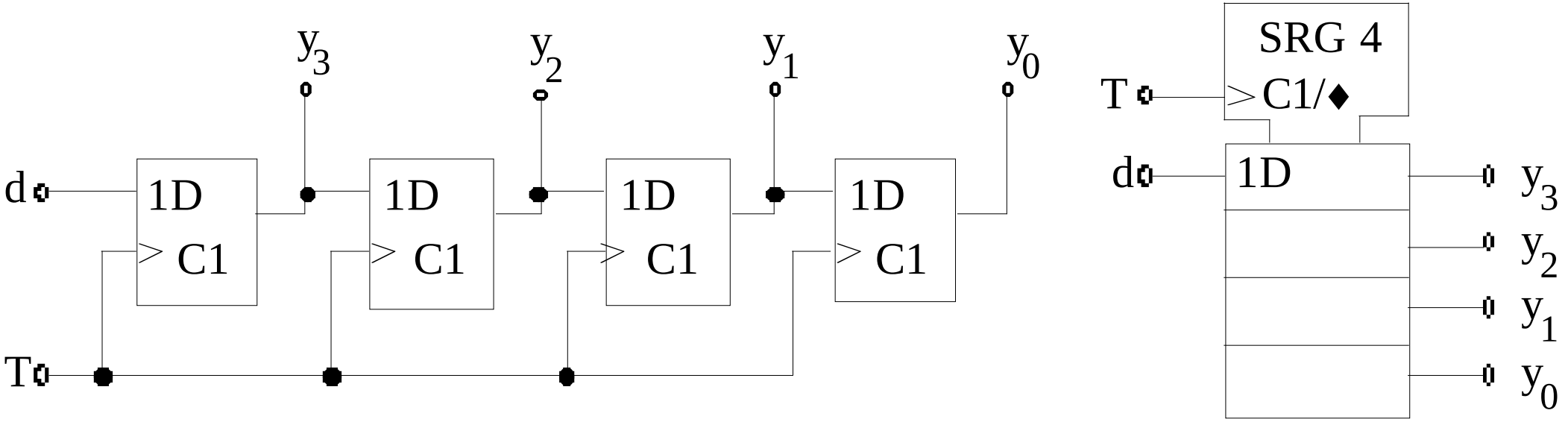


Data is only transferred
if input “enable” = 1

Shift Registers (1)

- Shift registers "shift" their content across interconnected memory elements
  - Implemented as a chain of registers or D flipflops connected in series
  - Thus the output of one memory element is connected to the input of the next

- Example of a 4-bit shift register:



© 2017 Burkhard Stiller

Shift Registers (2)

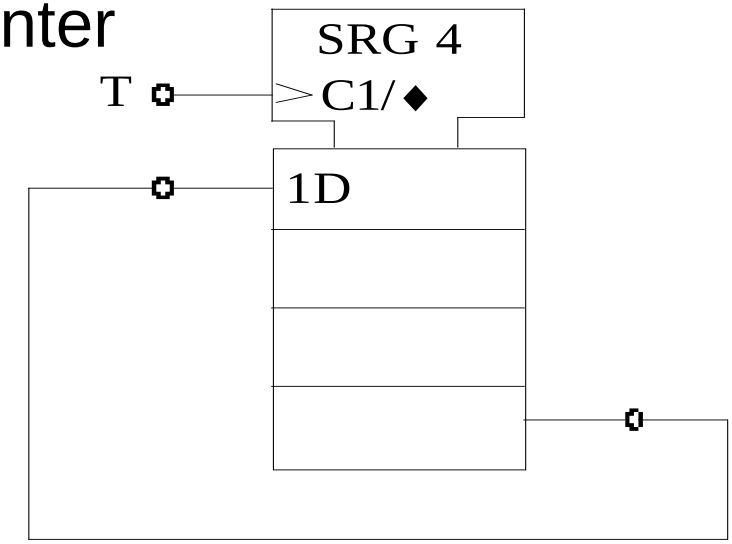
- Main purpose: Division/multiplication by/with "2_{10}"
- If a bit sequence $y_3\ y_2\ y_1\ y_0$ is interpreted as a binary number, a shift to the right by one memory element (place) corresponds to a division by 2 (without remainder)
- If bit sequence is shift to the left by one memory element (with 0 as the newly introduced last bit to the right) the result corresponds to a multiplication with 2

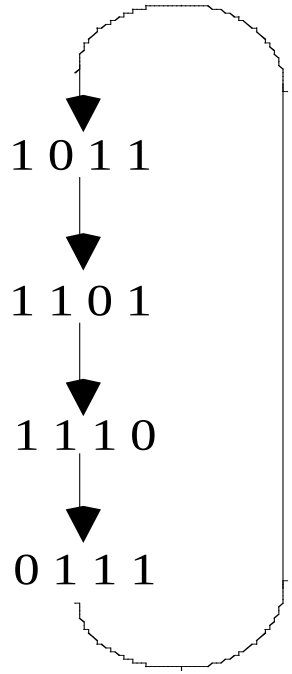
© 2017 Burkhard Stiller

Shift Registers (3)

- Additional purpose: Ring counter (Ringzähler) or circular buffer (Umlaufspeicher)
  - By connecting the serial output of a shift register with its serial input a ring counter is achieved
  - Ring counters can store a bit sequence for an unlimited amount of time and move it in continuous circles

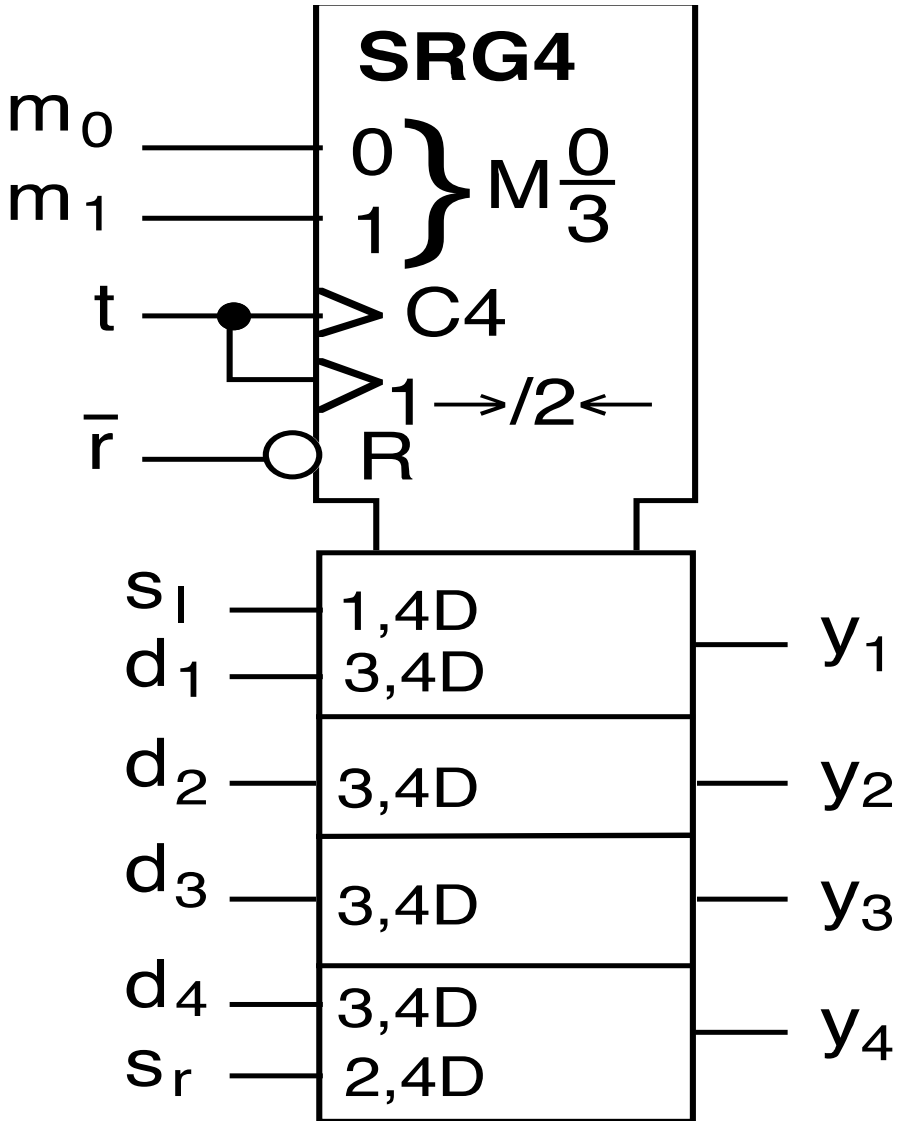
- Example of a ring counter with a circulation of 4 bit long content





© 2017 Burkhard Stiller

4-bit Shift Register Chip 74LS194 (1)

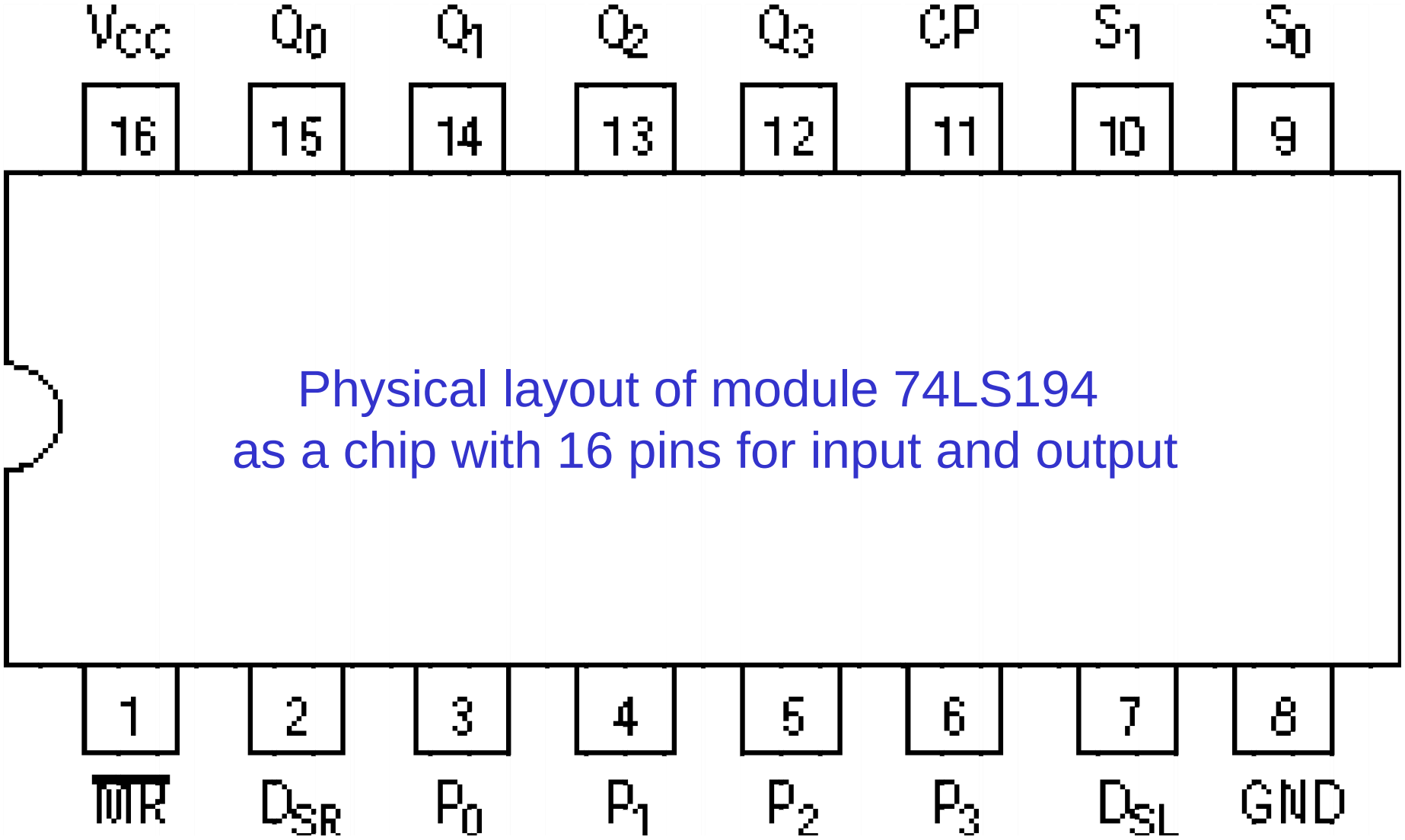


Functions

1. Data input at the 4 parallel inputs $\mathbf{d_1}$ to $\mathbf{d_4}$ are loaded in parallel
2. Data at input $\mathbf{s_l}$ are serially included and old register content is moved further down
3. Data at input $\mathbf{s_r}$ are serially included and old register content is moved upward
4. Current register content is stored along multiple clock periods
5. The reset input $\overline{\mathbf{r}}$ is used to reset asynchronously the full shift register

© 2017 Burkhard Stiller

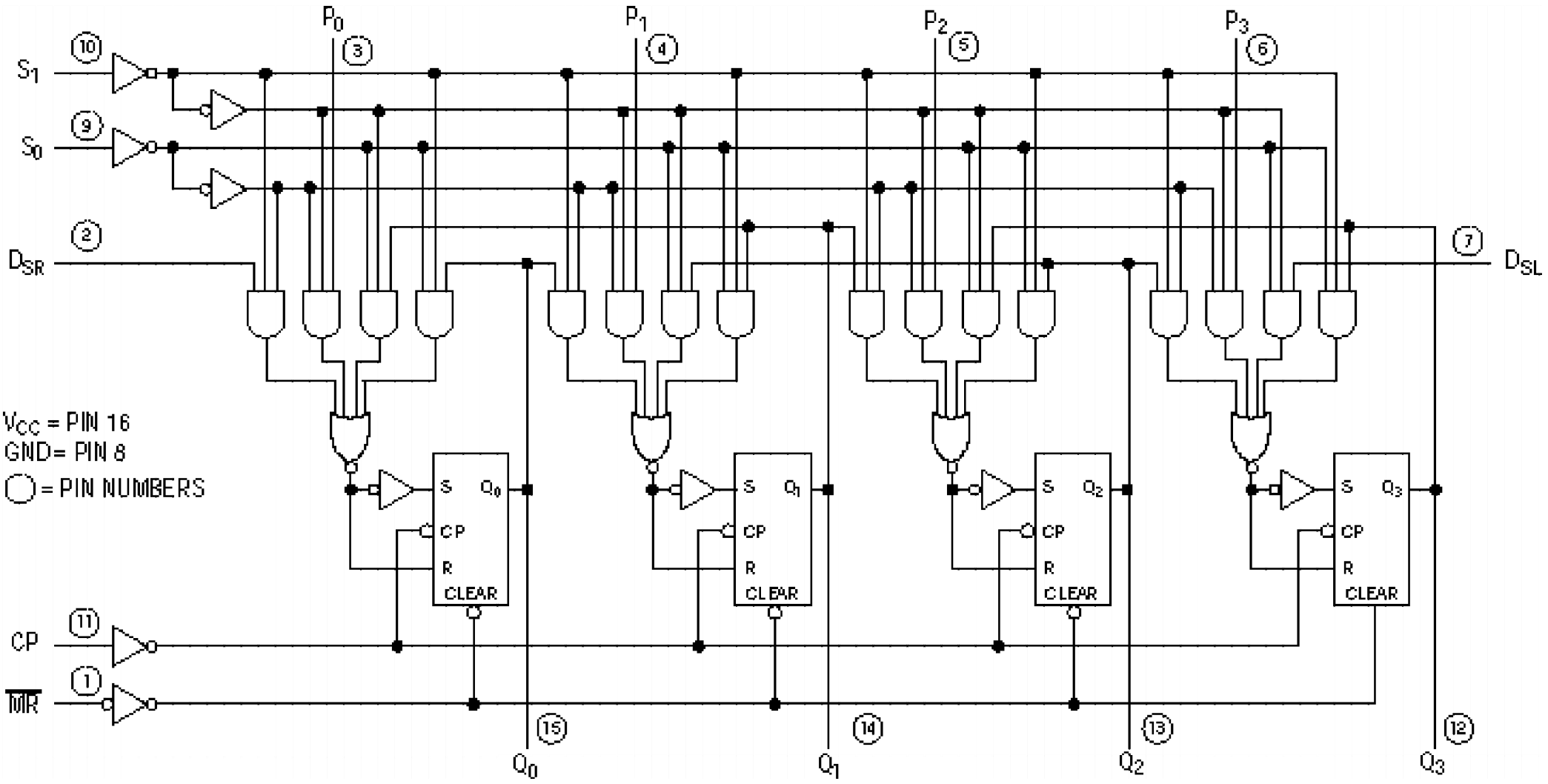
4-bit Shift Register Chip 74LS194 (2)



© 2017 Burkhard Stiller

4-bit Shift Register Chip 74LS194 (3)

Internal circuit details of module 74LS194 (US gate symbols)



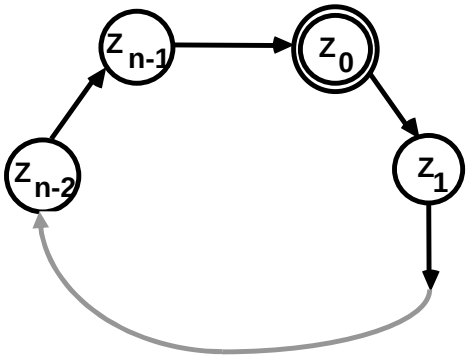
© 2017 Burkhard Stiller

ifi

Counters (1)

- Counters implement the function of counting of “events”

Basic FSM



- Examples
  - Counting of pulses or state changes
  - Counters can
    - point to successive addresses for a given memory, *e.g.*, in their instantiation of program counters
    - check successive work steps for control units
  - Counters can act as frequency dividers
    - A given pulse sequence can be reduced in frequency
    - In doing so, one takes advantage of the fact that the bit *i* of a number $z_n \ldots z_i \ldots z_0$ changes only 2^{-i} times as often as bit 0, if this number is incremented continuously

© 2017 Burkhard Stiller

Counters (2)

- By modifying the basic structure additional functions possible
  - Setting or resetting inputs
  - Enabling input
  - Counting forward or backward

- If states are binary-coded (state Z_i is coded with the binary number i), the resulting circuit is termed binary counter

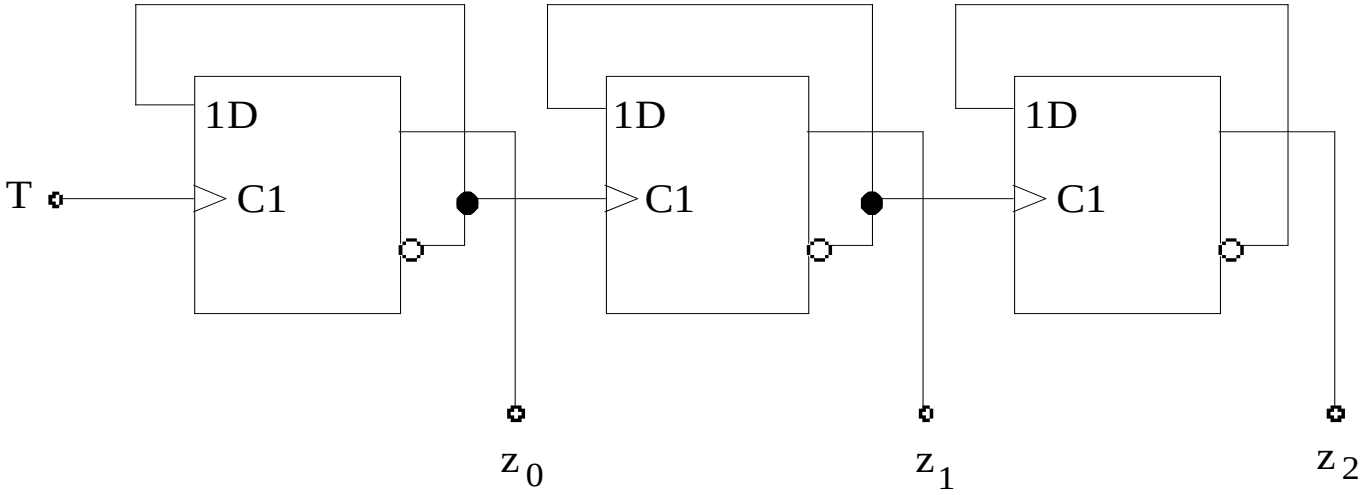
- Depending on the length n of the counting cycle, a counter is called a modulo-n-counter

- If $n = 2^m$ holds, the circuit is termed m-ary counter

© 2017 Burkhard Stiller

Asynchronous Counters

- The size of asynchronous counters' control logic (Ansteuerlogik) grows rapidly as the number of bits increases
- Thus, ripple counters (asynchronous counters) are attractive
  - Example implementation of a 3-ary binary ripple counter:



- Such a counter's circuit is slow, since each flipflop only reacts when the previous flipflop has switched from 1 to 0
- Outputs of this circuit do not change simultaneously

© 2017 Burkhard Stiller

Half-adder (1)

- For adding two binary 1-ary numbers *a* and *b*
  - The sum *s* and the carry *c* determine the result

- Function table of half-adder:

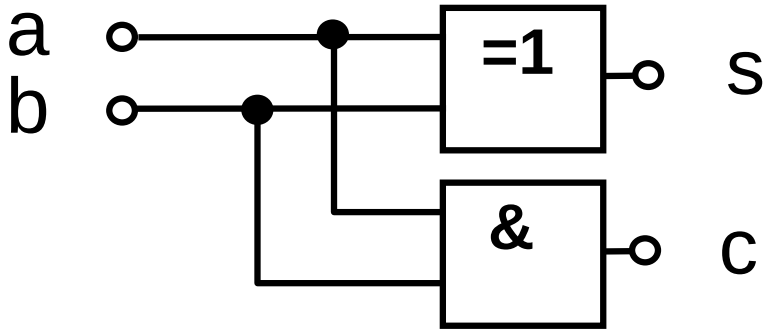
a	b	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

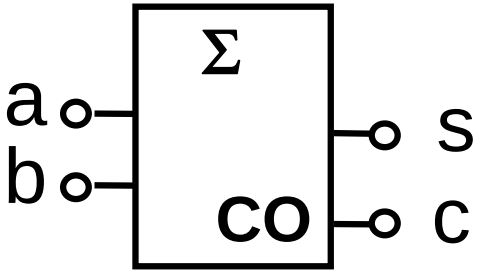
© 2017 Burkhard Stiller

Half-adder (2)

- Equations: $s = a \overline{b} \vee \overline{a}\, b = a \not\leftrightarrow b$
  $c = a\, b$

- The circuit and its symbol for 1-bit half-adder:





© 2017 Burkhard Stiller

Full-adder (1)

- Adding n-ary binary numbers
  - An additional input variable for carry of the preceding place required

a_i	b_i	$ü_i$	s_i	c_{i+1}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

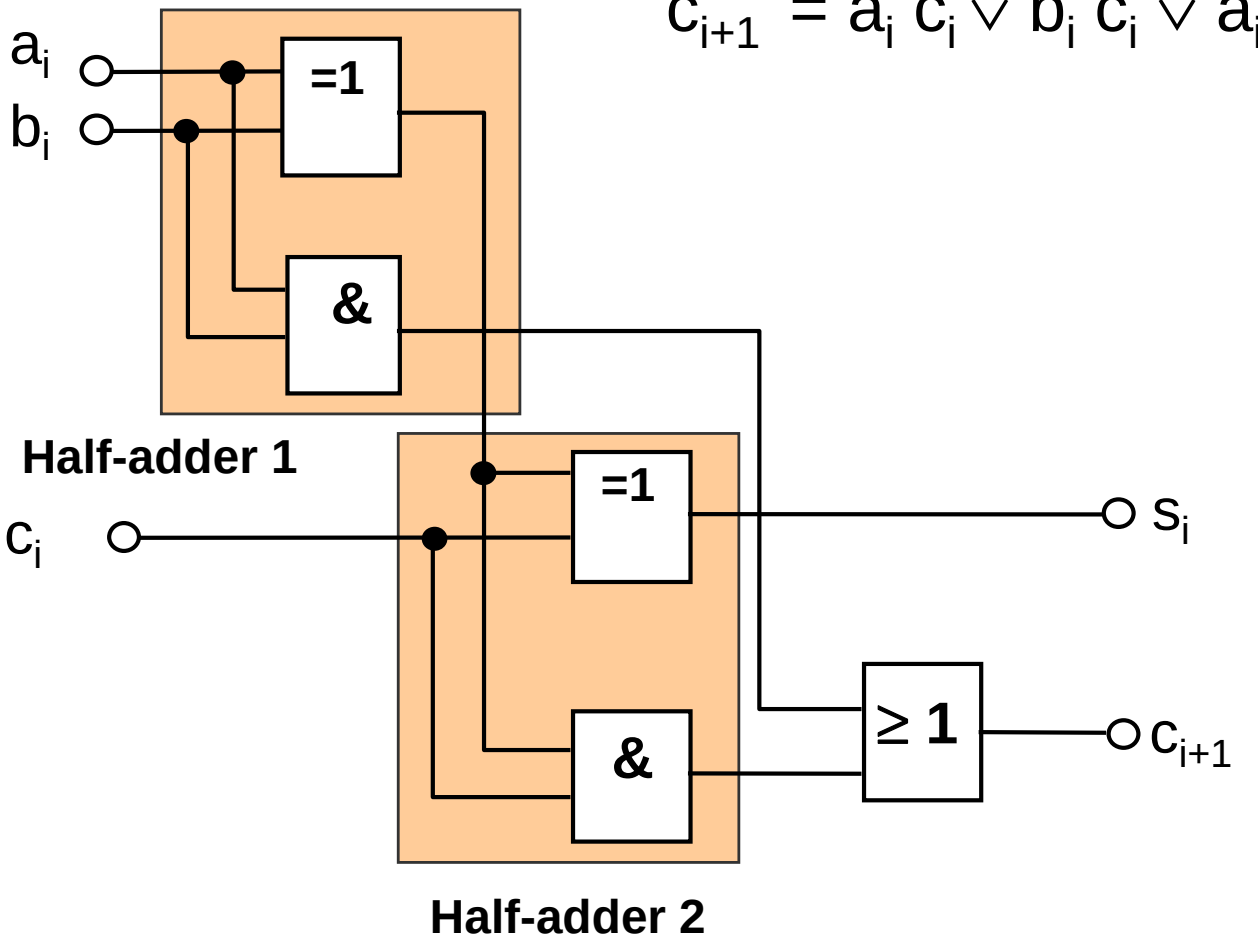
© 2017 Burkhard Stiller

Full-adder (2)

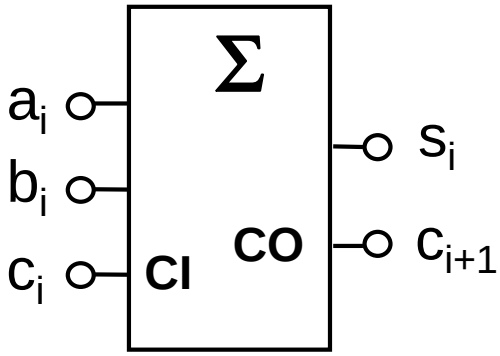
Equations:

$$s_i = a_i \leftrightarrow b_i \leftrightarrow c_i$$

$$c_{i+1} = a_i\, c_i \vee b_i\, c_i \vee a_i\, b_i = (a_i \leftrightarrow b_i)\, c_i \vee a_i\, b_i$$



Symbol



Full-adder

© 2017 Burkhard Stiller

ifi

Carry-lookahead-adder (1)

- Carry-lookahead-adders calculate all input configurations directly from input variables
  - Avoids the drawback of carry-ripple adder's large addition times

- It applies:

$$c_{i+1} = a_i\, b_i \vee (a_i \not\leftrightarrow b_i)\, c_i$$
$$= g_i \vee p_i\, c_i$$
$$s_i = (a_i \not\leftrightarrow b_i) \not\leftrightarrow c_i$$
$$= p_i \not\leftrightarrow c_i$$

- with

$$g_i = a_i\, b_i \quad \text{(generate carry)}$$
$$p_i = (a_i \not\leftrightarrow b_i) \quad \text{(propagate carry)}$$

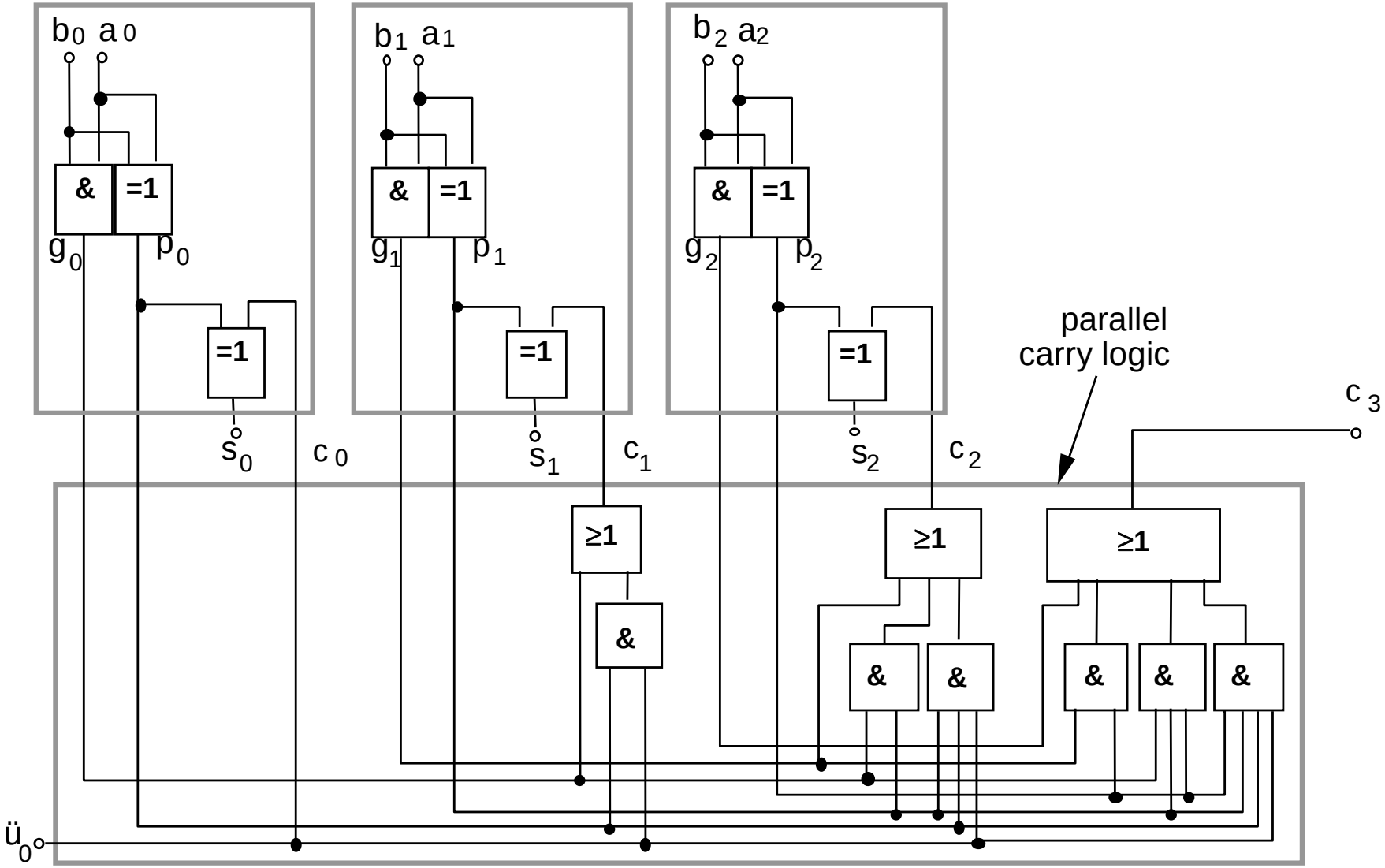
- g_i and p_i can be generated directly from input variables

Carry-lookahead-adder (2)

- Problem
  - The size of control logic increases with an increasing number of digits
- Solutions
  - Smaller carry-lookahead-adders with parallel carry generation and serially cascaded
  - Processing all carries of many smaller blocks in parallel
  - Hierarchy of carry-lookahead-adders

© 2017 Burkhard Stiller

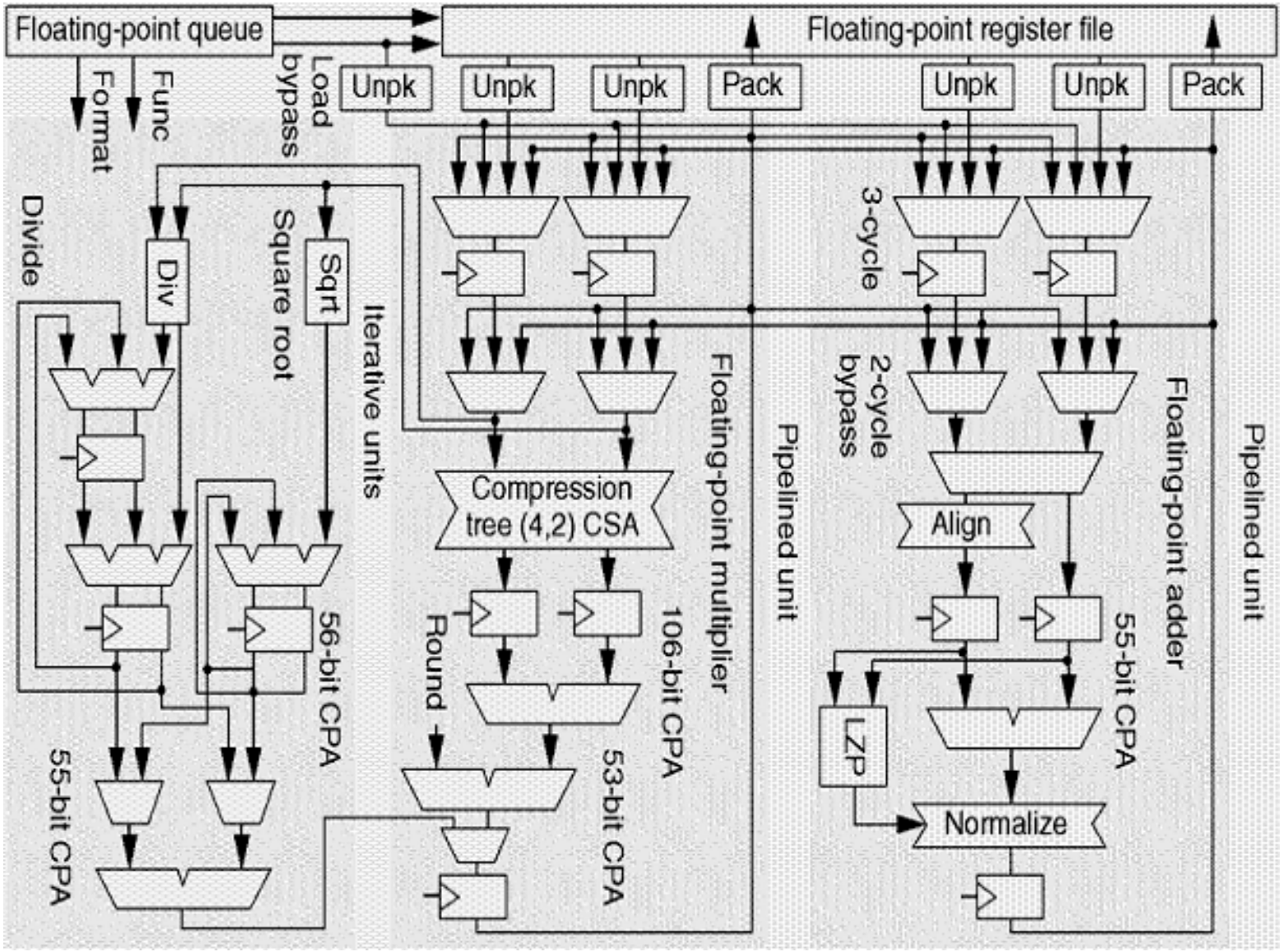
Example of a 3-bit Carry-lookahead-adder



© 2017 Burkhard Stiller

ifi

Example MIPS R10000 Floating-point Unit



© 2017 Burkhard Stiller

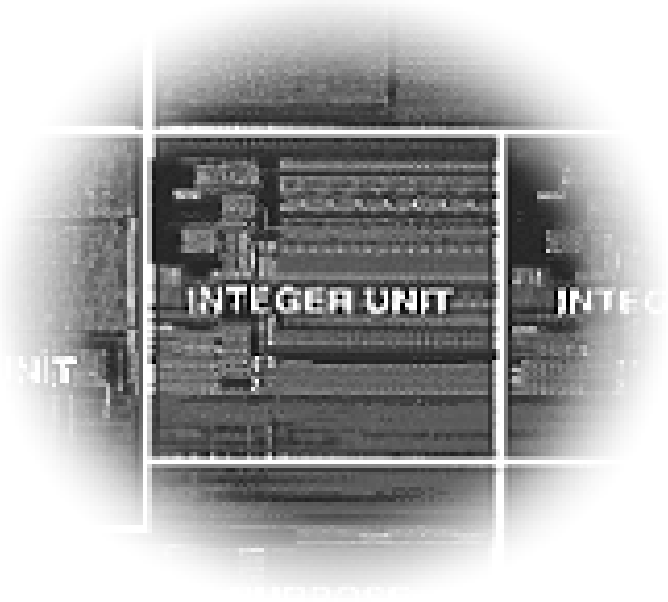
Module 4: Sequential Circuits

- Basics
  - Finite State Machines (FSM)
- Timing and Flipflops
  - (A)synchronous Circuits
  - Level-triggered, Edge-triggered Flipflop Types
- Design of Synchronous Sequential Circuits
- Dedicated Sequential Circuits
  - Registers, Counters, Shift Registers
  - Half-adder, Full-adder
  - Floating Point Unit
- Arithmetic Logic Unit (ALU)

© 2017 Burkhard Stiller

Arithmetic Logic Unit (ALU)

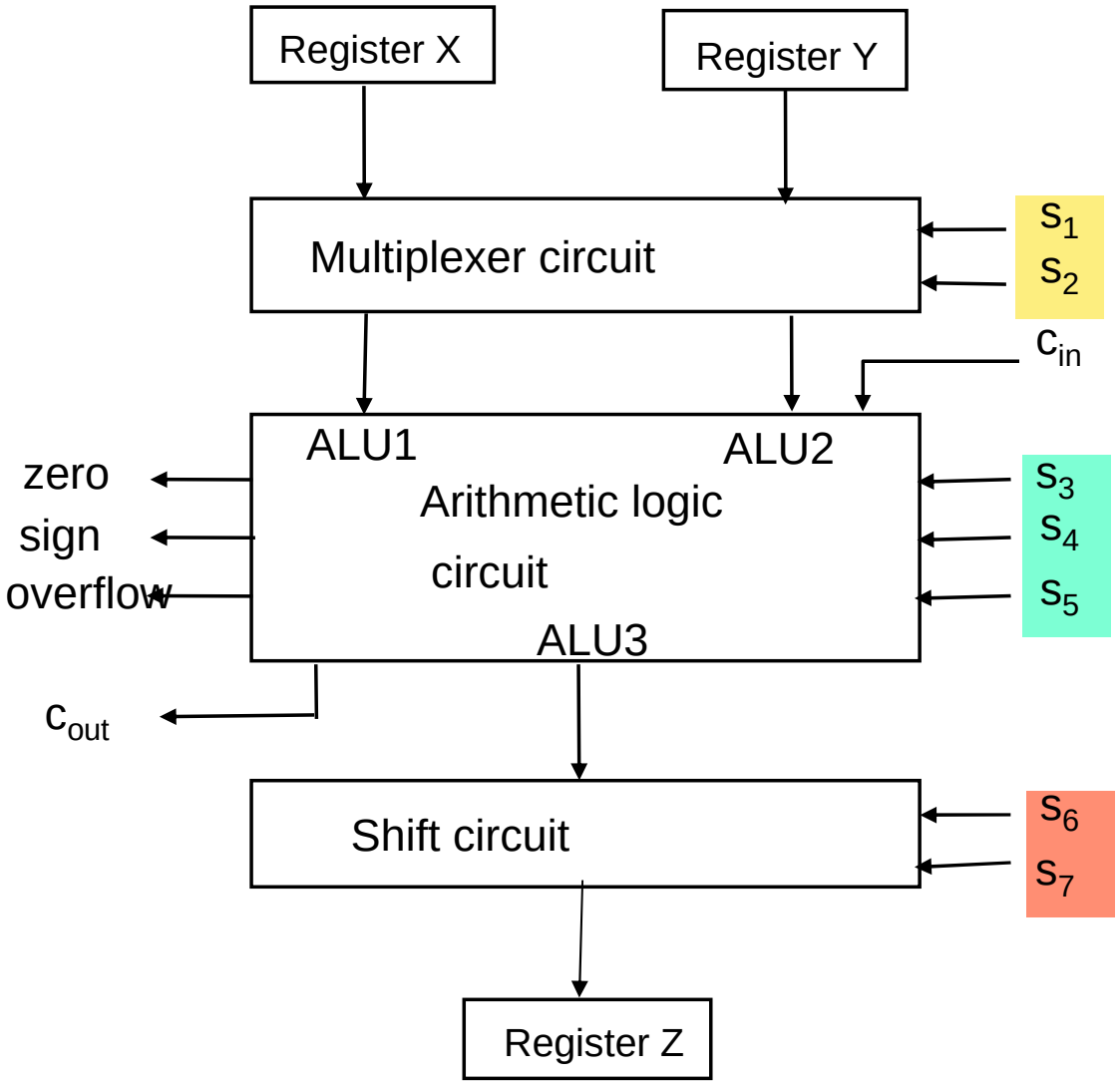
- Arithmetic Logic Unit (ALU)
  - Calculator, functional core of a digital computer
  - Performing logical and arithmetic operations.
- ALU input data
  - Data and control signals from processor
- ALU output data
  - Results and status signals to processor



- Almost always ALUs integrated into a processor process fixed-point numbers only
  - Floating point operations are either executed by a floating point unit or are converted by software into a sequence of fixed point operations

© 2017 Burkhard Stiller

A Simple ALU (Schema)



s_1	s_2	ALU1	ALU2
0	0	X	Y
0	1	X	0
1	0	Y	0
1	1	Y	X

s_3	s_4	s_5	ALU3
0	0	0	ALU1 + ALU2 $+c_{in}$
0	0	1	ALU1 – ALU2 - $\overline{c_{in}}$
0	1	0	ALU2 – ALU1 - $\overline{c_{in}}$
0	1	1	ALU1 $\vee$ ALU2
1	0	0	ALU1 $\wedge$ ALU2
1	0	1	$\overline{\text{ALU1}} \wedge$ ALU2
1	1	0	ALU1 $\not\leftrightarrow$ ALU2
1	1	1	ALU1 $\leftrightarrow$ ALU2

s_6	s_7	Z
0	0	ALU3
0	1	ALU3 $\div$ 2
1	0	ALU3 $\times$ 2
1	1	Z store

© 2017 Burkhard Stiller

Components of an ALU

- Set of registers
  - Stores variable values of input and output variables
- Multiplexer circuit
  - Determines which input variable will be stored in which ALU memory
- Arithmetic logic circuit
  - Performs arithmetic and logical operations
- Shift circuit
  - Performs multiplications, divisions, or leaves inputs unchanges

- Inputs:
  - Data words *X* and *Y*, control signals $s_1 \ldots s_7$ for selecting the selected operation
- Outputs:
  - Status signals (a) zero, (b) sign, and (c) overflow
  - Control unit is enabled to detect dedicated ALU states and to react accordingly

© 2017 Burkhard Stiller

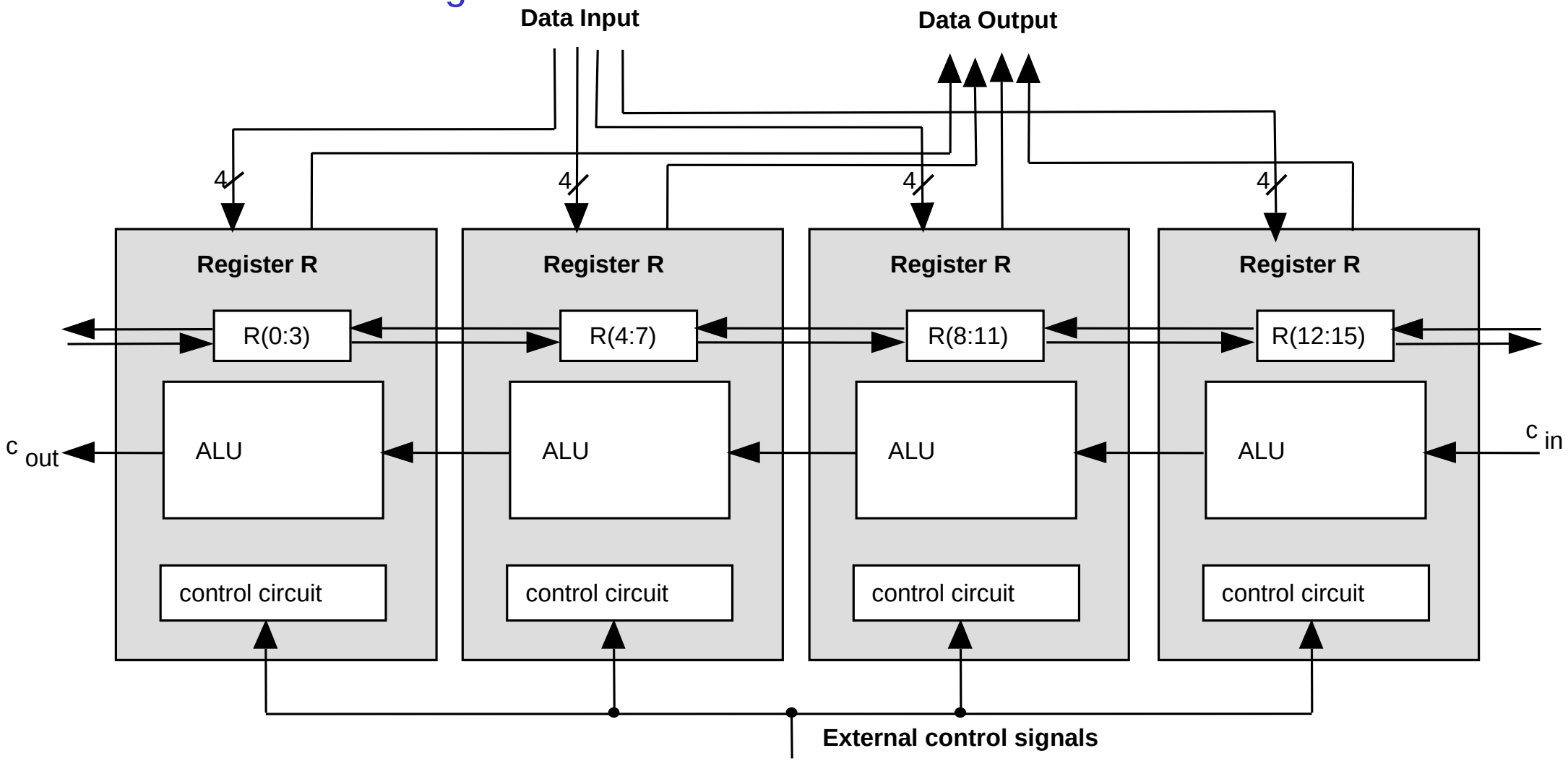
Bit-slice ALU

- Bit-slice ALU
  - Expandable structure on a single chip with a shorter word length
  - Typically with *m* = 4 or 8 → m-bit ALU
  - Concatenation of *k* similar blocks of, i.e., an m-bit ALU
  - Enables the processing of operands with a word length of *k* * *m*
- All chips receive identical control signals
  - They perform in parallel the identical operation on different parts of the operand
- All carries are considered by the neighboring chip
- A common control is provided, *e.g.*, by micro-programmable sequences and its control unit

© 2017 Burkhard Stiller

Example AMD 2901 Chip (1)

4 bit slices of 4 bit length



Example AMD 2901 Chip (2)

- Components
  - Register set of 16 registers of 4 bits each (16x4-bit RAM)
  - ALU can perform 3 arithmetic and 5 logic functions
  - Command bus *I* for selecting an operation
  - Shift circuit to right/left shift
- ALU inputs
  - From the register memory or register *Q*
  - From external data input *D* or “null”
  - Memory addresses *A* and *B* determine registers for source and destination
- ALU outputs (results)
  - Internal in register set or at data output *Y*
  - Control lines *I*, *A*, and *B* are connected in parallel to all bit slices
  - Each bit slice processes the carry c_{in} and transmits it to *cout*,
    - Enables a group carry in terms of a serial processing
  - Flags F_0, *OVR*, and *Z* indicate the processor status: sign, overflow, and zero

© 2017 Burkhard Stiller

— Computer Engineering —
Module 5: Computer Engineering



Universität Zürich[UZH]

CSG
Communication Systems Group

© 2017 Burkhard Stiller

ifi

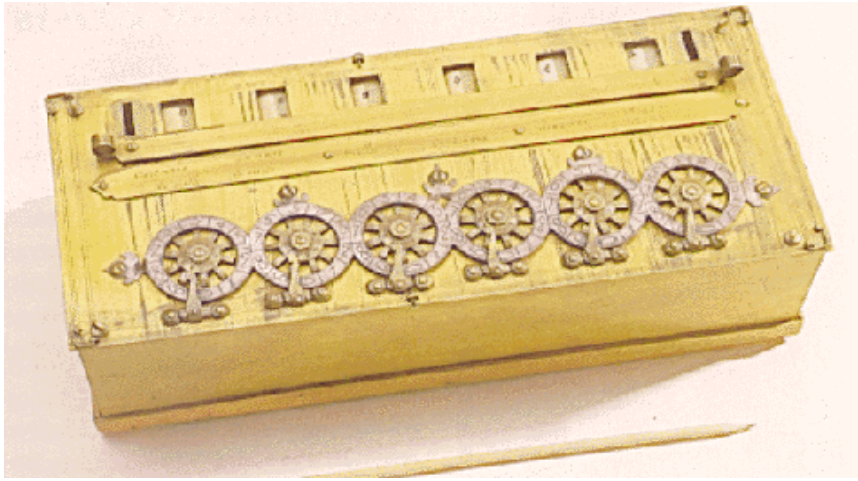
Modul 5: Computer Engineering

- von-Neumann Architecture
- Set-up and Operations
- Organization
- Periphery

© 2017 Burkhard Stiller

Overview on Computer Developmemts

- 1642: Blaise Pascal (1623-1662)
  - Addition
- 1672: Gottfried Wilhelm Leibniz (1646-1716)
  - +, -, *, /
- 1774: Philipp Matthäus Hahn (1739-1790)
  - Reliability
- 1792-1871: Charles Babbage
  Difference Engine and Analytical Engine
  - Only plans, never fully operational
- 1936: Konrad Zuse (Z1, Z2, Z3, Z4)
  - 1st programmable and operational computer (Z3)
- 1944: Howard Aiken (Havard Mark I)
  - Additions 0,3 s, multiplications 6 s
- 1946: J. P. Eckert, J. W. Mauchly
  ENIAC (Electronic Numerical Integrator And Computer)
  - Multiplications 3 ms
- 1946-1952: J. von Neumann, A. W. Burcks, H. H. Goldstine
  EDVAC (Electronic Discrete Variable Automatic Computer)

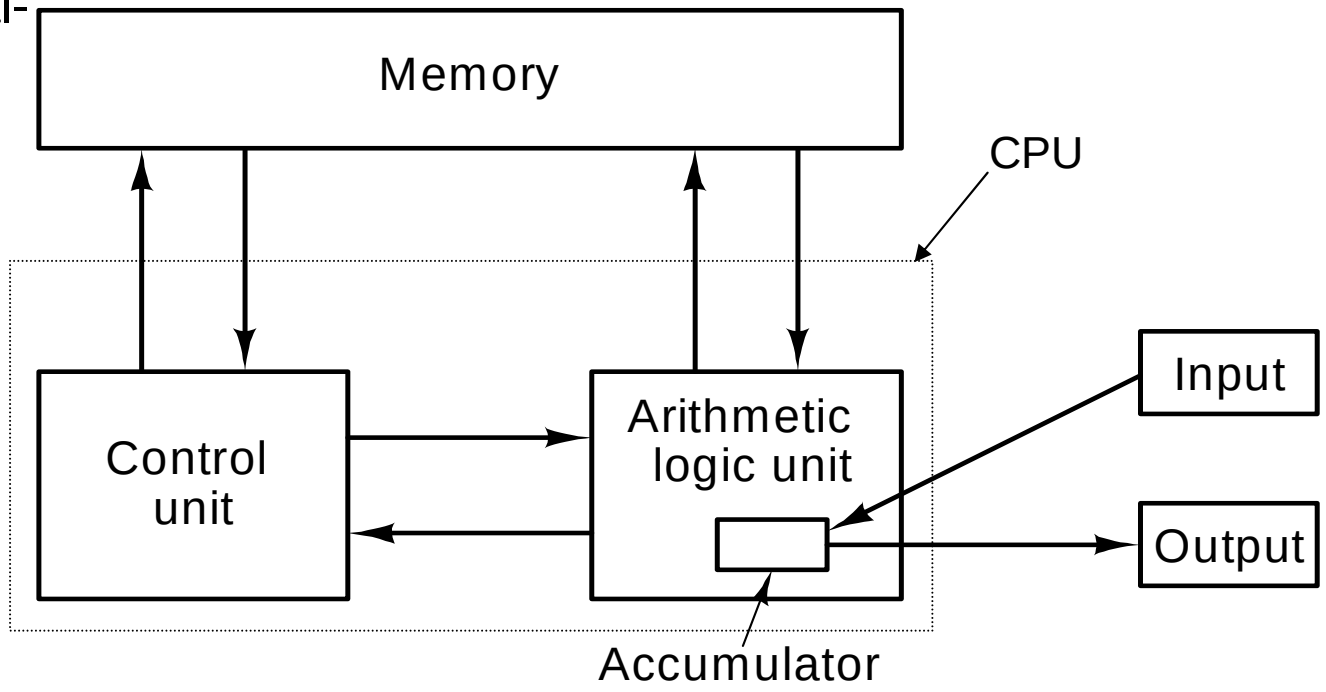


© 2017 Burkhard Stiller

von-Neumann Architecture (1)

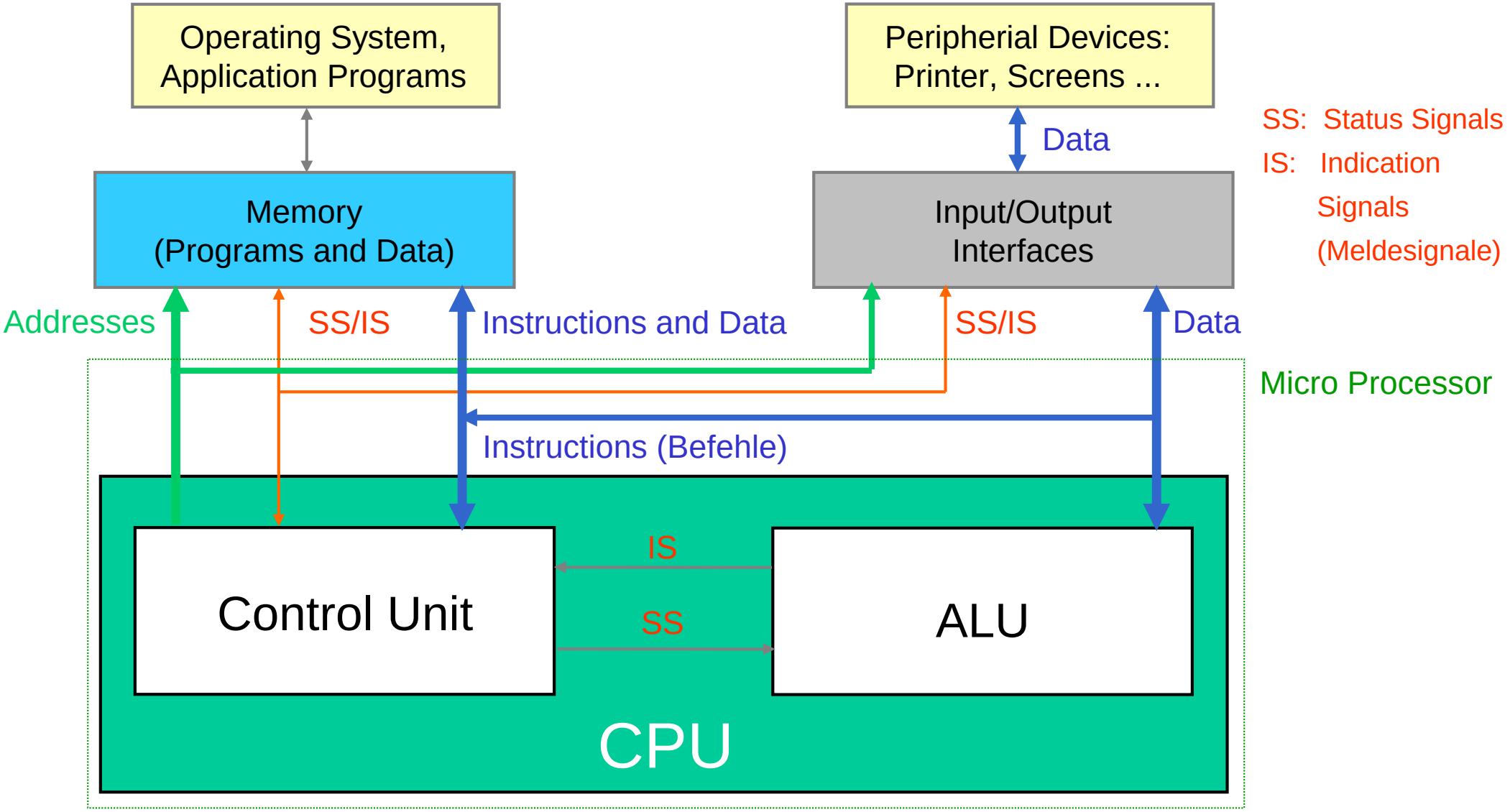
- The von-Neumann architecture defines the basics of basically all hardware architectures as of today

- This architecture comprises the following major components:
  - Central Processing Unit (CPU, Rechner or Zentral-einheit)
    - Control Unit (Steuerungseinheit or Leitwerk)
    - Arithmetic Logic Unit (ALU, Rechenwerk)
  - Memory (Speicher)
  - Input/Output Units (Ein/Ausgabeeinheiten)



© 2017 Burkhard Stiller

von-Neumann Architecture (2)



Modul 5: Computer Engineering

- von-Neumann Architecture
- Set-up and Operations
  - Mainboard
  - Registers and Stack
  - Instructions
  - Memory
- Organization
- Periphery

© 2017 Burkhard Stiller

Mainboard

- The mainboard (or motherboard) of a computer carries all components on a (printed circuit) board (Platine):
  - Micro processor (a dedicated instance of a CPU): Operations of programs, control and management of hardware
  - RAM memory (Arbeitsspeicher) stores programs currently in operation and respective data
  - ROM memory maintains programms (such as the *BIOS* for PCs), which check available hardware and boots the operating system of the harddrive, a CD/DVD, or a memory stick
  - Busses and interfaces enable the communication between all components of the main board and the access to the periphery, devices such as graphic cards, network cards, hard drives, printers
  - The chip set maintains all functionality of the mainboard, which is always required. It includes all sequential circuits for the CPU, the ALU, the input-/output interfaces, the control of those, and the access to various memory and storage units

© 2017 Burkhard Stiller

Micro Processor

- Micro processors (µP) are implemented as sequential circuits
  - Originally designed with a few hundreds of transistors, todays instances show millions of transitors at an extremely high integration density (still increasing)
- Micro processor components:
  - ALU for mathematical and logical operations
  - Control unit operates with two dedicated registers, the instruction pointer (Befehlszählerregister) and the instruction register (Befehlsregister) to control the operation of programs, maintains stack pointer (Kellerzeiger)
    - *Instruction table* (Befehlstabellen) decodes machine instruction code (Maschinenbefehl) of porgrams
  - Registers offer dedicated strorage within the processor, *e.g.*, an ALU operates on values in working registers, which exists in smaller numnbers, 2-8 per processor
  - Busses *(Datenleitungen)* interconnect the processor with its components:
    - *Data bus*: Exchange of data with RAM memory (Arbeitsspeicher)
    - *Address bus:* Feeding relevant memory data (addresses)
    - *Control bus:* Activates peripheral devices

© 2017 Burkhard Stiller

Register

- Register
  - Processor-internal storage for binary data of fixed length (*e.g.*, 32 bit)
  - Show physically short distances to control unit and ALU
  - Only with data stored in registers any operations are possible

- Types
  - Working registers are identified by IDs, such as D1 … D7 or EAX … EDX
    - Depending on processor type data registers (Datenregister) or address registers (Adreßregister) are distinguished
  - The instruction pointer (Befehlszählerregister) contains the address of the next instruction to be executed
  - The instruction register stores a single, binary machine instruction code
  - The stack register stores the processor‘s state and the instruction pointer
  - The flag register (Statusregister) contains dedicated execution states

© 2017 Burkhard Stiller

Stack (Kellerspeicher)

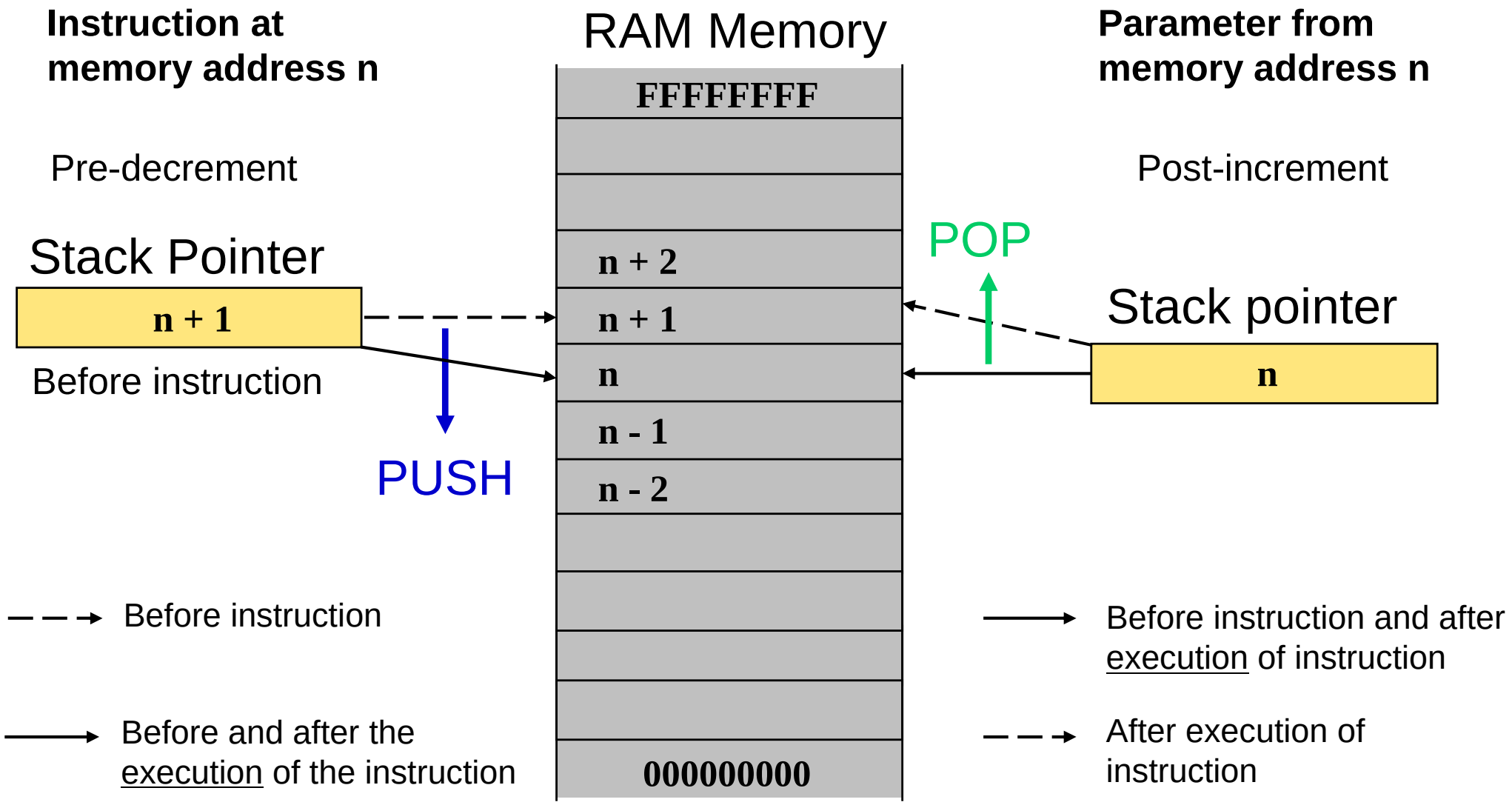
- A dedicated storage, typically located in the RAM memory, operates as a (software) stack (Keller), which means the the LIFO principle (Last-in-first-out) is applied for storing/fetching data.

- Function:
  - Processor state and instruction pointer are saved just before a sub-routine or an interrupt routine is called
  - Parameters are actively stored
  - Data may be stored during execution of instructions

- Modern processors operate typicaly with multiple (software or even hardware) stack:
  - System Stack, User Stack, Data Stack

© 2017 Burkhard Stiller

Operations of a Stack

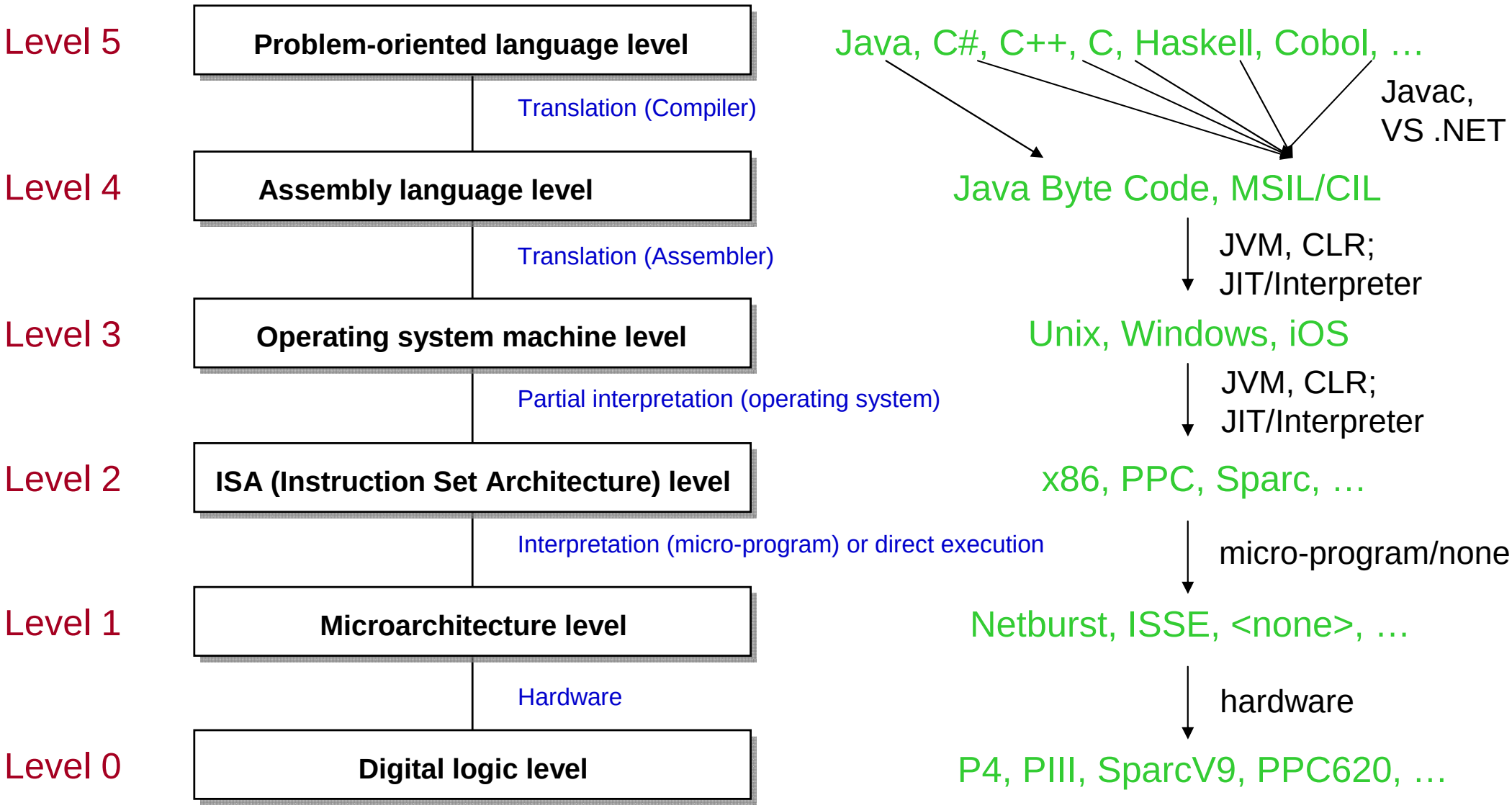


Operations of a Processor

1. The instruction counter of the control unit contains the address of the next machine instruction.
2. The address of this instruction will be received from the address register via the address bus.
3. The instruction will be transferred from the RAM memory to the instruction register.
4. The decodingt logic analyzes this instruction‘s content and starts its execution.
5. During the execution of this instruction, additional data may be read from the RAM memory, indication signals may be issued to the periphery, a calculation of the ALU will be started, or a program counter‘s jump will be indicated. The state of each operation will become availabe in the flag register (Statusregister).
6. In case of a jump, the instruction pointer will be adjusted to the new address, otherwise the instruction pointer will be increased by 1.
7. The processor continues with step 1 above.

© 2017 Burkhard Stiller

General Six Level Model of Instructions



© 2017 Burkhard Stiller

Machine Instruction Codes (1)

- Binary machine instruction codes (Binäre Maschinenbefehle) are processed from the instruction pointer, but are not “readable by humans”
- Thus, a symbolic language for machine instruction codes was defined to simplify programming
  - Symbolic machine instruction codes are termed Assembler instructions
- Almost every type of processor runs a different decoder logic, different machine instruction codes exist.
  - For the program execution Assembler instructions need to be translated into machine instruction codes of the target processor
  - For each processor type automated assemblers (translation programs) are available, which transfer the binary machine instruction code from its symbolic Assembler representation such that the processor can interpret them correctly

Assembler code	Machine code	Comment
`add edX, 0Fh`	`83 C2 0F`	**Add 15 to the content of the EDX register**

© 2017 Burkhard Stiller

Machine Instruction Codes (2)

- Examples for symbolic machine instruction codes (Assembler instructions):

  - **`MOV BX, $7A35`**

    Move from the RAM memory that value located at the address 7A35 to the working register BX
  - **`ADD BX, 20`**

    Add the value 20 to the current content of the working register BX
  - **`CMP BX, 50`**

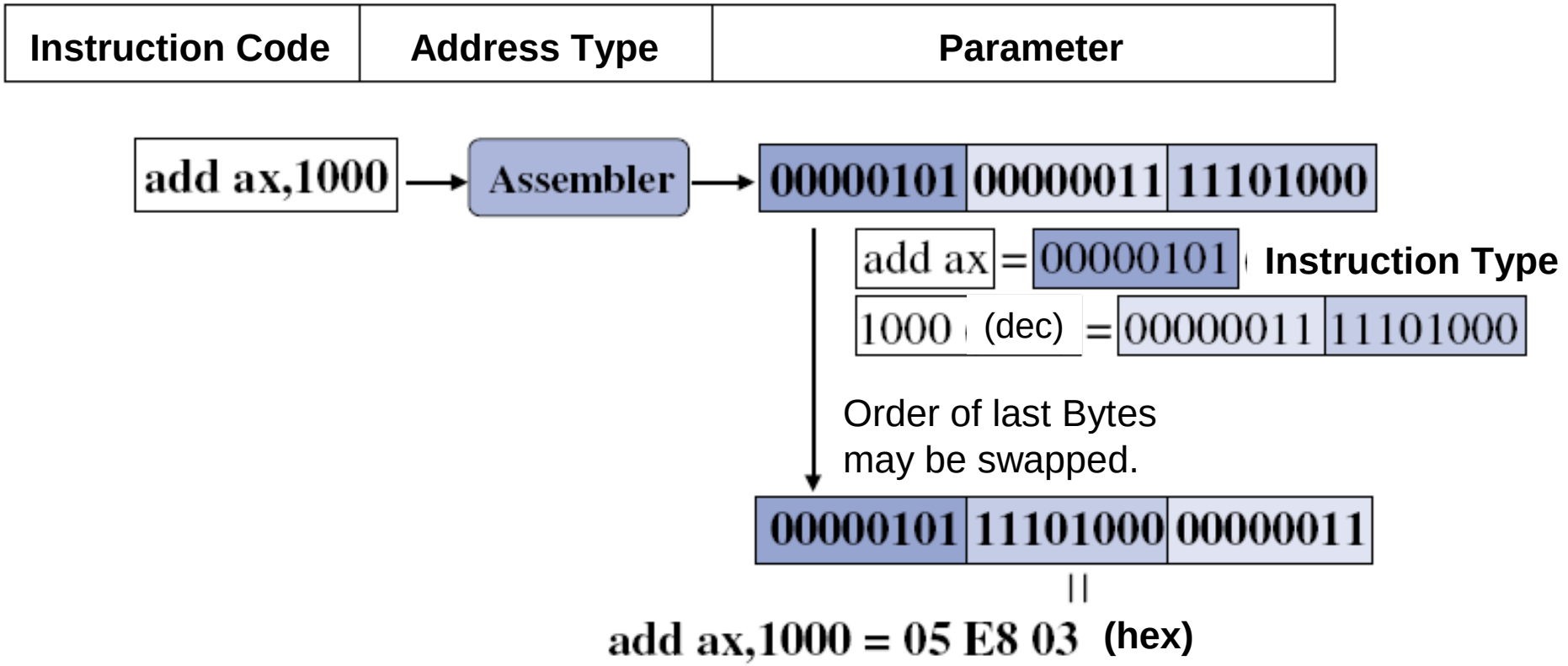
    Compare the value in register BX with 50, iff BX contains the value 50, the compare flag within the processor‘s status register will be set
  - **`JE $B7F4`**

    In case of an “equal” result of the preceding comparison (the flag in the processor’s status register was set), jump to program address B7F4
    JE defines the semantics of “jump if equal” (“Springe zu …, wenn gleich”).

© 2017 Burkhard Stiller

Machine Instruction Codes (3)

- Syntax and content of machine instruction codes vary depending on processor types and processor vendors. However, they follow common grounds and principles:



Machine Instruction Codes (4)

- Thus machine instruction codes typically contain:
  - The instruction (OP CODE)
  - A parameter (Operand) with the information of the address type
  - A value for one parameter or an address
- Each and every bit defines a dedicated bearing, which instructs the control unit to start the appropriate action
- Depending on the type of the instruction, these may show different lengths and a different number of parameters

© 2017 Burkhard Stiller

Data Types

Pentium II	8 bit	16 bit	32 bit	64 bit	128 bit
Signed integer	x	x	x		
Unsigned integer	x	x	x		
Float			x	x	

UltraSPARC II	8 bit	16 bit	32 bit	64 bit	128 bit
Signed integer	x	x	x	x	
Unsigned integer	x	x	x	x	
Float			x	x	x

JVM	8 bit	16 bit	32 bit	64 bit	128 bit
Signed integer	x	x	x	x	
Unsigned integer					
Float			x	x	

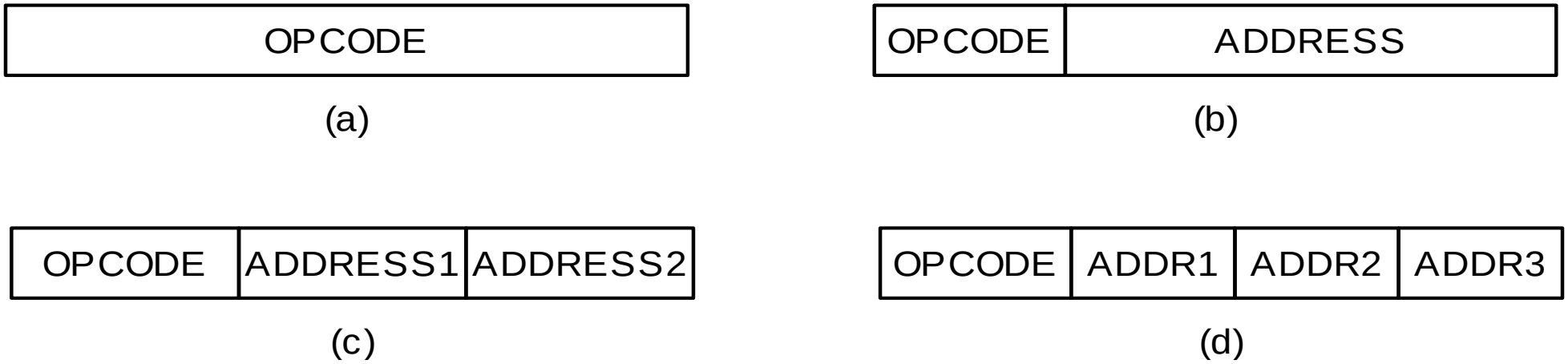
© 2017 Burkhard Stiller

Instruction Types

- Arithmetic and logic instruction enable calculations and logical decisions to be taken by the program
  - ADD, INC
- Jump instructions enable, while maintaining correct logical or arithmetical constraints, deviations from the linear processing of a program
  - JMP, CALL
  - In combination with arithmetical and logical instructions these jumps and calls determine the “intelligence” of a machine
- Transport instructions enable the transport of data between processor, RAM memory, and input/output units
  - MOV
- Processor control instructions prioritize tasks are applied to internally organize and maintain state of a processor

© 2017 Burkhard Stiller

Basic Instruction Types



- Example:
  C:= A + B

a) "Zero address" instruction
   - Stack operations: `push A; push B; ADD; pop C`

b) "One address" instruction
   - The accumulator defines the parameter and result: `load A; ADD B; st C`

c) "Two address" instruction
   - A parameter mutates to the result: `ADD B,A; move A,C`

d) "Three address" instruction
   - C := A + B: `ADD C,A,B`

© 2017 Burkhard Stiller

Examples for Instruction Set Architectures

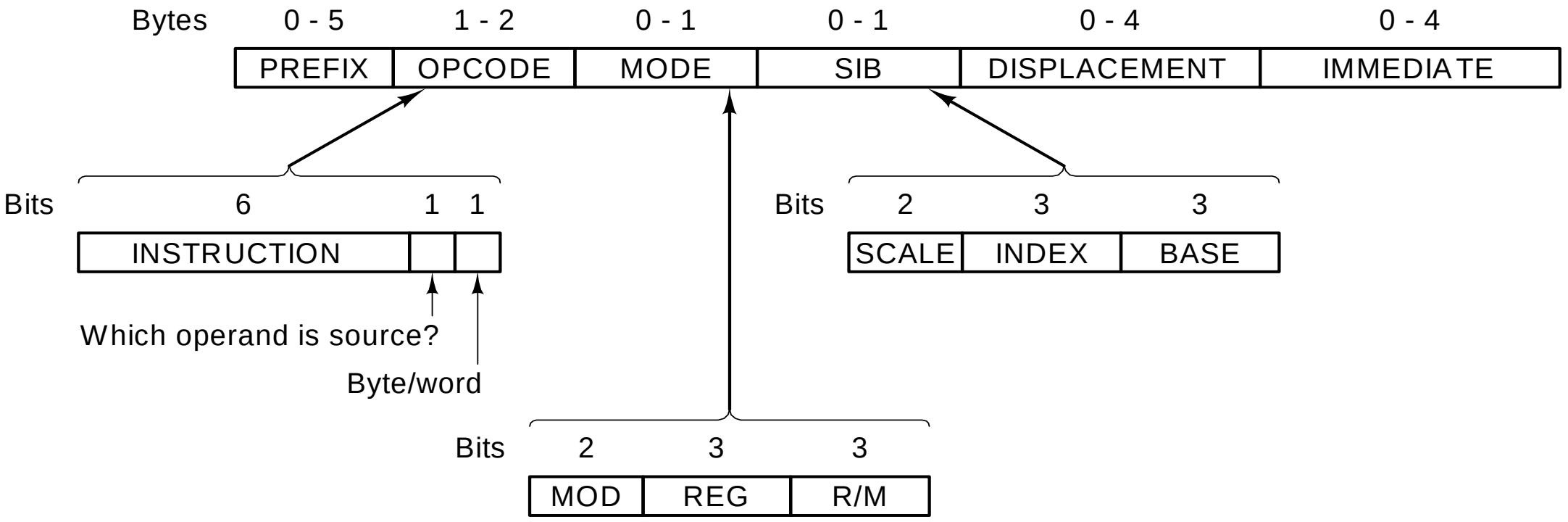
- Pentium:
  - Classical x86 CISC architecture (Common Instruction Set Computer)
  - CISC to the outside, but RISC (Reduced Instruction Set Computer) internally
  - Other CISC examples: Athlon and many older processors (VAX, IBM, ...)

- UltraSPARC (Ultra Scalable Processor Architecture):
  - Originates from early RISC projects (such as for the MIPS processor)
  - Extended RISC
  - Examples available for former SUN machines and industrial control systems
  - Other RISC examples: Alpha, MIPS, Power, PowerPC

- JVM (Java Virtual Machine):
  - Either a virtual processor or real hardware (*e.g.*, picoJava)
  - Computer with a stack (all instructions are executed in the stack)
  - Here: JVM heavily dependent on the Java programming language
  - Other examples for virtual machines: CLR, P-Code

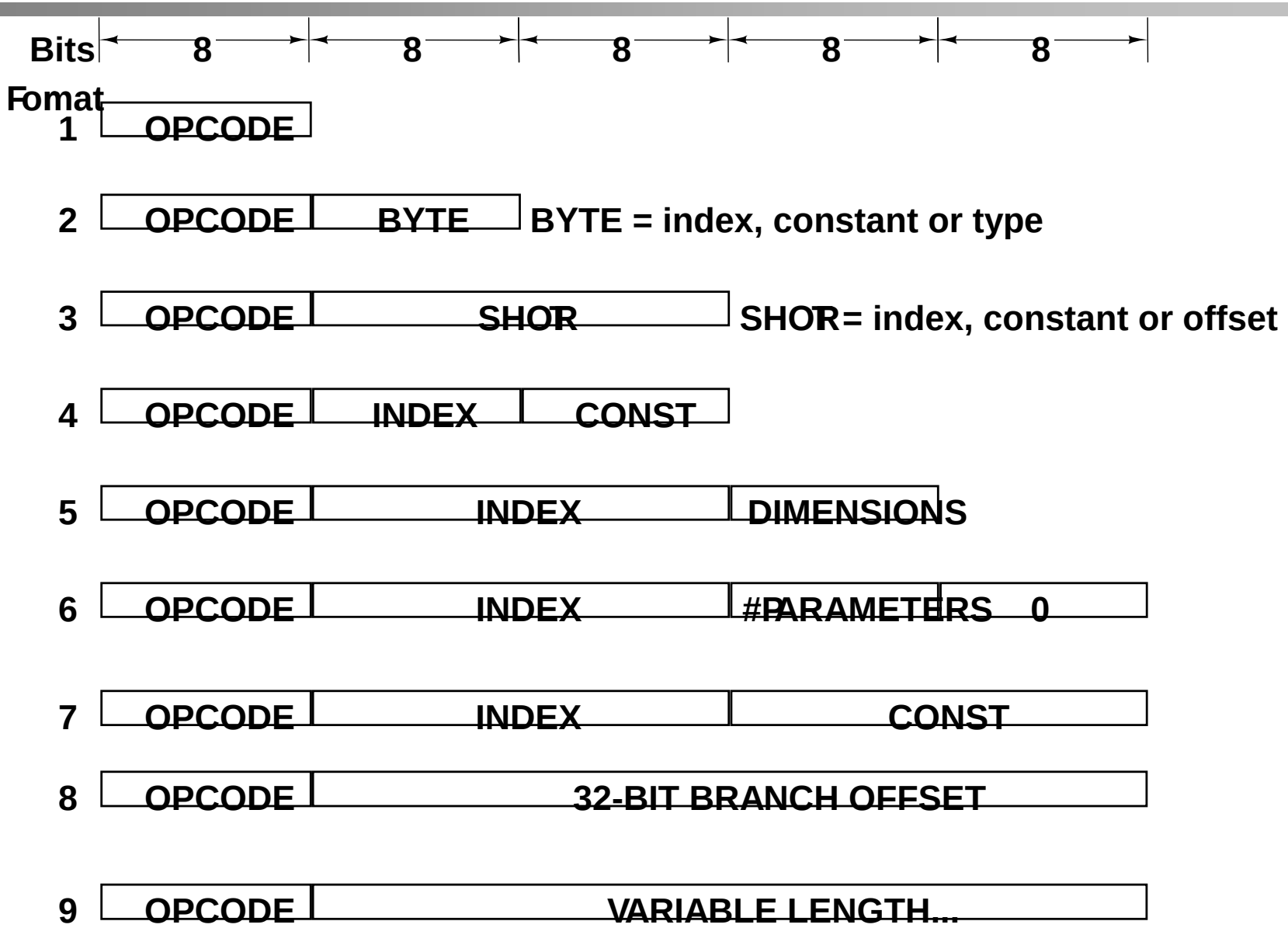
© 2017 Burkhard Stiller

Pentium II Instruction Set Architecture



© 2017 Burkhard Stiller

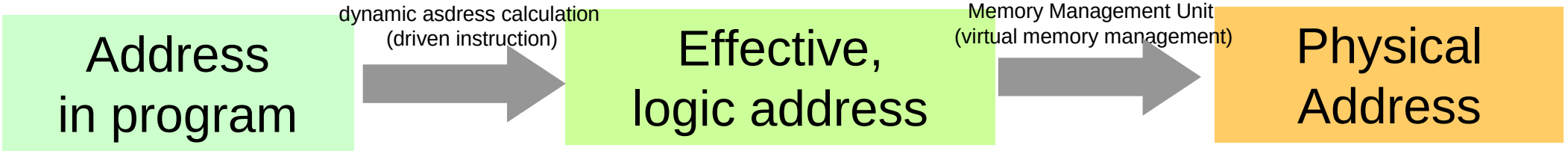
JVM Instruction Set Architecture



© 2017 Burkhard Stiller

Addressing Types

- Different addressing types offer alternatives of a processor to determine the address of a parameter or a jump
- Early processor generations:
  - Address of parameter and jump had been defined in the instruction absolutely
- Drawbacks:
  - Absolute addresses have to be known at the time of programming
    → But, programs may be placed at different locations within the memory
  - In case of table accesses within memory, the address has to be changed within the instruction
    → No read-only memory (Festwertspeicher) applicable as program memory!
- Today's processors generation:
  - Address is calculated at run-time (Laufzeit) – dynamic addressing



© 2017 Burkhard Stiller

Addressing Types — Overview (1)

- Register addressing
  - Implicit X — Flag
  - Explicit X
- One step memory addressing
  - Immediate — absolute
  - Direct — Zero page
  - Register-indirect — Page register
  - Indexed — Memory-relative, Register-relative, Indexed register-relative
  - Program counter-relative
- Two step memory addressing
  - Indirect, absolute
  - Indirect, register-indirect
  - Indirect indexed — Memory-relative, Register-relative X, Indexed register-relative
  - Indexed indirect
  - Indirect, program counter-relative X

© 2017 Burkhard Stiller

Addressing Types — Overview (2)

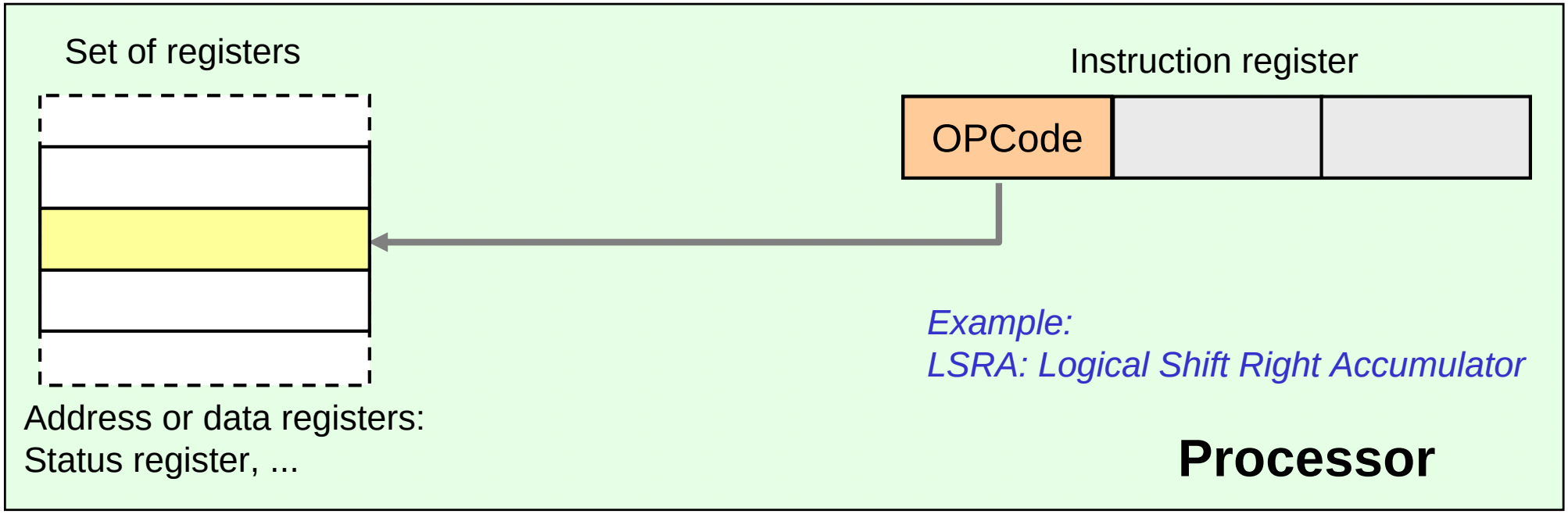
- Register-based addressing
  - Parameter already available in register
    - ➔ no memory access needed
- Single step memory addressing
  - One address calculation needed to determine the effective address, thus, to determine the address no multiple memory accesses required
- Two step memory addressing
  - Multiple, sequential address calculations and memory accesses needed
  - Result of first step determines the address of a memory cell, which contains an address (or an offset) to be used in the second step

© 2017 Burkhard Stiller

Implied Addressing

(implizite or inhärente Adressierung)

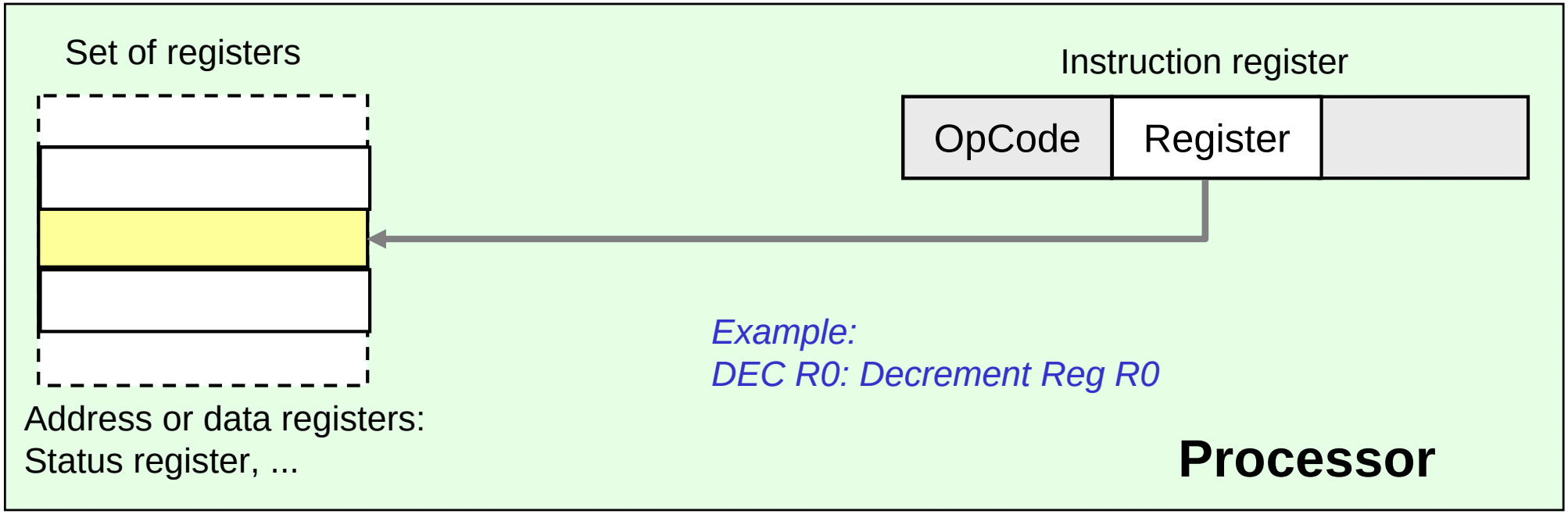
- The ID, that is the effective address of the required parameter of the addressed registers is directly encoded in the parameter‘s field (OpCode)
- Assembler representation: <Mnemo> A (A Accumulator)
- Effective address: EA it coded directly into the OpCode



Register Operand Addressing

(Explizite Register-Adressierung)

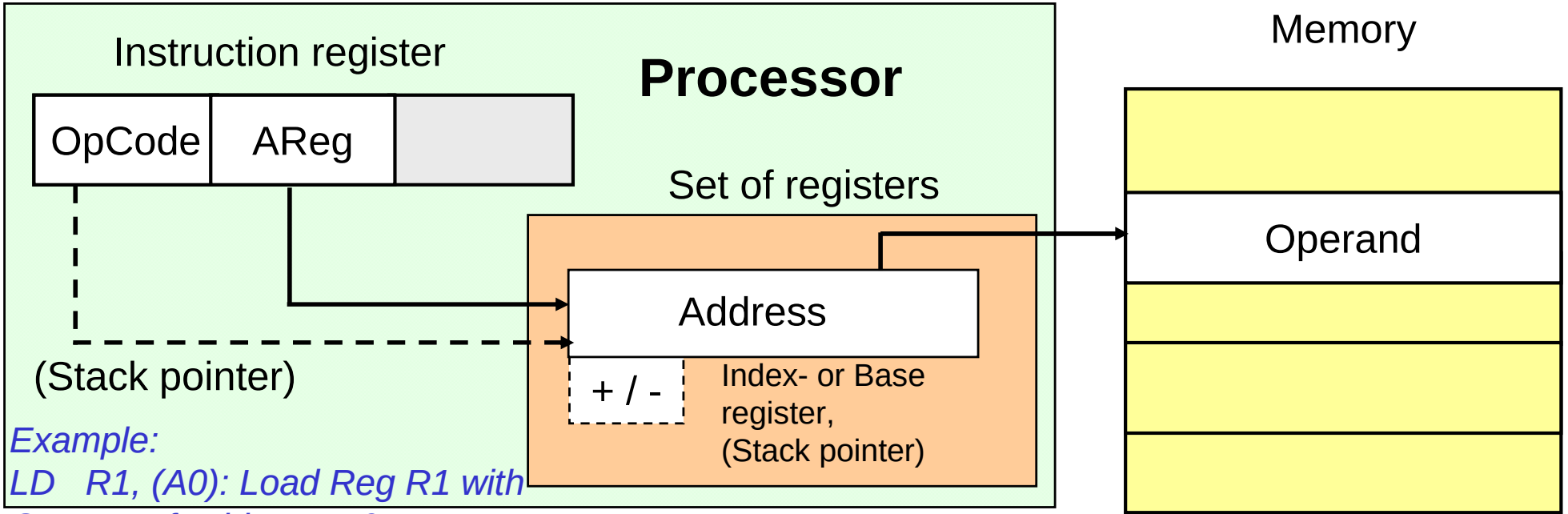
- The address (ID) of the register is part of the parameter‘s field
- Assembler representaion: <Mnemo> Ri (Register i)
- Effective address: EA = i



Register-indirect Addressing

(*Register-indirekte Adressierung or Zeigeradressierung*)

- The parameter‘s address is determined by the address register‘s ID, which is part of the OpCode‘s register field
- Assembler representaion: <Mnemo> (Ri)
- Effective address: EA = (Ri)

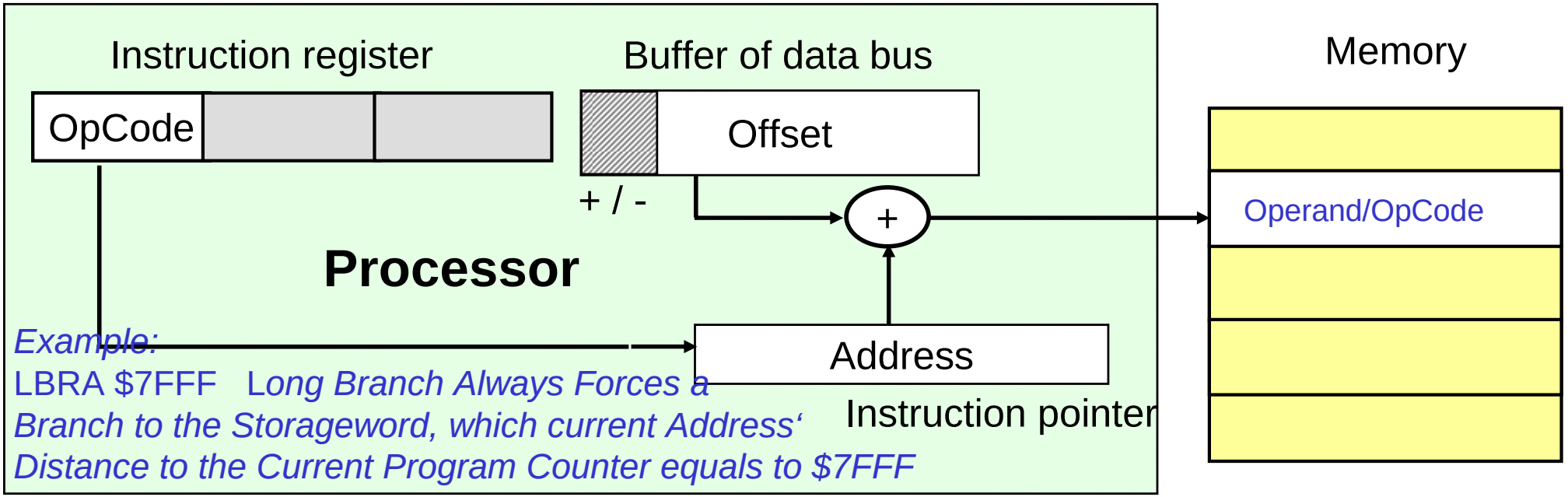


Example:
LD R1, (A0): Load Reg R1 with Content of Add Reg A0

Program Counter-relative Addressing

(Programmzähler-relative Adressierung)

- Effectiv address calculated by adding an offset from the current instruction to the current instruction pointer
- This addressing enables programs to be moved in main memory (Hauptspeicher) at any available position
- Assembler representaion: <Mnemo> <Offset>(PC)
- Effective address : EA = (PC) + 2 + ((PC) + 1)



© 2017 Burkhard Stiller

ifi

Comparison of Major Addressing Types

Type	Pentium II	UltraSPARC II	JVM
Immediate	x	x	x
Direct	x		
Register	x	x	
Register indirect	x		
Indexed	x	x	x
Based-indexed		x	
Stack			x

© 2017 Burkhard Stiller

Memory

- Two alternative memory types exist within a machine:
  - RAM memory (Arbeitsspeicher) and mass storage (Massenspeicher)
- RAM memory:
  - The *memory* of a machine
  - Stores programs and data, which are accessible at any time, immediately
- Permanent storage of data → Long-term memory
    - ROM, non-volatile, *e.g.*, for Operating System Kernel, system tables
- Vorübergehender Ablage von Daten → Short-term memory
    - Read-write RAM, volatile, *e.g.*, for applications and user-programs
- Technology:
  - Today: solid state memory
    - Fast, small, reliable, cheap (≈ 6-12 CHF/GByte for DRAM)
  - Past: toroidal core memory (Ringkernspeicher, Magnetspeicher), tubes, relays:
    - Slow, large, unreliable, expensive, *e.g.*, ¼ MByte memory for IBM/360-75 was in 1967 at 1,6 Million CHF

© 2017 Burkhard Stiller

Terms and Definitions (1)

- Storage element (Speicherelement):
  - 1 Bit memory
- Memory cell (Speicherzelle, -platz, -stelle):
  - Feste Anzahl von Speicherelementen, durch eine einzige Adresse ausgewählt, z.B. 8, 16, 32 Bit
- Storage word (Speicherwort):
  - Maximum number of storage elements, which can be transmitted within a given bus cycle between a µP and its memory ➔ Widths of storage word = Width of data bus
- Direct access (wahlfreier Zugriff):
  - Each memory cell can be accessed directly (no need to check others before)
  - Selection accordig to address decoder‘s result: 1-out-of-n code
- Organization:
  - Organization of memory circuit determined by the number *n* of its memory cells and the number *m* of storage elements per cell

© 2017 Burkhard Stiller

Terms and Definitions (2)

- Typically represented as: n x m Bit

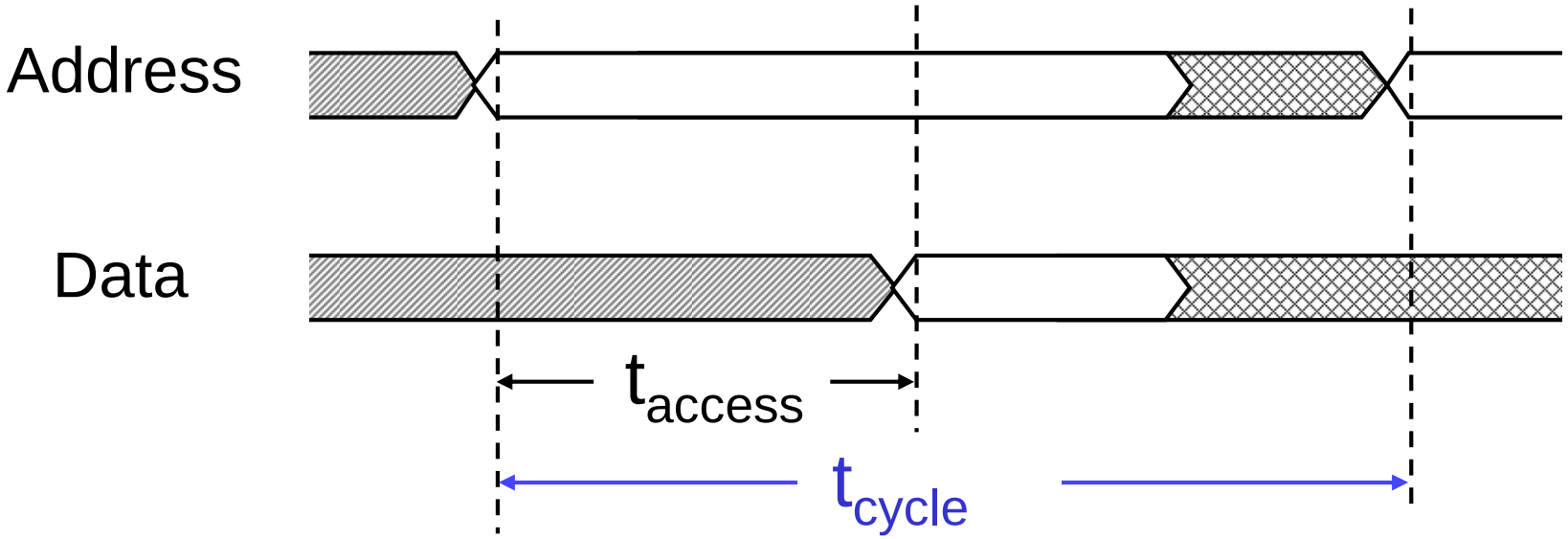
– Examples:
  - 4k x 8 Bit memory contains 4096 memory cell at 8 Bit each
  - Set-up with 8 4k x 1 Bit chipos or 2 4k x 4 chips or ...

- Capacity:
  – Information volume (measured in the number of bits), which can be stored within the memory: n x m Bit
- Majar parameters to characterize the speed of memory:
  – *Access time (*Zugriffszeit):
    - Maximum duration, which elapses after feeding the address to a memory and the availability of the respective data at its output lines
  – *Cycle time (*Zykluszeit):
    - Minimal duration, which has to elapse between two subsequent memory accesses driven by feeding the address to the memory

© 2017 Burkhard Stiller

Access Time and Cycle Time

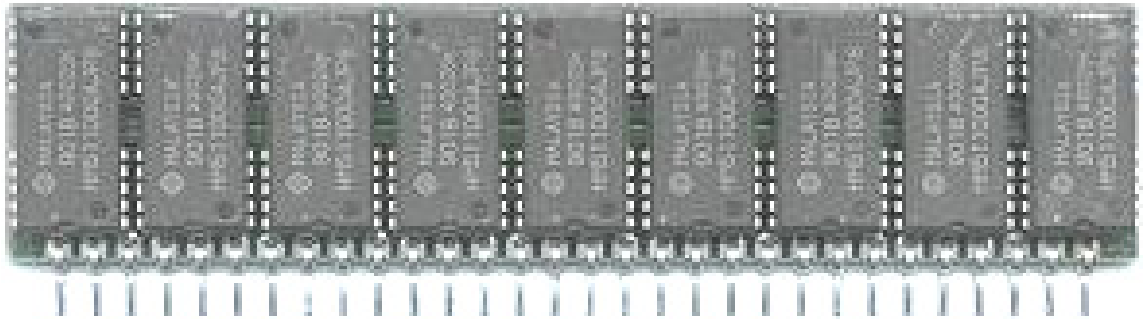
- The cycle time may be much longer then the access time!
- Reasons:
  - Memory cell needs to “recover” first from access
  - Some memory technologies destroy content at the read process such that a read always requires a write – here termed “refresh” – after an access
- Ideally: Cycle Time = Access Time
- Reality: (Often) Cycle Time > Access time (up to 80%)



© 2017 Burkhard Stiller

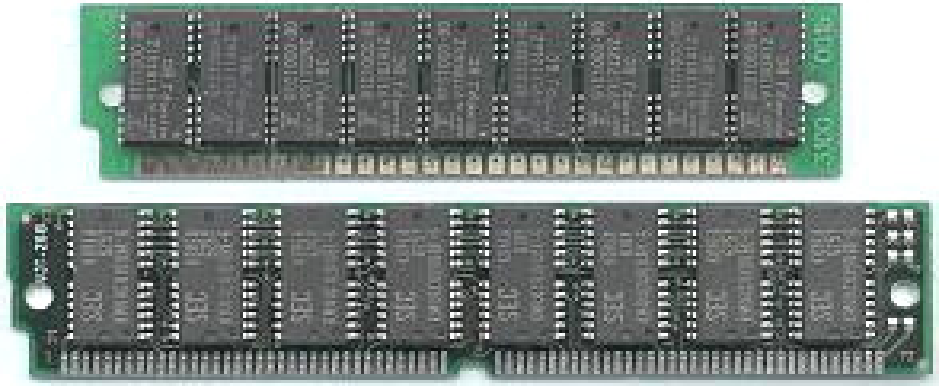
Memory Technology Alternatives (1)

- Single In-line Pin Package (SIPP): ***disappeared***



SIPP module in a 30 pinversion and a data bus width at 8 Bit

- Single In-line Memory Module (SIMM): PS/2 module ***disappeared***



SIMM module in 30- and 72-pin version with data bus width at 8 and 32 Bit

© 2017 Burkhard Stiller

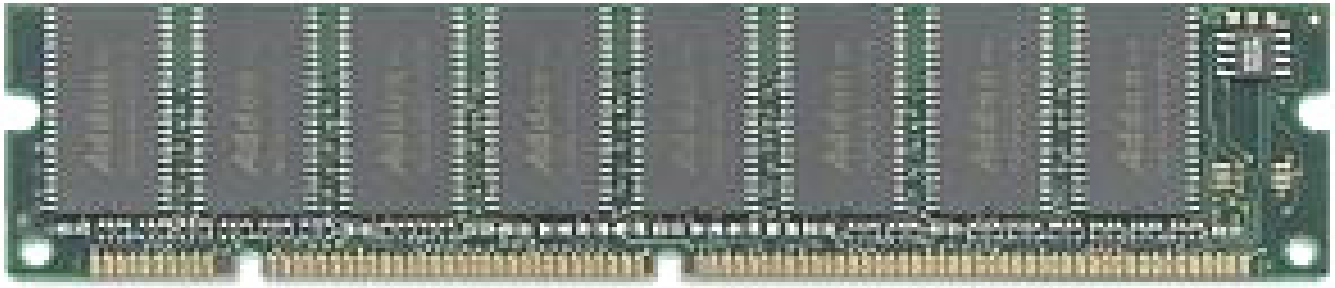
Memory Technology Alternatives (2)

- Rambus In-line Memory Modul (RIMM): **disappeared**



Memory technology by Intel. 400 bis 800 MHz with a metal cover to cool Rambus DRAMs

- Dual In-line Memory Module (DIMM):

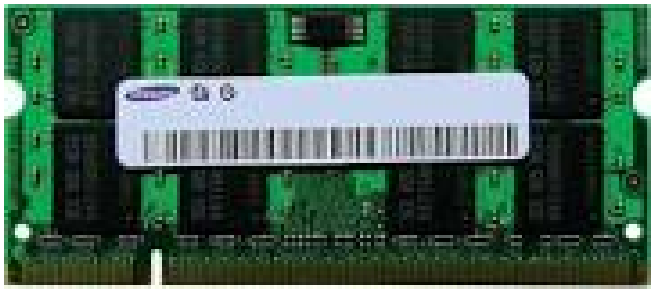


168 pin DIMMs offer a data bus width of 64 Bit; today, Double Data Rate (DDR4 SDRAM) at 1600 MHZ in use

© 2017 Burkhard Stiller

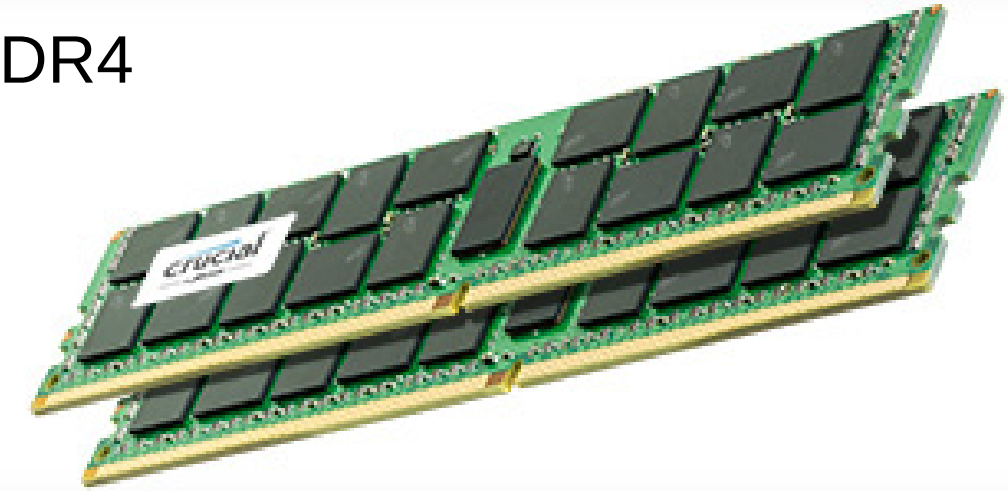
Memory Technology Alternatives (3)

- Double Data Rate (DDR) memory (since app. 2002):
  - Data transfer operated edge-triggered (flankengesteuert)
- DDR3



- 240 pins
- 333 MHz clock
- Max 2,7 Gbit/s
- 1,5 V operational voltage
- Refreshrate fixed

- DDR4

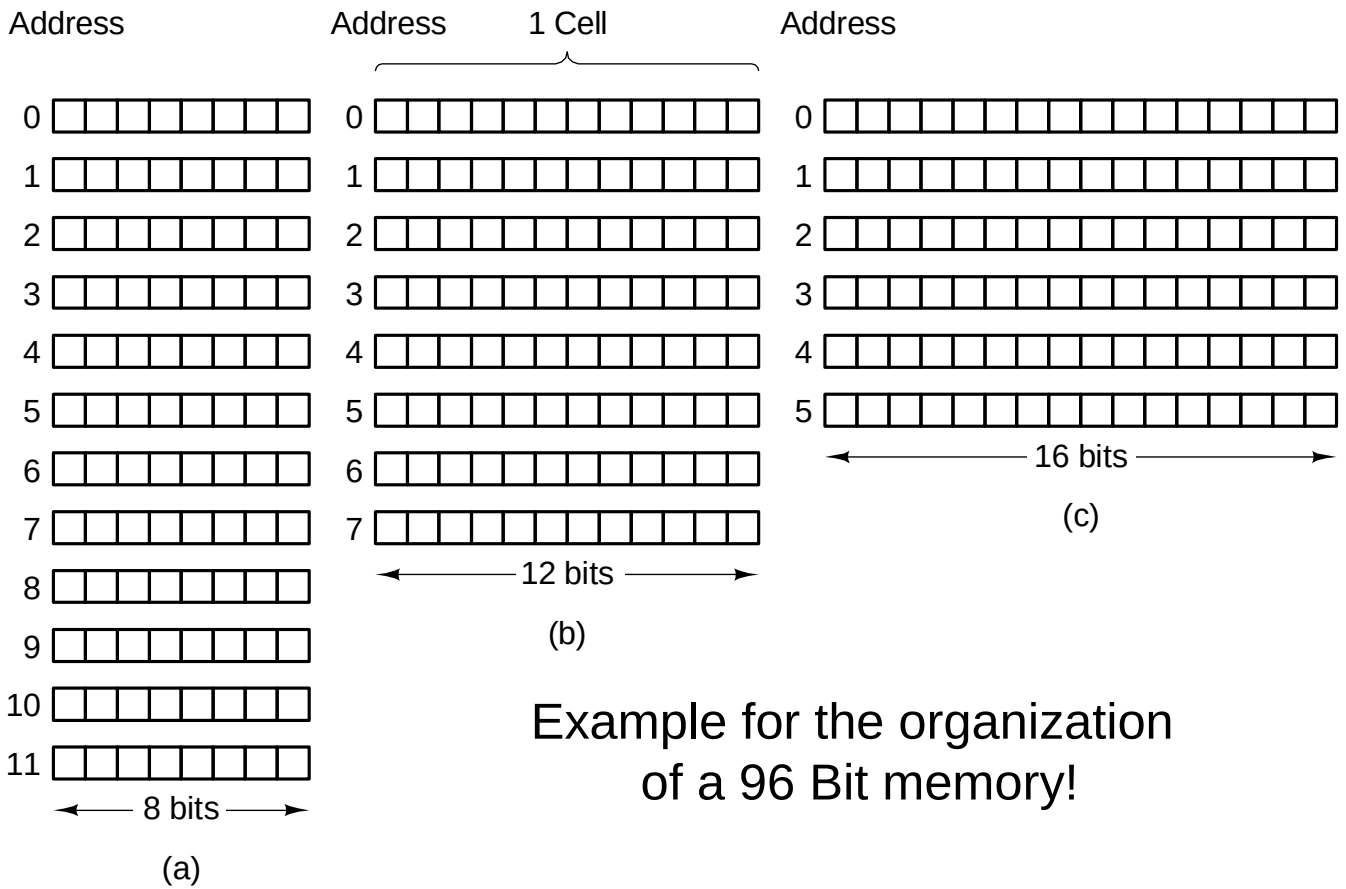


- 260 ... 288 pins
- 400 ...1600 MHz clock
- Max 3,2 MT/s (mega transfers)
- 1,2 V operational voltage
- Dynamic refresh rate:
  max all 64 ms (temperature)
- Embedded error detection
  and corretion

© 2017 Burkhard Stiller

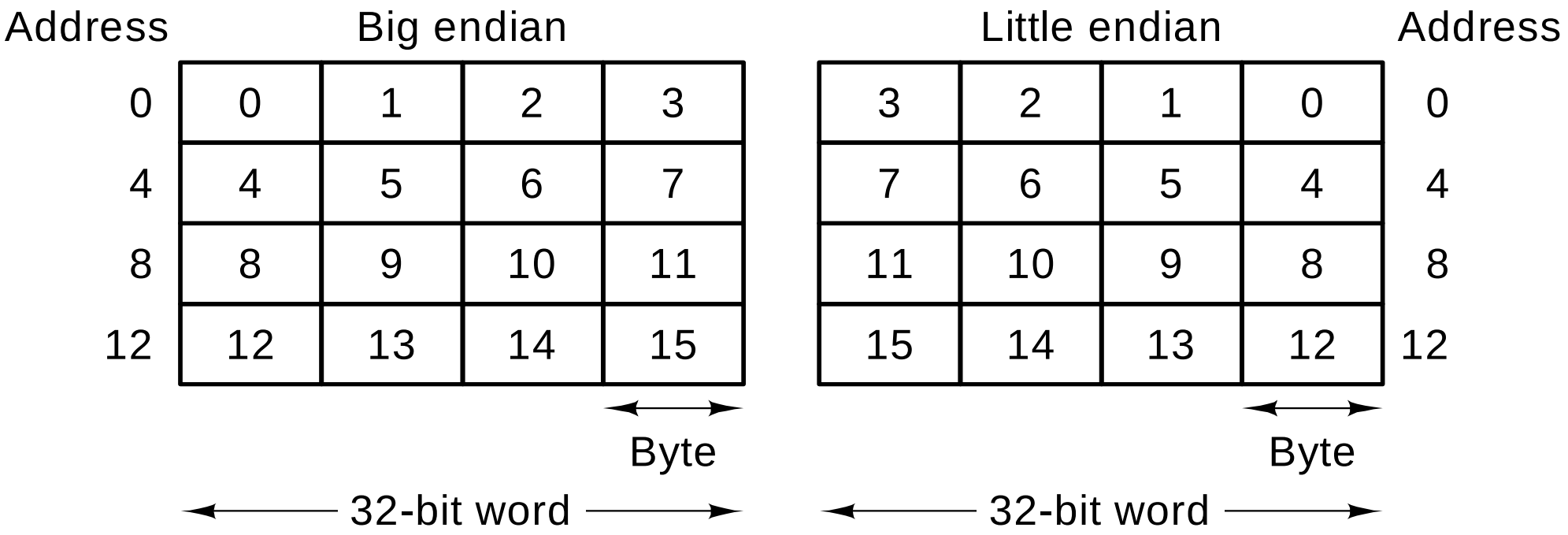
Organization of Main Meory

- Linear list of storage words
  - Implemented by memory chips
  - Access times depend on chips in use
  - Width of main memory typically the same as the width of the data bus (8, 16, 32, 64, or 128 Bit), which determines the maximum of information accessible within a single bus cycle



Example for the organization of a 96 Bit memory!

Representations — Big Endian vs. Little Endian



SPARC, IBM mainframes

Intel (thus the PC world)

Hours:Minute:Second

Day.Month.Year

Bulk Storage

- Comparing bulk storage is viable by distinguishing the physical read and write pocesses:
  - *Magnetic data storage*: Bits are representatd by magnetic areas with mit antithetic polarity: *E.g*., Hard Disks (HD), das Floppy Disk Drive (Diskettenlaufwerk), ZIP-/JAZ drives and streamers (Bandlaufwerk).
  - *Optical Data storage*: Daa are stored via reflecting metal surfaces, which is sampled from a laser, bits are represented in terms of pits *(*gebohrte Löcher) or unchanged land (nicht gebohrte Stellen)
  - *Magnetic-optical (MO) data storage:* A mix of magnetic and optical write and read operations is applied

- Technology:
  - Hard Disks (HD) are fast, small (3.5‘‘, 2.5‘‘, 1.8‘‘), quite reliable, cheap (app. 30 CHF/TByte for 3.5‘‘ HDs)
  - Solid State Disks (SSD) do not contain any rotating elements anymore, very fast, relatively expensive (app. 5-10 CHF/GByte for 2.5‘‘ SSDs)

© 2017 Burkhard Stiller

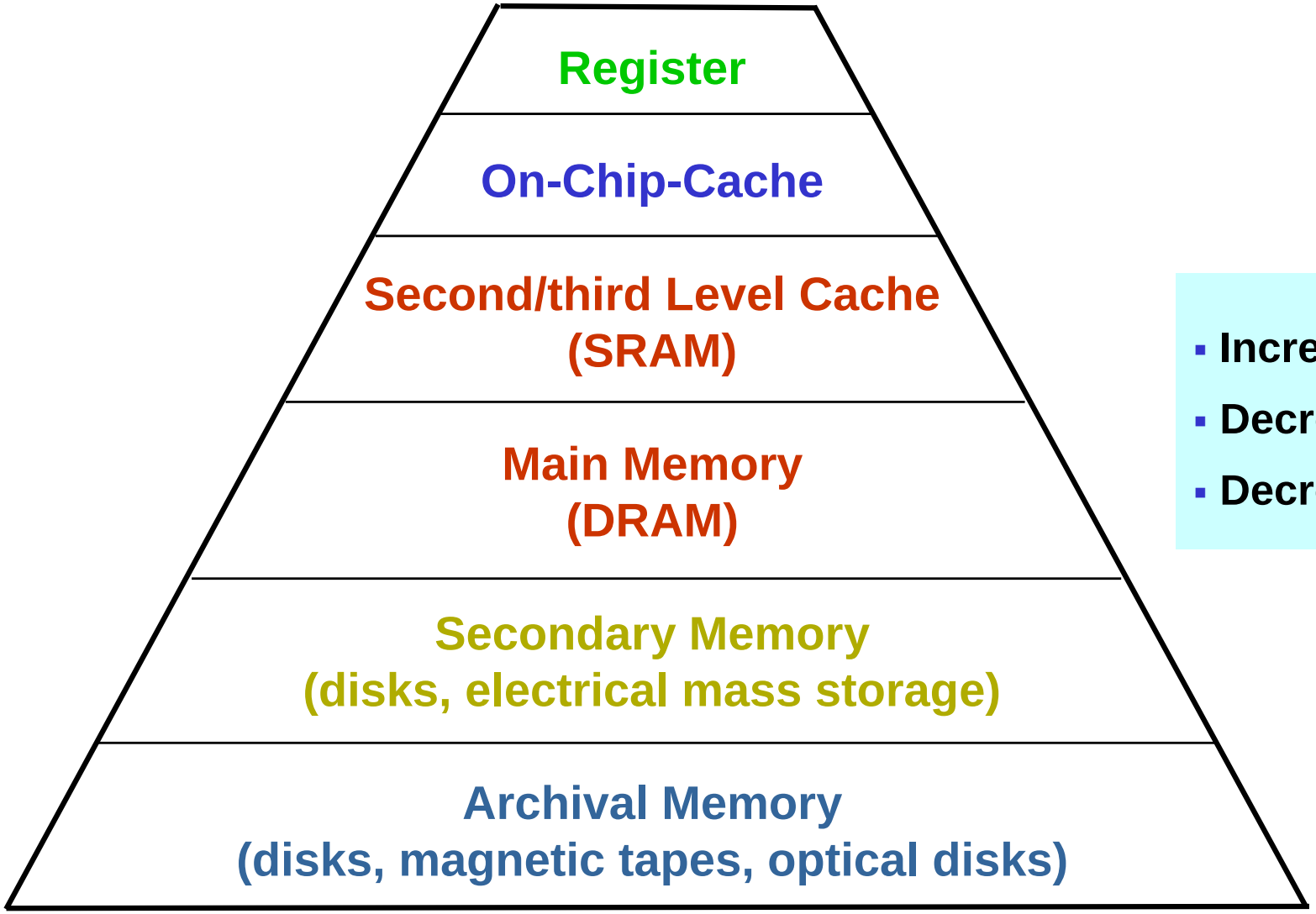
Storage Hierarchy (1)

- Optimium: A uniform storage with short access times and large capacity for all computer-related needs.
  - Problem: only the cost factor is considered to a limiting factor!

- Solution (in motion every year):
  - Layered approach of different storage technologies and copying and pasting of data between those layers → Storage hierarchy
  - Hierarchy balances the different accesss times of differet technologies such that storage close to the CPU is always the fastest
- However, two strategies are applicable:
  - Caches: Short access times
    → Improving the processor‘s access times to the data
  - Virtual memory:
    → Enlarging the existing main memory by a larger address space
    - *E.g.*, multiple processes may be executed at the same time

© 2017 Burkhard Stiller

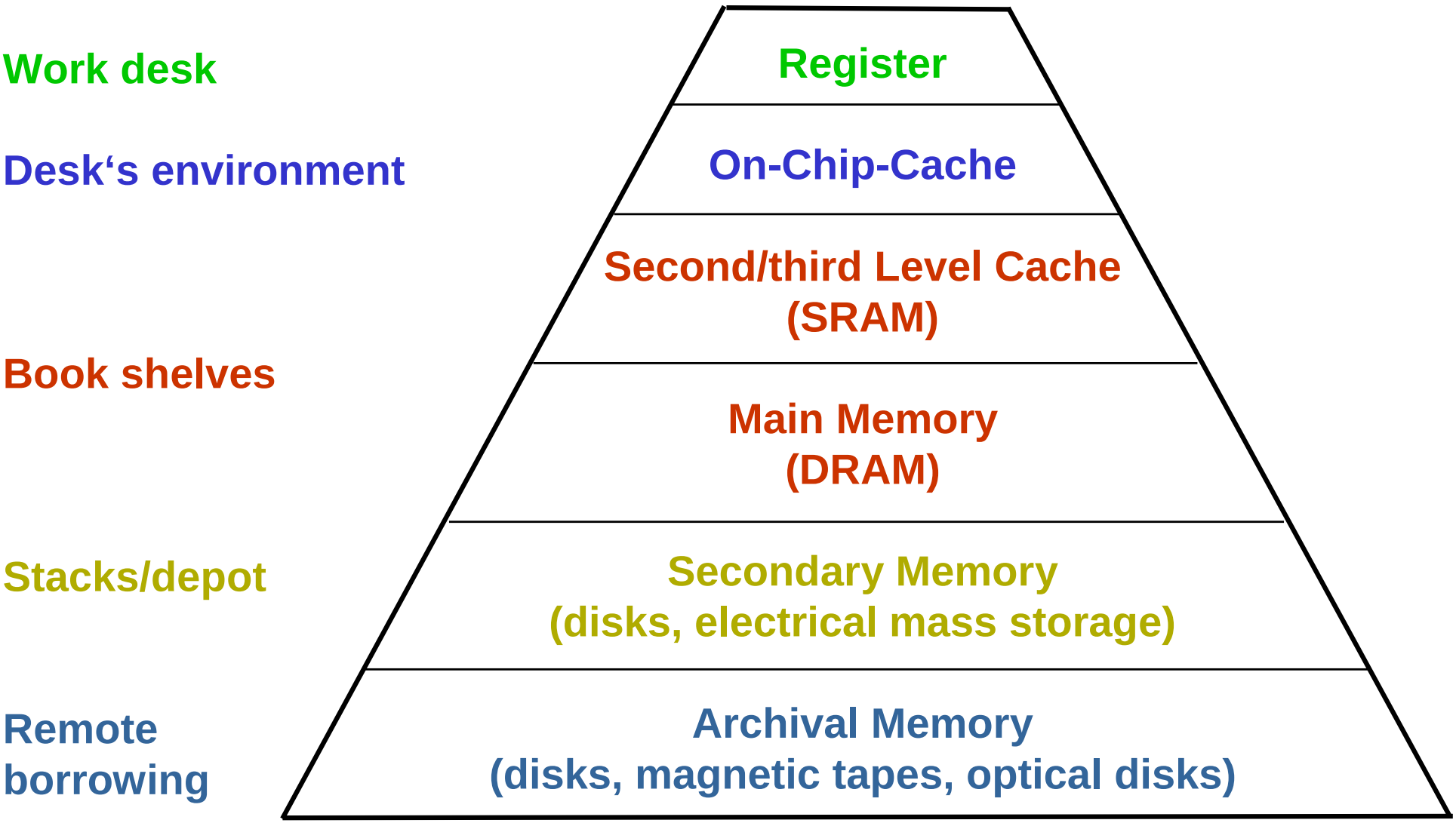
Storage Hierarchy (2)



- Increasing costs/Byte
- Decreasing capacity
- Decreasing access time

© 2017 Burkhard Stiller

Storage Hierarchy (3) — Comparison



© 2017 Burkhard Stiller

Cache — Problem, Idea, and Solution

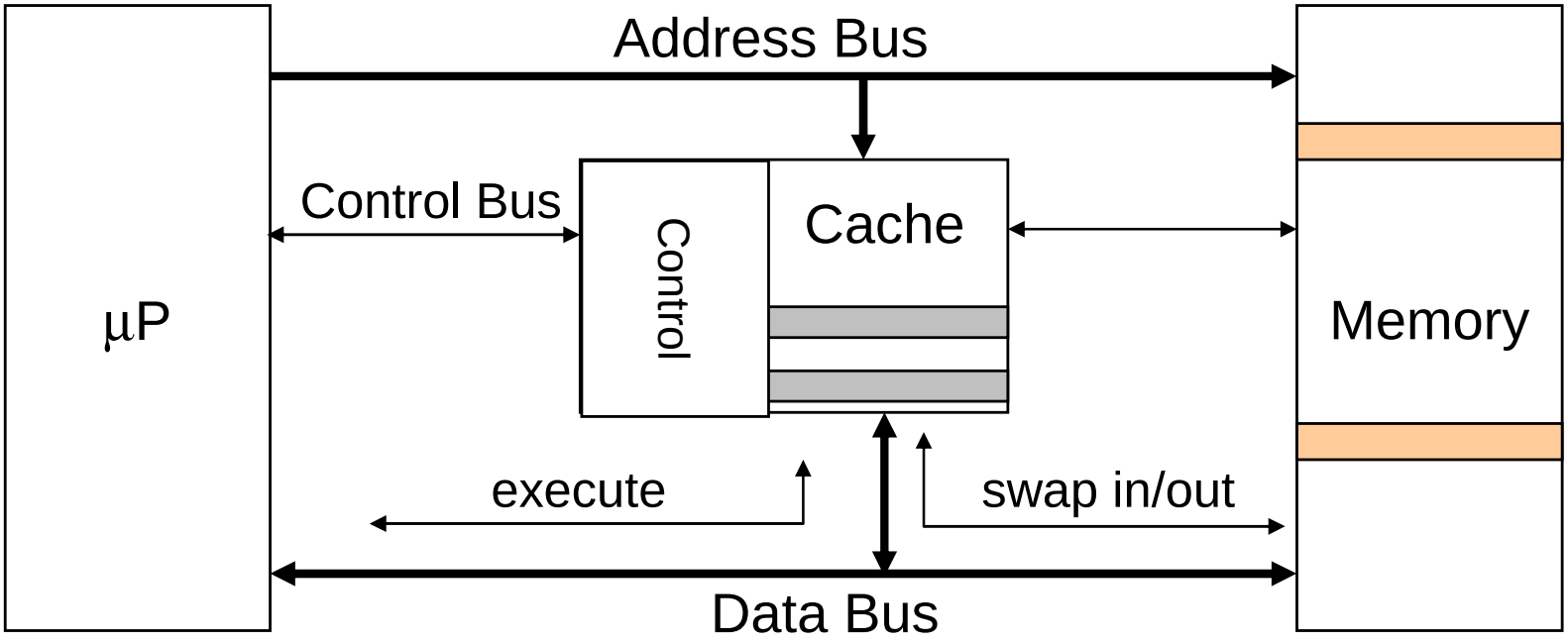
- Problem:
  - Cycle times of busses attached to modern processors is much shorter that the cycle time of large, cheaper DRAMs
  - This situation requires to introduce wait cycles
  - SRAMs can be operated without wait cycles, however, they are small, expensive, and more energy hungry with a larger dissipation power (Verlustleistung)

  → Thus, only smaller storage is implemented in that type
- Idea:
  - A small, but fast memory of SRAMs is located between the processor and the relatively slow, but much cheaper main memory (implemented in terms of DRAMs)
- Solution:
  - Caches (Cache-Speicher) are small, but fast buffer-style storage cells, which are located just before larger and slower storage, to increase with a certain likelihood the successfull access of data with a shorter access time.

© 2017 Burkhard Stiller

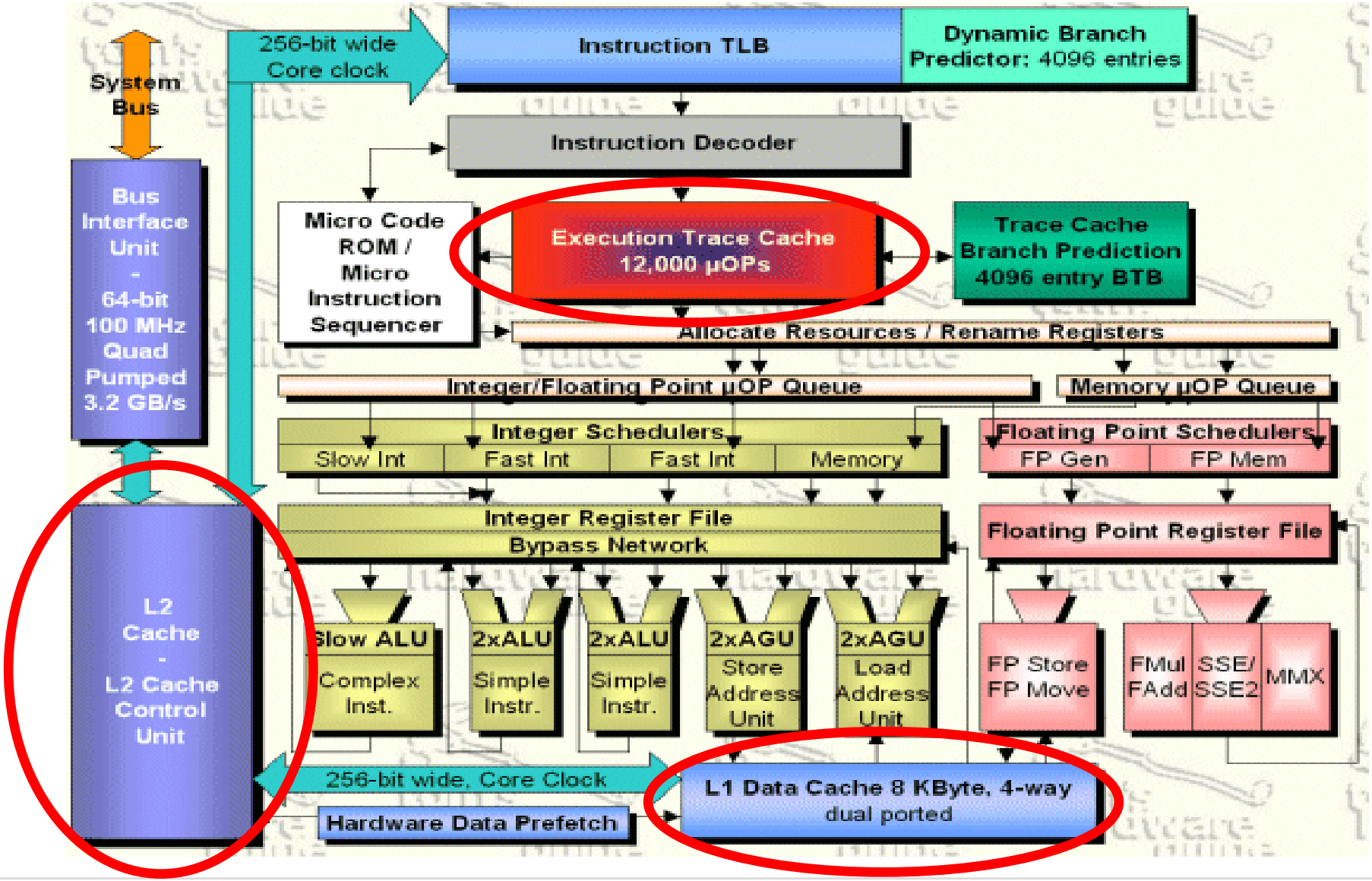
Principle of a Cache

- A buffer maintains copies of those parts of the main memory, which the CPU will access next with a high likelihood



  - The Cache may integrated already on the processor chip (on-chip cache) or
  - is implemented in the fastest and expsenive SRAM technology (off-chip cache).

Pentium 4 Caches



© 2017 Burkhard Stiller

ifi

Modul 5: Computer Engineering

- von-Neumann Architecture
- Set-up and Operations
- Organization
  - Components
  - Busses and Interfaces
- Periphery

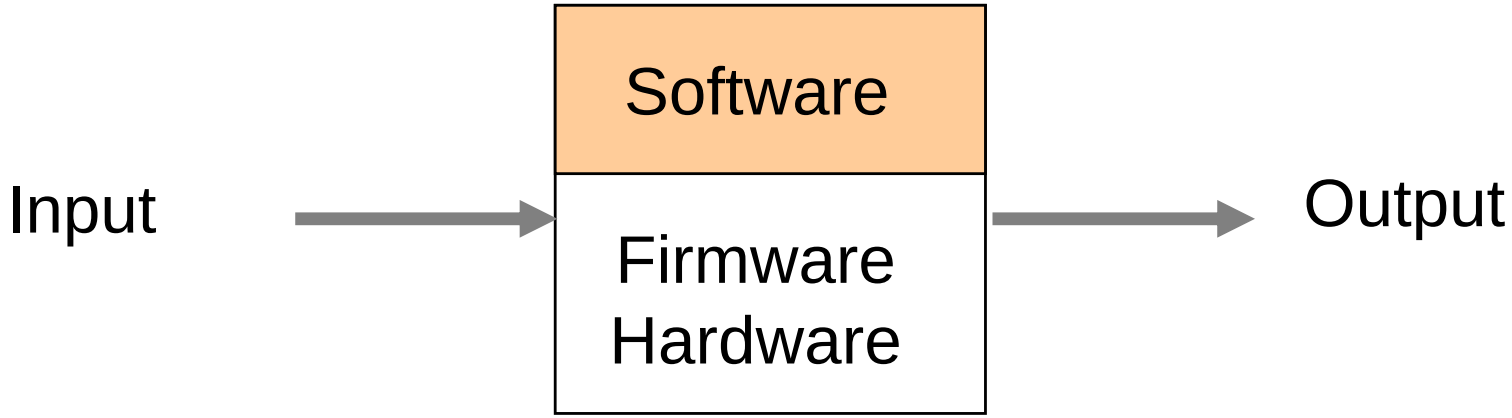
© 2017 Burkhard Stiller

Computer Components

- Hardware (HW)
  All mechanical and electronical units, elements, and chips

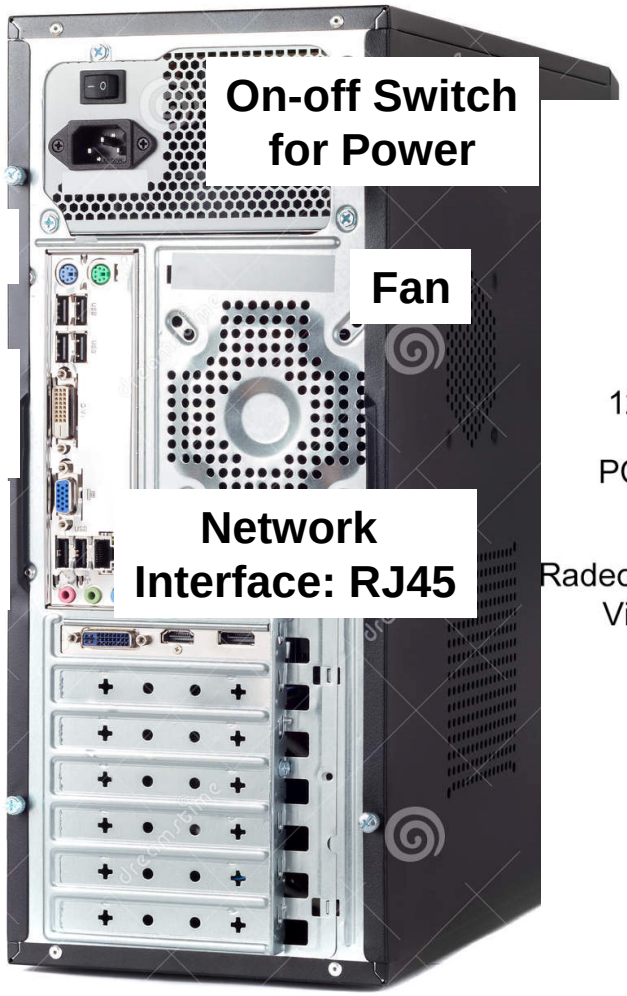
- Software (SW)
  All programs being able to be operated on the computer

- Firmware (FW)
  Microprograms (typically stored on ROMs), intermediary between hardware and software



© 2017 Burkhard Stiller

Personal Computer (PC)



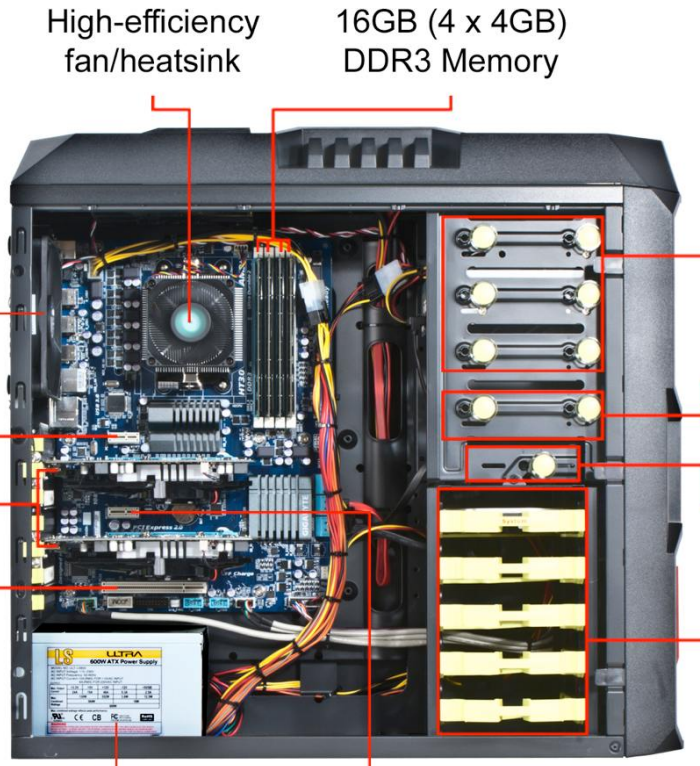
A/V Interfaces

USB 3.0

Screen Interface: HDMI

Screen Interface: VGA

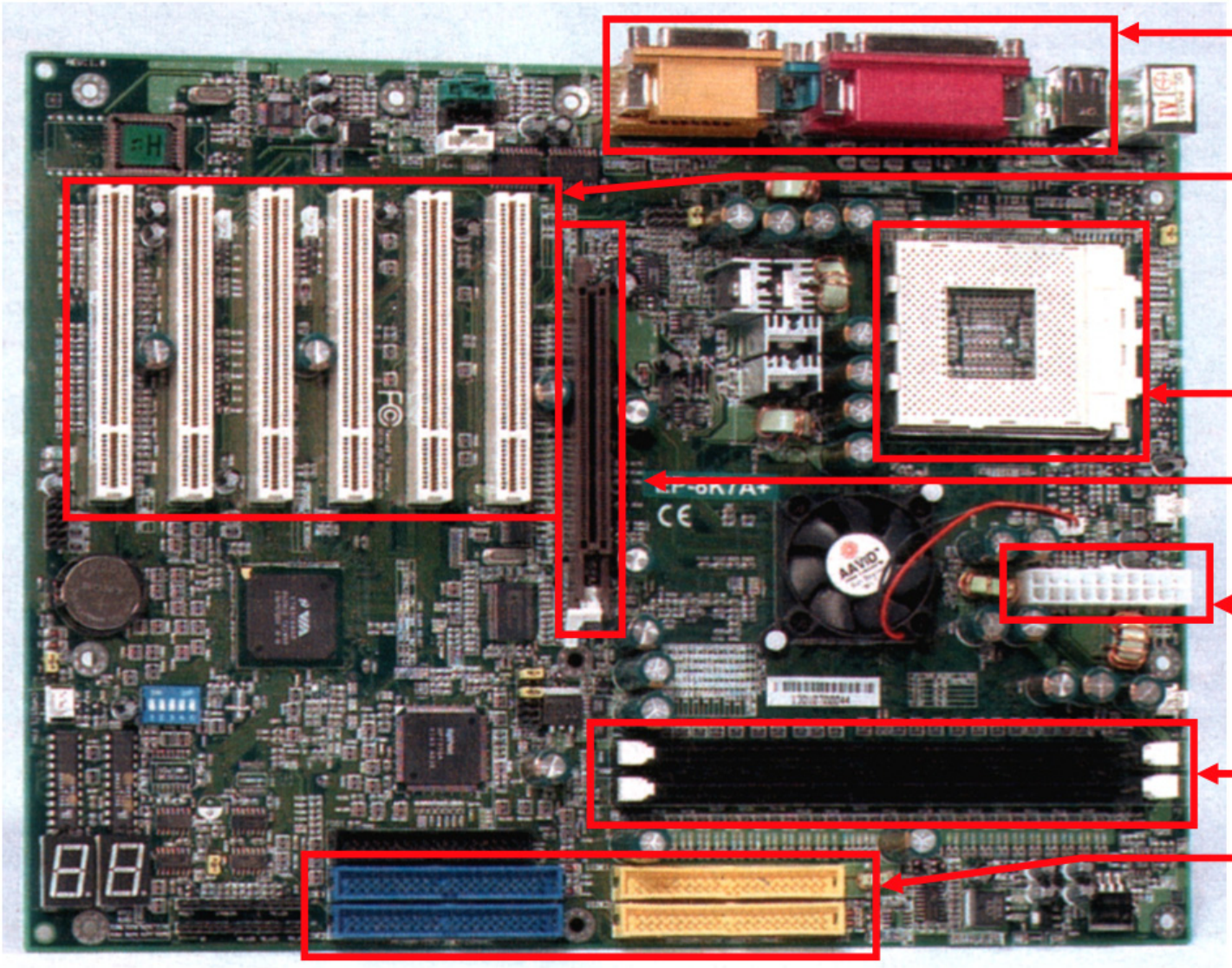
Empty Slots



© 2017 Burkhard Stiller

ifi

Motherboard/Mainboard



I/O Interfaces
(USB [Universal Serial Bus], seriell, parallel)

PCI slots (Peripheral Component Interconnect)

CPU Slot

AGP Slot
(Accelerated Graphics Port)

Energy Supply

Additional Memory Slots

EIDE Interface (Enhanced Intelligent/Integrated Drive Electronics): Disk, CD, DVD

© 2017 Burkhard Stiller

ifi

BIOS (Basic Input/Output System)

- Originally, the BIOS was implemented as a chip, which was located for IBM PCs on the mainboard, containing *firmware*, the key base of the control logic to boot a machine
  - Today, the BIOS is located in a flash PROM supported by a small and battery-buffered RAM, in which all configuration parameters and the system's time are stored, operational without external power as long as the battery works
  - Upon a computer start a dedicated program is executed in the BIOS to run various tests, displaying indication messages at the screen; afterwards the operating system is loaded into the main memory from the storage specified and the computer is started
- The operating system (SW) extends all HW and FW capabilities to a "useful" system according to user demand

© 2017 Burkhard Stiller

Main Sample BIOS Functions

- *POST – Power-On Self Test* (*Selbsttest beim Einschalten*):
  - At first, tests the key hardware components (RAM, graphics ...)
- *Simple Communications with Hardware*
  - Setting of systems clock
  - Booting from hard disk, CD, or USB stick
- *Control Handover to Data Storage*:
  - After a successful POST test, BIOS hands-over the control to the booting storage
  - The program stored within the *Master Boot Record MBR (Startsektor) of the booting drive will be loaded into main memory and executed.*

© 2017 Burkhard Stiller

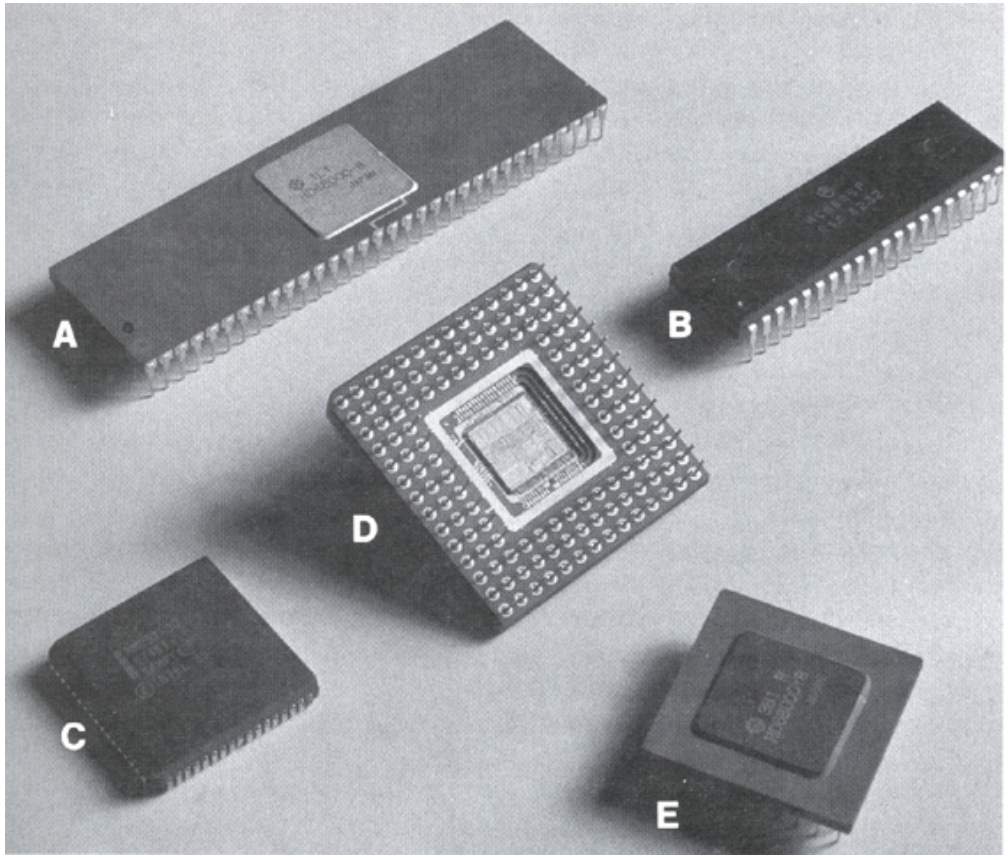
System Chip Set

- A Chip set determines the physical interconnection between all hardware components of a computer
- Functionally comparble chip sets of different vendors can easily vary in their performance (even above 10%)
- The chip set determines, which hardware components may be used and how they do have to communicate:
  - System bus
  - Storage type
  - Interfaces
  - Type of processor

© 2017 Burkhard Stiller

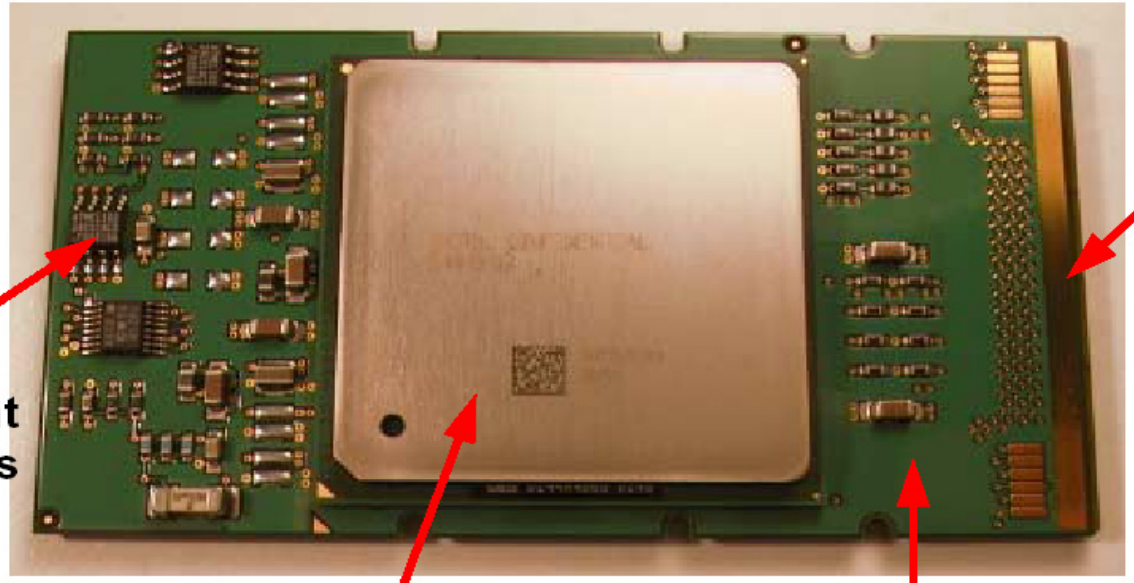
Processor, μP, and Chip Hardware

- Packages and electrical connections
  - Any chip needs to be embedded into a package such that is can be safeguarded against mechanical impact, that electrical connections can be provided reliably, and such that heat generated can be discharged (up to 150 W)
- Materials for packages:
  - Plastic
  - Ceramics
- Main packaging types
  - A to E



© 2017 Burkhard Stiller

Intel Itanium CPU Package and Xeon 5600 Series



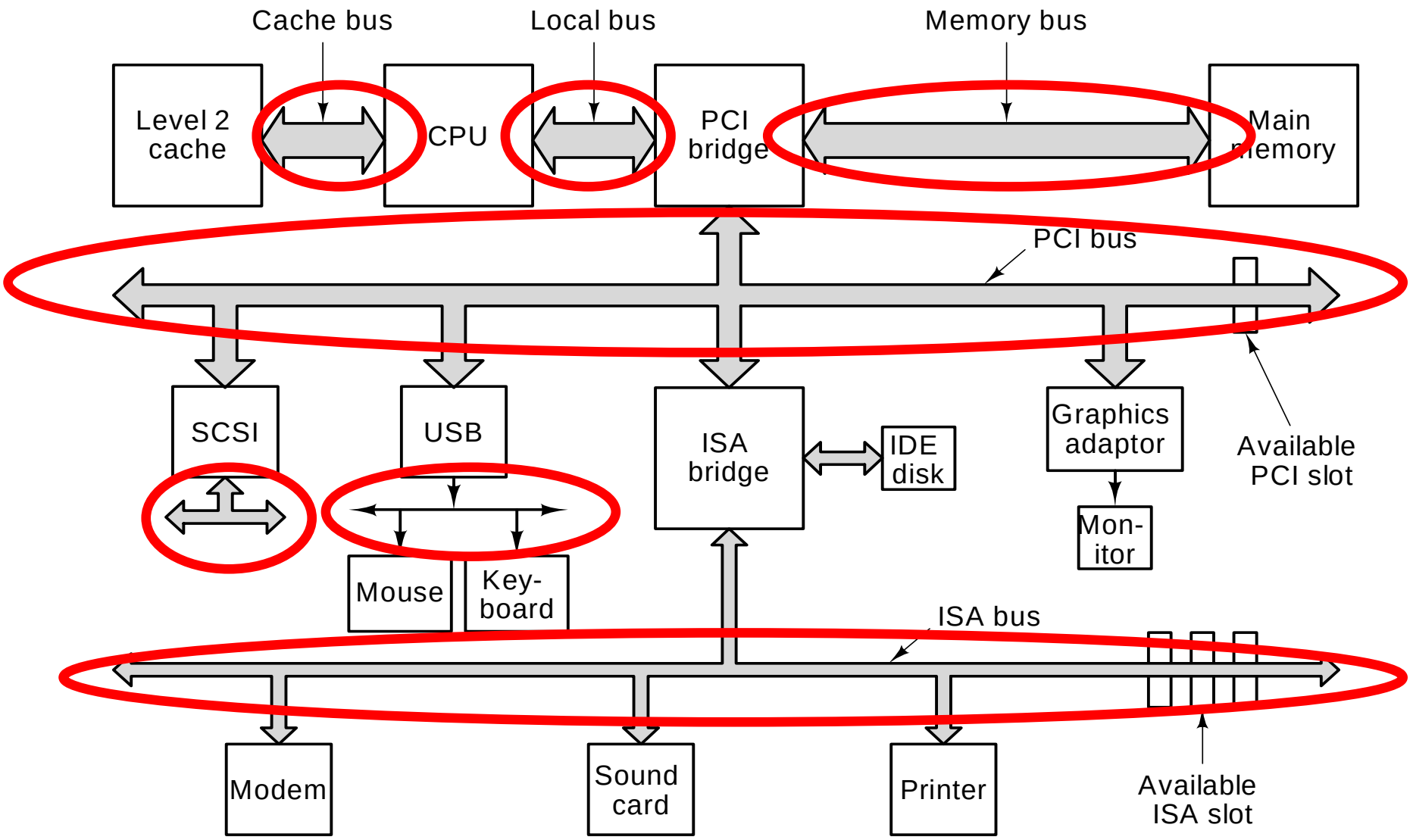
Introduction into the market: 2010/11

Main characteristics

- 1,6-3,6 GHz clock
- 4-6 cores
- 4-12 MB Cache
- 60-130 W dissipation power
- Costs: 195-1600 US$

© 2017 Burkhard Stiller

The Family of Buses



© 2017 Burkhard Stiller

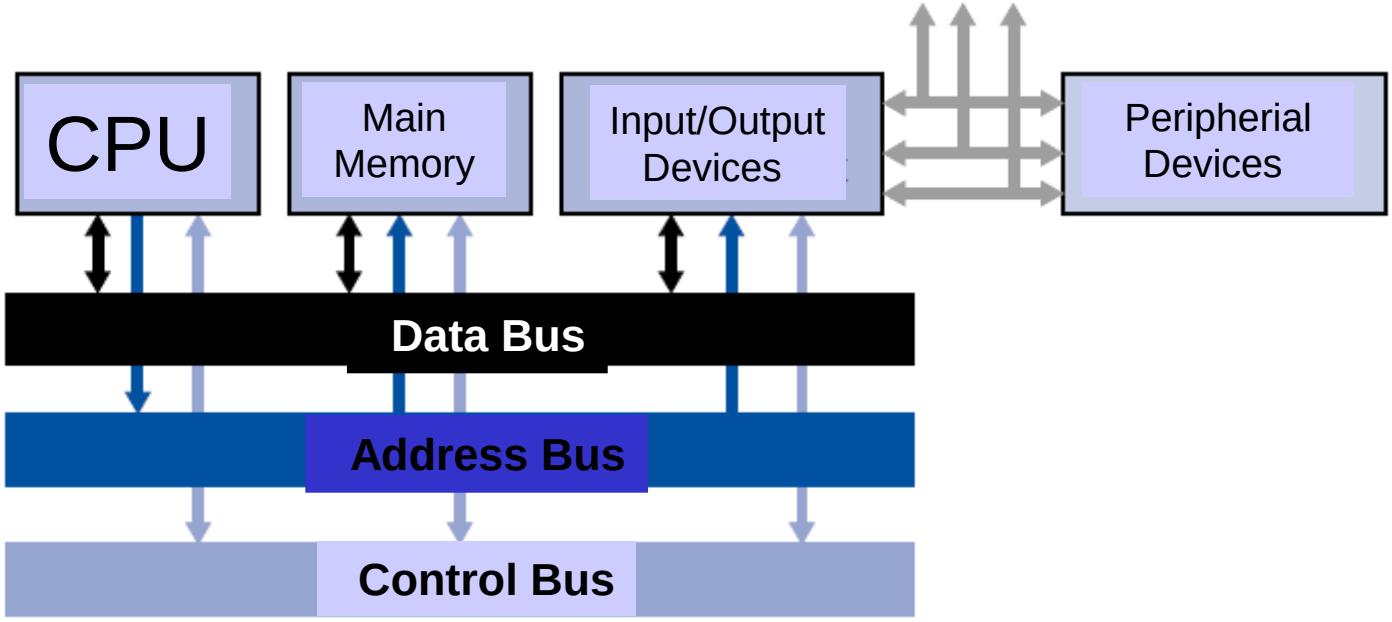
Buses and Interfaces

- Buses and interfaces are needed to communicate between all mainboard components as well as to external peripherials, such as external graphic processors, hard disks, printers
- All transport of data between mainboard components, the processor, the main memory, the I/O devices is operated by internal buses
- To reach a high speed communication, internal buses uses multiple parallel lines, one each per bit needed; the chip set determines the maximally available number of bus lines, which correlates at best with the processor‘s register width measured in bits such that all storage words can be processed in parallel

© 2017 Burkhard Stiller

Bus Types

- *Data bus: Bidirectional transmission of data between any number of devices*
- *Address bus: Unidirectional transmission of addresses to the storage or input/output devices*
- *Control bus:* Coordination of exclusive access rightsonto the data and address bus (reservation of, release of bus)



© 2017 Burkhard Stiller

Communications

- The communication between the processor, mainboard components, and attached peripherial devices is run by
  - *Polling*

    The processor polls in well determined, periodical time frames a device, if it shows data to be communicated, devices cannot start any communication
  - *Interrupt Request (IRQ)*

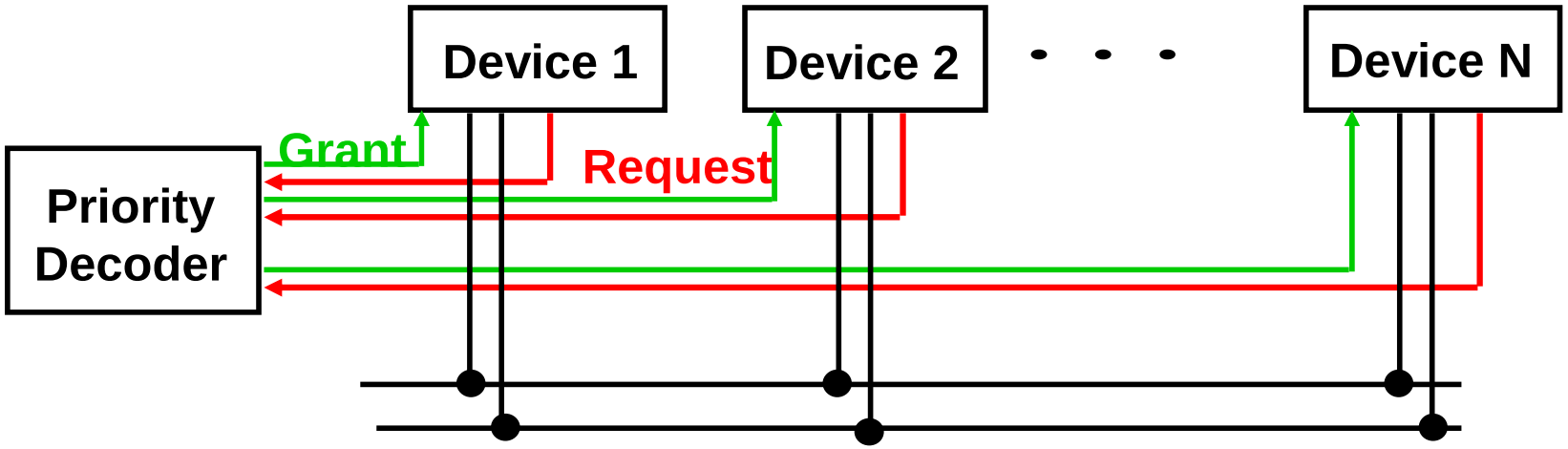
    A communication with the processor may be requested or started by any device by issueing an interrupt request (a deciated signal)

- A dedicated system control unit is required to decide as the bus master (bus arbiter), which of the components attached can use the bus
  - Arbitration determines the hierarchy of access rights and the access itself

© 2017 Burkhard Stiller

System Bus Arbitration (1)

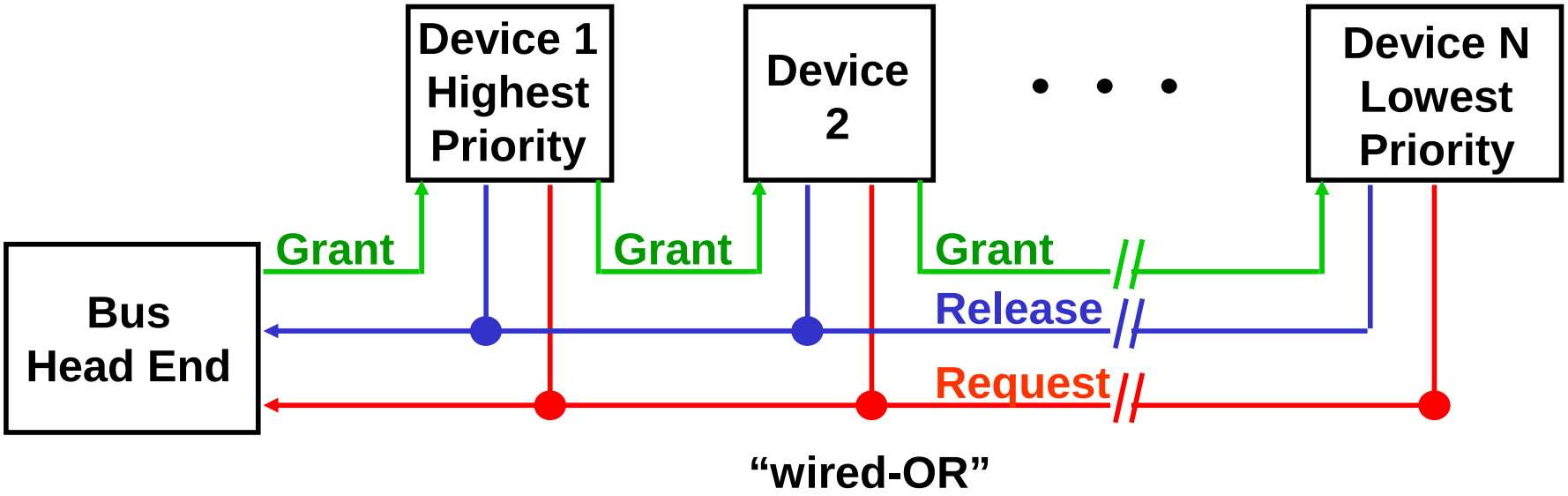
- Centralized approach (independent requests):
  - Each device requests via its bus arbiter (one per device) the access to the system bus
  - The *centralized priority decoder* decides in case of multiple simultaneous requests according to a pre-defined algorithm, which device comes first, second, third, ...
    - The decoder operates as a “umpire”
  - An arbitration works with two messages
    - Request = request for an arbitration
    - Grant = grating the arbitration



© 2017 Burkhard Stiller

System Bus Arbitration (2)

- Decentralized approach (Daisy Chaining):
  - All arbiters are chained according to their physical position
  - Each arbiter forwards the *Bus Grant message*, which is not needed locally (thus., no local *Bus Request* exists), to the next arbiter down the line
    ➔ The first arbiter in the chain shows the highest priority
  - Thus, a third message for arbitration needed
    - Release = Forwarding of not needed grant

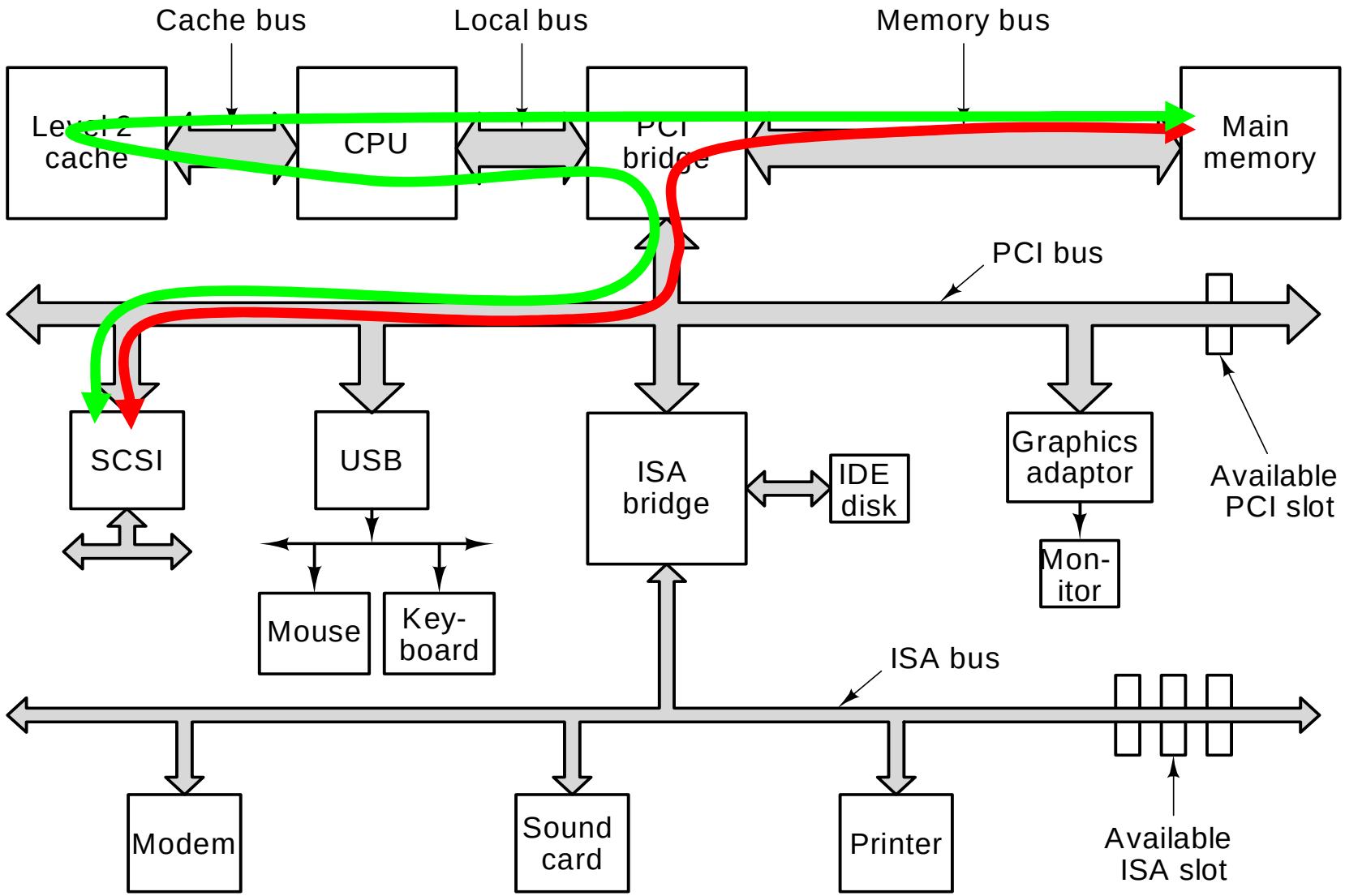


Interfaces for Internal Communications

- Alternatives:
  - *Programmed I/O (E/A-Port- bzw. Basisadressen)*
    A pre-negotiated address block defines the memory, which is used to exchange data between the processor and the respective device
  - *Memory-mapped I/O*
    Processors may not offer dedicated address blocks for input/output devices, thus, addresses for these I/O devices are mapped into the addresses used within the main memory
  - *Direct Memory Access (DMA) and Bus Mastering*
    Dedicated approaches allow for a direct transmission of data between devices and the main memory and vice versa without the processor being involved

© 2017 Burkhard Stiller

PIO (Programmed I/O) und DMA (Direct Memory Access)



© 2017 Burkhard Stiller

ifi

Direct Memory Access (DMA)

- Data transfer performed between the relevant components without involving the processor
  - A DMA Controller coordinates the data transfer
- Advantages:
  - No memory access in terms of accessing LOAD, STORE, and LOOP instructions necessary, since the data transfer runs in the hardware without any programmed control

    ➔ only two (sometimes only one) memory access per data required only
    - The processor‘s load is reduced and can run tasks, which do not need the system bus
- Drawback:
  - Inconsitencies of the cache‘s content need to be handled
- In general:
  - None of these approaches is always better or worse! Application-dependencies!

Interfaces for Extension Slots

- PCs offer a wider range of interfaces for hardware extensions:
  - *PCI Interface (Peripheral Component Interface)* was a standard PC extension interface (32-Bit, 33 MHz).
  - *AGP Interface (Accelerated Graphics Port)* defines a dedicated extension for graphics (64-Bit, >= 66 MHz).
  - *PCMCIA Interface (Personal Computer Memory Card International Association) is* used also as PC Card Interface and is applied for laptops as external interface for small extension devices

© 2017 Burkhard Stiller

Interfaces for Drives

- Hard drives, CD-ROM drives, and other bulk storage devicesuse two main interface standards:
  - An *EIDE (Enhanced Integrated Device Electronics)* interface is integrated on PC mainboards, which can connect to two devicess, one *master a*nd one *slave*
  - The *SCSI (Small Computer System Interface)* interface enables the connection to seven devices, while *Wide SCSI* enables 15 devices to be attached
- Next generation:
  - The *S-ATA (Serial-Advanced Technology Attachment)* interface follows the IDE/EIDE (also the P-ATA) interface with a faster and serial data transfer
  - The 2008 standard CFast replaced the CompactFlash standard

© 2017 Burkhard Stiller

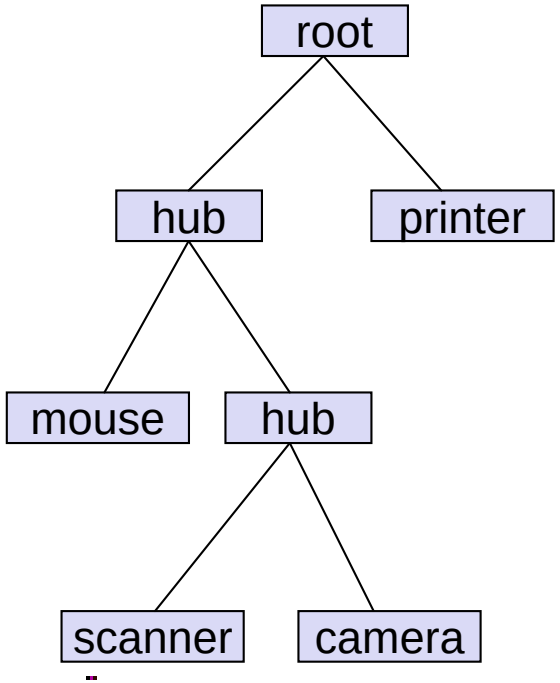
Interfaces for Additional Peripherials

- Serial interfaces for external devices are considered standard
- Support of the *Hot Plugging* Approach:
  - Devices can be added and removed in operation of the machine
  - *USB interface* (*Universal Serial Bus*) started as *USB 1.0 and 1.1* with a Data transfer rate of 12 Mbit/s and as *USB 2.0* with 480 Mbit/s. USB 3.0 reaches up to 5 Gbit/s
  - The *IEEE-1394 Interface* (termed *FireWire*) reaches a data transfer rate of 400 Mbit/s , the newer *FireWire 800* of 800 Mbit/s

© 2017 Burkhard Stiller

USB (Universal Serial Bus) – Basic Idea

- Connection of low-bandwidth peripherals
  - Mouse, keyboard, scanner, printer, camera, PDA, …
- Plug-and-Play
  - Simple connection, every time possible, no configuration
- Services
  - Shared bandwidth (USB 1.0: 12 Mbit/s, USB2.0: 480 Mbit/s, USB 3.0: 4,8 Gbit/s)
  - Isochronous and bulk data transfer
- System
  - Tree topology
  - Root hub at root polls devices (leaves)
  - 4 wires/cable: 2 power supply, 2 data
- See http://www.usb.org for more detai



© 2017 Burkhard Stiller

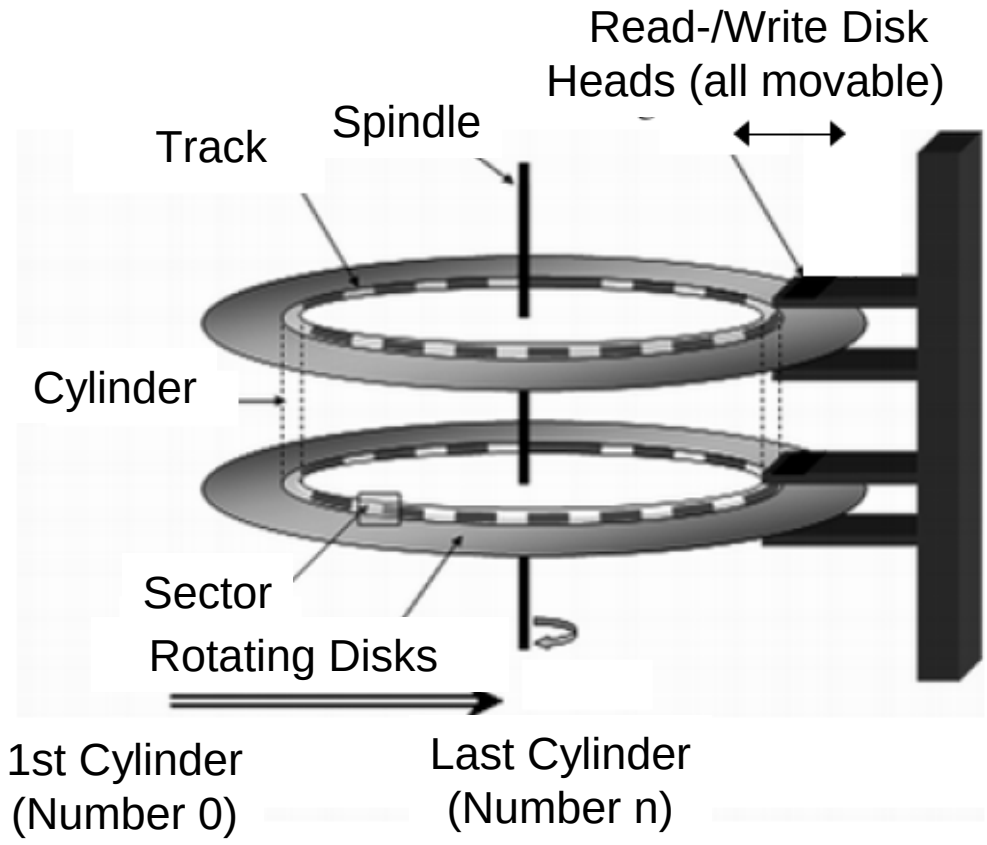
Modul 5: Computer Engineering

- von-Neumann Architecture
- Set-up and Operations
- Organization
  - Components
  - Busses and Interfaces
- Periphery (Samples only)
  - Hard Disks
  - CDs and DVDs
  - Modems

© 2017 Burkhard Stiller

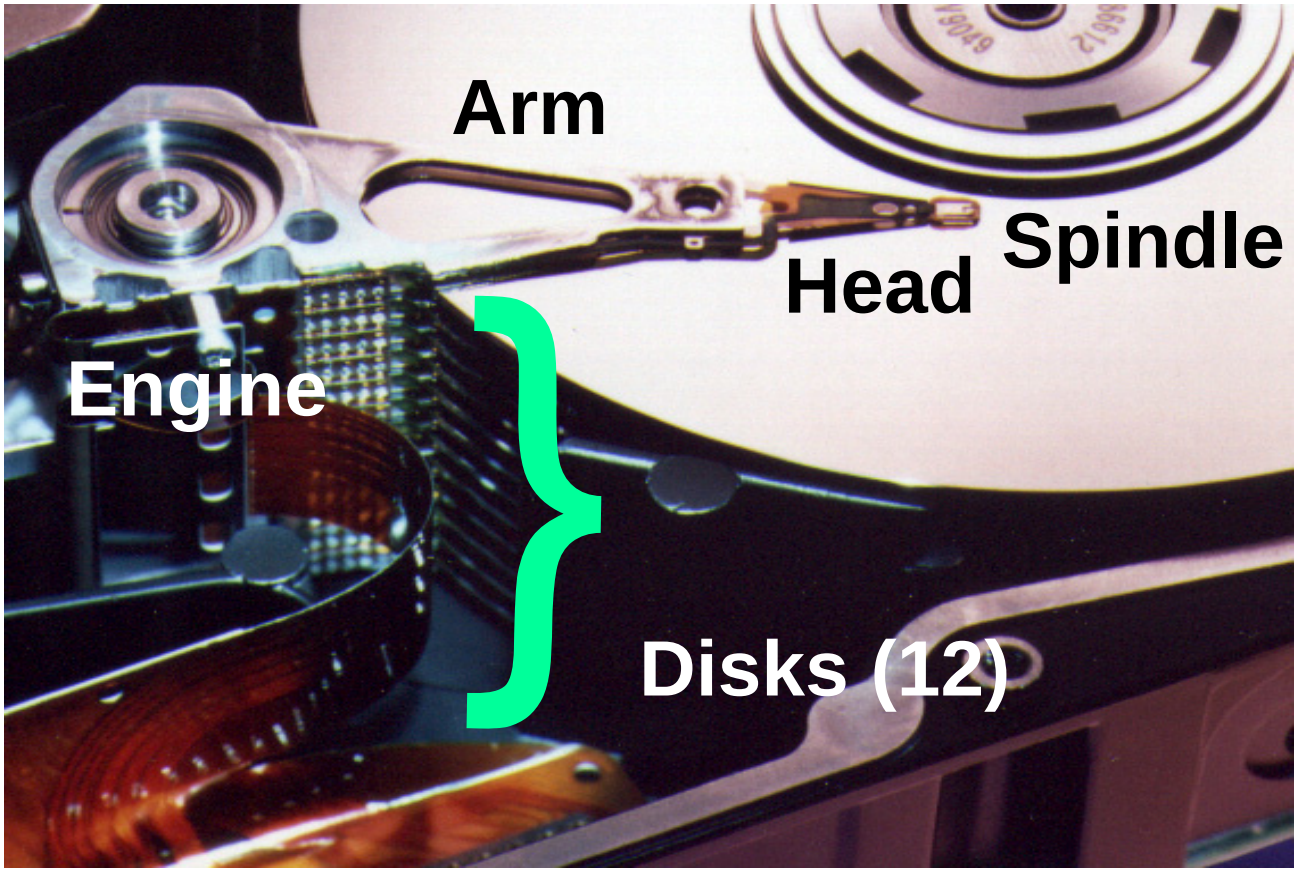
Hard Disks (HD) (1)

- Disks of different technology
  - Information (bit) is stored via magnetics on both sides of a disk, which are bundled and centered across a rotating spindle (Spindel)
  - Bits are organized in tracks (Spur), which are structured in sectors (Sektor) of, *e.g.*, 512 Byte each
  - Engines drive disk heads (Plattenköpfe) (>= 1/disk side) across the tracks to the required disk side and wait until the sector to be read or written appears
  - Cylinders are defined as all tracks accessible at the disk heads



© 2017 Burkhard Stiller

Hard Disks (HD) (2)



Sizes:
(5,25‘‘), 3,5‘‘, 2,5‘‘, 1,8‘‘



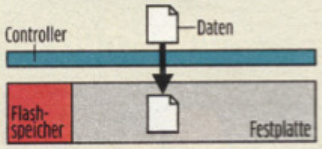
Development: 1956 (IBM) 24‘‘, 5 MB, 1 t weigth
In the 70s: 8‘‘
Today (2017): up to multiple TByte per disk at 3.5‘‘

© 2017 Burkhard Stiller

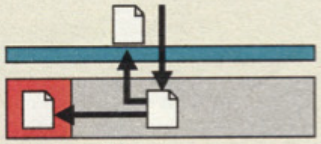
Hard Disks (HD) (3)

Tempo durch Controlling

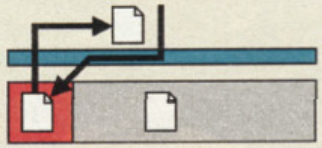
Die Hybridplatte besitzt einen speziellen Controller, der wichtige Dateien für einen schnellen Zugriff in den Flash kopiert.



Schreiben
Zu speichernde Daten leitet der Controller an die HDD weiter. Die SSD ist für Schreibzugriffe gesperrt



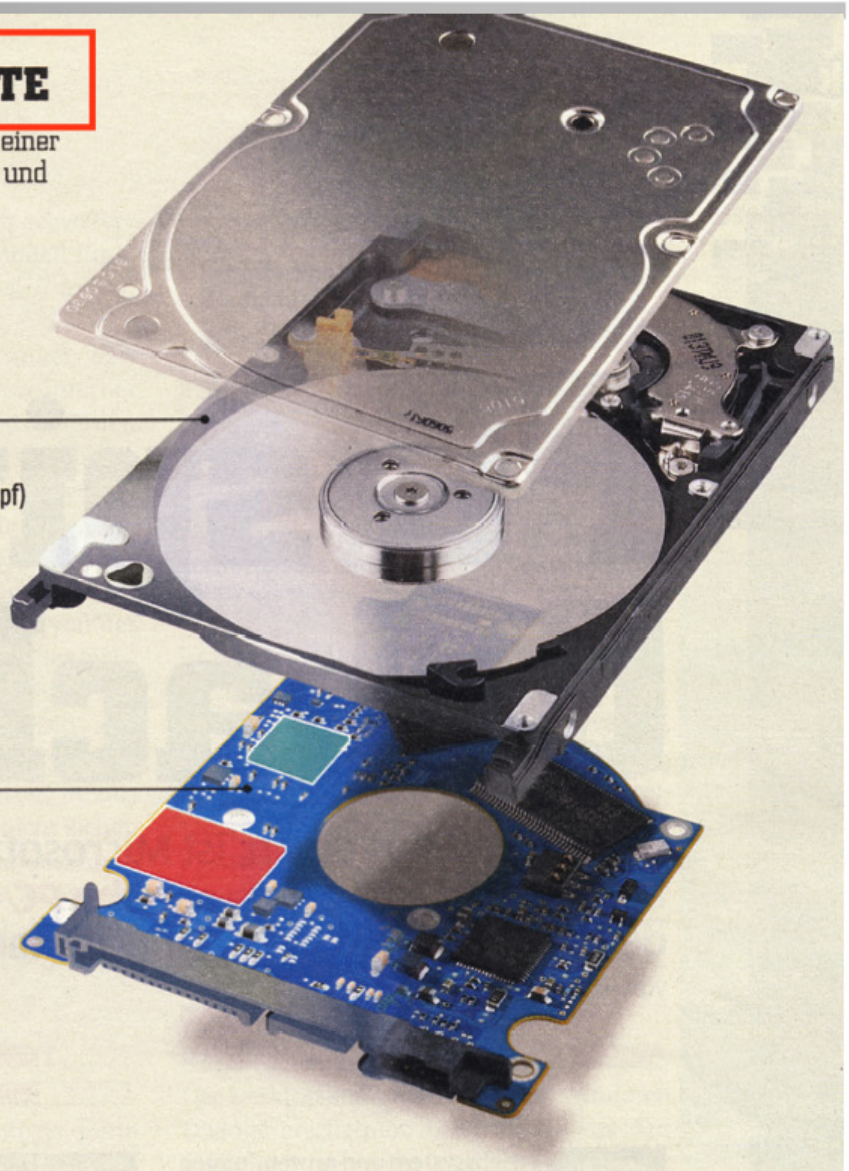
1. Zugriff
Ein Counter im Controller zählt die Dateizugriffe. Angeforderte Files werden in den Flash kopiert



2. Zugriff
Ruft der Nutzer die Dateien nochmals auf, leitet der Controller die Anfrage zum schnellen Flashspeicher um

DIE HYBRIDPLATTE

Die Momentus XT besteht aus einer 500 GByte großen HDD-Platte und vier GByte SSD-Speicher



HDD

Datenspeicher: Magnetscheibe
Zugriff: mechanisch (Schreib-Lese-Kopf)

- ⊕ günstig (ca. 0,08 Euro pro GByte)
- ⊕ unbegrenzte Schreibvorgänge
- ⊖ schockempfindlich
- ⊖ langsam
- ⊖ hoher Stromverbrauch
- ⊖ hörbares Arbeitsgeräusch

SSD

Datenspeicher: Halbleiter
Zugriff: elektronisch

- ⊖ teuer (ca. 2,60 Euro pro GByte)
- ⊖ begrenzte Schreibvorgänge
- ⊕ schockresistent
- ⊕ schnell
- ⊕ geringer Stromverbrauch
- ⊕ geräuschlos

HDD: Hard Disk Drive
SSD: Solid State Disk

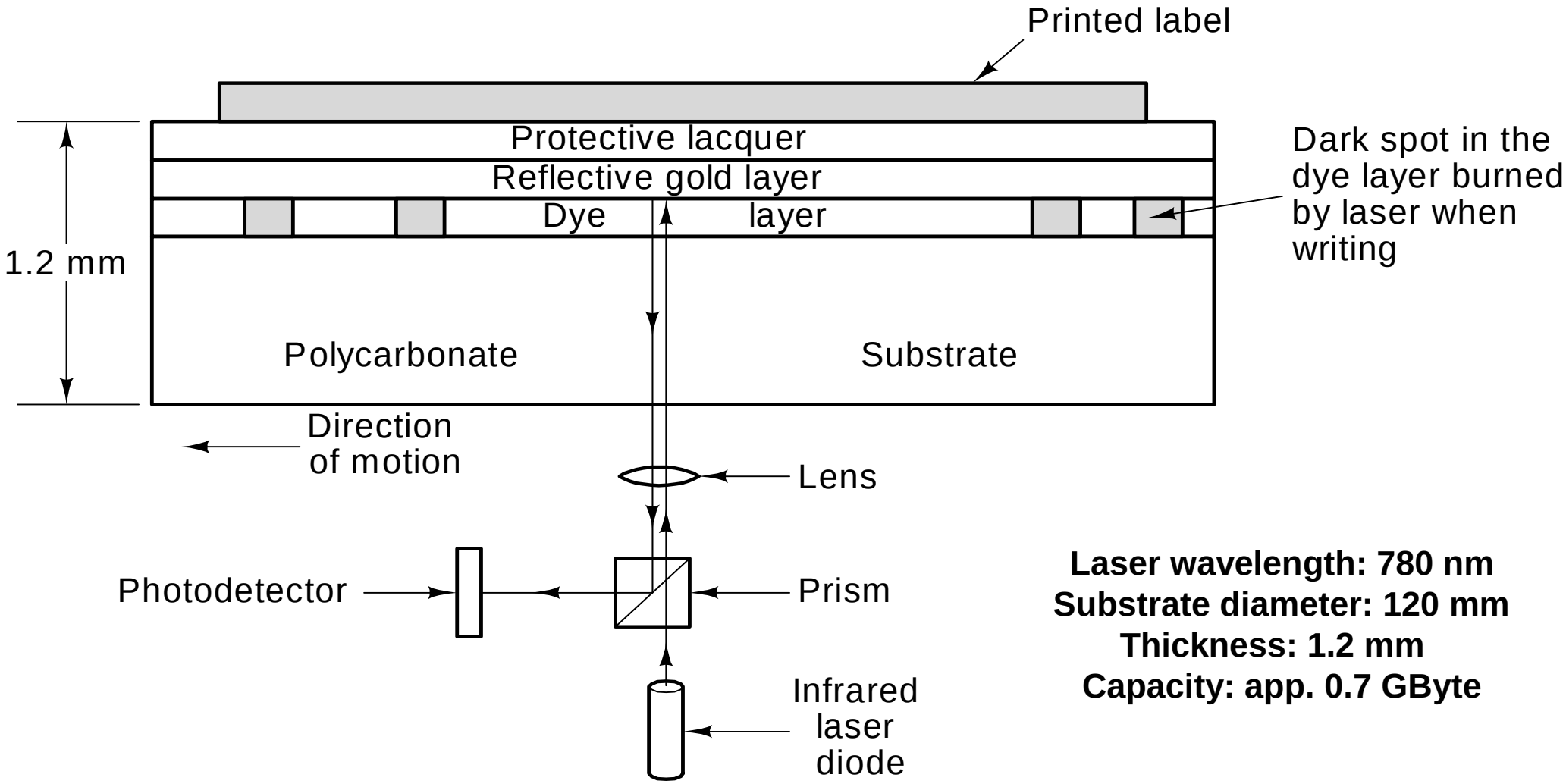
© 2017 Burkhard Stiller

Optical Disks

- The *CD* (Compact Disc) determines an optical data storage
  - Introduced in 1982 by Sony and Philips to replace the record (music)
- The CD surface holds a thin metal layer, which is sampled by a laser, pits (Vertiefungen/Löcher) and land represent bits stored
  - Digitized data of music and storage of any computer data
  - Multiple data formats exists for CD-ROMs
  - Storage capacity of 650 ... 900 MByte, which corresponds to CD-quality music of 74 to 99 min.
- DVD (Digital Versatile Disc) offers 4,7 GByte storage capacity
  - Many incompatible formats originally in operation
  - Since 2004 DVD drives should be able to read the competing recordable formats

© 2017 Burkhard Stiller

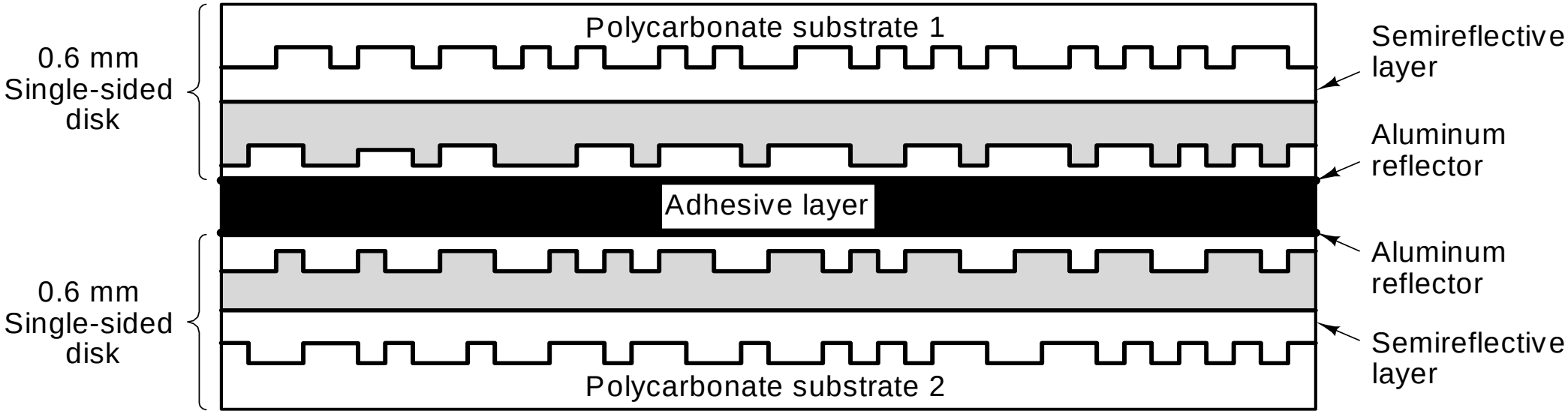
Compact Disk (CD)



Laser wavelength: 780 nm
Substrate diameter: 120 mm
Thickness: 1.2 mm
Capacity: app. 0.7 GByte

© 2017 Burkhard Stiller

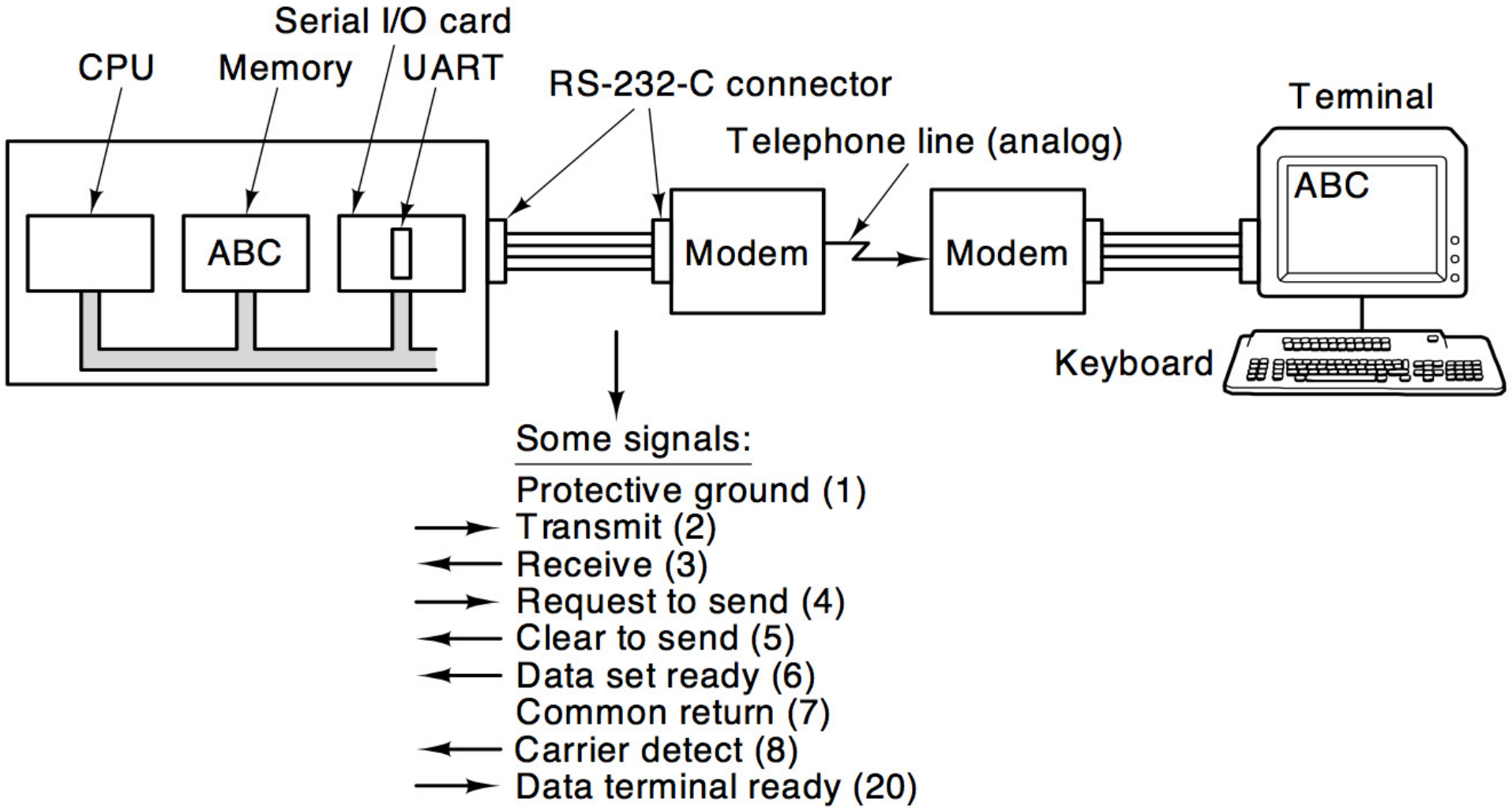
DVD (Double-sided, Dual-layer)



Laser wavelength: 635 or 650 nm
Substrate diameter: 120 mm
Thickness: 1.2 mm
Capacity: 4.7, 8.54, 9.4, or 17 GByte

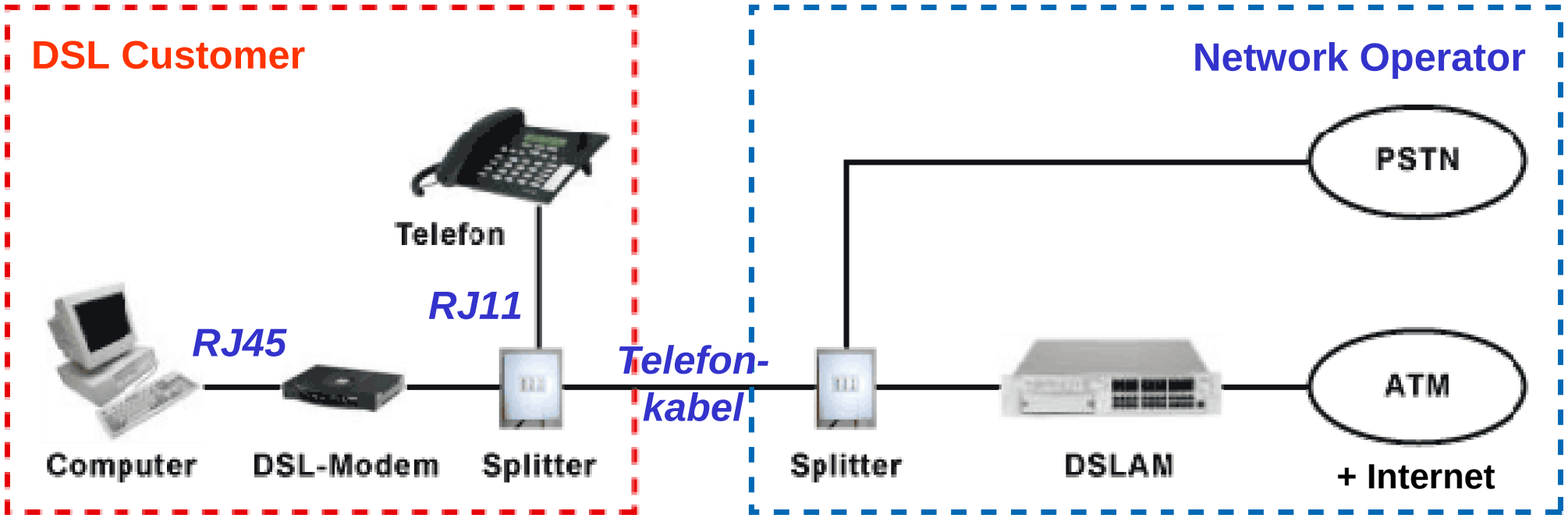
© 2017 Burkhard Stiller

Modem – Interconnection of Computers (old)



© 2017 Burkhard Stiller

ADSL Modem – Interconnection of Computers (new)



(A)DSL: (Asymmetric) Digital Subsrciber Line
DSLM: DSL Line Multiplexer
PSTN: Public Switched Telephony Network
ATM: Asynchronous Transfer Mode

© 2017 Burkhard Stiller

ifi

6. Introduction

Motivation and History

Basic Components and Terminology

Transmissions

Network Classification

Standardization

© 2017 Burkhard Stiller

Motivation

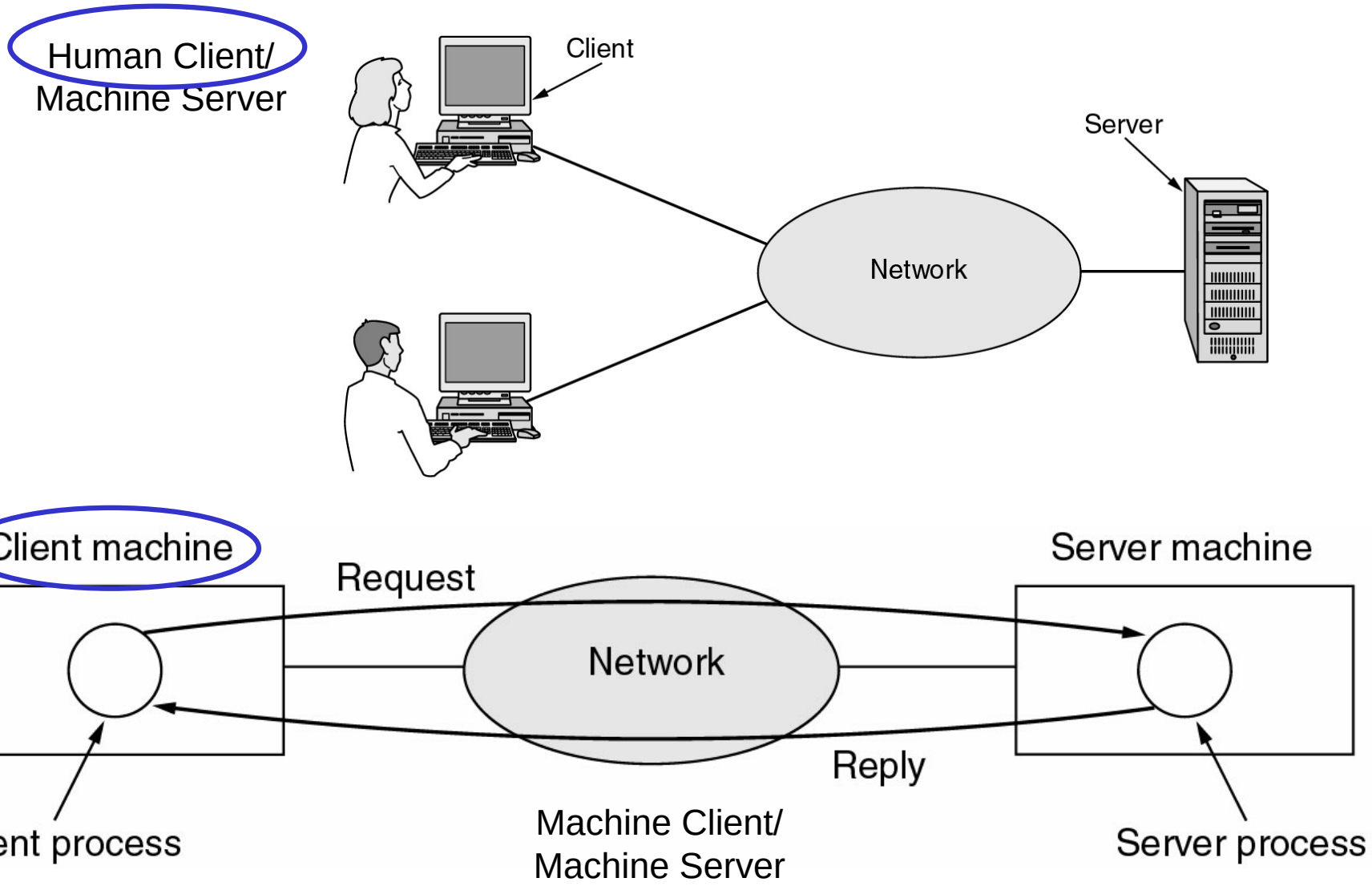
- Main goal of computer networks:
  - Exchange of information
- Purpose:
  - Human-to-human communication
  - Human-to-machine communication
  - Machine-to-machine communication



- Means:
  - Language, signals, characters, bits, waves …
- Problems:
  - Semantics and mappings
  - Transmission media

© 2017 Burkhard Stiller

Human Client/Server and Client/Server



History …

History — Antique Systems

- Antique messaging systems:
  - (Riding) messenger with letters or oral messages
  - Requires:
    - Roads
    - Relay stations (horse changes, restocking)
  - Postal services (letter, package) operate similarly, still today



- Historic optical systems:
  - Prerequisite
    - Coding of messages
  - Optical signals
  - Message transported by technical means!





History — Electric Systems (1)

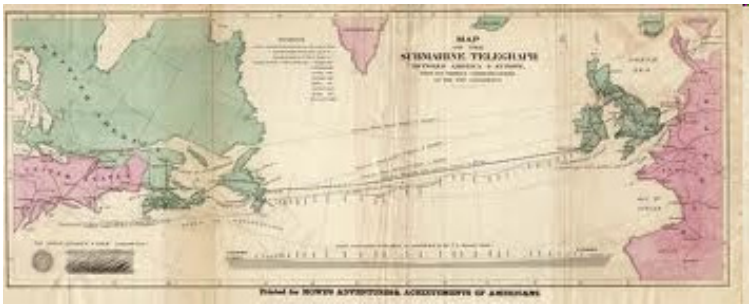
Fixed network transmissions (cable-based):

1833: C.F. Gauss and W. Weber:
Telegraph connecting the observatory and physics institute (1 km)



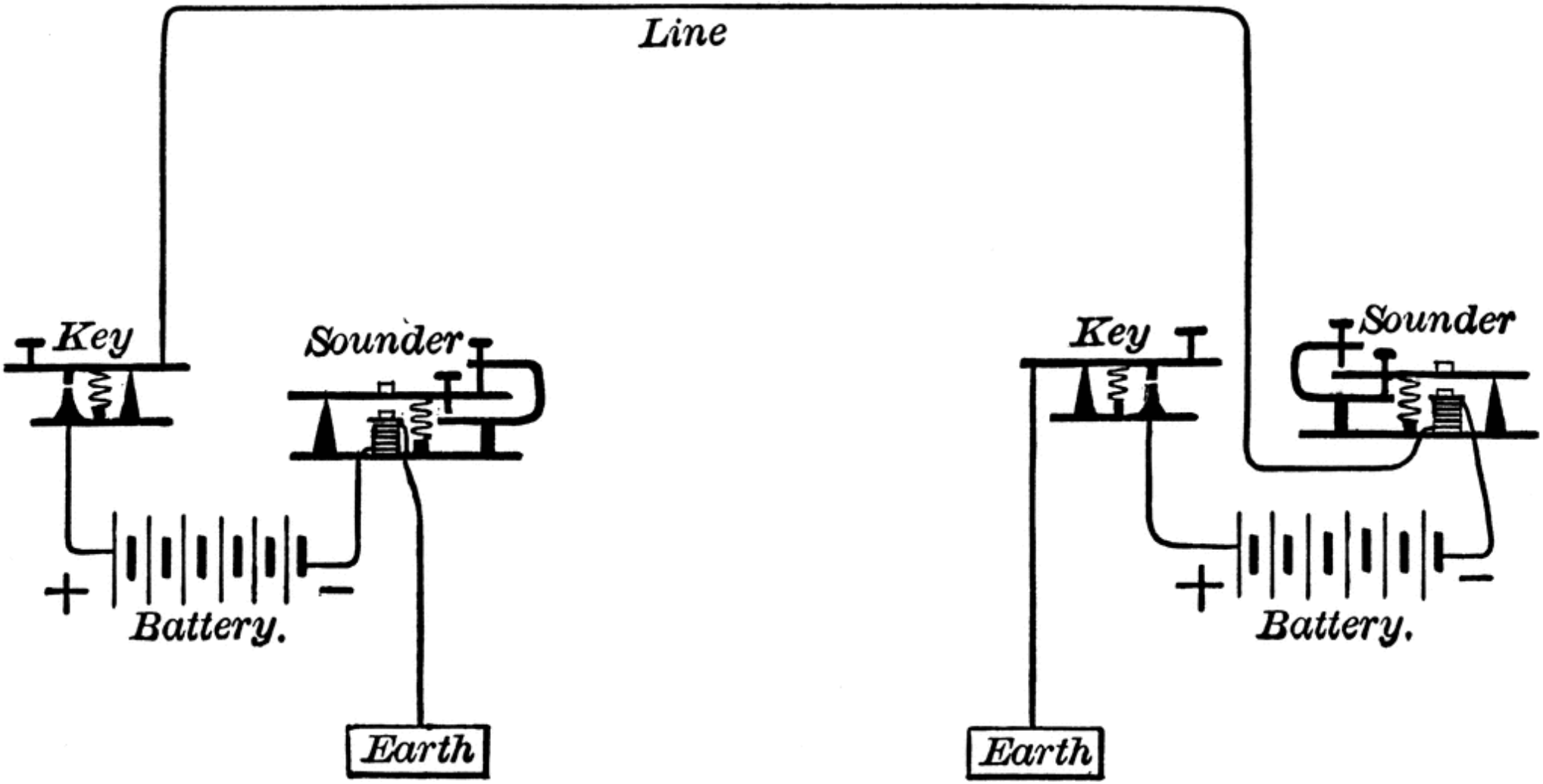
1851: First trans-sea cable between Dover and Calais

1858: Transatlantic line-based telegraphy between Europe and America



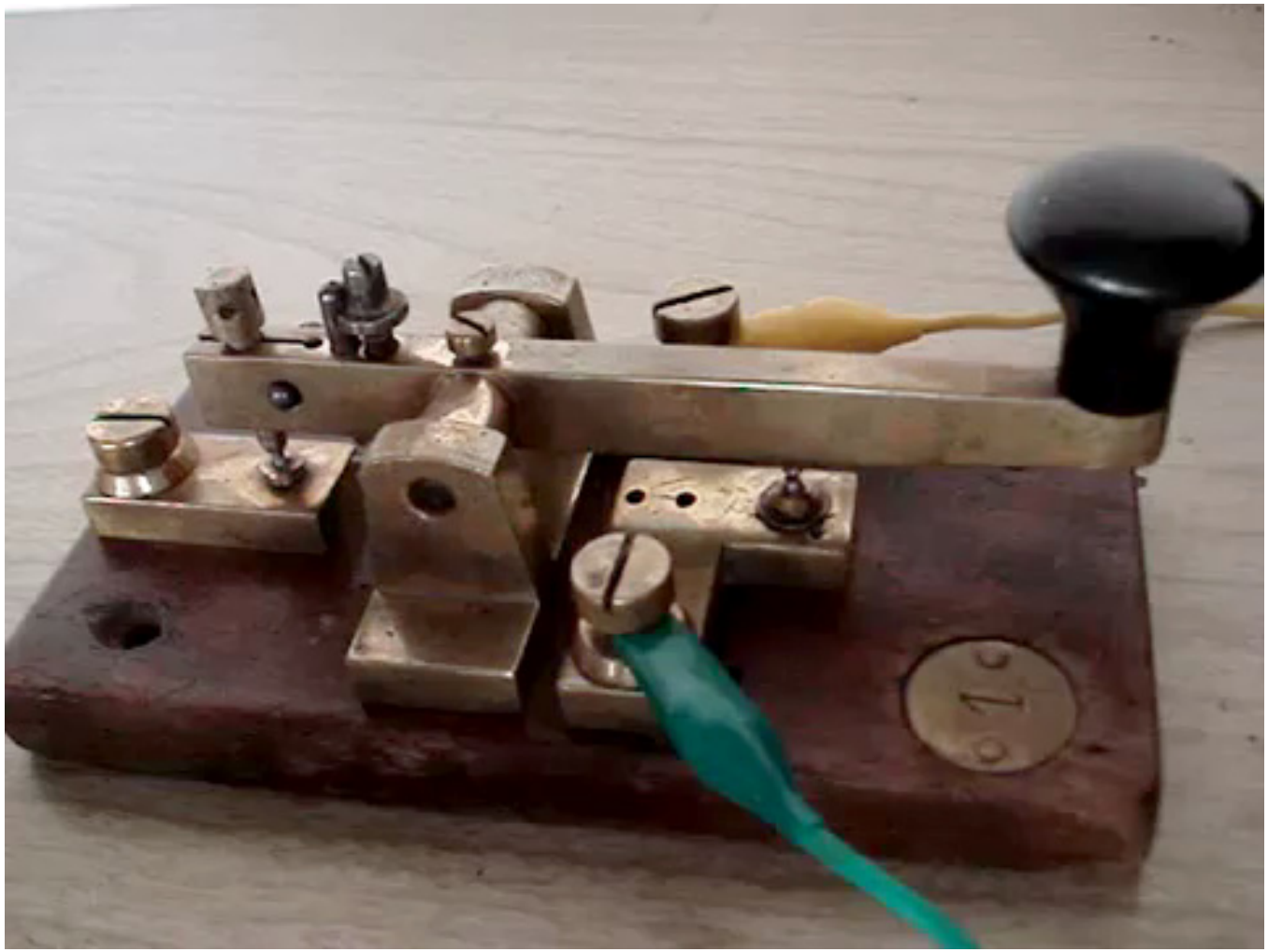
1866: Durable transatlantic cable

History – Schematic Telegraph System



© 2017 Burkhard Stiller

Telegraph Key and Sounder in Action



© 2017 Burkhard Stiller

History — Electric Systems (2)

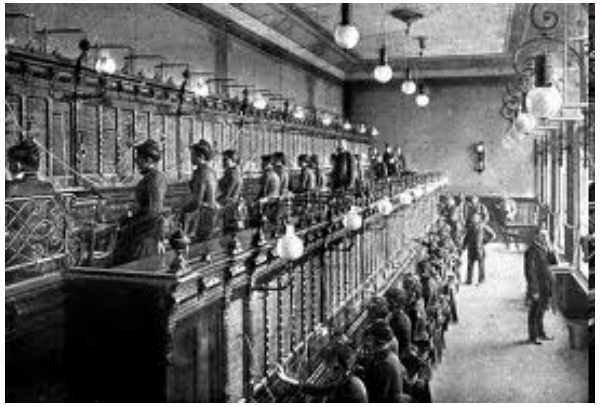
1876: Bell patents the “phone” (Reiss in Germany)





1880: 50.000 participants in US phone network

1881: Berlin opens the
first “Fernsprechamt”
(here Fernsprechamt 1)



© 2017 Burkhard Stiller

History — Electric Systems (3)

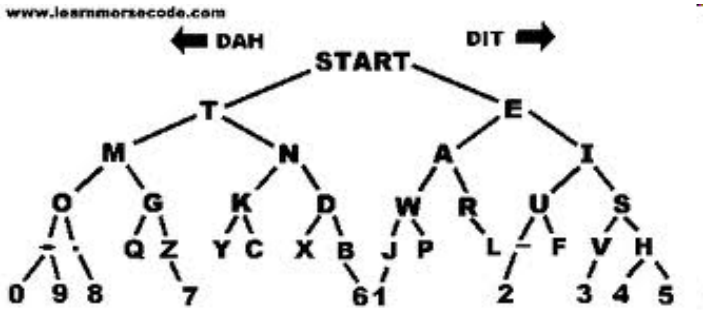
Radio transmissions:

1897: Morse codes transmitted by radio (Marconi)

1901: Radio-based telegraphy between Europe and US

1914: Introducing the teletype/telex system (Fernschreiber)

1920: First public radio transmission in Königs-Wusterhausen (next to Berlin, Germany)



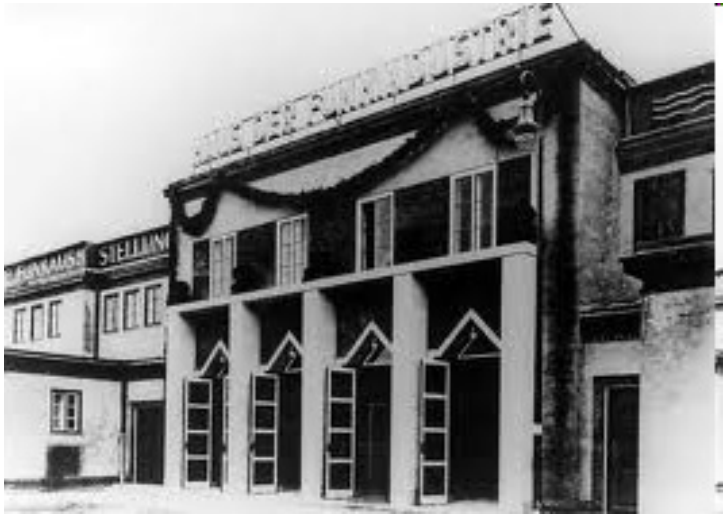




History — Electric Systems (4)

1923: Start of entertaining radio in Berlin

1929: First radio-based TV transmission
(Funkausstellung in Berlin)

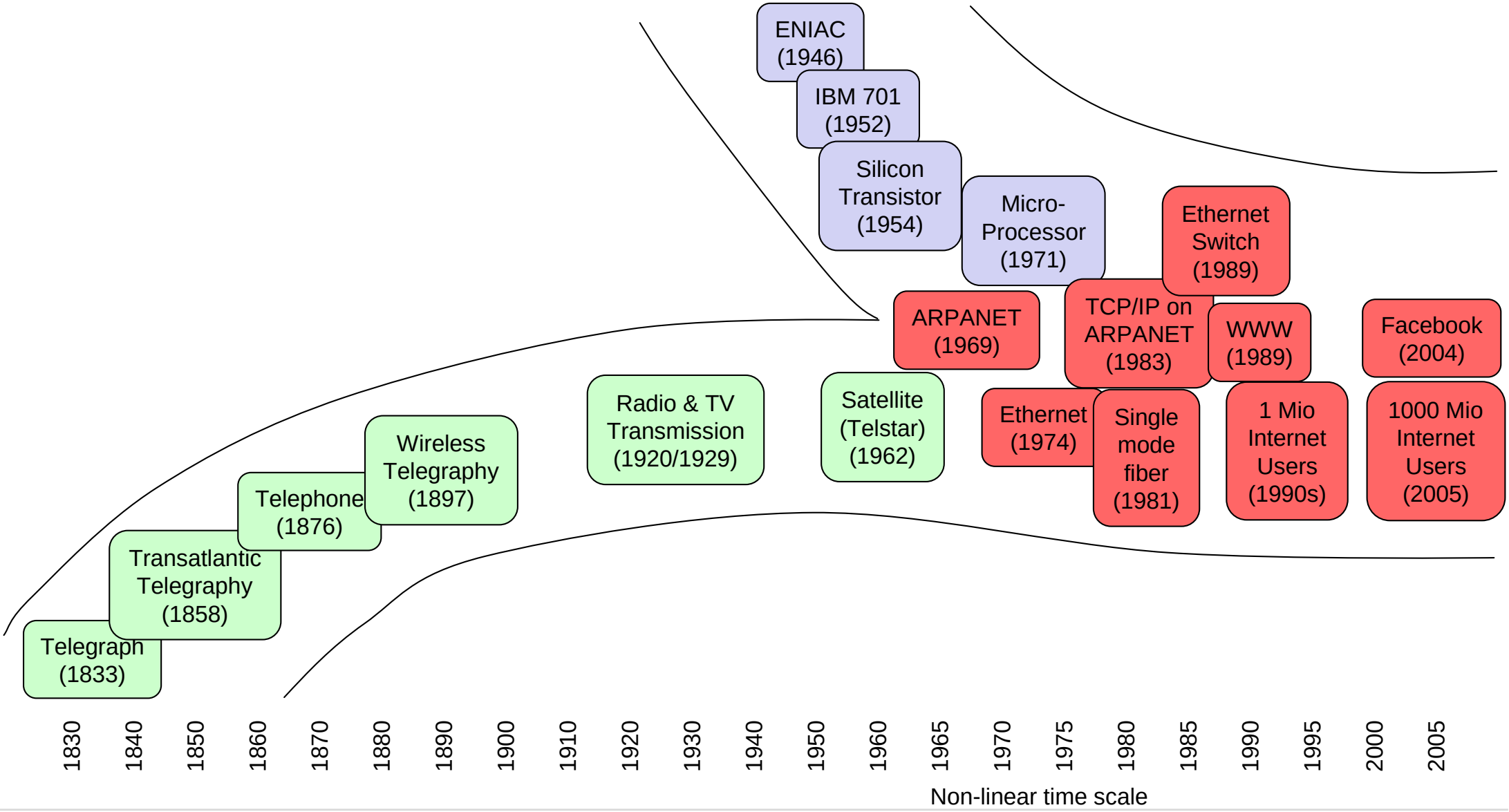


IFA
CONSUMER ELECTRONICS UNLIMITED
Berlin, 29.8. – 3.9.2008

1935: First regular public TV transmissions in Berlin

© 2017 Burkhard Stiller

Evolution of Computer Networks



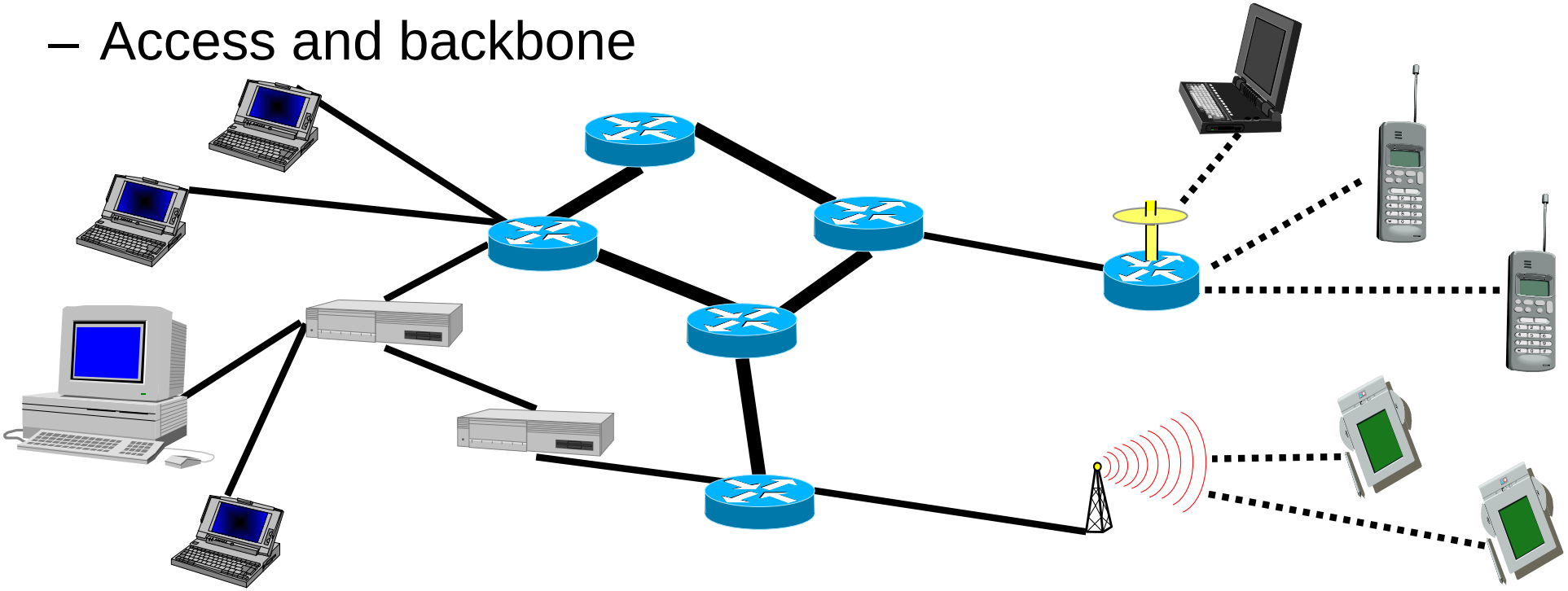
© 2017 Burkhard Stiller

ifi

Basic …

Major Network Components (1)

- **End-systems**, terminals, computer, nodes, **hosts** …
  - Fixed or mobile
- **Intermediate systems**, bridges, **routers**, gateways …
- **Links**, lines, cables, circuits, “radio links”, channels …
  - Access and backbone



© 2017 Burkhard Stiller

ifi

Major Network Components (2)

- End-systems (OSI) or hosts (IETF):
  - Offer services to users (customers)
  - Interfacing humans
- Intermediate systems (OSI) or routers (IETF):
  - Simple: Concentrators, bridges
  - Complex: Switches, IP routers
- Links (see Module 03):
  - Phone lines
  - Coaxial cables
  - Fiber optics
  - Radio links

OSI: Open System Interconnect
IETF: Internet Engineering Task Force
IP: Internet Protocol

© 2017 Burkhard Stiller

Terminology

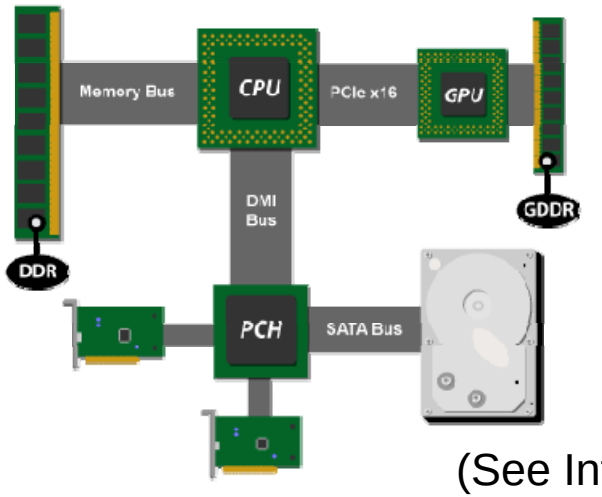
Network, System, Application (1)

- Computer **network**:
  - A set of *autonomous* computers exchanging information, where each machine operates on a dedicated memory, processor, and operating system



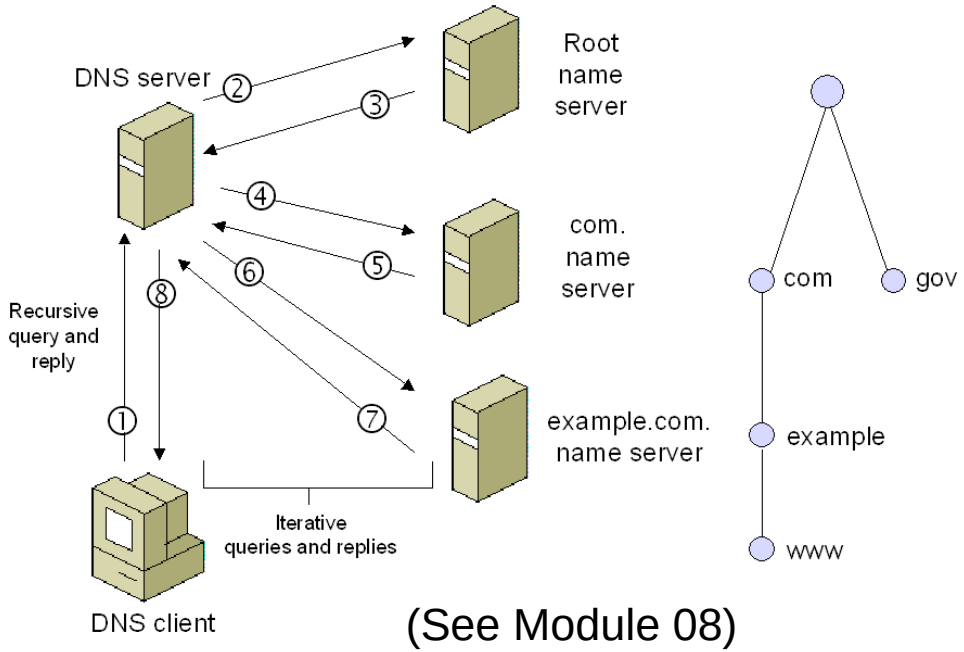
Network, System, Application (2)

- **System**:
  - A regularly interacting or interdependent group of components forming a unified whole



(See Informatik I)

- **Distributed system**:
  - A set of geographically distributed, autonomous, and interconnected computers, which offer a service based on cooperation



(See Module 08)

Network, System, Application (3)

- **Application**:
  - Single host-based service offering
  - Examples: Word processing, drawing, …

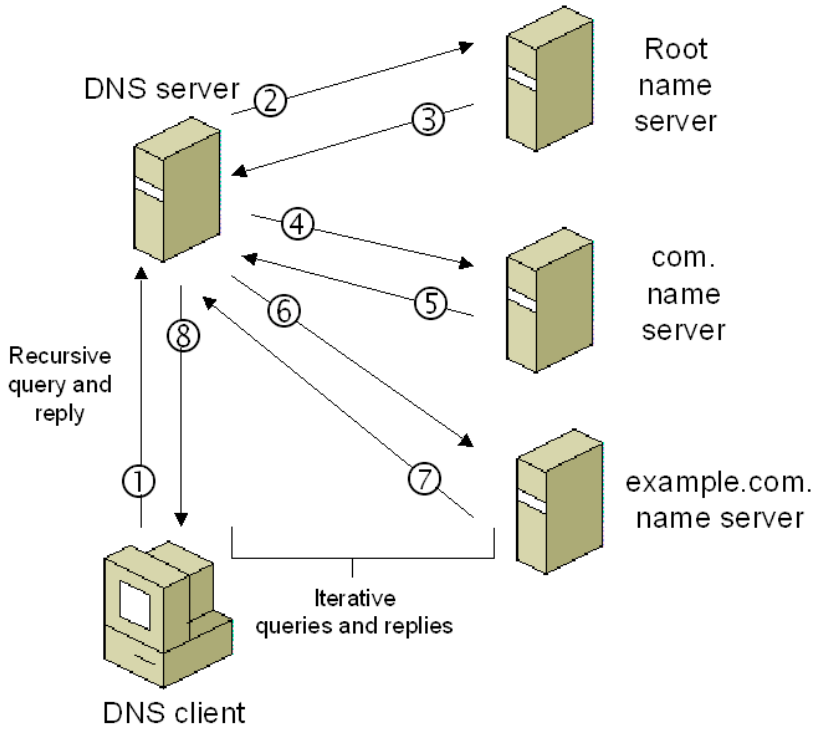
- **Distributed application**:
  - An application running on multiple, cooperating, geographically distributed computers and delivering a *specific* service to a user
  - A distributed application implements a special instance of a service in a distributed system

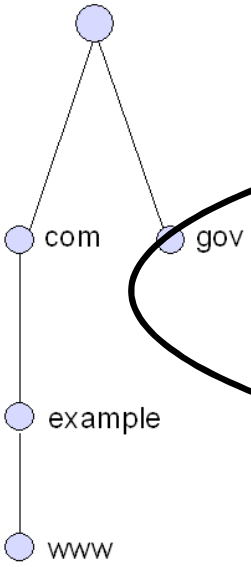
© 2017 Burkhard Stiller

Network, System, Application (4)

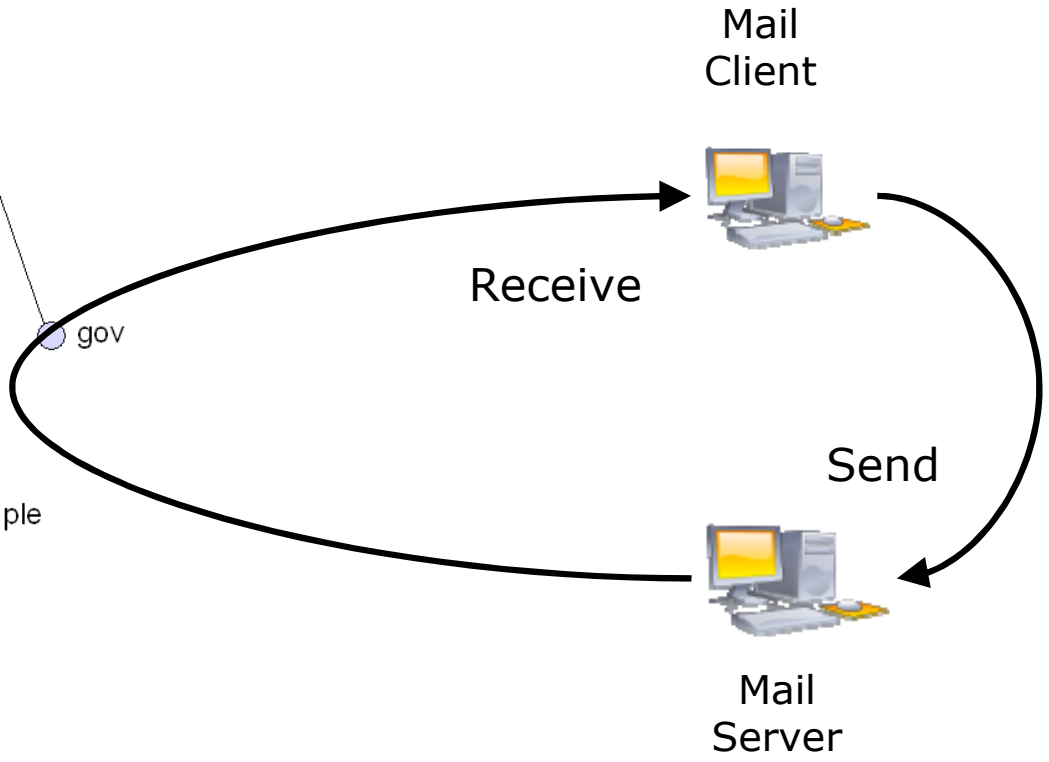
- Major **difference** between “network” and “distributed system”: Transparency of the distribution

Distributed system example: Domain Name Service



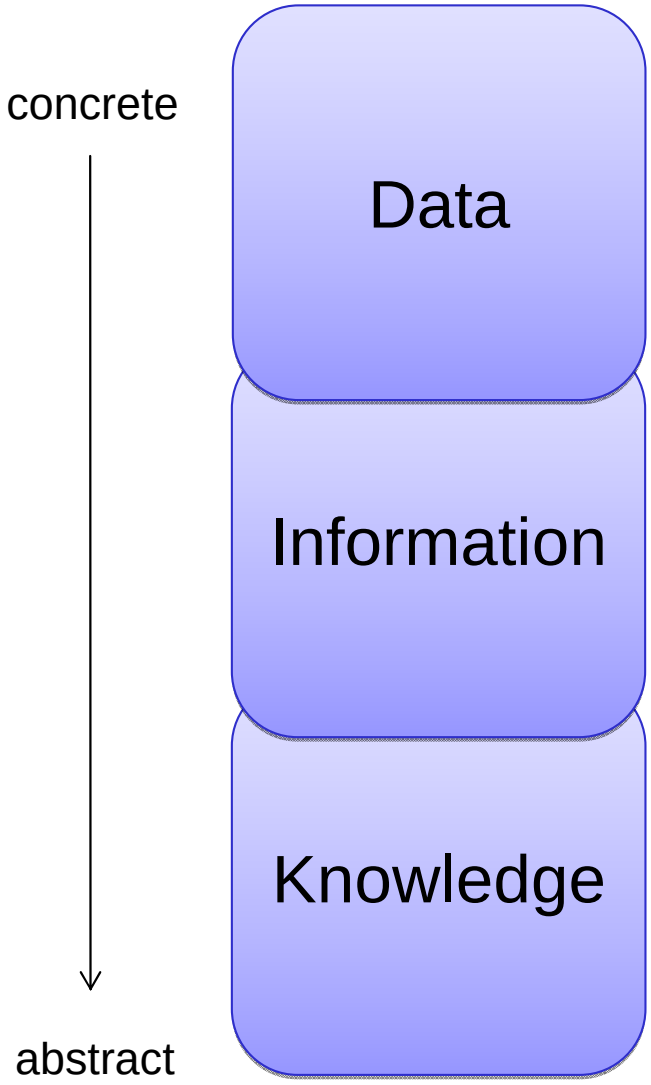


Network Application: Mail



© 2017 Burkhard Stiller

Terms (1)

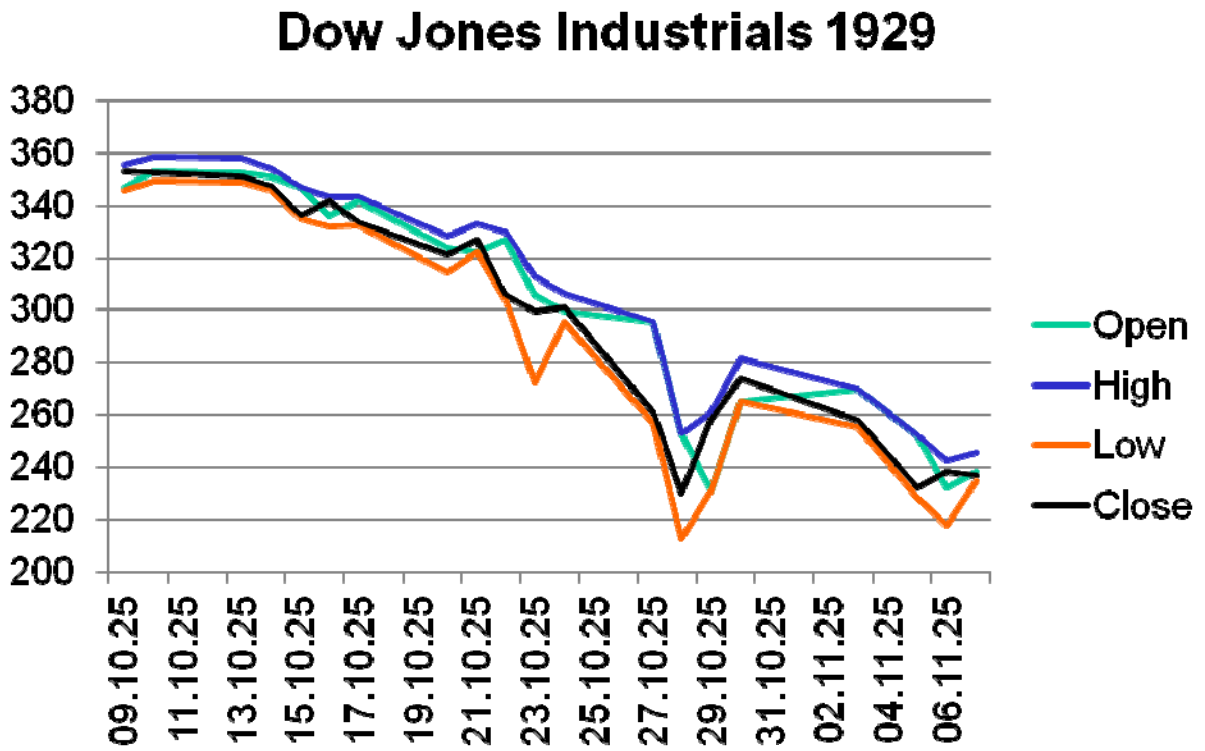


- Facts, data means “that what is given”
  - Digital data are coded in discrete characters
  - Analog data are represented as continuous functions
- Data that is processed, interpreted, structured, and organized in a context to make it to useful information
  - Involves human rules and activities
- Highest abstraction level: knowledge
  - Practical understanding of a subject
  - “Justified, true belief”

© 2017 Burkhard Stiller

Example

10.10.29	346.66	352.86	4000000
11.10.29	352.86	352.69	3960000
14.10.29	352.69	350.97	2760000
15.10.29	350.97	347.24	3110000
16.10.29	346.99	336.13	4090000
17.10.29	336.13	341.86	3860000
18.10.29	341.86	333.29	3510000
21.10.29	323.87	320.91	6090000
22.10.29	322.03	326.51	4130000
23.10.29	326.51	305.85	6370000
24.10.29	305.85	299.47	12900000
25.10.29	299.47	301.22	5920000
28.10.29	295.18	260.64	9210000
29.10.29	252.38	230.07	16410000
30.10.29	230.98	258.47	10730000
31.10.29	264.97	273.51	7150000
04.11.29	269.75	257.68	6200000
06.11.29	252.2	232.13	5920000
07.11.29	232.13	238.19	7180000
08.11.29	238.19	236.53	3220000



Data

Knowledge:
Crashes may occur.

Information

© 2017 Burkhard Stiller

ifi

Terms (2)

- **Signals** define a physical representation of data by well-defined space- or time-based changes of physical parameters
  - Signals represent physically abstract data.

- **Messages** are data in digital representation to be used for transmission
  - The structure of messages and its exchange rules are defined in *communication protocols*

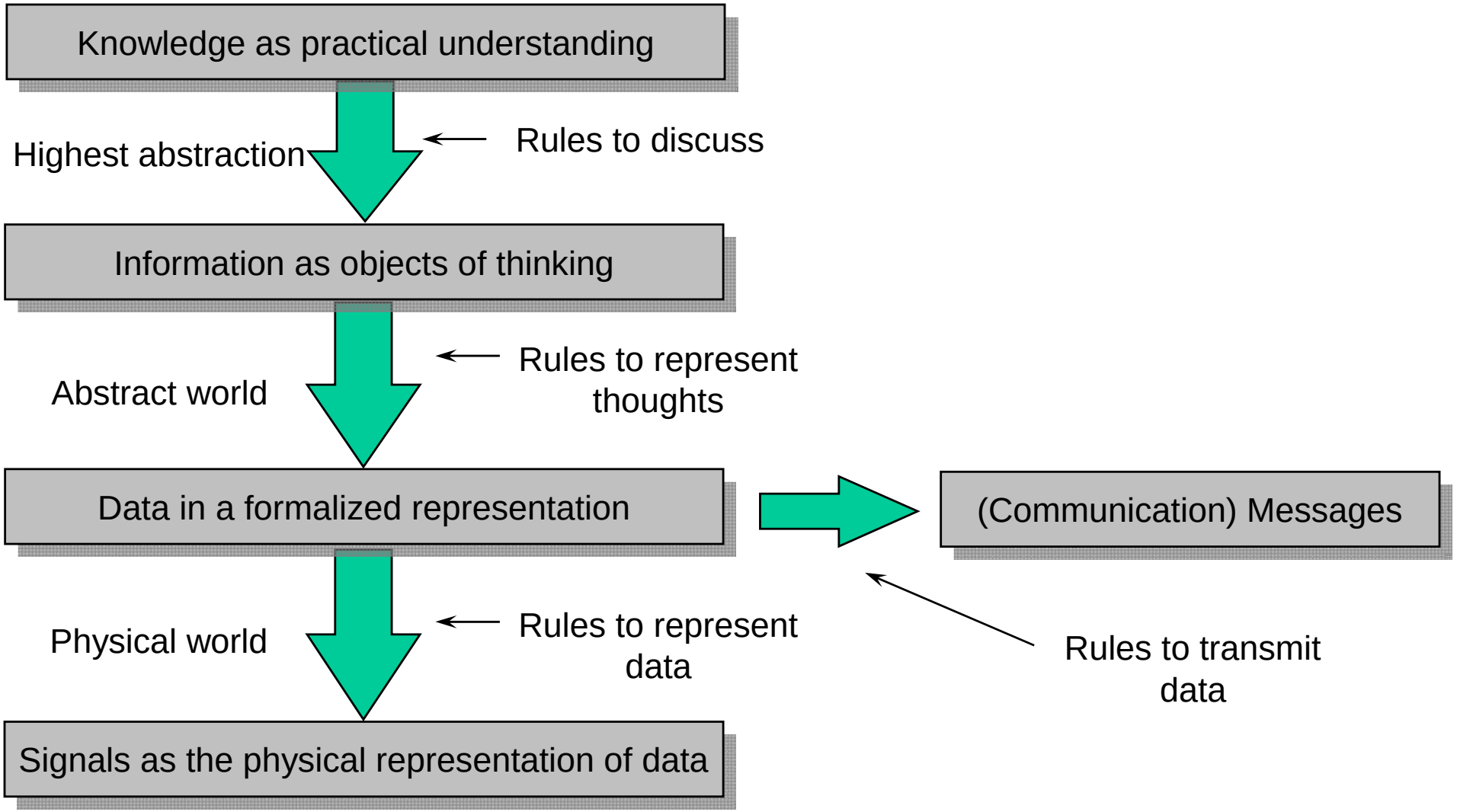
© 2017 Burkhard Stiller

Terms (3)

- **Communication:** The imparting or exchange of information
  - Origins from Latin “communis”, meaning “to share”
- **Tele-Communication:** Communicate with technology support
- **Data (Tele-)Communication:** The exchange of data using intangible carriers
  - Intangible carriers: flows of energy, current, waves
  - Opposite: tangible carriers, such as letter, floppy, or CD
- Note: The topic of this lecture is “communications” of “data”.
  The term “information” shouldn’t be used in this context!

© 2017 Burkhard Stiller

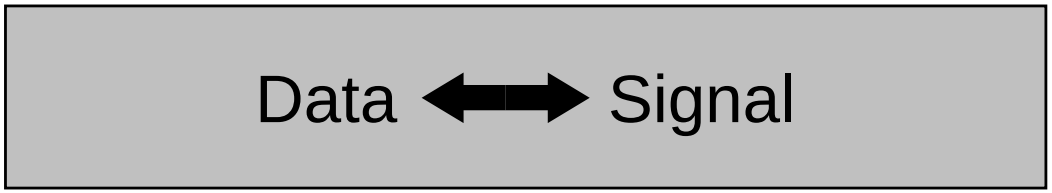
Terms (4)



© 2017 Burkhard Stiller

Terms (5)

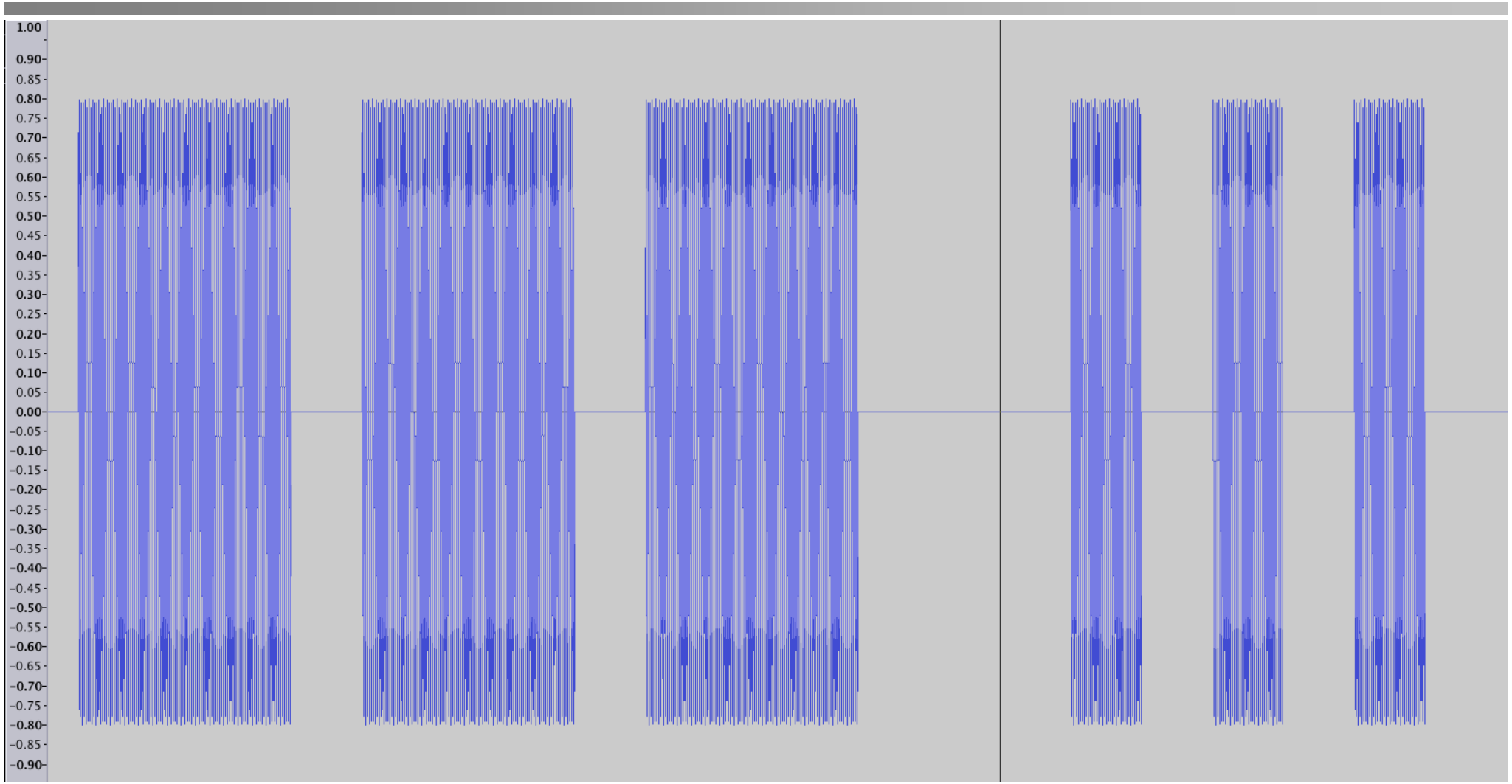
- Avoid any confusion of “data“ and “signal”!
  - Data may be captured, stored, transmitted, or handled only in signal form
  - Every data representation requires a signal representation



  - Examples:
    - Sounds of a language (data), which are transmitted as waves during talking (signals)
    - Representation of sounds (data) by means of electrical currents (signals)
    - Printed characters on paper as optical signals of abstract alphabet characters (data)

© 2017 Burkhard Stiller

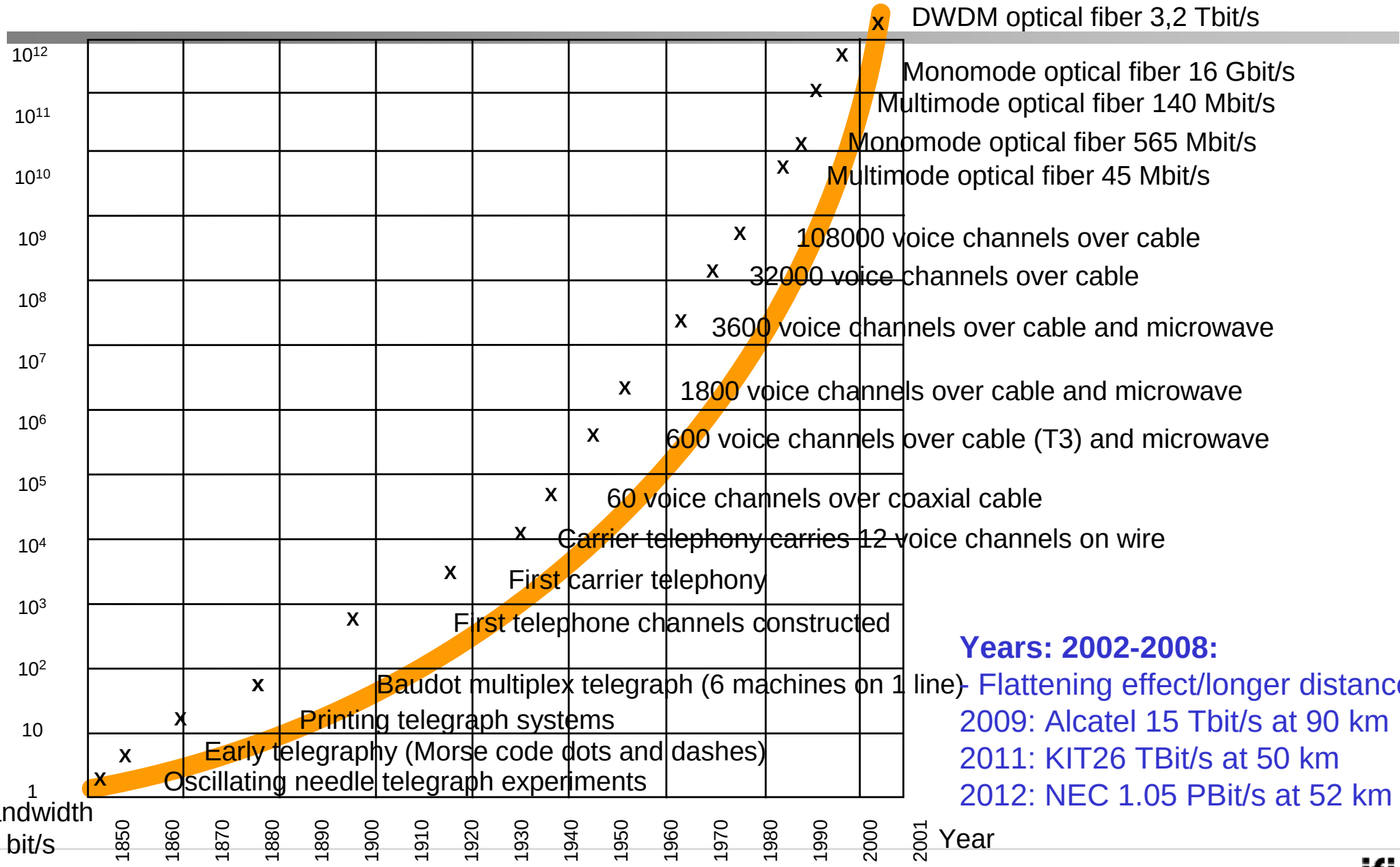
Signal – Example



© 2017 Burkhard Stiller

Transmissions

Inventions in Telecommunications



Years: 2002-2008:

- Flattening effect/longer distances

2009: Alcatel 15 Tbit/s at 90 km

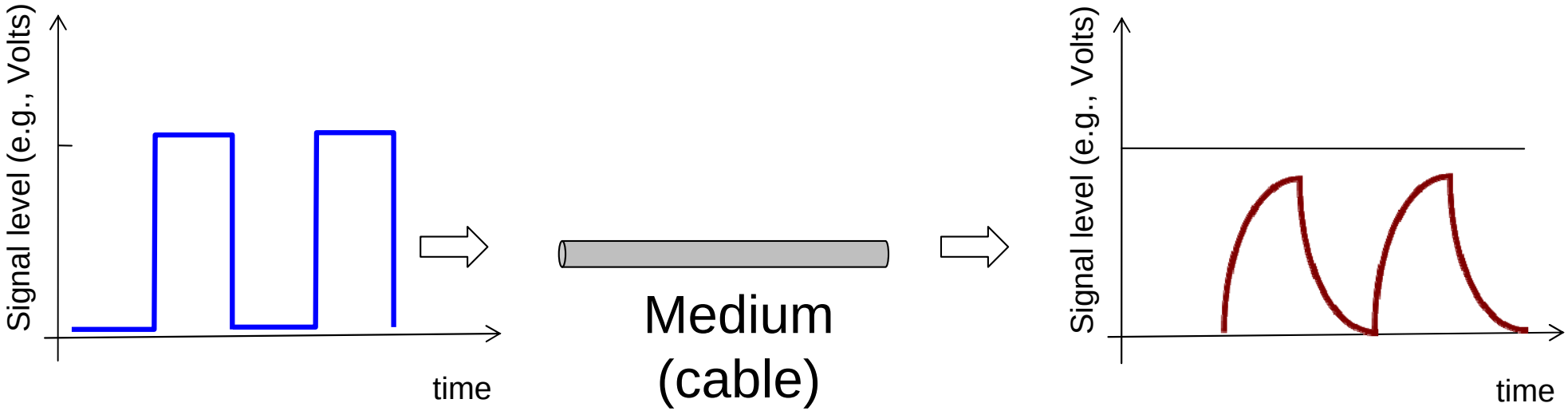
2011: KIT26 TBit/s at 50 km

2012: NEC 1.05 PBit/s at 52 km

© 2017 Burkhard Stiller

ifi

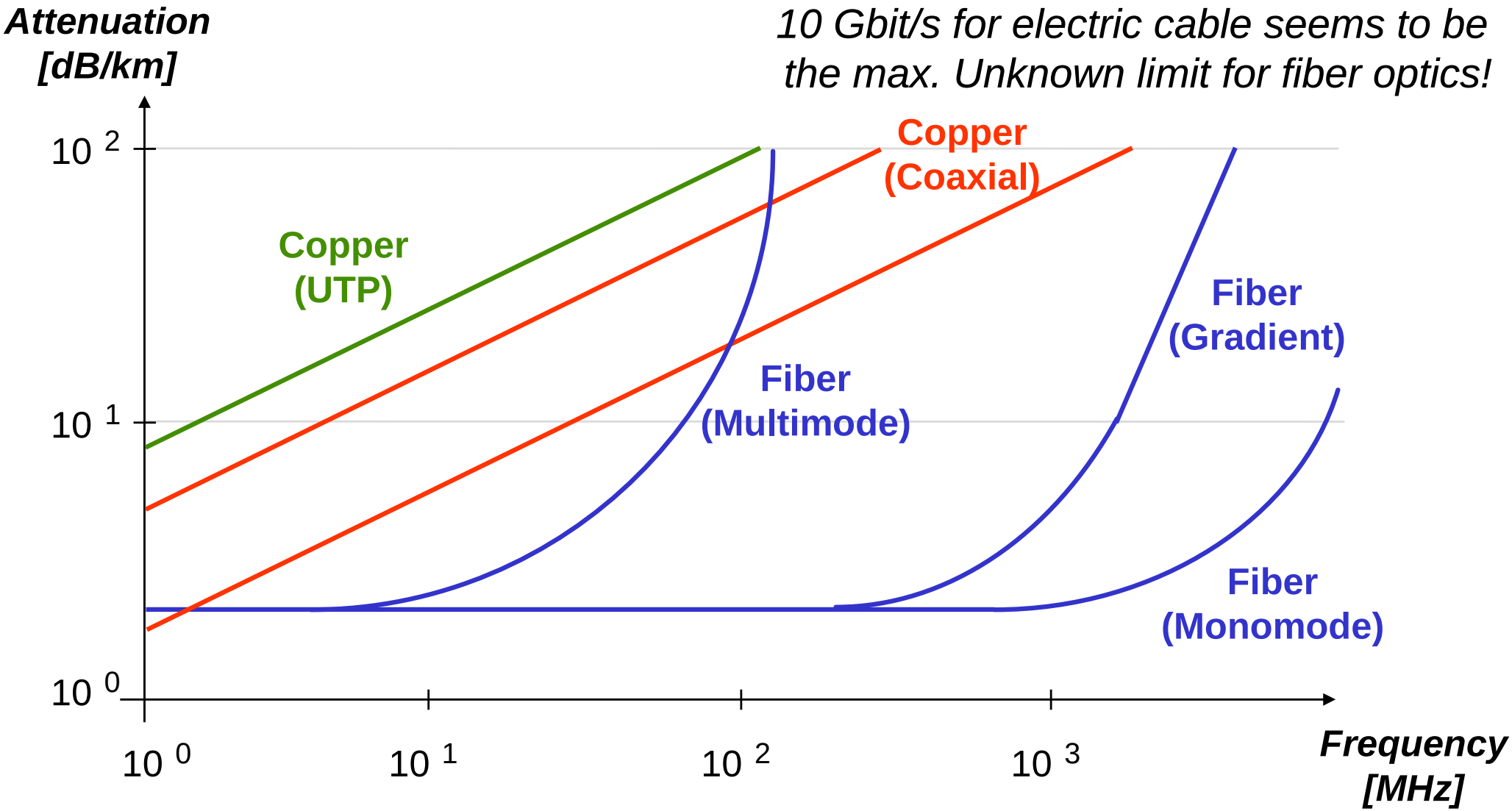
Attenuation



Input Signal **Output Signal**

Transmission Links

10 Gbit/s for electric cable seems to be the max. Unknown limit for fiber optics!



© 2017 Burkhard Stiller

ifi

Class…Traditional Network Classification

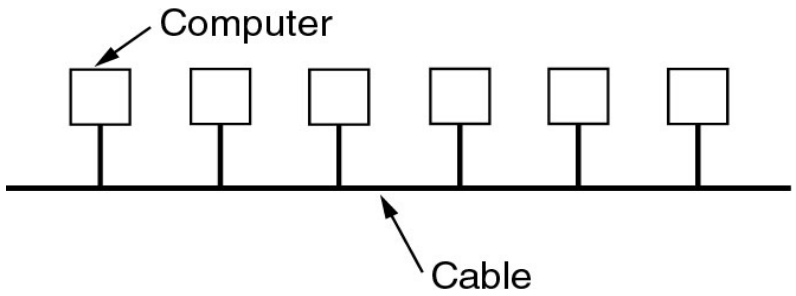
Distance	Coverage	Type	
1 m	Desk	PAN	Wireless keyboard and mouse
10 m	Room	LAN/SAN	
100 m	Building	LAN	
1 km	Campus	LAN	Interconnected LAN
10 km	City	MAN	
100 km	Country	WAN	
1,000 km	Continent	WAN	
10,000 km	Planet	GAN	The Internet

Personal Area Network: PAN
Storage Area Network: SAN
Local Area Network: LAN
Metropolitan Area Network: MAN
Wide Area Network: WAN
Global Area Network: GAN

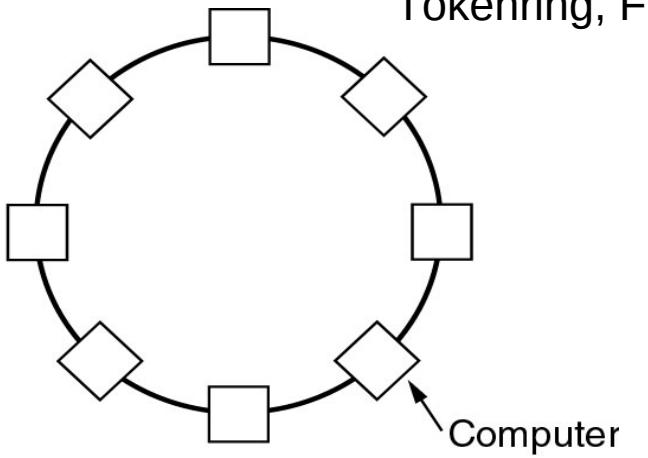
© 2017 Burkhard Stiller

Local Area Network (Examples)

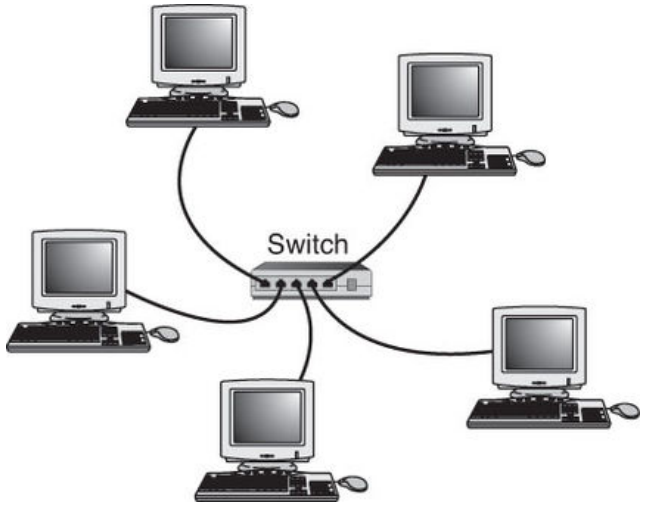
Bus Topology:
Original Ethernet (10Base-2, 10Base-5)



Ring Topology:
Tokenring, FDDI

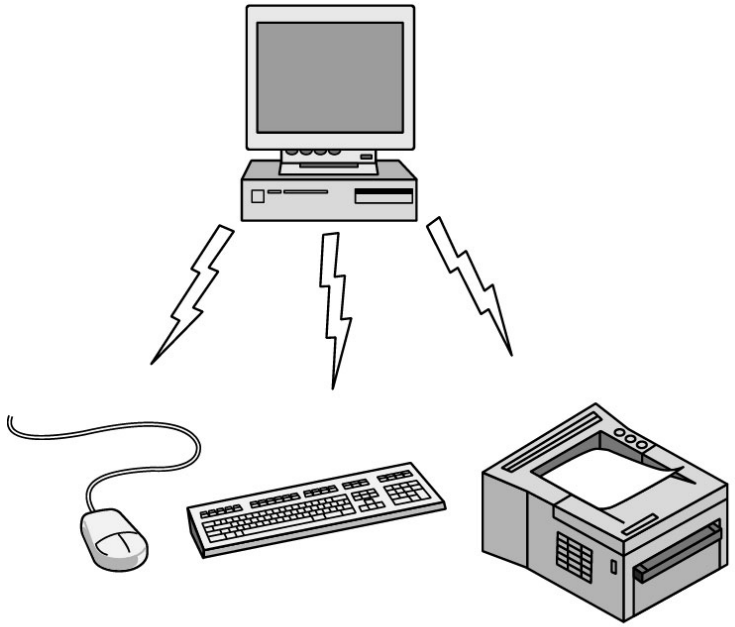


Star Topology:
Ethernet (10-BaseT,
100-Base-TX, 1000-Base-TX)

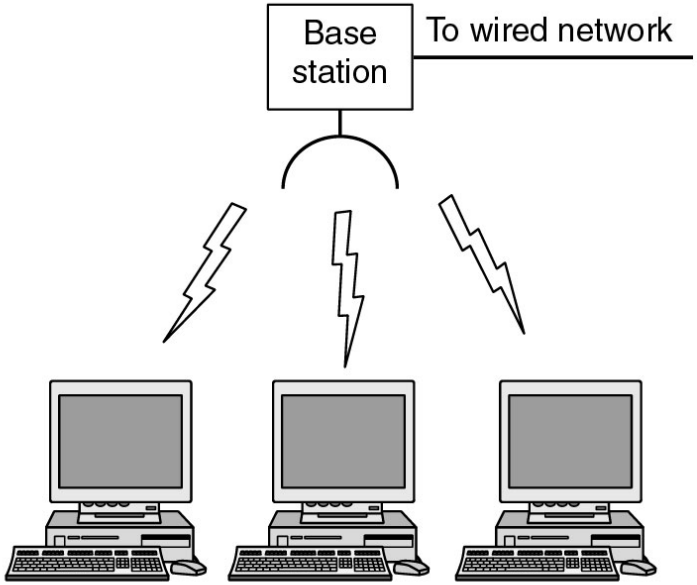


© 2017 Burkhard Stiller

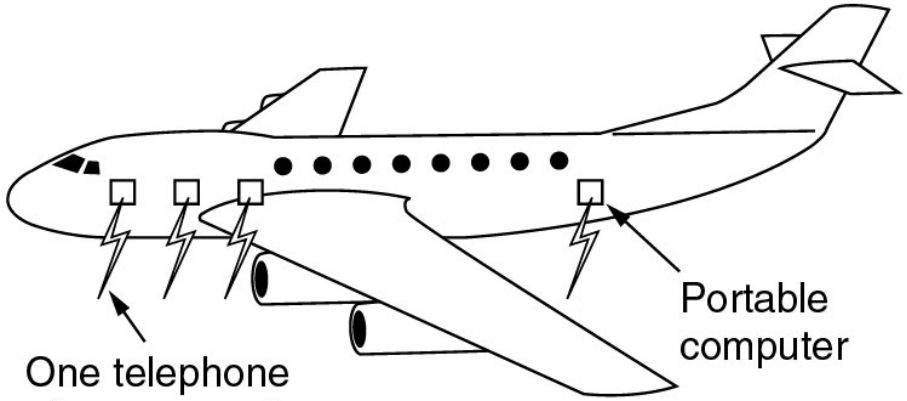
Wireless Networks (Examples)



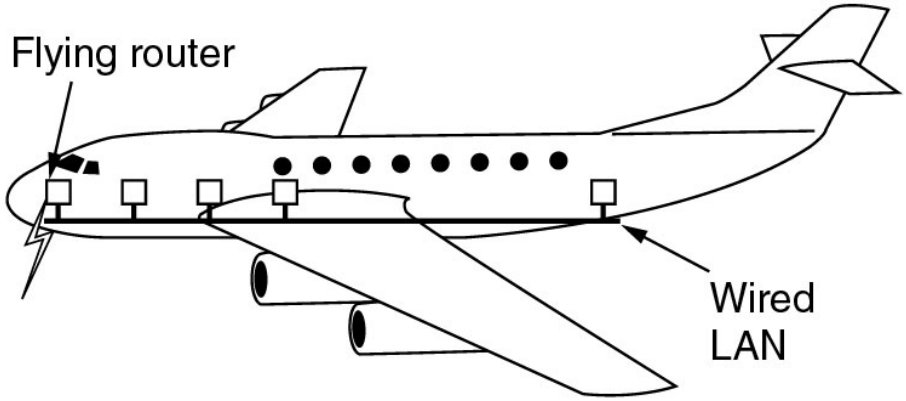
(a)



(b)



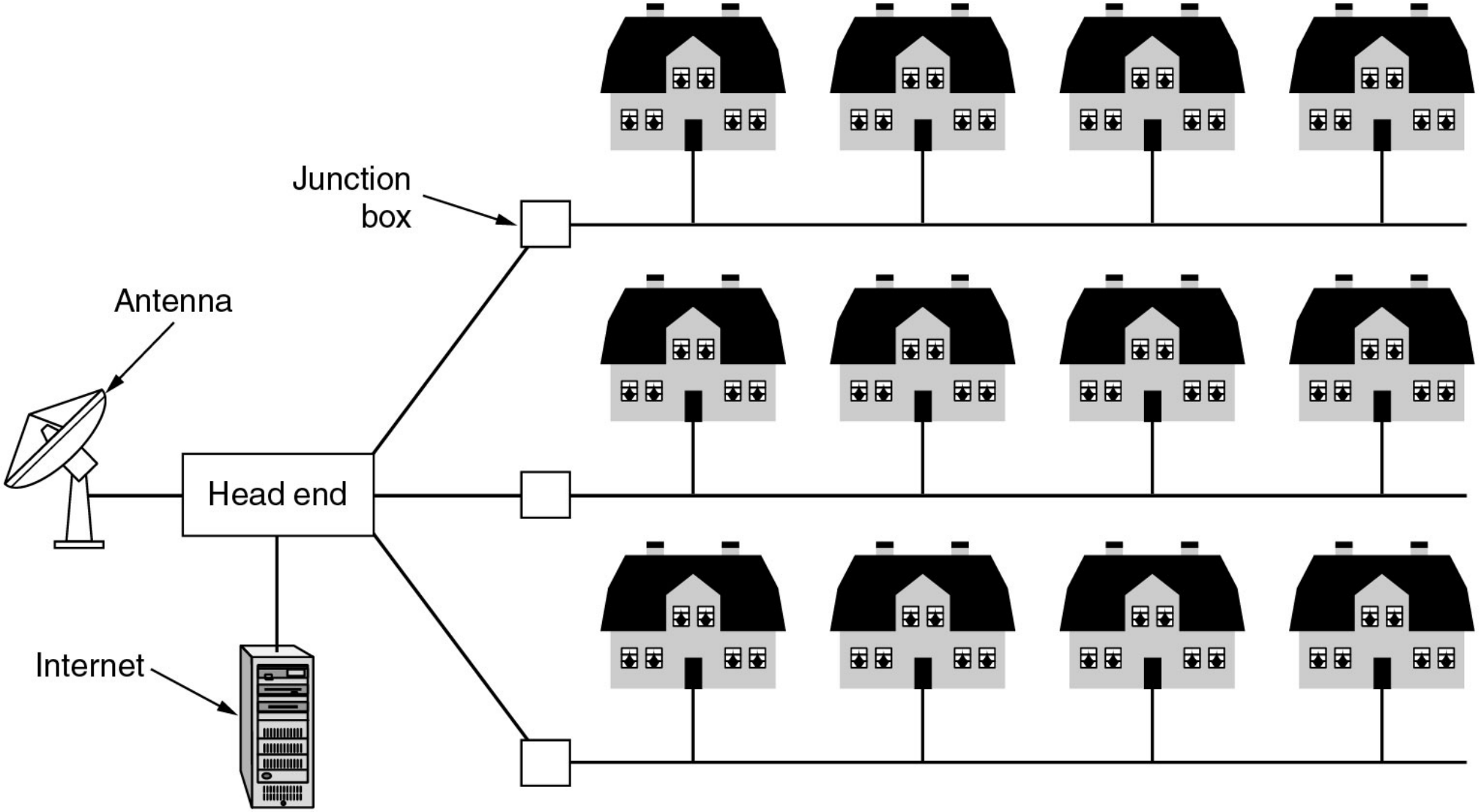
(a)



(b)

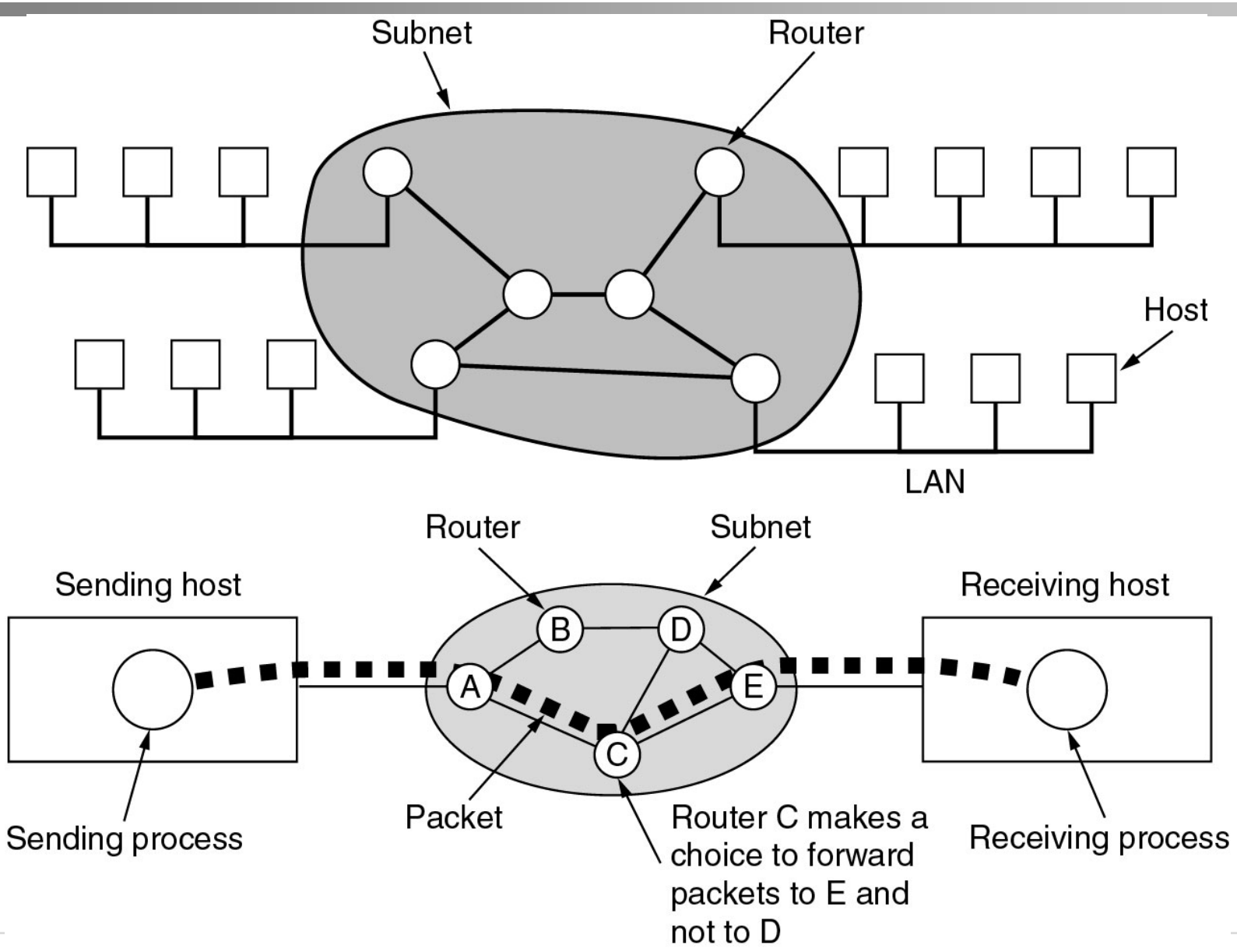
© 2017 Burkhard Stiller

Metropolitan Area Network (Examples)



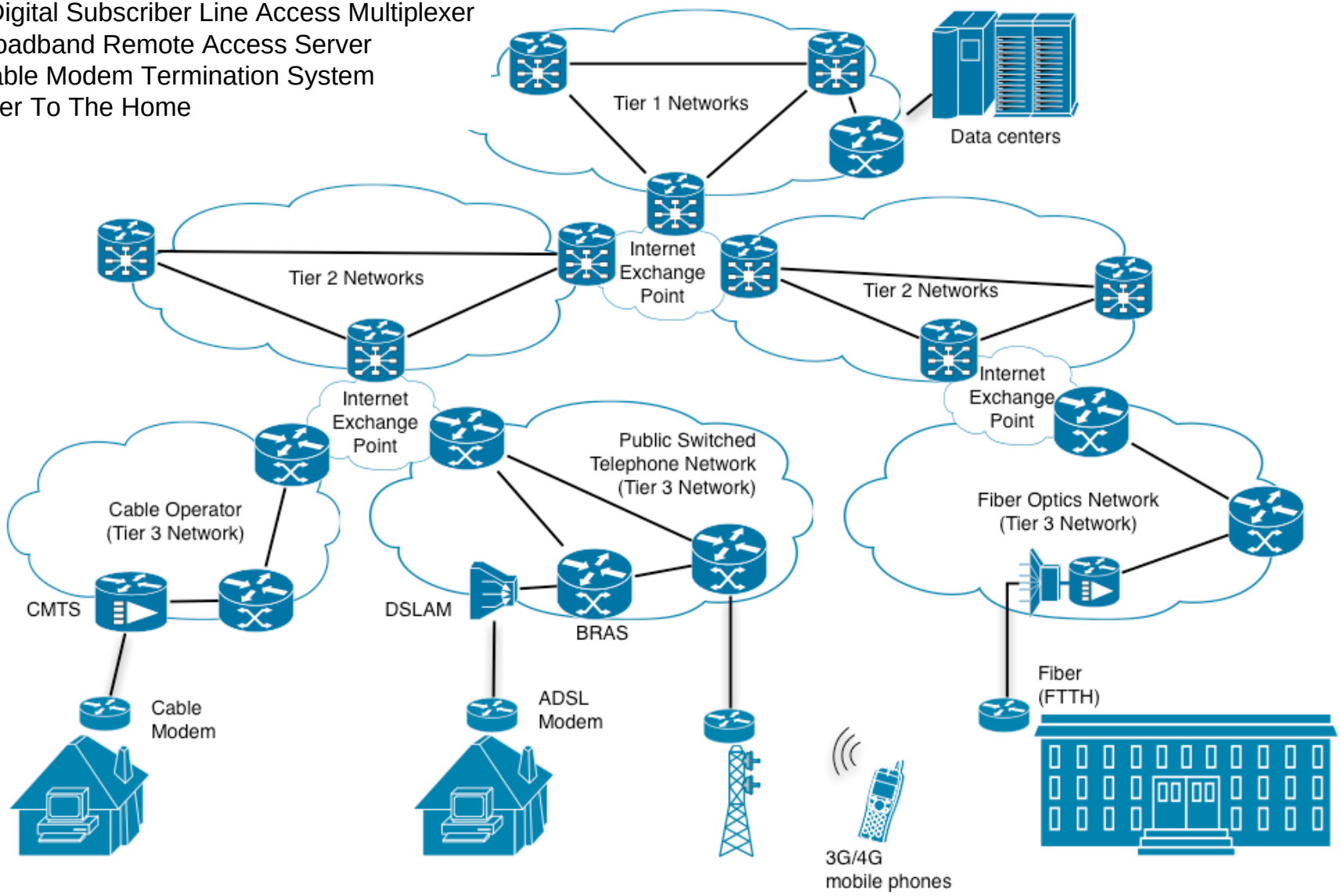
© 2017 Burkhard Stiller

Wide Area Network (Examples)



The Internet – A LAN, MAN, WAN

DSLAM: Digital Subscriber Line Access Multiplexer
BRAS: Broadband Remote Access Server
CMTS: Cable Modem Termination System
FTTH: Fiber To The Home



Standardization (1)

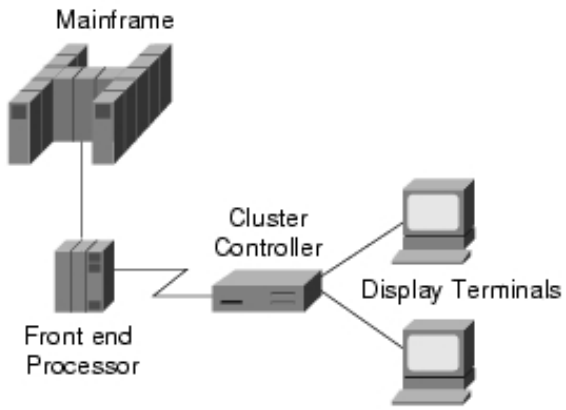
- Standardization is based on joint and public agreements on norms or requirements
  - Standards define what is needed for *interoperability*
  - Standards don't define how (and why) things work
- De facto standards
  - Informal convention or dominant use
  - HTTP and Bluetooth both started as de facto standards but are now also de jure standards
- De jure standards
  - Specified by a formal standardization body
  - Standardization process is *time-consuming* and *costly*
- **Standardization in communications is a MUST! Compatibility (interoperability) of communications!**

© 2017 Burkhard Stiller

Proprietary and Vanished Approaches

- DECNet (Digital Equipment Corporation), 1974
  - Invented to connect DEC-PDP-11 minicomputers
  - Evolved into a hybrid DECNet and OSI compatible protocol suite until 1987
- Systems Network Architecture (IBM), 1974
  - Hierarchical architecture to connect dumb terminals to a mainframe
  - In use until 2000
- Novell NetWare, 1983
  - OS and LAN-based protocol for connecting PCs with 90% market share in early 1990
  - Last release in 2004





© 2017 Burkhard Stiller

- S
  - –
  - –
    -
    -
    -
- S
- S
  -





© 2017 Burk

Example of Standards

- ITU-T
  - 1988: Open Systems Interconnect Model (X.200)
    see also: ISO/IEC 7498-1
  - 1988: Telefax (V.27, V.29, …)
  - 1992: JPEG image compression (T.81, T.83,…)
  - 1999: Asymmetric Digital Subscriber Line (G.992.1)



- ETSI
  - 1990: Digital cellular telecommunications system (GSM 900, phase 1, ...)



- ISO
  - 1975: Paper sizes (ISO 216)
  - 1984: Open Systems Interconnect Model (ISO/IEC 7498-1)
  - 1998: Single-byte character sets (ISO 8859-1, ...)



© 2017 Burkhard Stiller

Standards Bodies (2)

- Standardization of the Internet: protocols, components
  - End-user and task forces with sample implementations:
  - ***IETF***: Internet Engineering Task Force
  - ***IRTF***: Internet Research Task Force
- Standardization of Local- and Metropolitan Area Networks
  - ***IEEE-SA***: Institute of Electrical and Electronics Engineers Standards Association
  - IEEE-SA also works in other areas:
    - Power and energy industries, biomedical and health care, information technology, transportation, nanotechnology etc.

© 2017 Burkhard Stiller

IEEE 802 LAN/MAN Standards

Number	Topic
802.1	**Bridging and Network Management, Overview and Architecture (active)**
802.2	**Logical Link Control (hibernating)**
802.3	**Ethernet (active)**
802.4	Token bus (disbanded, standards withdrawn)
802.5	Token ring (hibernating)
802.6	DQDB, early MAN standard (disbanded, standards withdrawn)
802.10	Virtual LANs and Security (disbanded, standards withdrawn)
802.11	**Wireless LANs; WiFi (active)**
802.14	Cable-TV modems (disbanded, industry consortium got there first!)
802.15	Wireless personal area networks; Bluetooth, ZigBee (active)
802.16	Wireless metropolitan area network; WiMAX (active)

© 2017 Burkhard Stiller

Internet Standards

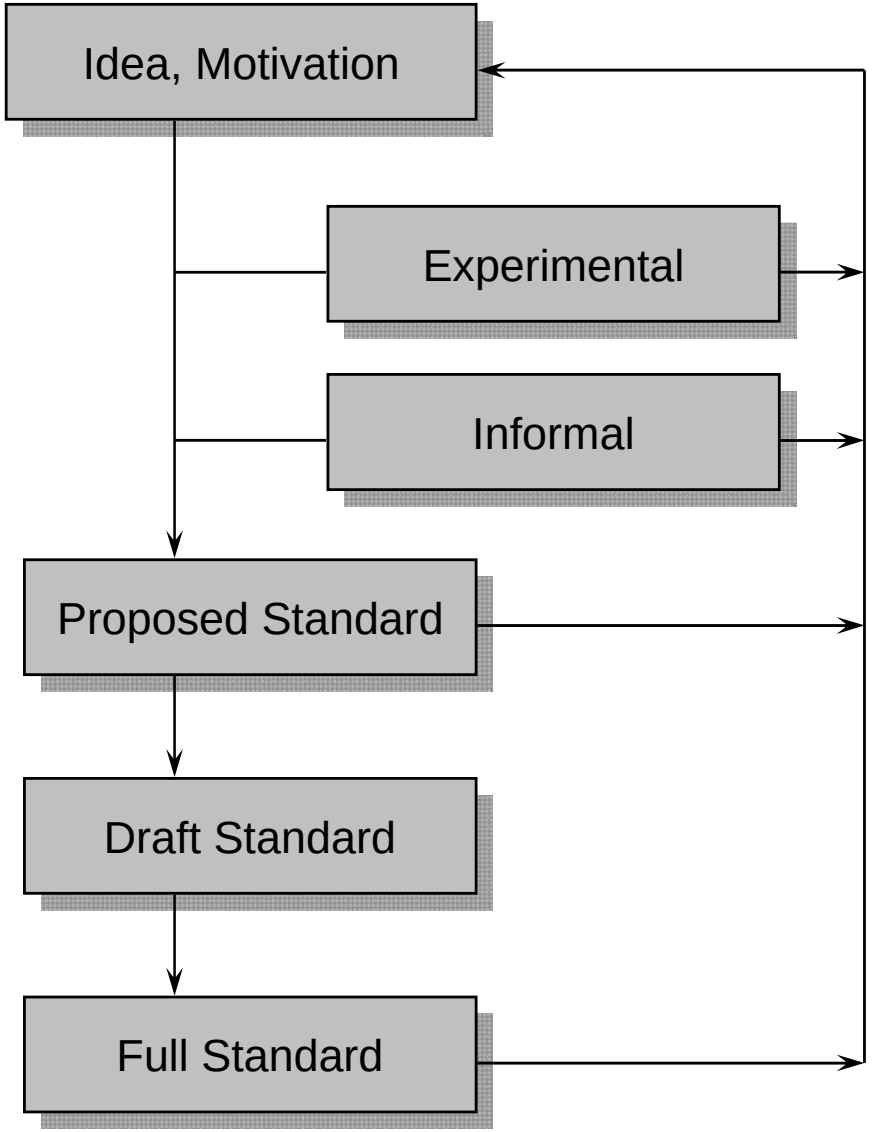
Number	Topic
RFC 768	**User Datagram Protocol (UDP) - Aug 1980**
RFC 791	**Internet Protocol (IP) - Sep 1981**
RFC 792	**Internet Control Message Protocol (ICMP) - Sep 1981**
RFC 793	**Transmission Control Protocol (TCP) - Sep 1981**
RFC 959	File Transfer Protocol (FTP) - Oct 1985
RFC 1149	Standard for the Transmission of IP Datagrams on Avian Carriers – Apr 1990 (Experimental)
RFC 2616	**Hypertext Transfer Protocol -- HTTP/1.1 - Jun 1999**
RFC 6860	Hiding Transit-Only Networks in OSPF – Jan 2013

- Published documents are called „Request for Comments“ (RFC).
- An RFC may define a standard or a best-current-practice or are informational altogether.

© 2017 Burkhard Stiller

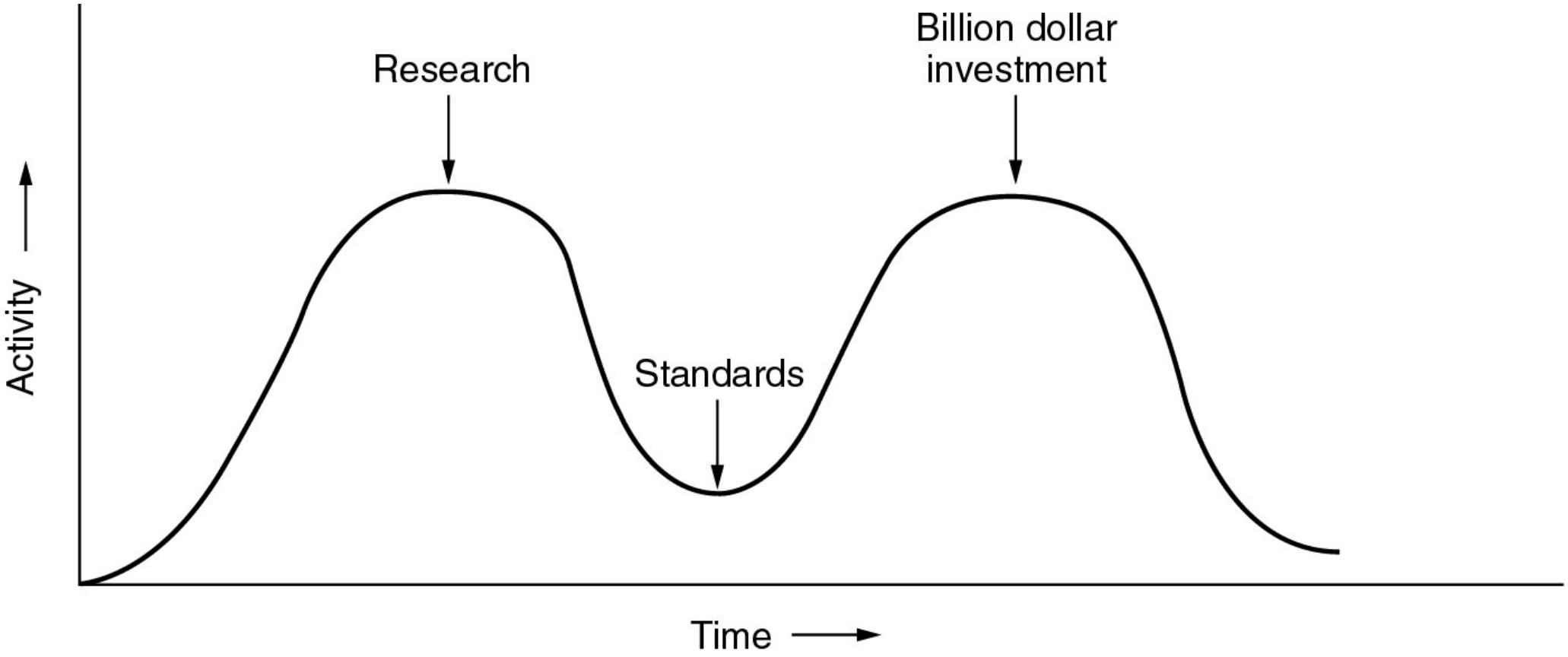
Standardization: Example Internet

- Process operated by the Internet Engineering Task Force (IETF):
  - Internet Engineering Steering Group (IESG) leads discussion
- Two results:
  - Standard
  - Informal/Experimental
- Already for a Draft Standard two independent and interoperable implementations are required



© 2017 Burkhard Stiller

“Apocalypse of Two Elephants”



© 2017 Burkhard Stiller

7. Architectures, Services, and Protocols

Communication Characteristics

Architectures

Service and Protocol

Internet and Layers

Network Examples

© 2017 Burkhard Stiller

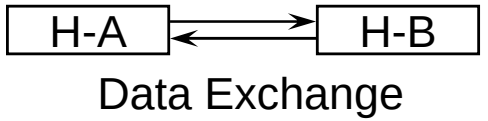
Characteristics

Communication Partners – Hosts (1)

- Communications define the technical approach of communicating messages between participating partners
  - 4 major ***characteristics*** exist

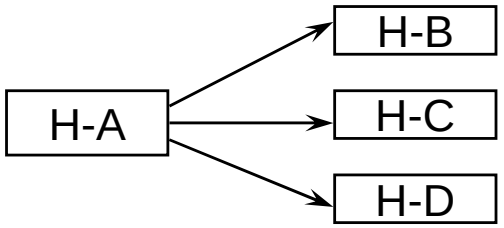
1. Communication Partners

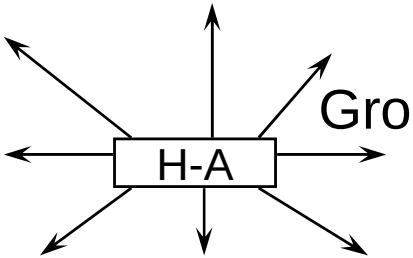
- Dialog: Two hosts exchange data across a point-to-point link
- Multicast: A single host addresses multiple (known) other hosts
- Broadcast: A single hosts addresses many (regularly unknown) other hosts



Data Exchange

Individual communications





Group/Mass communications

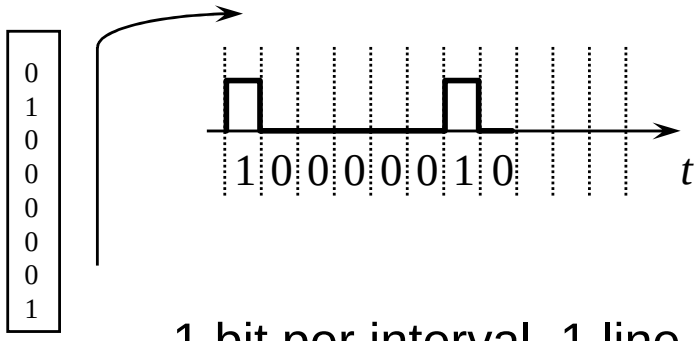
H: Host

© 2017 Burkhard Stiller

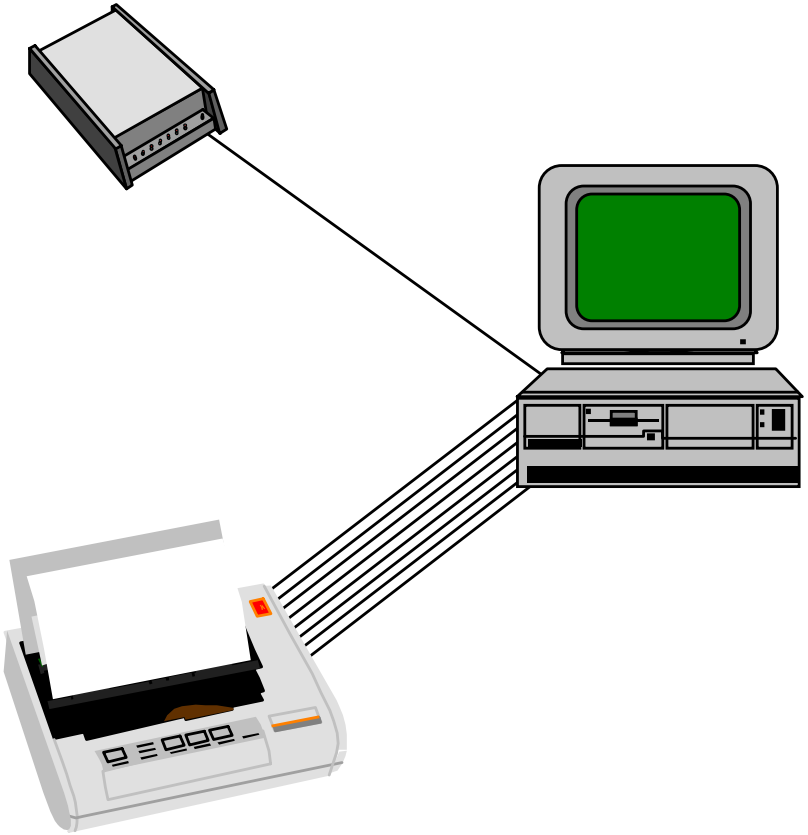
ifi

2. Transmission Schemes (2)

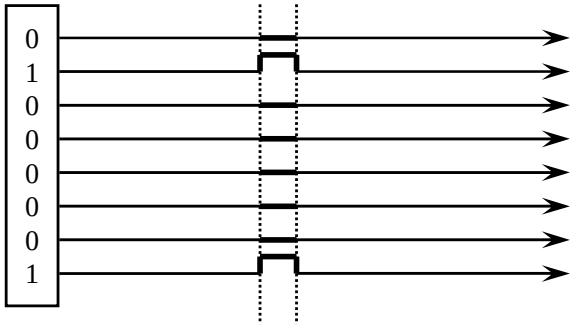
– Serial transmission



1 bit per interval, 1 line



– Parallel transmission

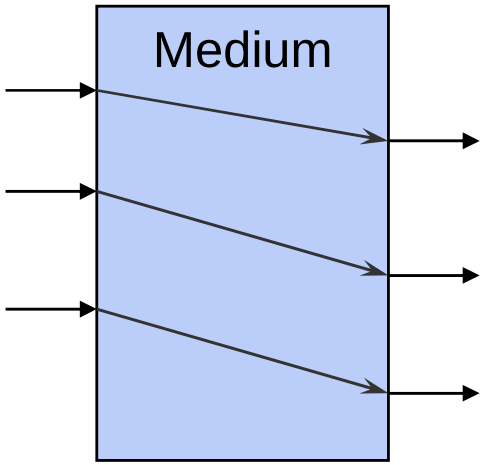


8 bit per interval, 8 lines

© 2017 Burkhard Stiller

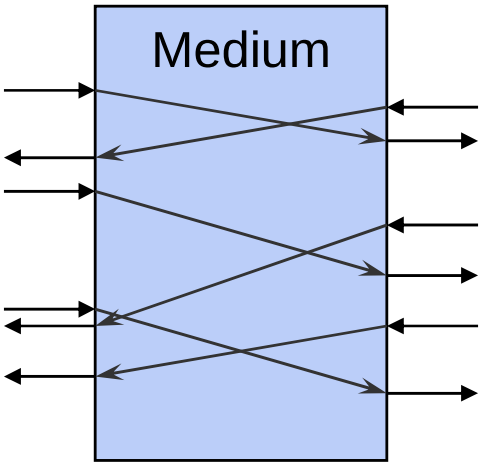
3. Directions of Use (3)

Simplex



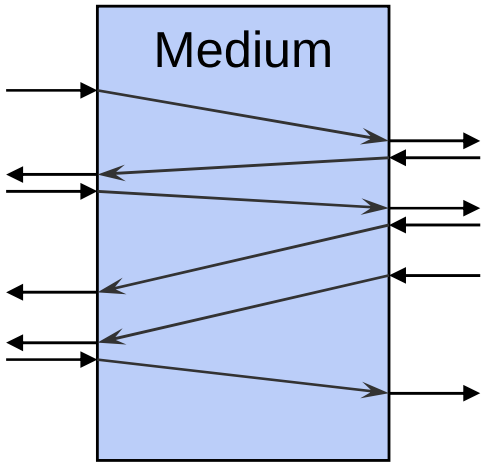
- Fire alarm
- Sensors
- Pager

Duplex



- Phone

Half duplex



- Inter phone
- Some GSM connections

GSM: Global System for Mobile Communications

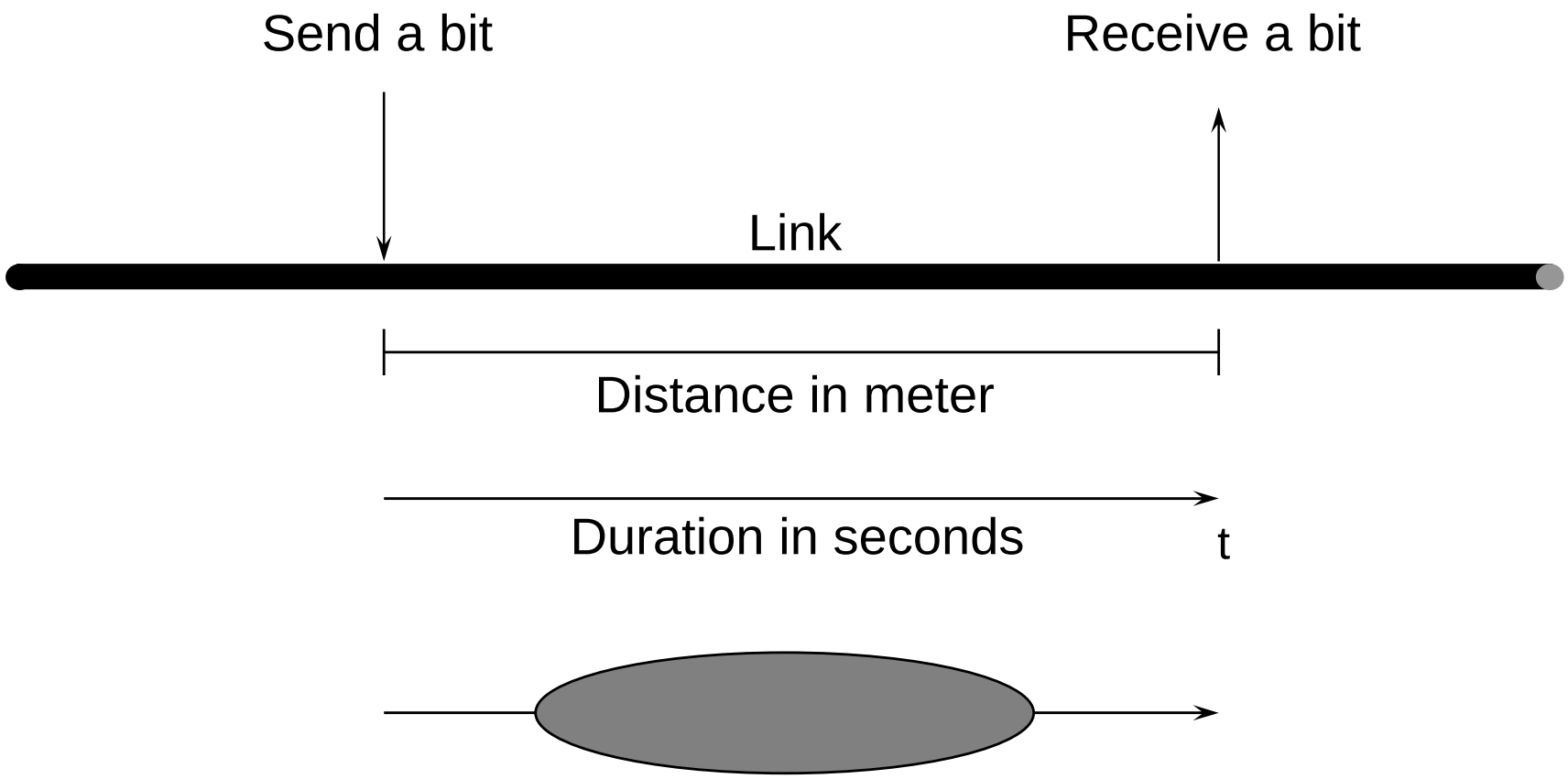
© 2017 Burkhard Stiller

4. Quality of Communications (4)

- User requirements on communications include:
  - Adequacy: ease-of-use, satisfaction of needs
  - Subjective ratings (MOS: Mean Opinion Score)
- Technical requirements cover:
  - Performance: Bandwidth or capacity, latency or delay, throughput, sending/receiving rate ….
  - Reliability: Error tolerance, resilience (Ausfallsicherheit), noise immunity (Störunanfälligkeit), availability (Verfügbarkeit) ...
  - Security: Integrity, privacy, authentication, authorization, denial-of-service, non-repudiation (Nichtabstreitbarkeit) …
  - Safety
  - Costs: Investment, operation, maintenance costs …

© 2017 Burkhard Stiller

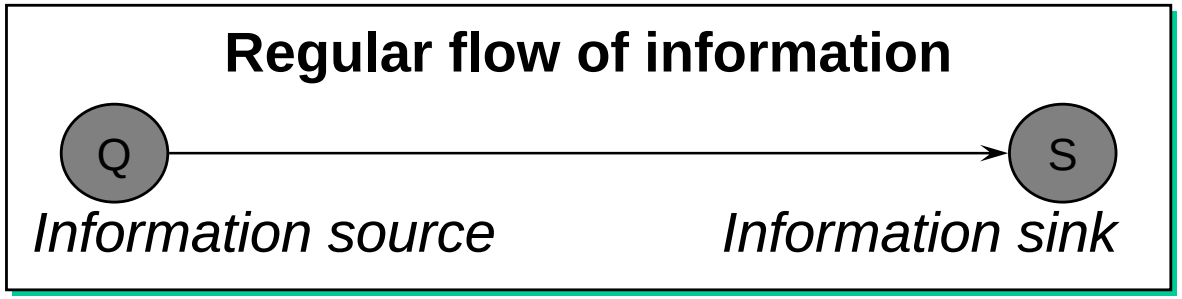
Quality Example (1): Technical Performance



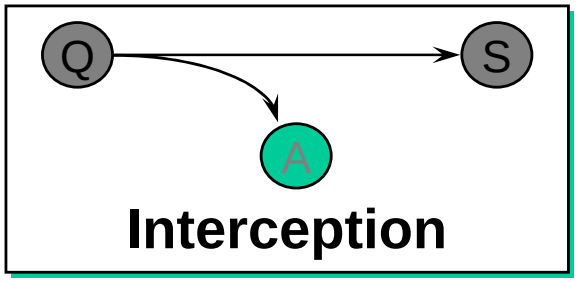
Throughput (or: capacity) =
Number of bits transmitted per second [unit bit/s]

Quality Example (2): Security

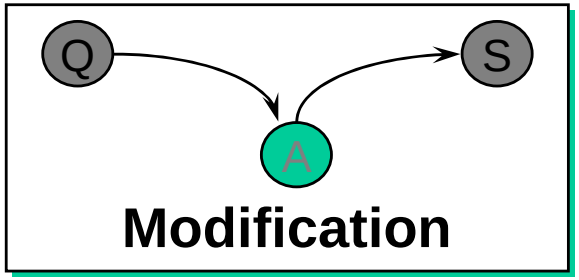
- Security
  - Encryption (cryptographic codes)
  - Trustworthy systems (authentication, authorization)

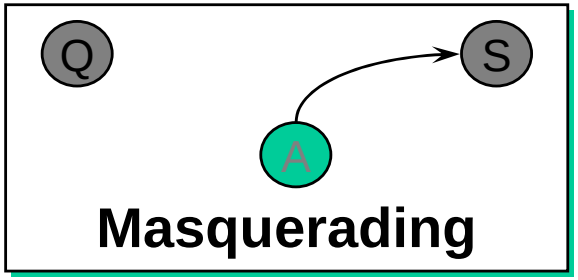


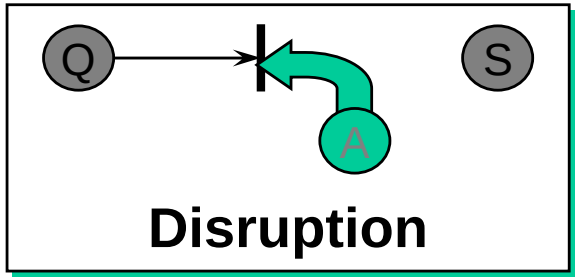
Passive:



Active:

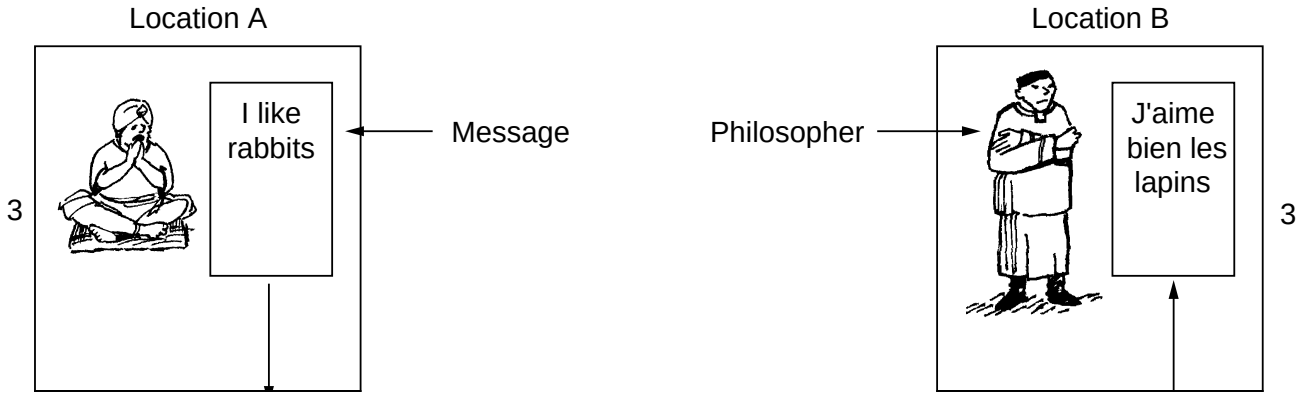






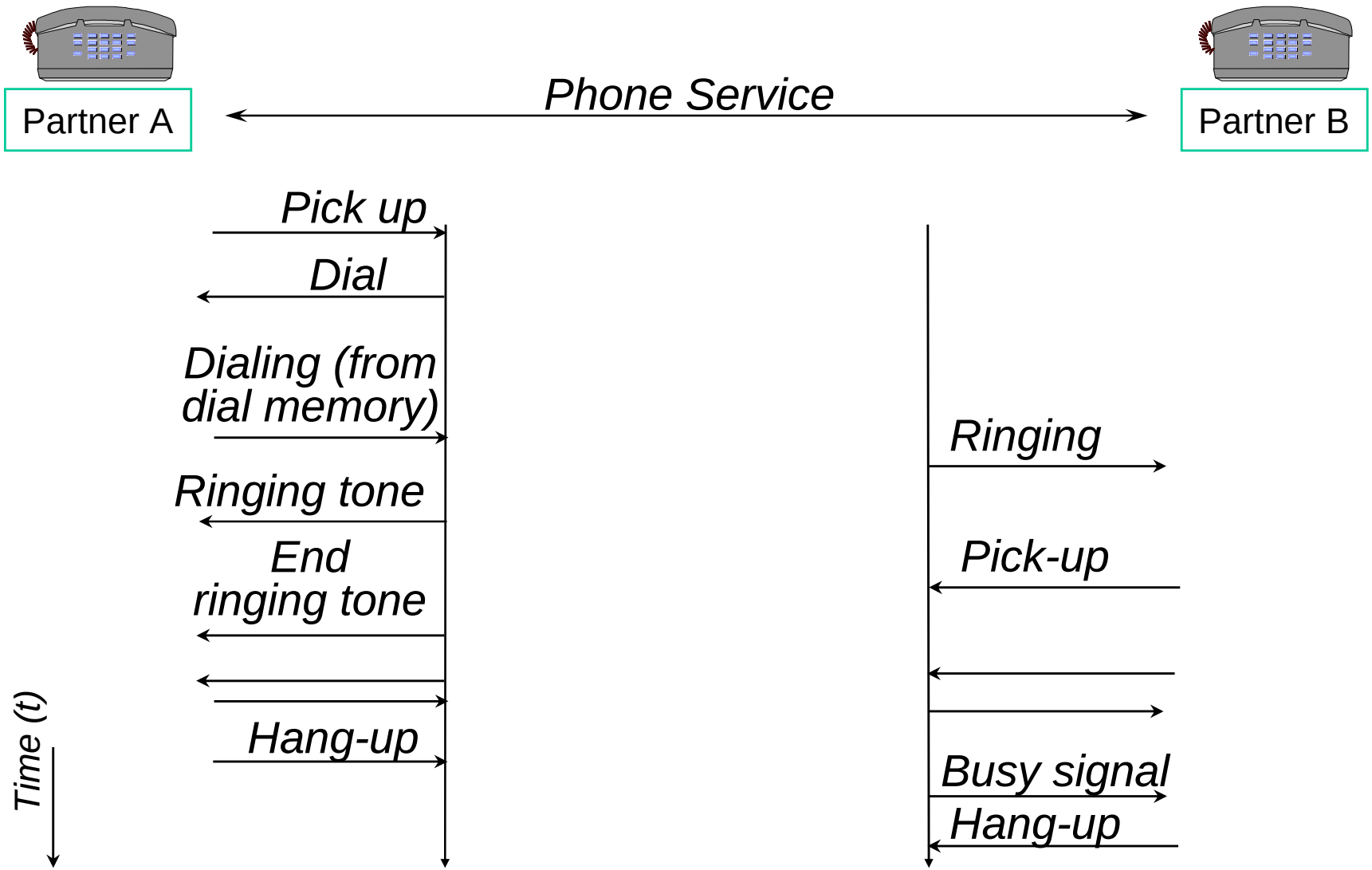
© 2017 Burkhard Stiller

Communications in Daily Life



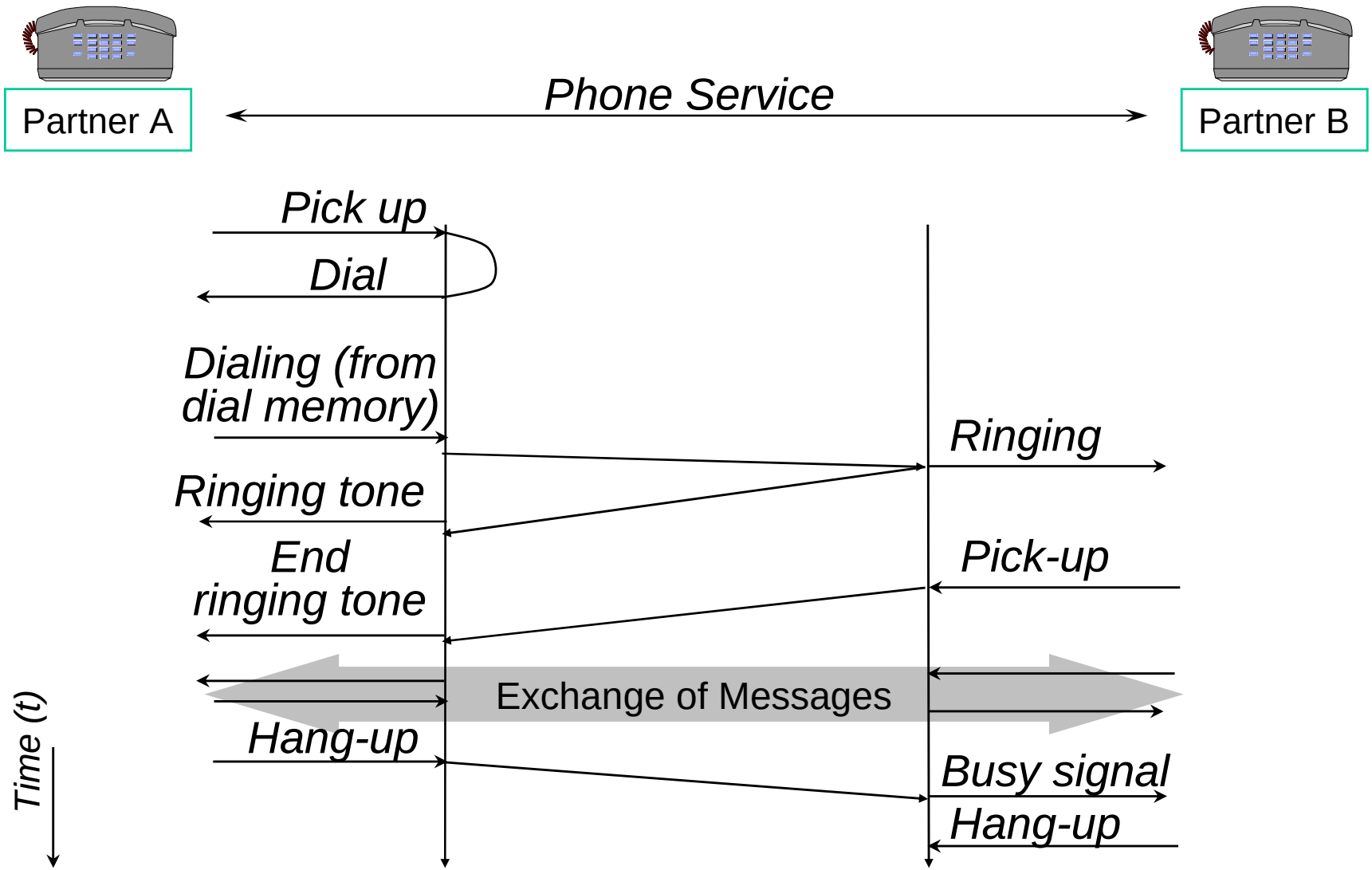
© 2017 Burkhard Stiller

Example Phone – Service User View (1)



© 2017 Burkhard Stiller

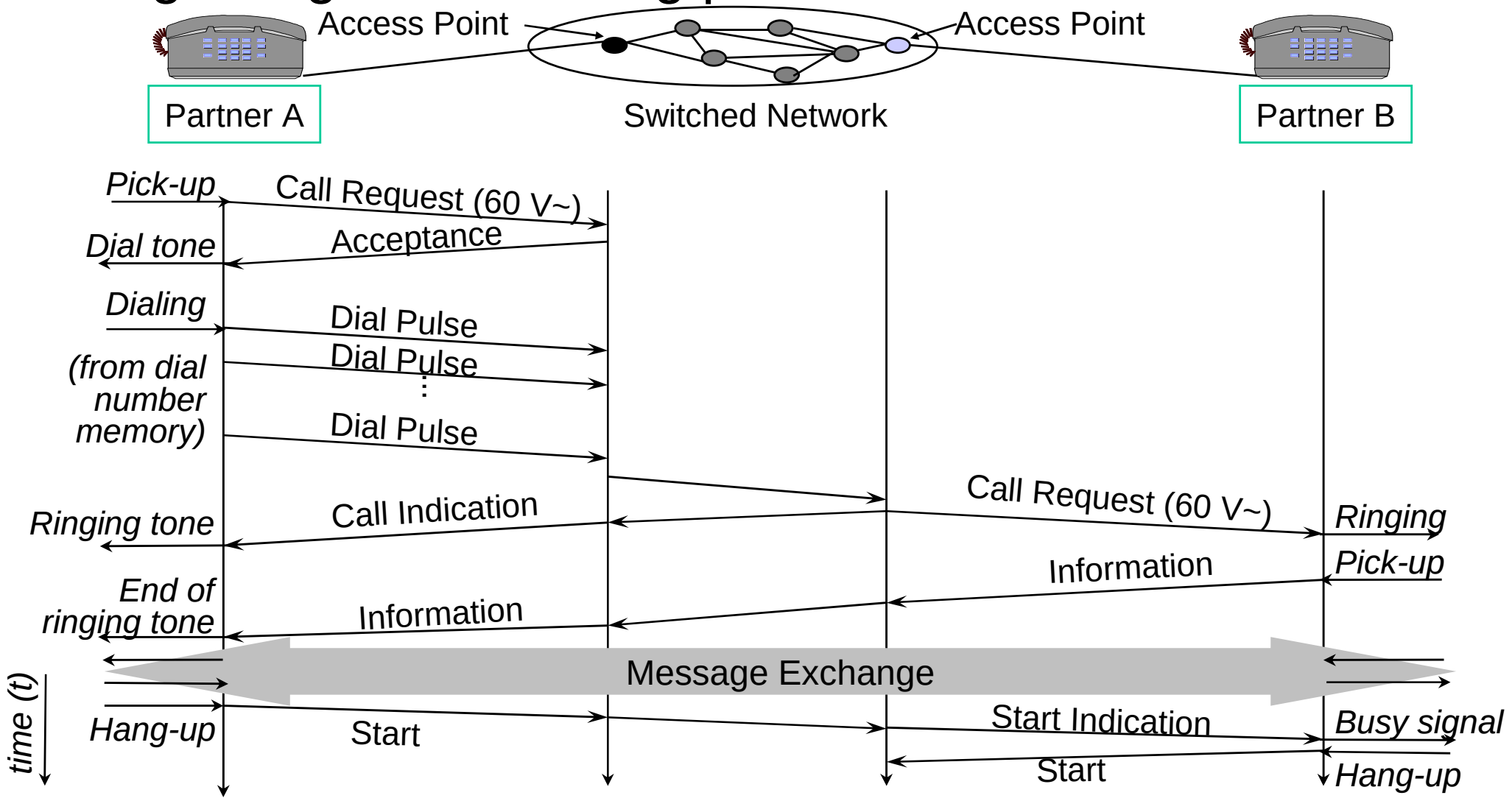
Example Phone – Service User View (2)



© 2017 Burkhard Stiller

Example Phone – Service and Protocol

- Signaling in the analog phone network:



© 2017 Burkhard Stiller

ISO/OSI Basic Reference Model (1)

Anwendungsschicht

Darstellungsschicht

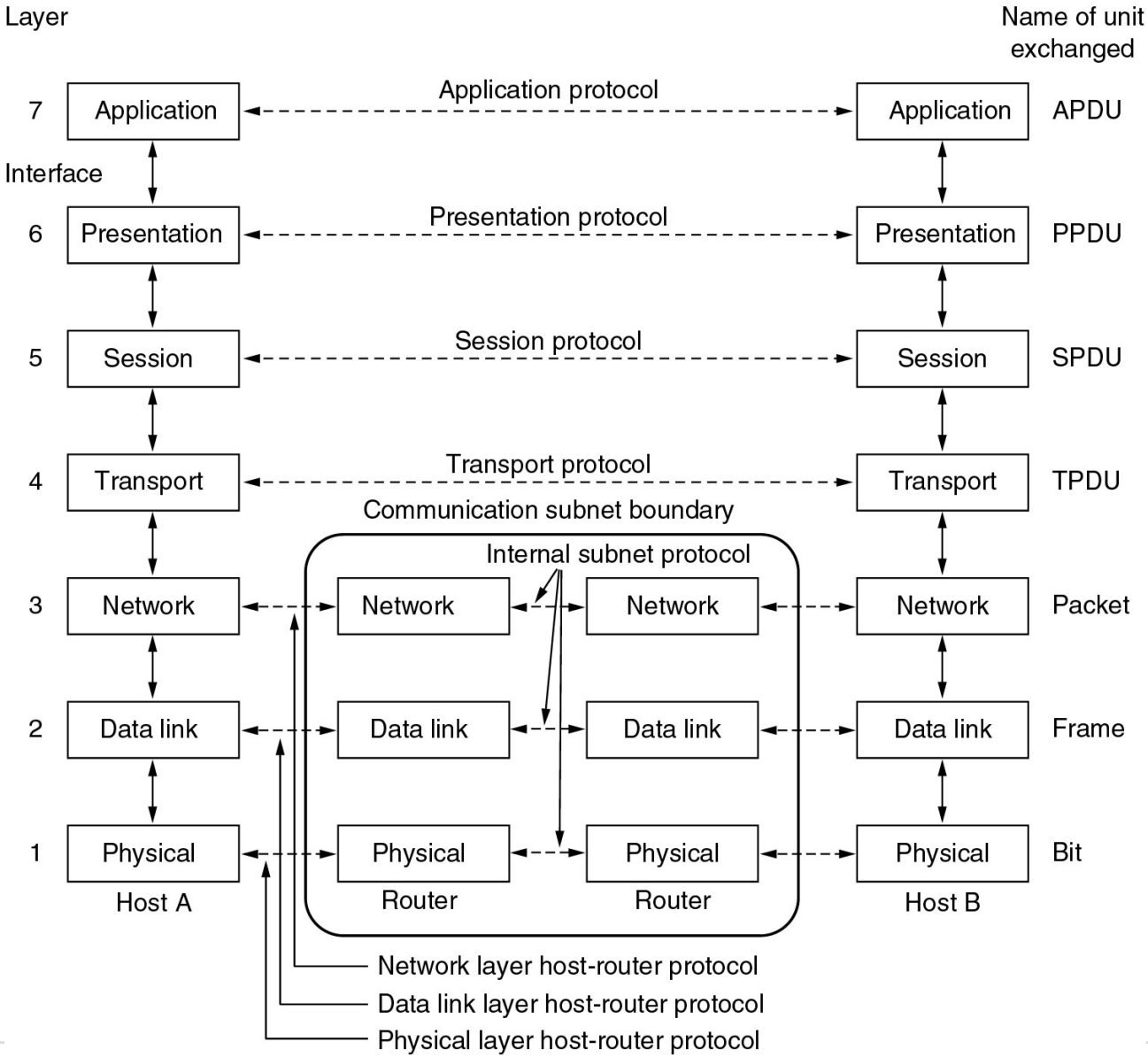
Kommunikations-steuerungsschicht

Transportschicht

Vermittlungsschicht

Sicherungsschicht

Bitübertragungsschicht



© 2017 Burkhard Stiller

ifi

ISO/OSI Basic Reference Model (2)

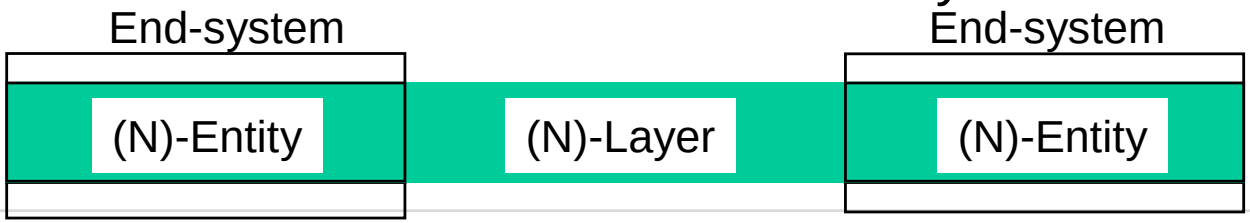
- International standardization of services and protocols to implement "Open System Interconnection (OSI)":
  - Basis for enabling the communication for systems of different vendors
  - BRM serves as a guideline to understand and model all tasks for communications
  - Major implementations existed in Europe in the late 80's
  - The Internet model inherited from there, took over in the 90's

  - ISO/IEC IS 7498: Information Processing Systems - Open Systems Interconnection - Basic Reference Model, International Standard, October 15, 1984
  - Included in ITU-T Recommendation X.200

© 2017 Burkhard Stiller

Basic Principles of the ISO/OSI BRM

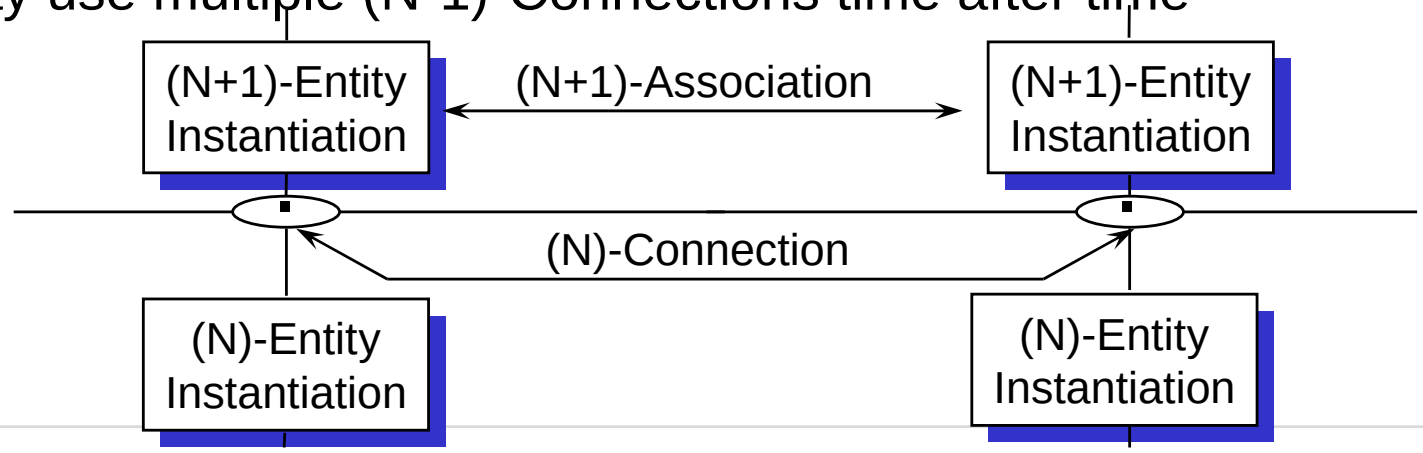
- OSI end-system:
  - System, which operates according to OSI standards
- (N)-Layer:
  - All units of an (N)-Hierarchy in all end-systems
- (N)-Entity:
  - Implementation of an (N)-Service in an end-system
  - Many types of (N)-Entities may exist ((N)-Entity Type), *e.g.*, to implement different (N)-Layer protocols, a copy of an (N)-Entity is termed (N)-Entity Instantiation ((N)-Instanz-Vorkommniß).
- Peer Entities:
  - Peer entities deliver functions of a service by data exchange



© 2017 Burkhard Stiller

Connection and Association

- (N)-Connection:
  - Relation between two (or more) (N+1)-Entity Instantiations in an (N)-Layer being supported by an (N)-Protocol
- (N)-Association:
  - Cooperative relation between two (N)-Entity Instantiations including the management of state information
  - (N)-Association being supported by (N-1)-Connections (or by a (N-1) connectionless service)
  - May use multiple (N-1)-Connections time after time

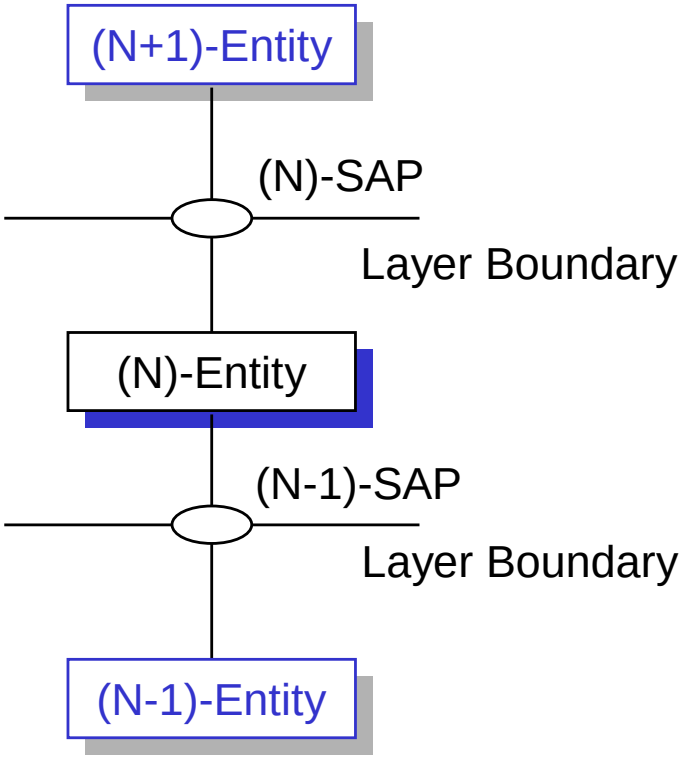


© 2017 Burkhard Stiller

(N)-Service Access Point — (N)-SAP

- Within layered communication systems (N+1)-Entities communicate across (N)-SAPs (Service Access Points)
  - An (N)-Entity offers its (N)-Service at the (N)-SAP
  - An (N)-Entity utilizes services, which are offered at the (N-1)-SAP

Relations between (N-1)-SAP, (N)-Entity, and (N)-SAP

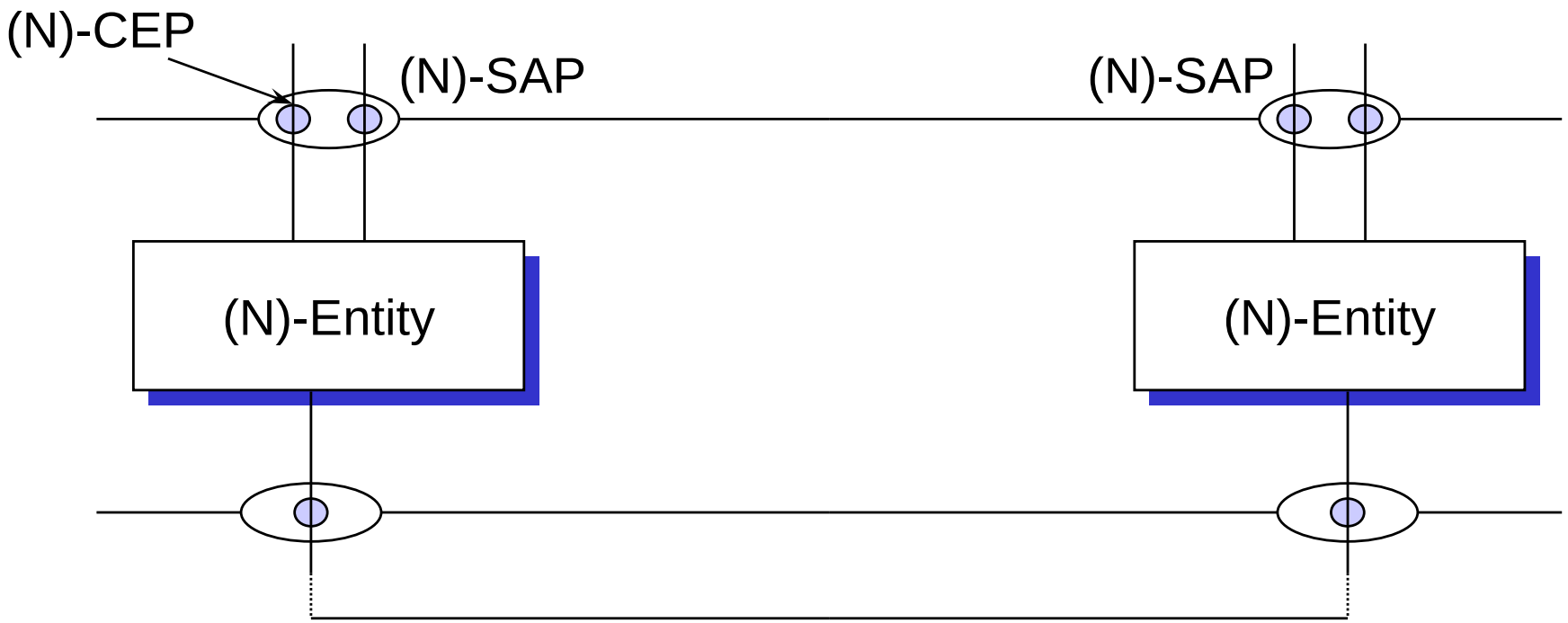


Internet example: A SAP is defined by an IP address

© 2017 Burkhard Stiller

(N)-Connection End Point

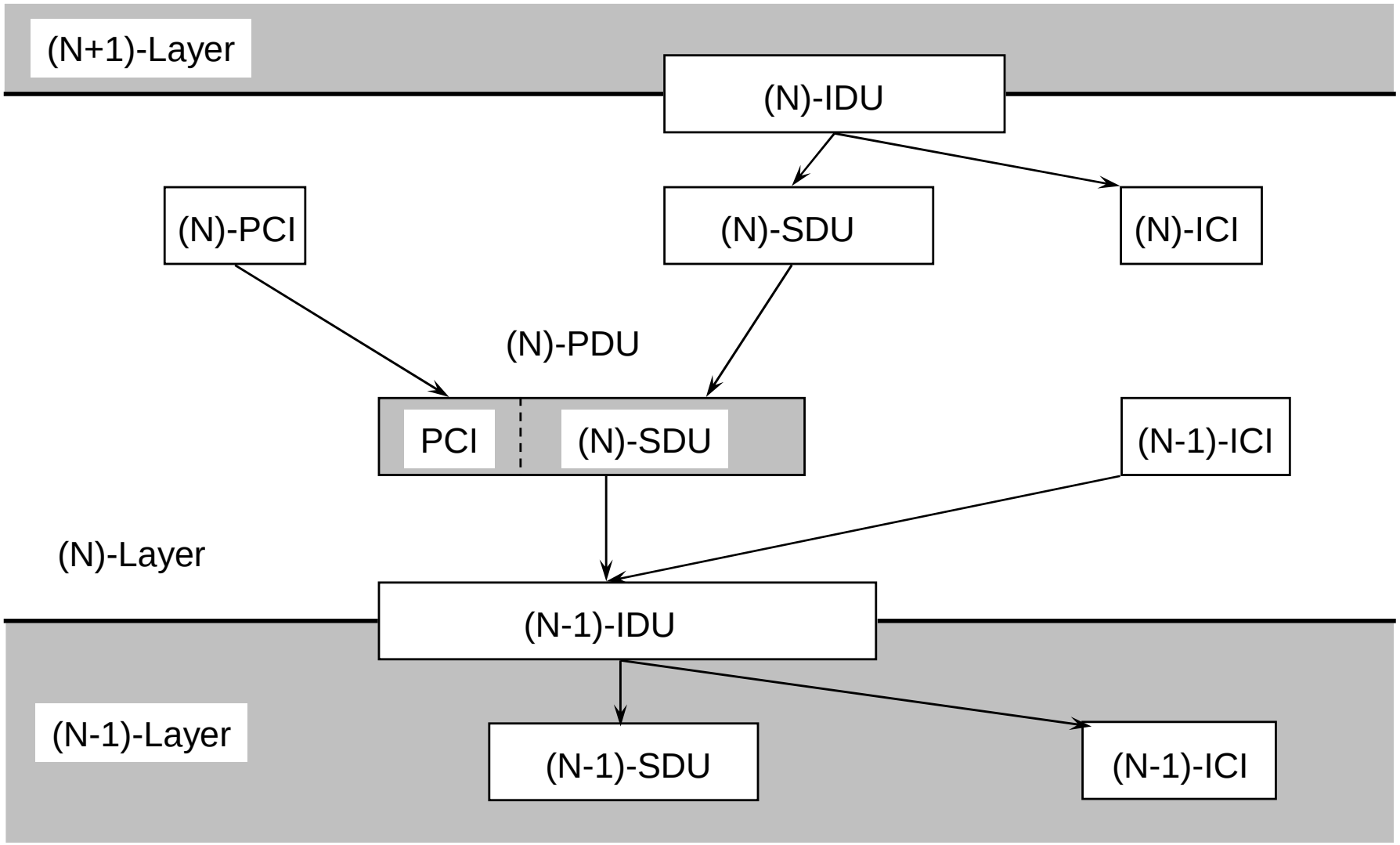
- An (N)-Connection End Point ((N)-CEP) determines the end point of an (N)-Connection within an (N)-SAP



Internet example: A CEP is defined by a port number

© 2017 Burkhard Stiller

Generic OSI Communication Data Units (1)



© 2017 Burkhard Stiller

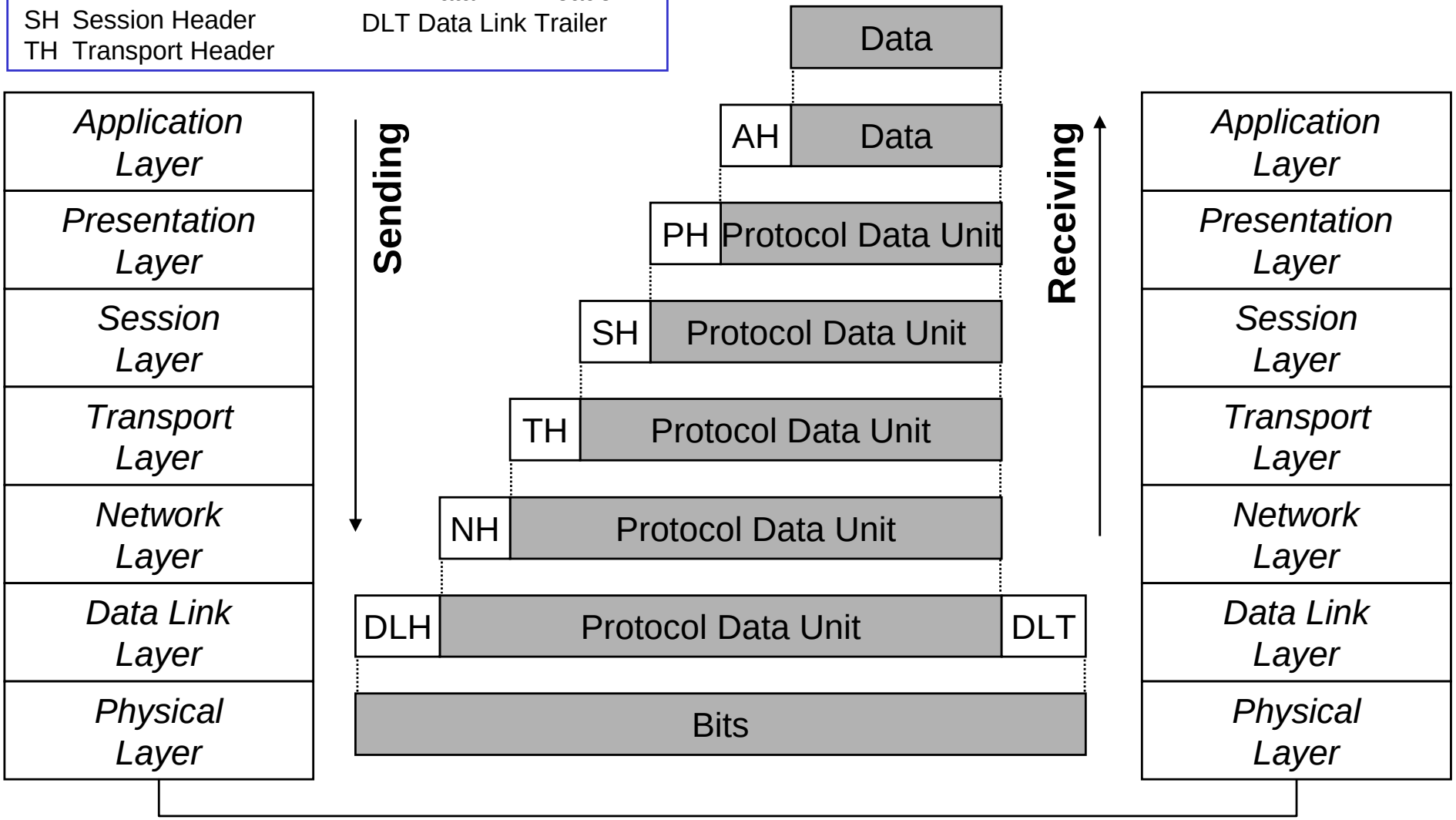
Generic OSI Communication Data Units (2)

- *(N)-PDU*
  - Protocol Data Unit, PDU
  - Data unit exchanged between (N)-Entities by applying a (N-1)-Layer service
  - Combined out of (N)-PCI and (N)-SDU
  - Equals an (N-1)-SDU
- *(N)-IDU*
  - Interface Data Unit, IDU
  - PDU exchanged between (N+1)- and (N)-Entity at an (N)-SAP
  - Combined out of (N)-ICI and (N)-SDU
- *(N)-SDU*
  - Service Data Unit, SDU
  - Data being transmitted between (N)-SAPs transparently
- *(N)-PCI*
  - Protocol Control Information, PCI
  - Data exchanged between (N)-Entities to control the execution of operations, *e.g.,* sequencing
- *(N)-ICI*
  - Interface Control Information, ICI
  - Parameter exchanged between (N)-Layer and (N+1)-Layer to manage service functions, *e.g.,* addresses

© 2017 Burkhard Stiller

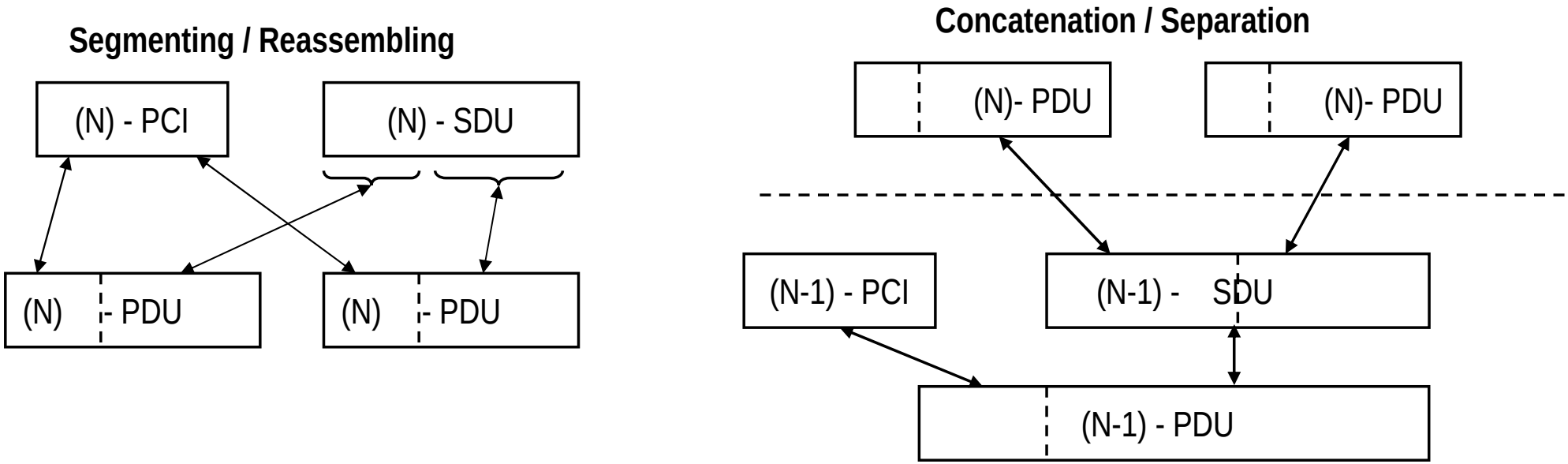
Encapsulation of Data

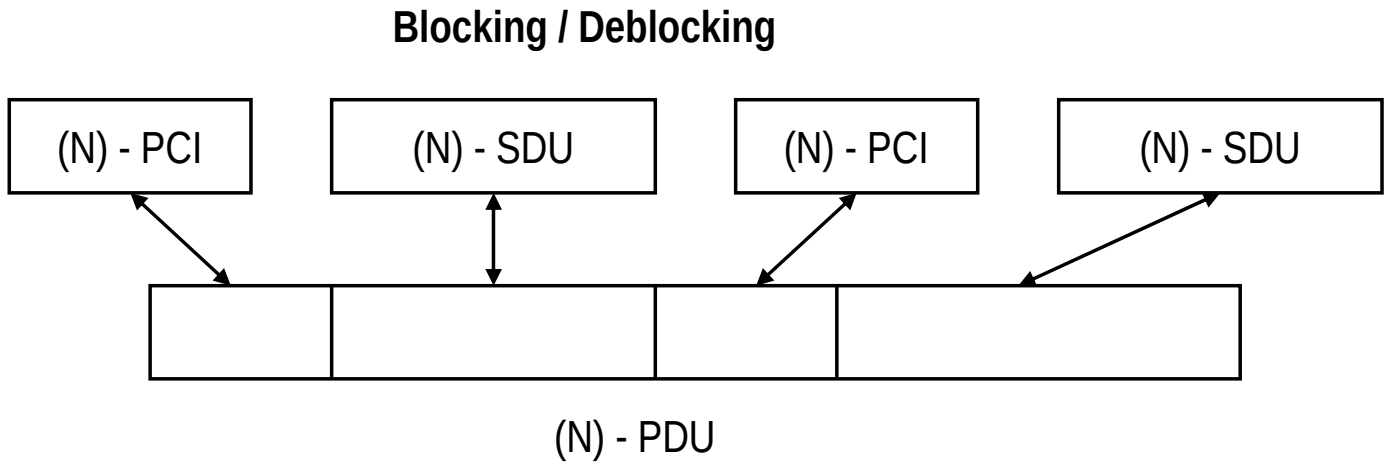
AH Application Header
PH Presentation Header
SH Session Header
TH Transport Header
NH Network Header
DLH Data Link Header
DLT Data Link Trailer



© 2017 Burkhard Stiller

Handling of Data Units

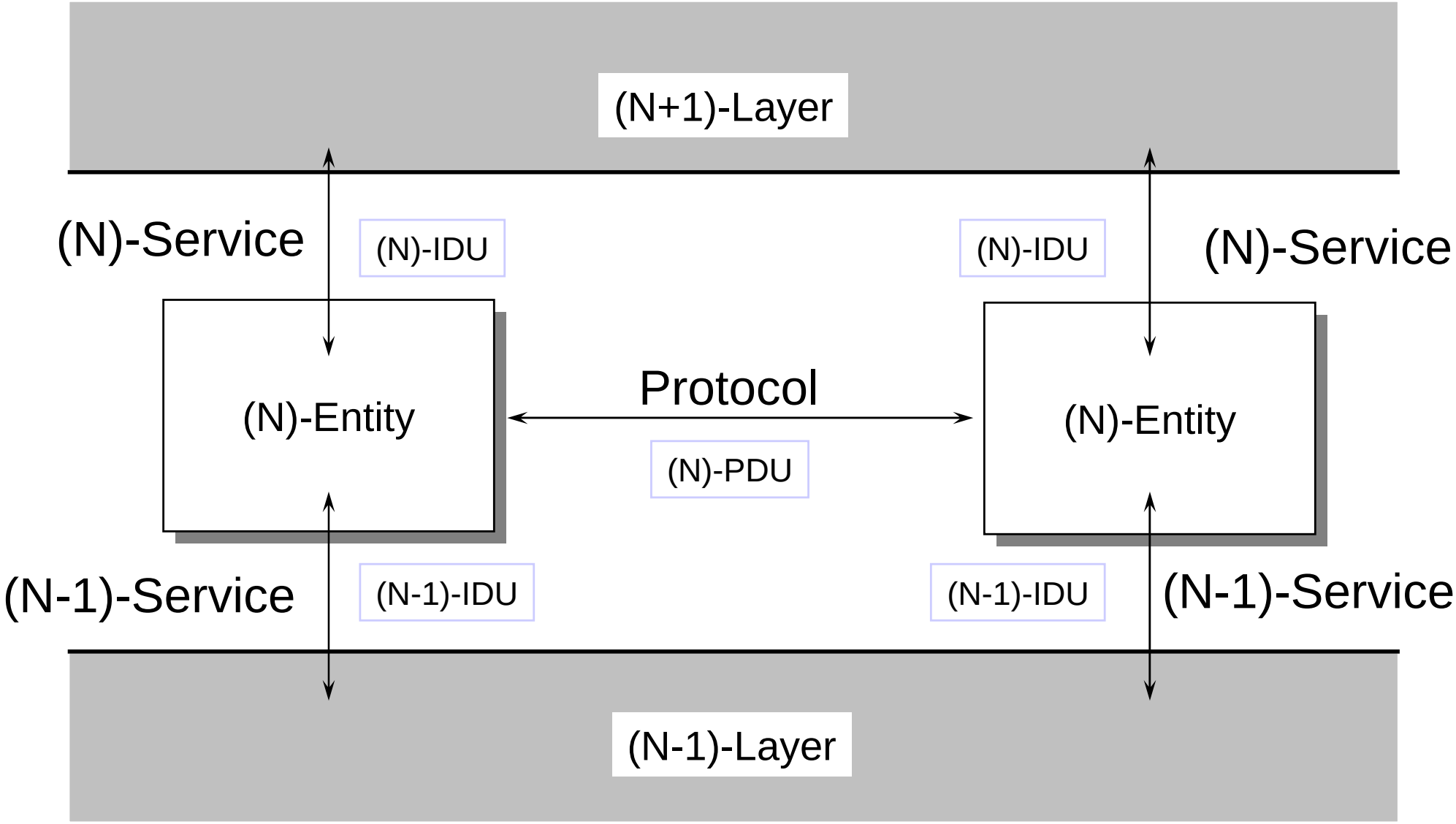




© 2017 Burkhard Stiller

Communication in/between OSI Systems

Service, Protocol



© 2017 Burkhard Stiller

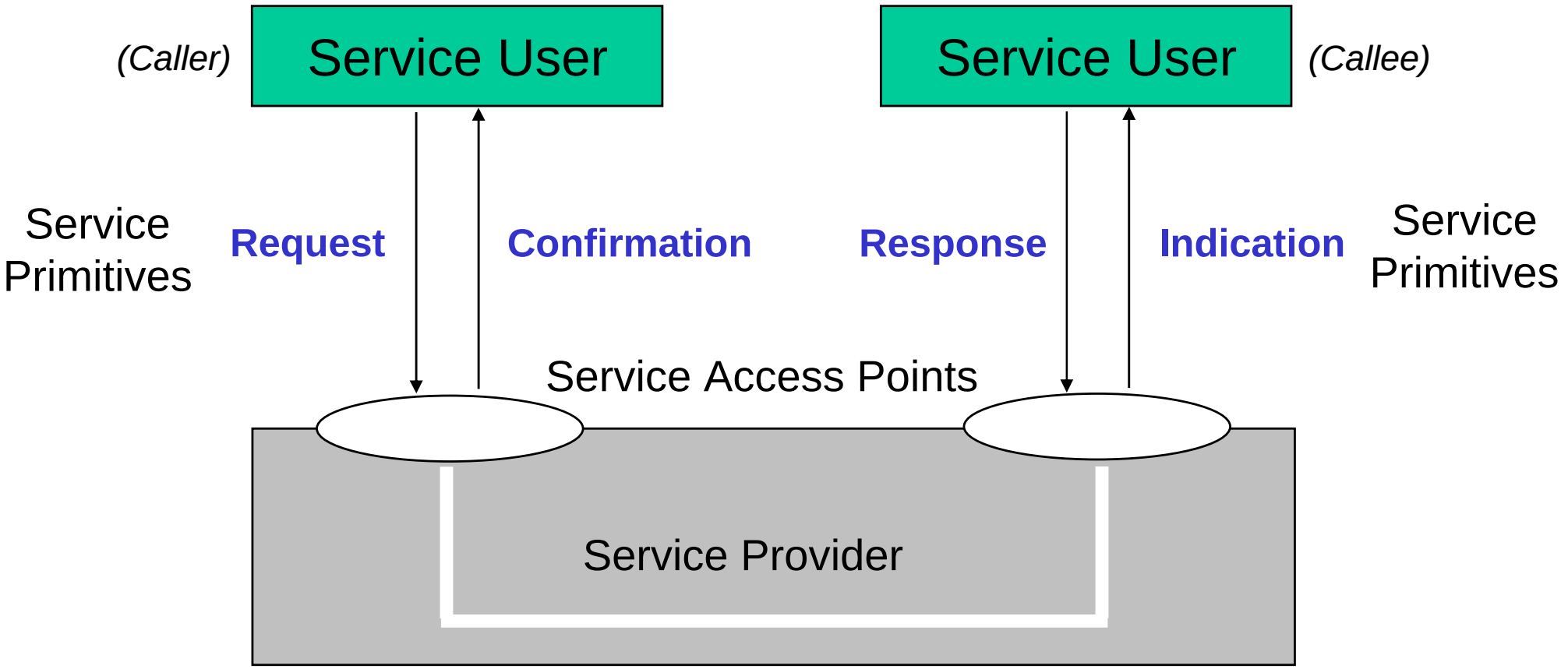
ifi

Service and Protocol (1)

- Single layer offers functionality in terms of services
  - Services provided by exchanging data between partner ***entities*** based on rules and formats defined in protocols.
    - A service is offered from a ***service provider*** at the Service Access Point (SAP) to a ***service user***.
  - (N)-Service:
    - Set of functions offered by the (N)-Layer to the (N+1)-Entities at the interface between the (N)-Layer and (N+1)-Layer
    - All details of providing services of the (N)-Layer are transparently hidden from the (N+1)-Layer
- A service definition includes services offered and their use specifications in the upper layer.
  - A service primitive defines this interaction in detail

© 2017 Burkhard Stiller

Service and Protocol (2)



(General view)

© 2017 Burkhard Stiller

Service Primitives – Naming

- Fields of a service primitive:

Layer/Application Name	Service	Service Type	Parameter
Physical (Ph) Data Link (DL) Network (N) Transport (T) HTTP FTP ...	Connect (Con) Data (Dat) Release (Rel) Abort (Abo) Provider Abort (PAbo) Disconnect (Dis) ...	Request (Req) Indication (Ind) Response (Rsp) Confirmation (Cnf)	*(defined)*

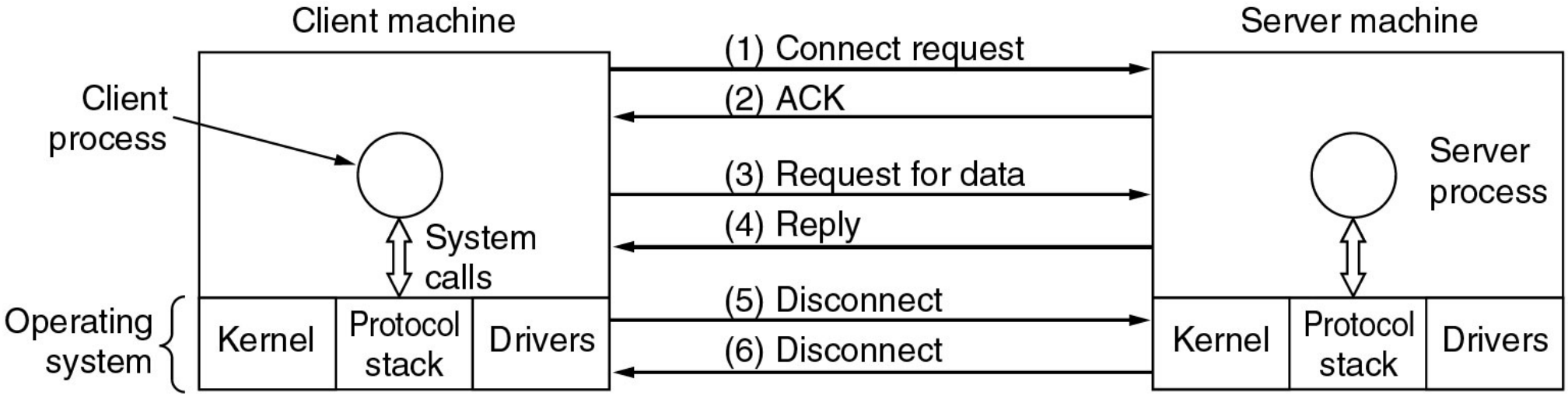
- Examples:
  - T-Con.Req(*Adresses*) = Connection request at the interface to the transport service
  - HTTP-Get.[Req](*URL*) = Request of an HTML page being identified by the *URL*.

(Specific instances)

© 2017 Burkhard Stiller

Service Primitives — Implementation

Primitive	Meaning
LISTEN	Block waiting for an incoming connection
CONNECT	Establish a connection with a waiting peer
RECEIVE	Block waiting for an incoming message
SEND	Send a message to the peer
DISCONNECT	Terminate a connection



(Implementation view)

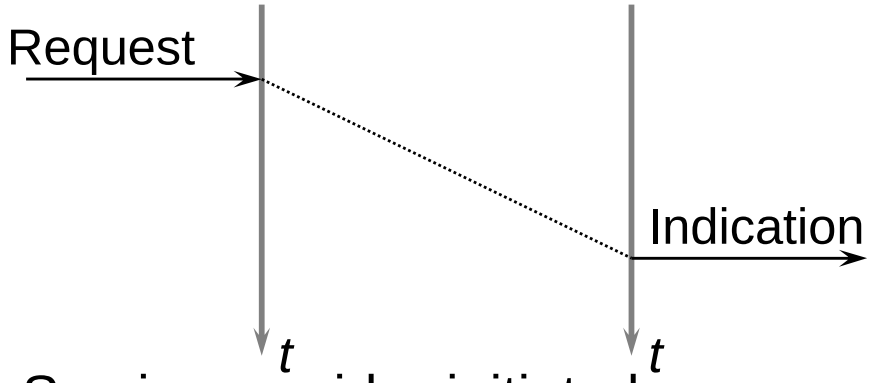
© 2017 Burkhard Stiller

ifi

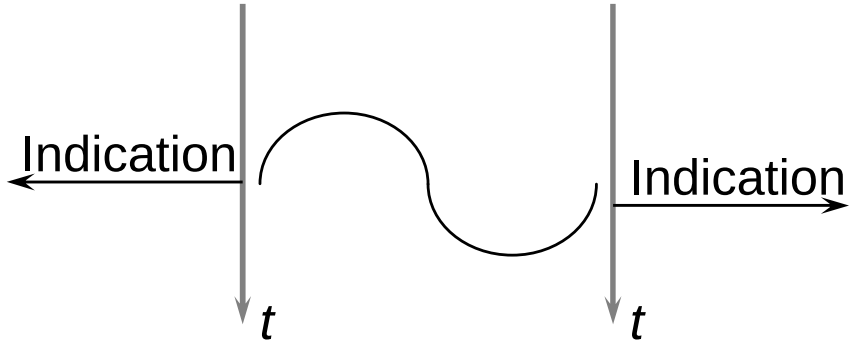
4 Service Types

1. Un-acknowledged Service

- Example: Postal letter service
- Service user-initiated:

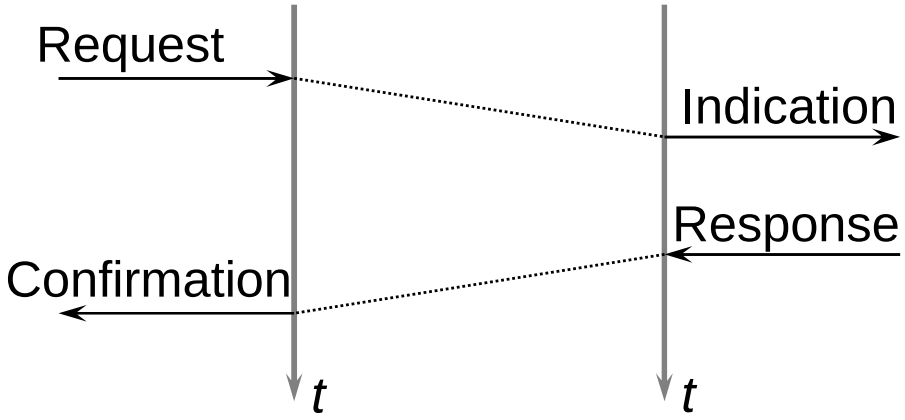


- Service provider-initiated



2. Acknowledged Service

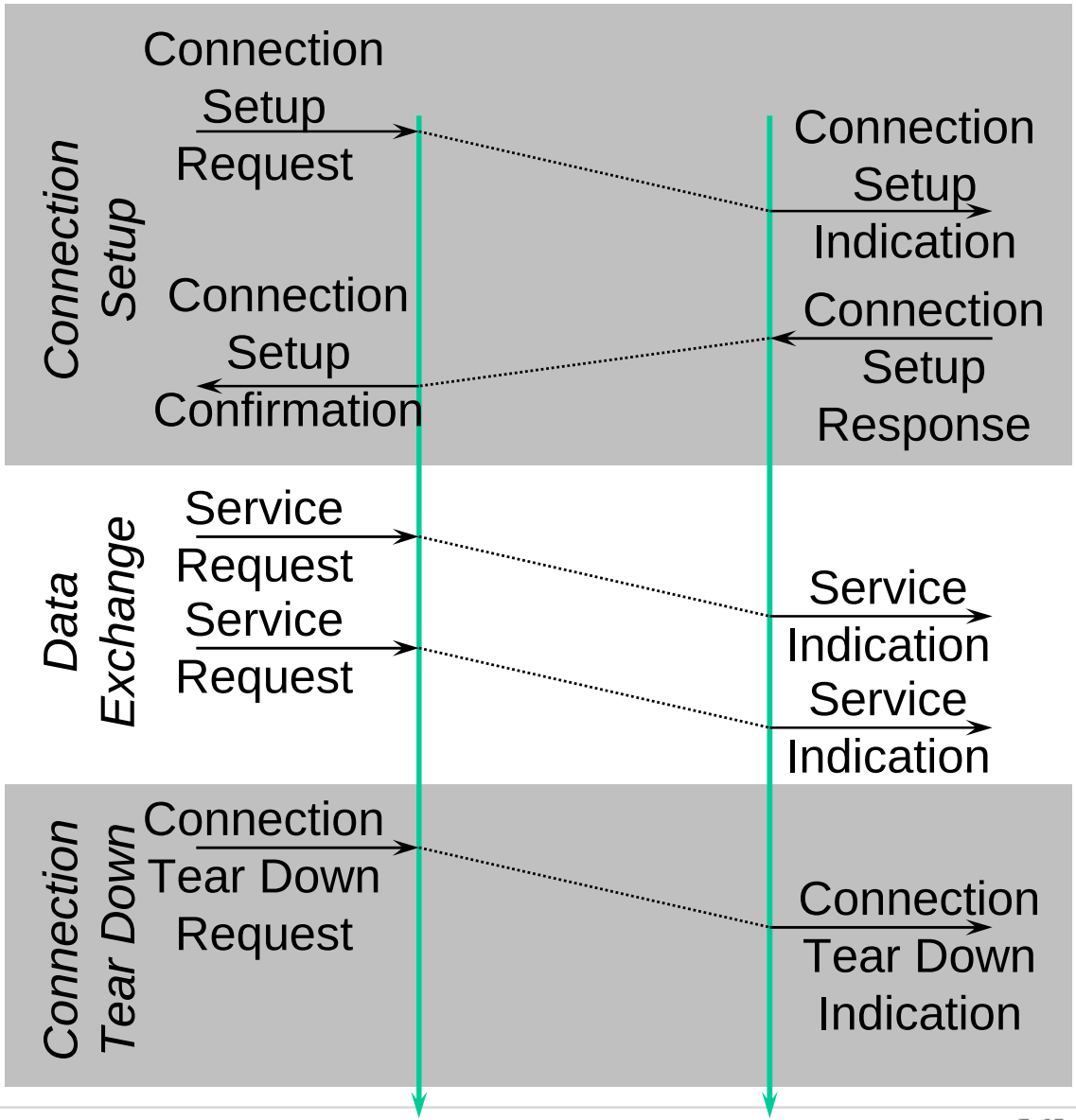
- Example: Booking



© 2017 Burkhard Stiller

3. Connection-oriented Services

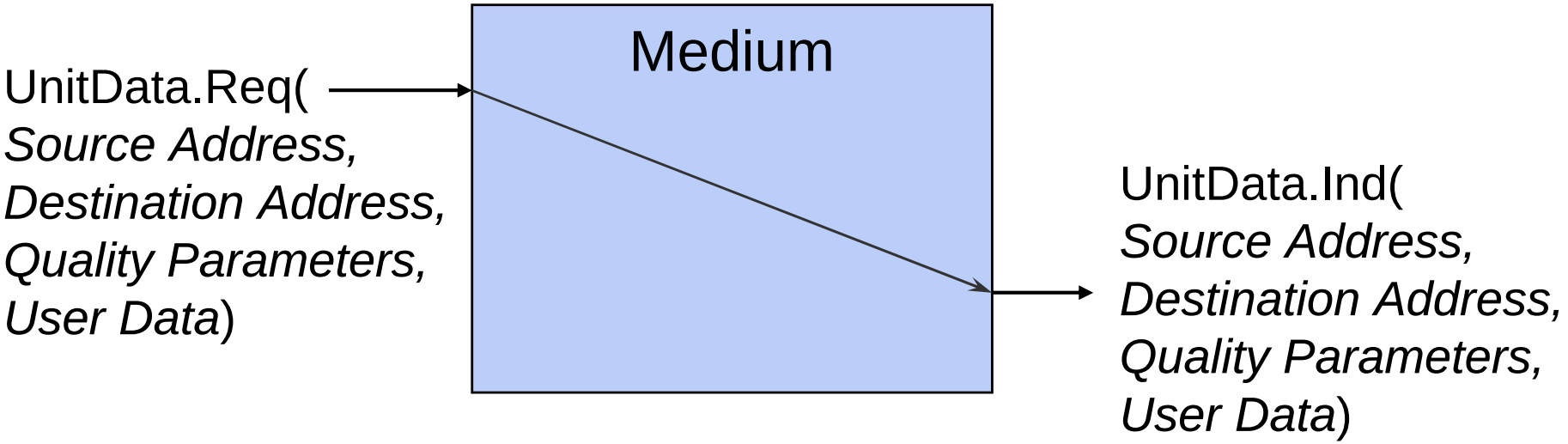
- Three phases concept:
  - Connection setup:
    - Generation of context in the network and host
    - Resource allocation
  - Data exchange:
    - Example: simplex, less context information required
  - Connection tear down:
    - Context deletion
    - Resource de-allocation



© 2017 Burkhard Stiller

ifi

4. Connectionless Service

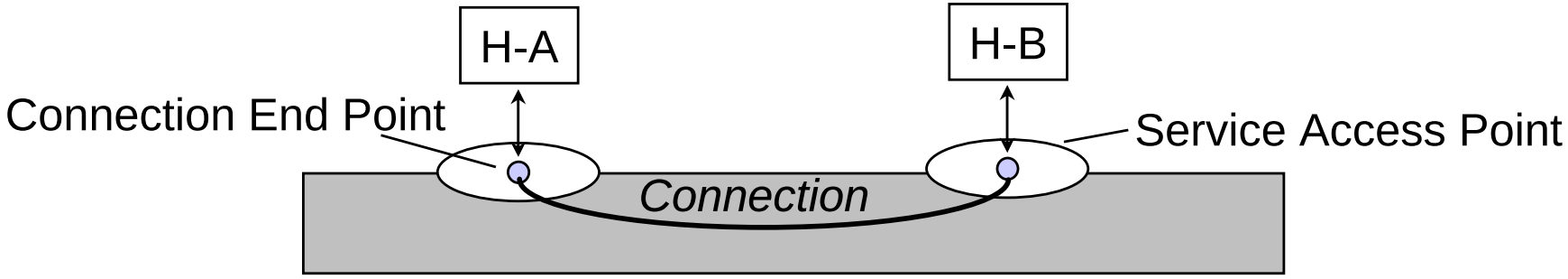


- A connectionless (datagram) service …
  - does not know any relation between any transmission service
  - does not support any sequence of delivery (arbitrary delivery)
  - does not enable any negotiation between partners
  - implements an unacknowledged service
  - Entities do not store any context related to previous datagrams

© 2017 Burkhard Stiller

Addressing

- An address defines a unique identification of a communication partner's host
  - Connectionless service:
    - Service request includes the destination address
    - Service indication includes the source address (optional)
  - Connection-oriented service:
    - Context being established during the connection setup phase includes addressing information
    - Multiple connections from the same SAP require a connection ID



© 2017 Burkhard Stiller

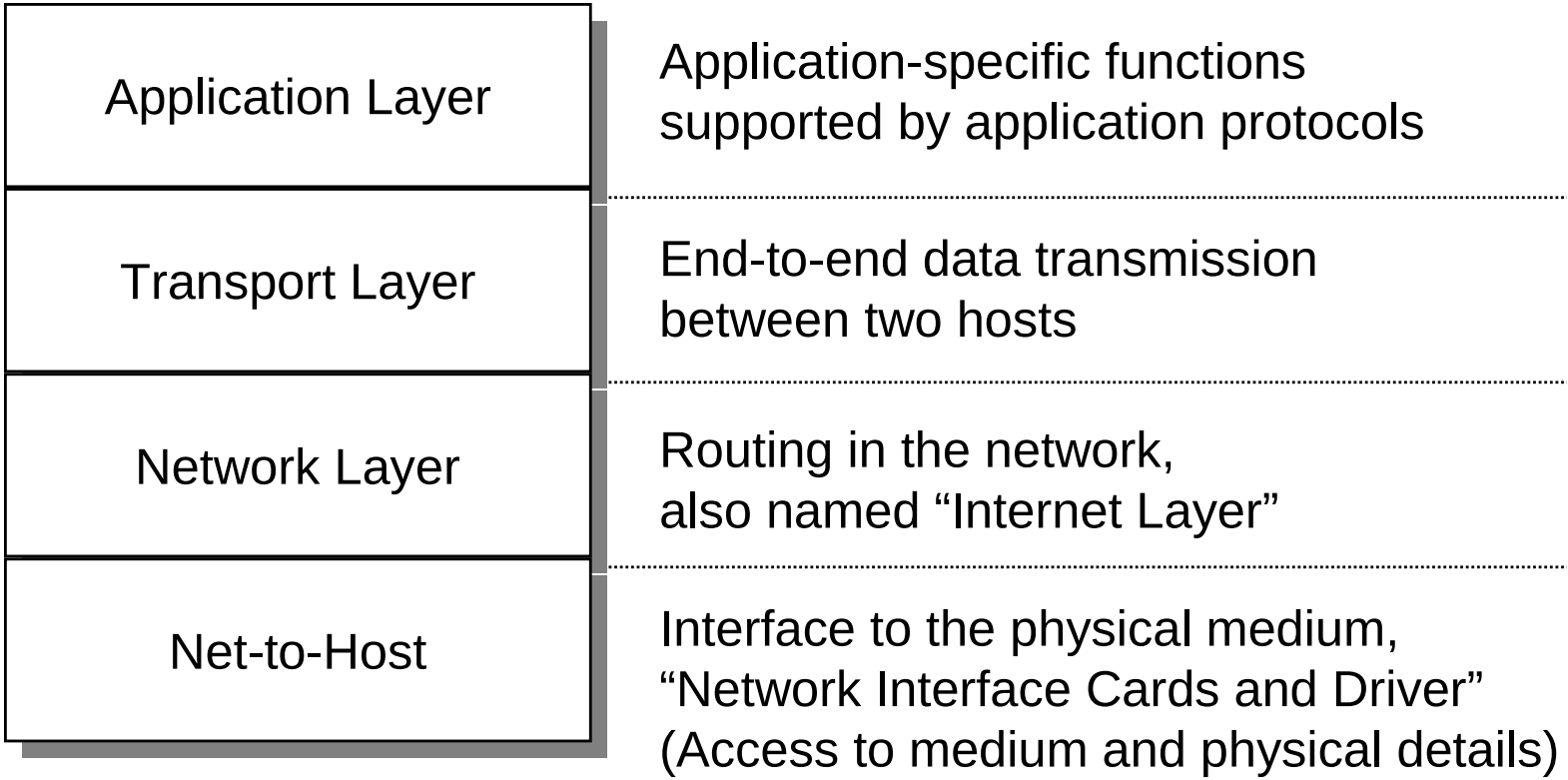
Discussion: Connectionless/oriented Services

- Connectionless service:
  - Each data transfer is considered separately
  - No connection status known or even required
- Connection-oriented service:
  - Set up of a connection between service users of the (N)-Layer by (N-1)-Entities
  - Request performed by (N-1)-Layer service primitives
  - Protocol-dependent negotiation of transmission parameters: partners, Quality-of-Service (QoS), route
  - Data transfer in this connection under the constraints of the context of the connection (connection status)
  - Tear down of the connection required (im- or explicit)

© 2017 Burkhard Stiller

Internet …

The Internet Architecture

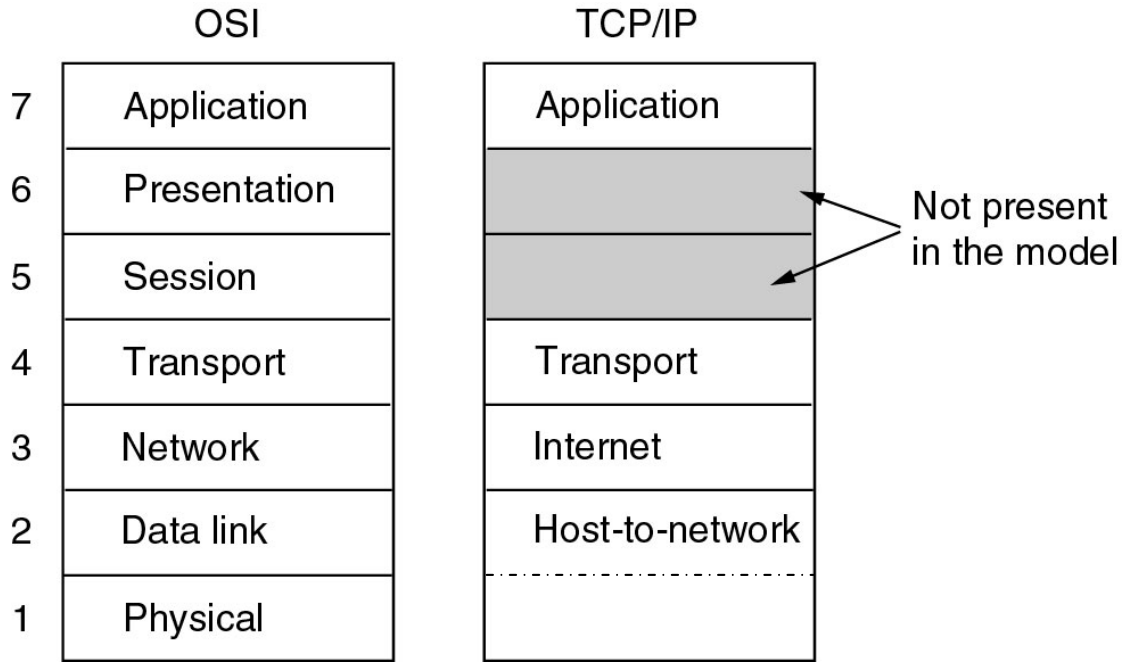


© 2017 Burkhard Stiller

ifi

Relation between OSI and the Internet (1)

- *Major differences:*
  - Tasks of the presentation and session layer (layer 5 and 6) are included in the application layer.
  - Tasks of the data link and physical (layer 1 and 2) are combined into a host access layer.



TCP: Transmission Control Protocol
IP: Internet Protocol

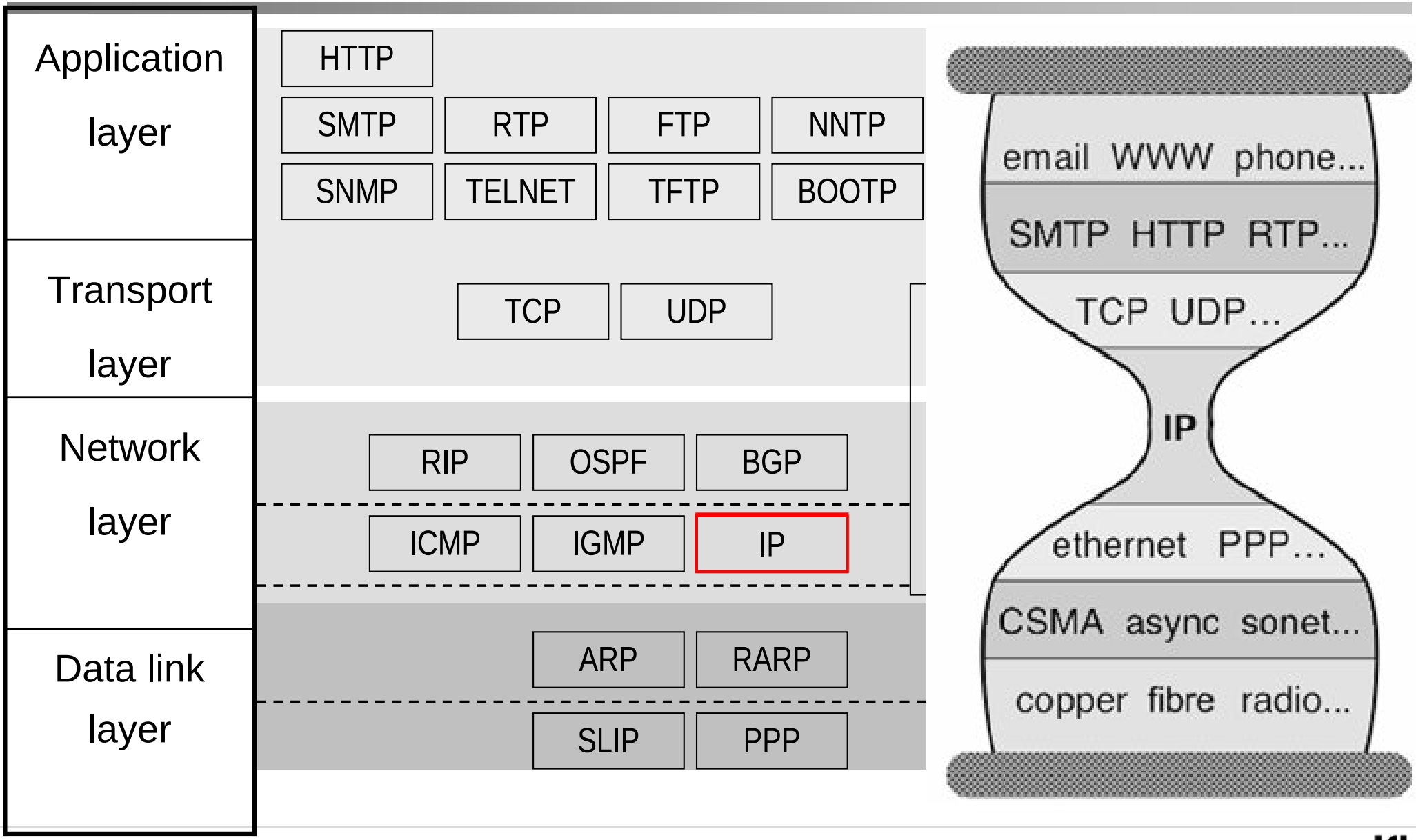
© 2017 Burkhard Stiller

Relation between OSI and the Internet (2)

- ISO/OSI BRM
  - Layers (7)
  - End-system
  - Intermediate System
  - Protocol
  - Service
    - Connectionless
    - Connection-oriented
  - Service Access Point
  - Connection End Point
  - Protocol Data Unit
  - Network
- Internet Architecture
  - Layers (5)
  - Host
  - Router (bridge, switch)
  - Protocol
  - Service
    - Connectionless
    - Connection-oriented
  - IP Address
  - Port number
  - Packet, Datagram, Byte Stream
  - Internet

© 2017 Burkhard Stiller

Internet Protocols — Examples



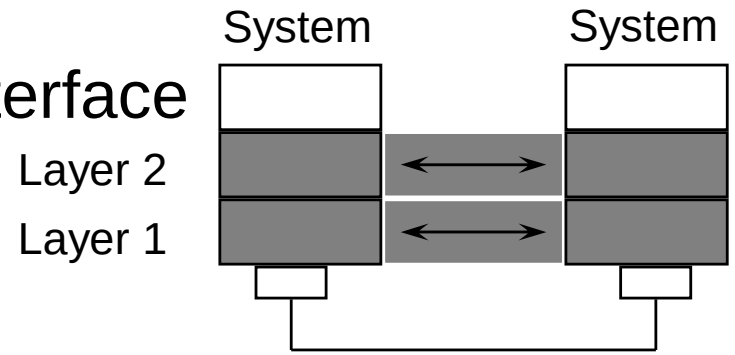
© 2017 Burkhard Stiller

ifi

Layers …

Physical and Data Link Layer

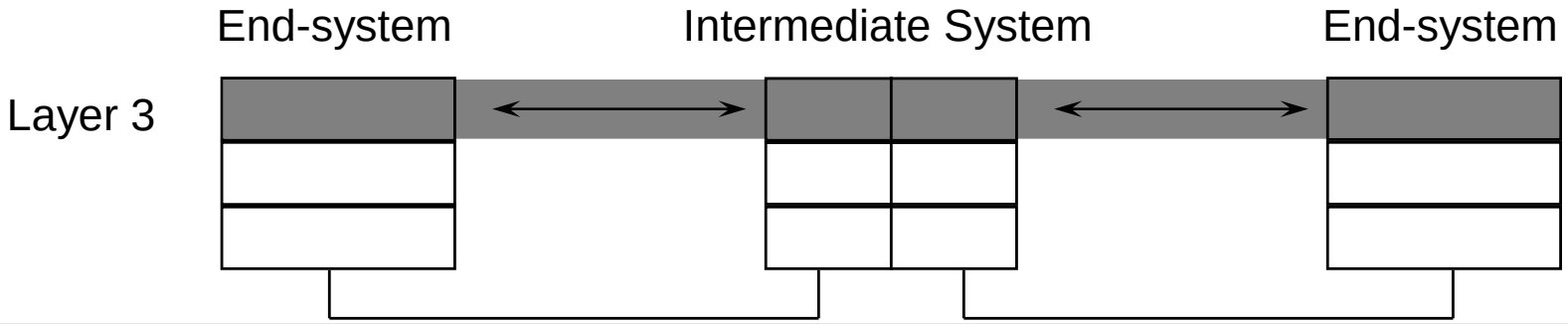
- Physical Layer:
  - Unsecured connection between hosts/routers
  - Transmission of bit sequences (not structured)
  - Covers physical connectivity, plugs, pins, and the mapping scheme of data to signals
  - Standardization of physical system interface
- Data Link Layer:
  - Error free data transfer
  - Separation of streams of bits of layer 1 into frames
    - Protocol functions: error detection and error handling, acknowledgements, time and sequence control, retransmission, flow control, and reset



© 2017 Burkhard Stiller

Network Layer

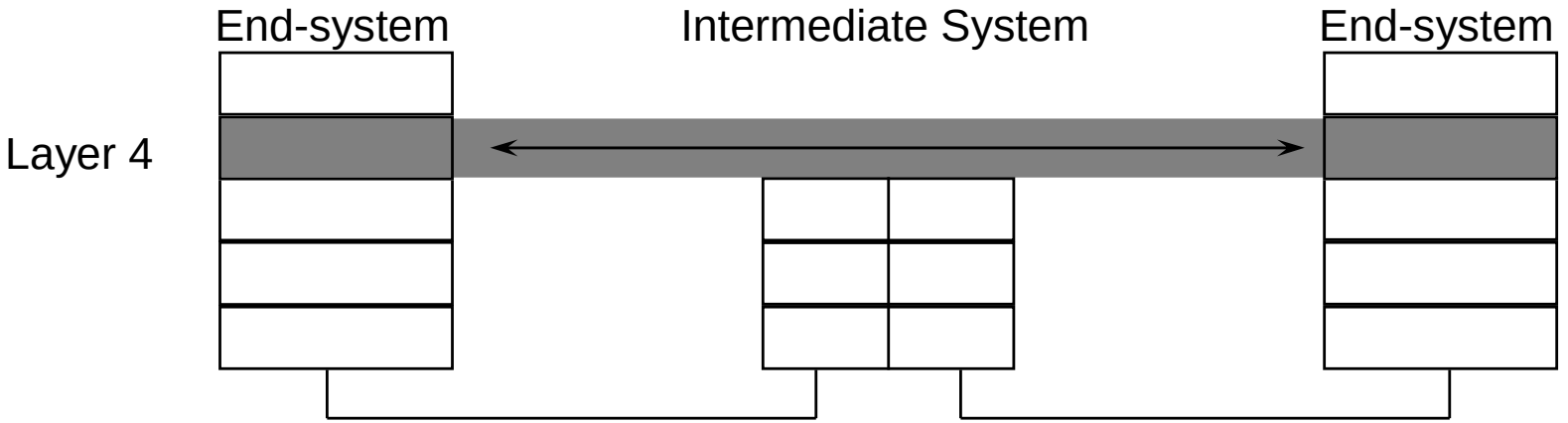
- Network Layer:
  - Interconnects link layer connections to host-to-host connections
  - Connectionless or connection-oriented services
  - May show: Inter-network, sub-network, or routing sub-layers
  - Protocol functions:
    - Routing in case of switching
    - Forwarding
    - Congestion control



© 2017 Burkhard Stiller

Transport Layer

- Transport Layer:
  - Addressing of transport service users
  - Data transfer between users in end-systems, e2e connection
  - Hides away transparently transmission and switching technology as well as any sub-network structure
  - Connection-oriented or connectionless services
    - Protocol functions: error control, flow control, acknowledgments ...



© 2017 Burkhard Stiller

Application-oriented Layers

- Session Layer:
  - Synchronization
  - Definition of sessions based on user requirements
- Presentation Layer:
  - Defines the syntax for data to be transferred
  - Abstract Syntax Notation (ASN.1)
- Application Layer:
  - Offers services to the user
    - A.o. file transfer, e-mail, remote terminal login ...
  - Offers generic services for applications:
    - A.o. reliable message transfer, remote procedure call (RPC) ...

© 2017 Burkhard Stiller

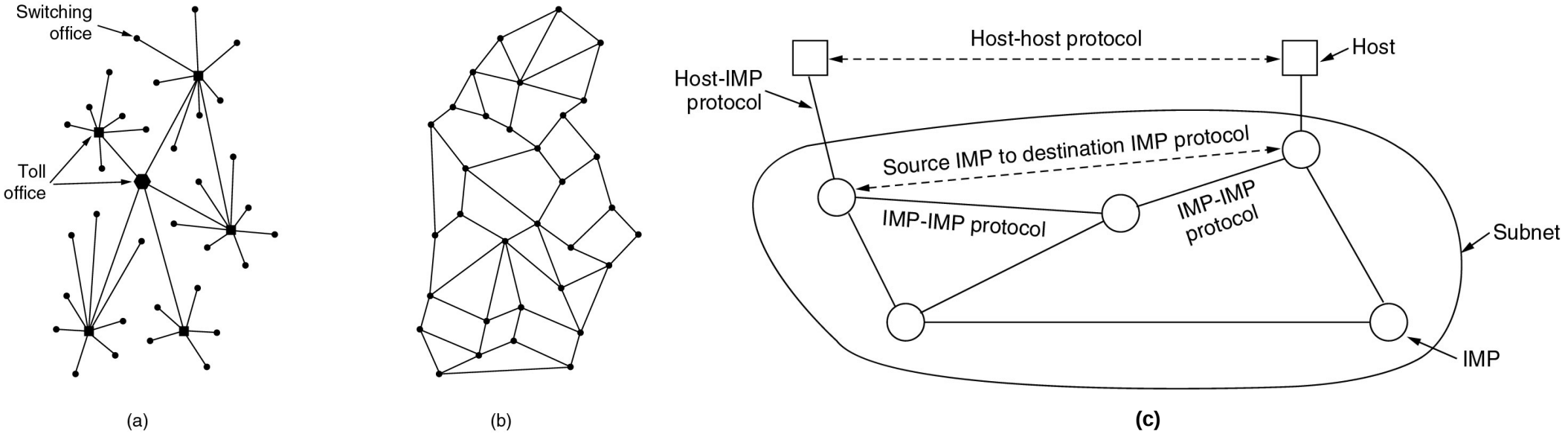
Network Examples

Communication Networks

(a) Public Switched Telephony Network (PSTN)

(b) Decentralized and distributed switching system

(c) Original ARPANET – Predecessor of the Internet



IMP: Interface Message Processor – an ARPANET packet-switching node

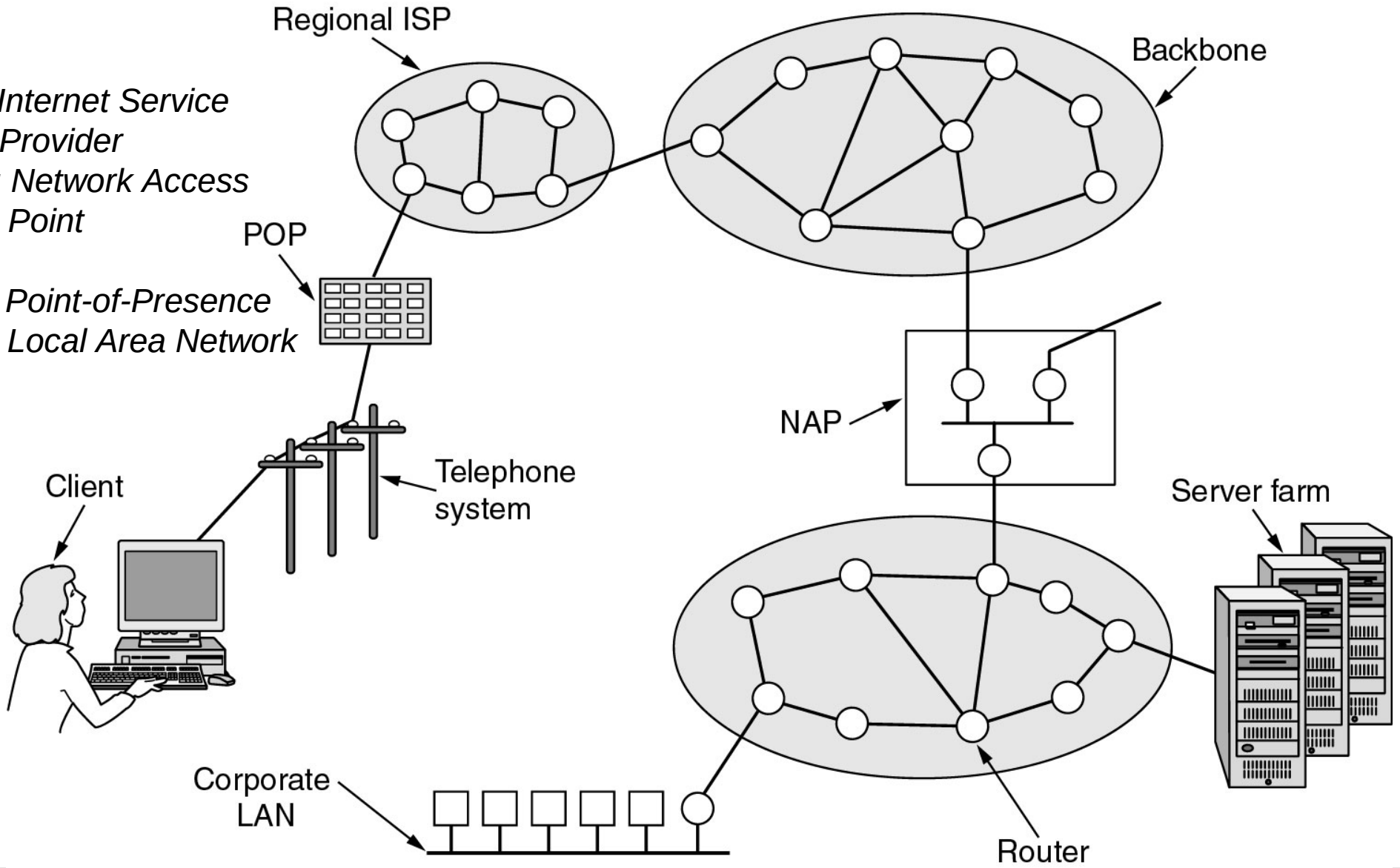
© 2017 Burkhard Stiller

ifi

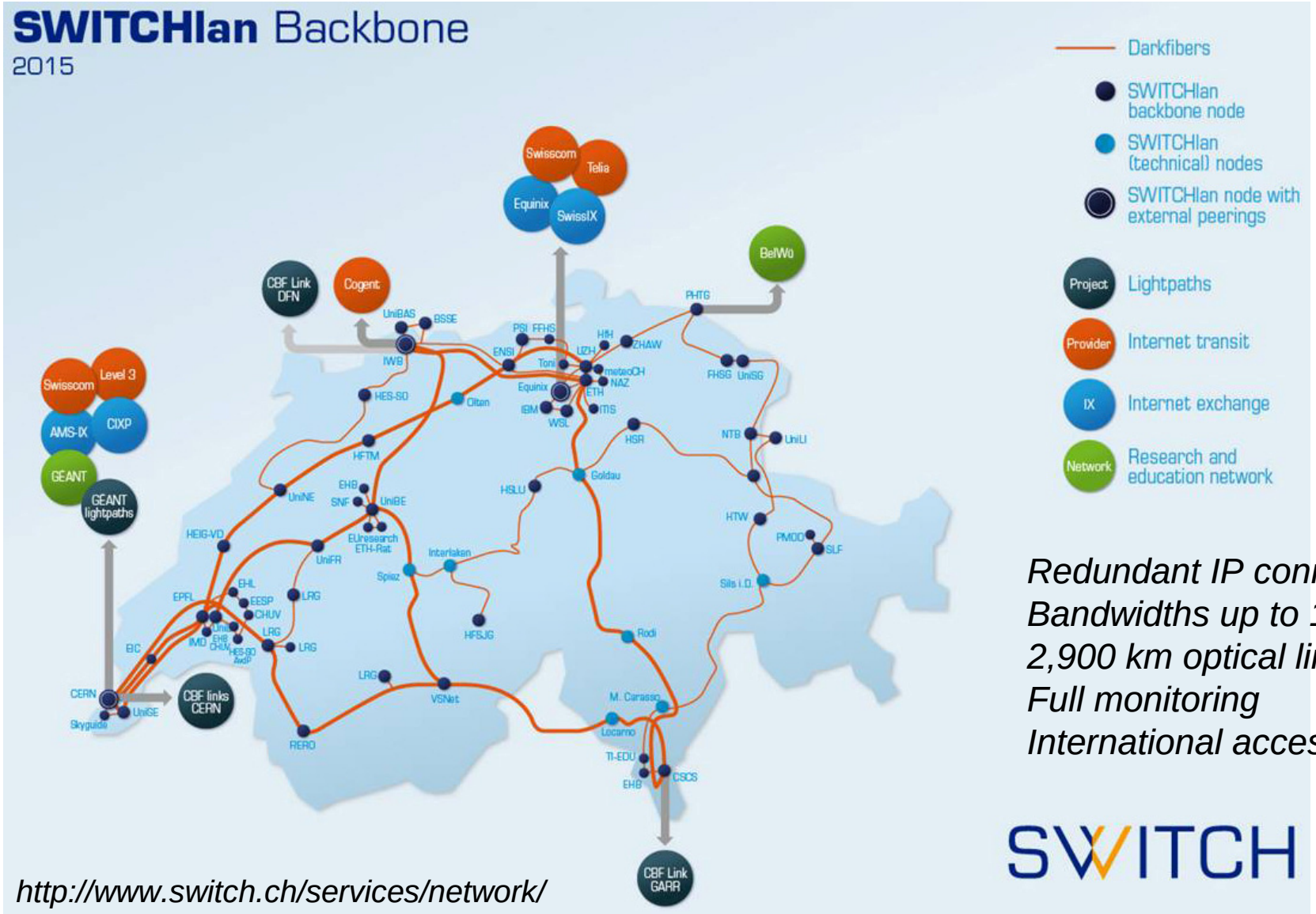
Sample of the Internet

ISP: Internet Service Provider
NAP: Network Access Point

PoP: Point-of-Presence
LAN: Local Area Network



SWITCH – Serving Swiss Universities



http://www.switch.ch/services/network/

Redundant IP connectivity
Bandwidths up to 100 Gbit/s
2,900 km optical link length
Full monitoring
International access points

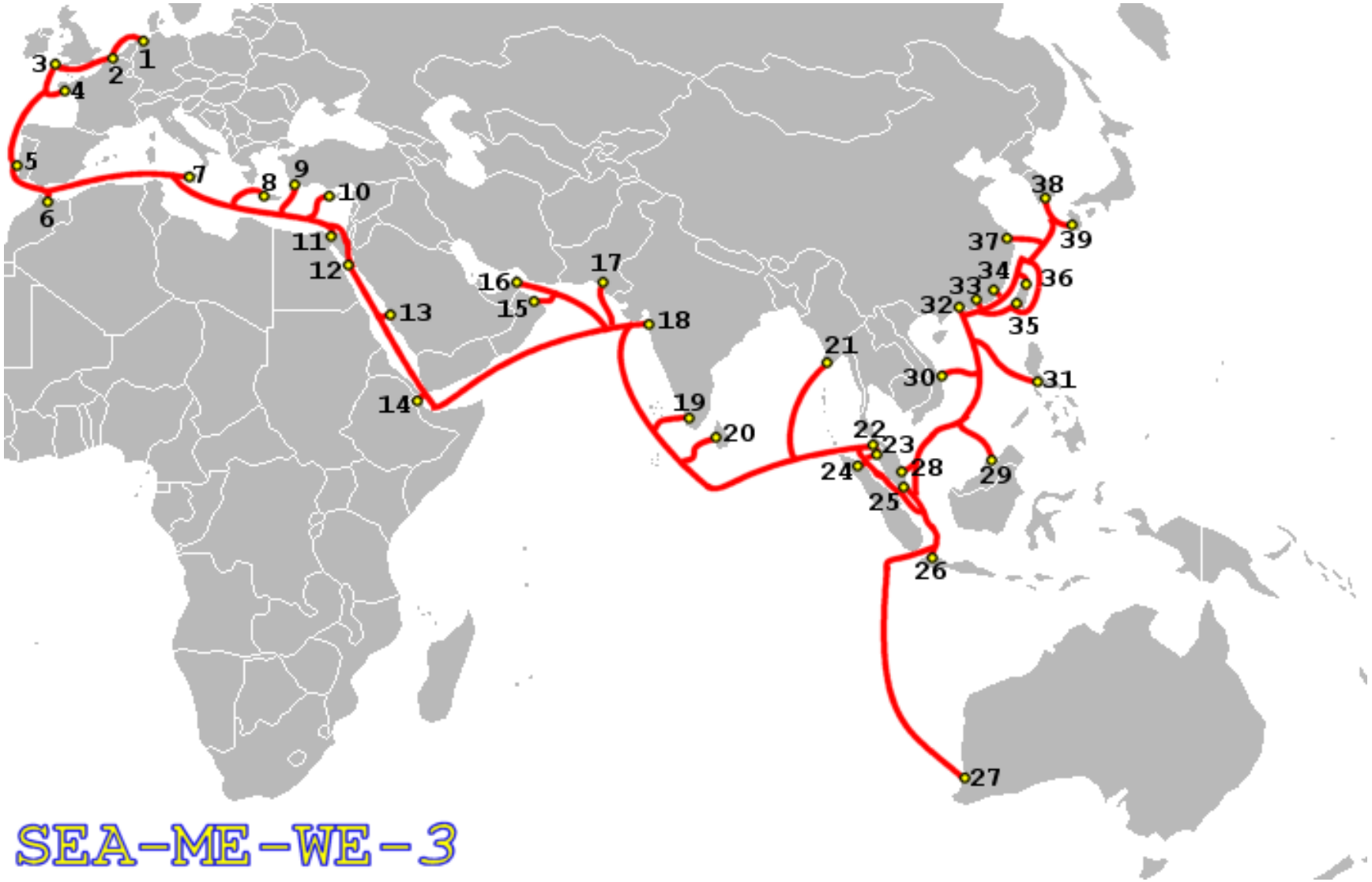
SWITCH

© 2017 Burkhard Stiller

ifi

France Telecom Network

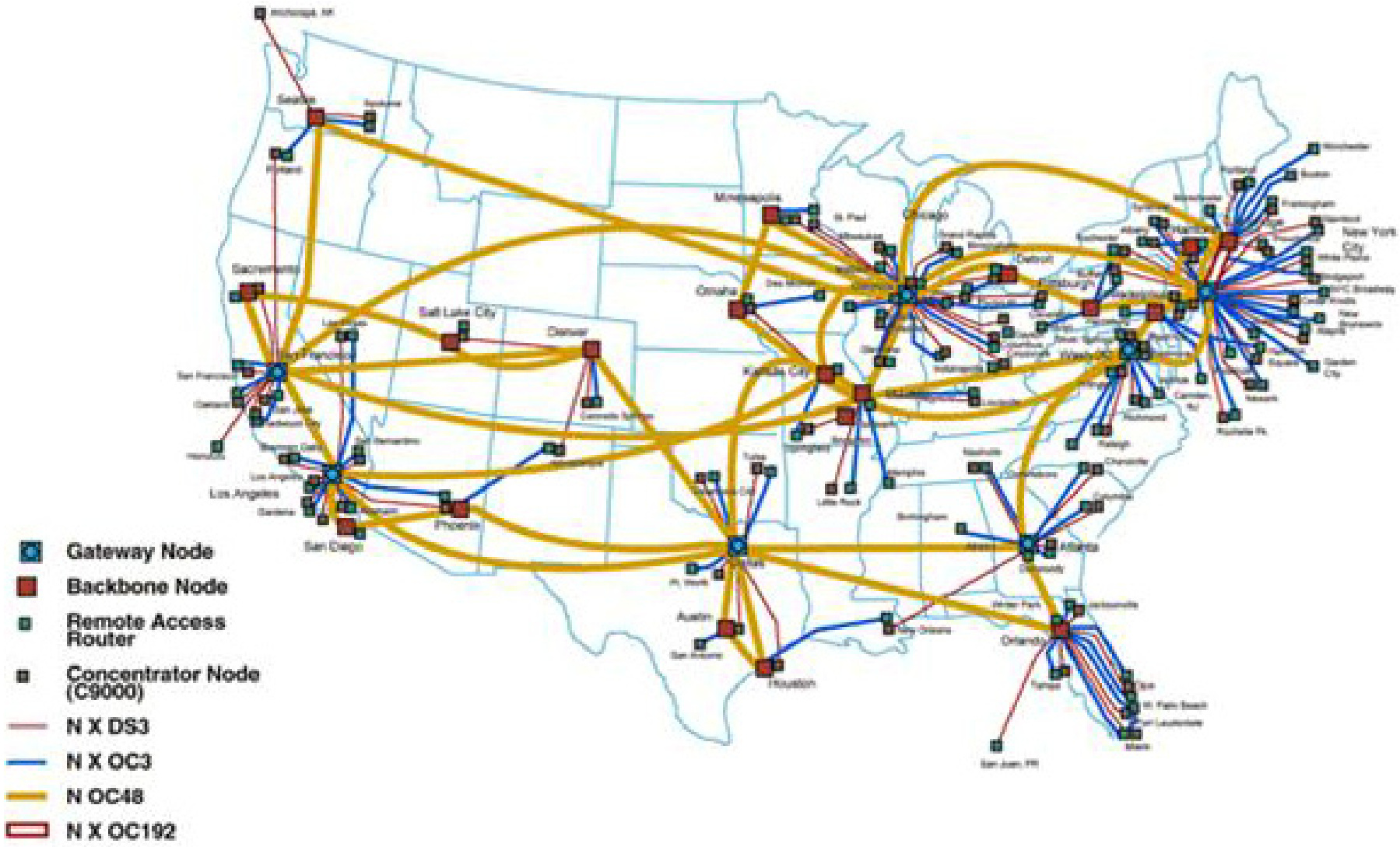
SEAMEWE 3: South-East Asia – Middle East – Western Europe 3



http://www.sec.gov/Archives/edgar/data/1038143/000130817909000044/registrationdocument.htm (2009)

© 2017 Burkhard Stiller

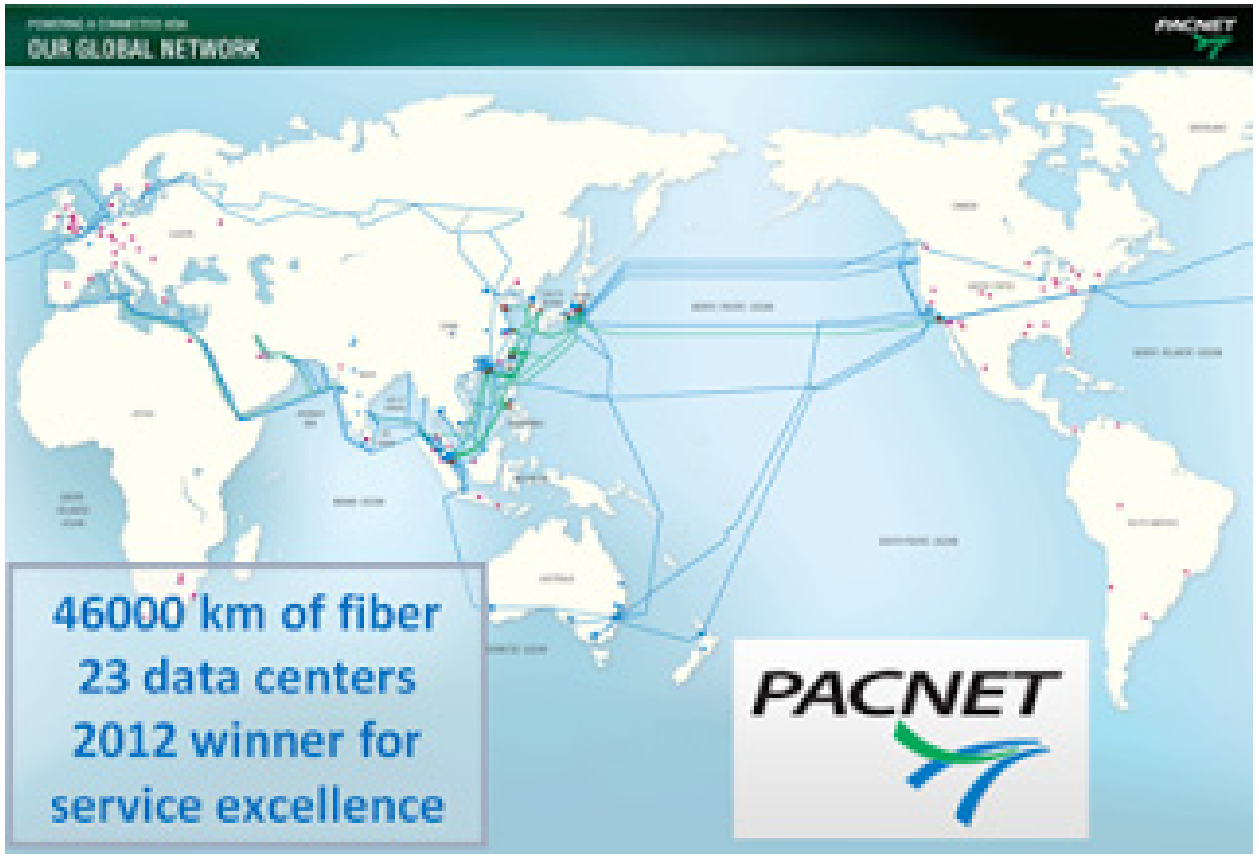
AT&T‘s IP Backbone Network (2000)



http://wheatcommunications.com/internet_marketing/fast-tier-1-premium-bandwidth

© 2017 Burkhard Stiller

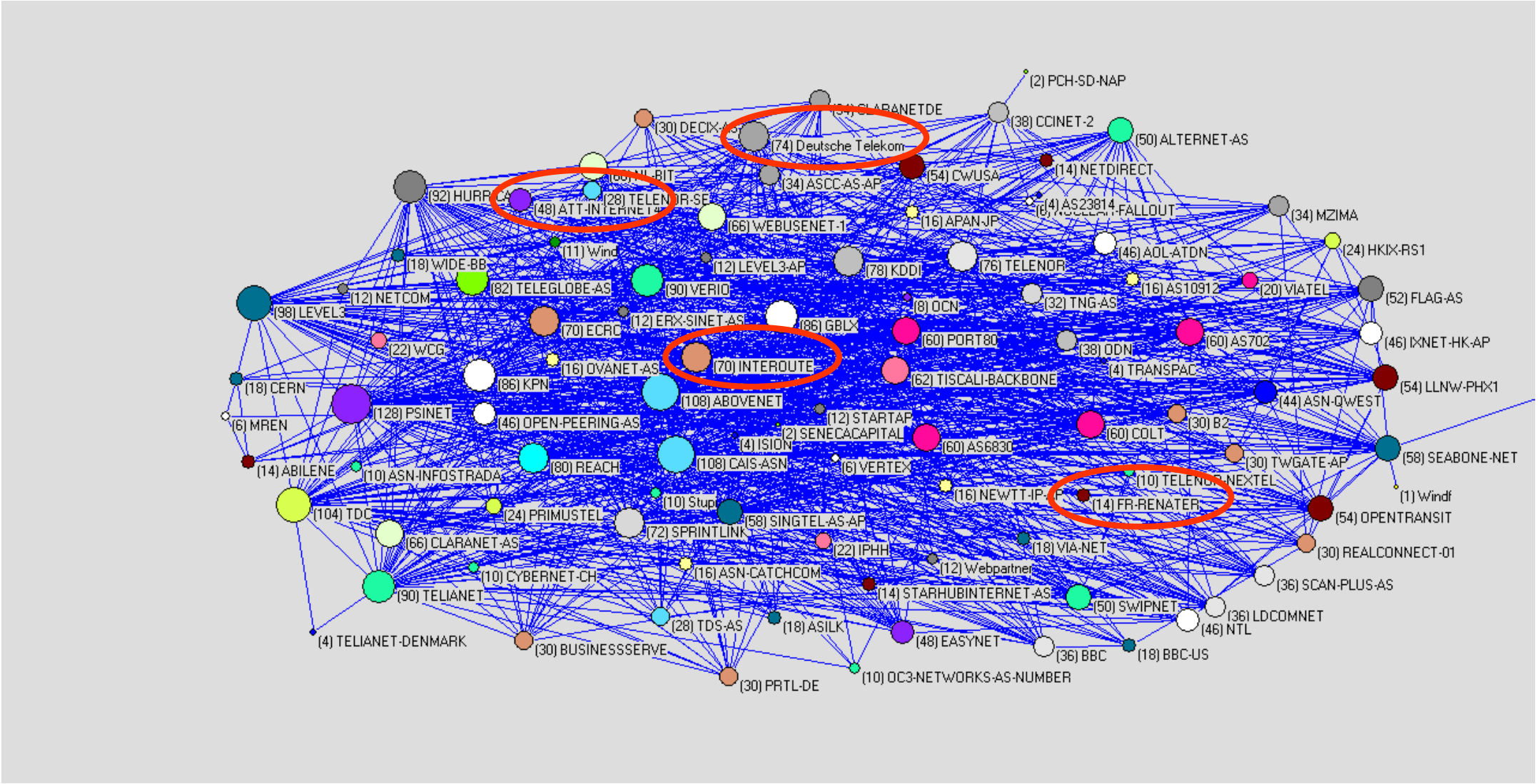
PACNET – Fiber Optics Submarine Network



http://www.convergedigest.com/2013/06/pacnet-signs-chinese-carriers-for-bgp.html (June 2013)

© 2017 Burkhard Stiller

Top 100 Network Interconnection Agreements



http://revel.unice.fr/eriep/?id=3203, July 2006

© 2017 Burkhard Stiller

ifi

8. Basic Concepts

Point-to-point Direct Links

Resource Sharing and Multiplexing

Physical Media

Reliable Transmissions

© 2017 Burkhard Stiller

Point …

Basic Network Building Blocks



Links (medium and modulation/encoding)



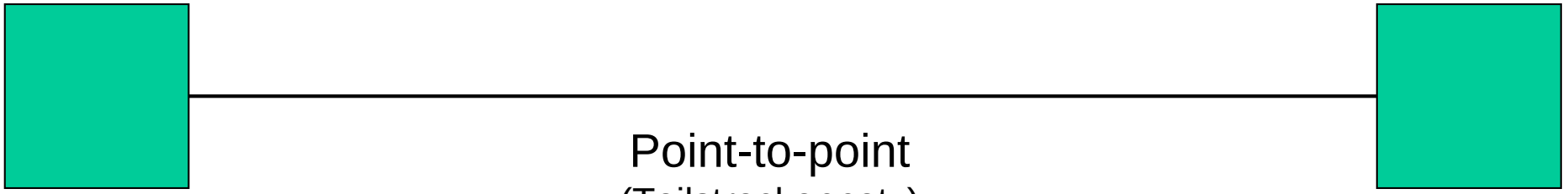
Nodes (PC, router, switch, embedded system)

- Nodes
  - Source
  - Sink
- Intermediate systems
  - Source and sink
- Abstraction of a Link
  - Channel

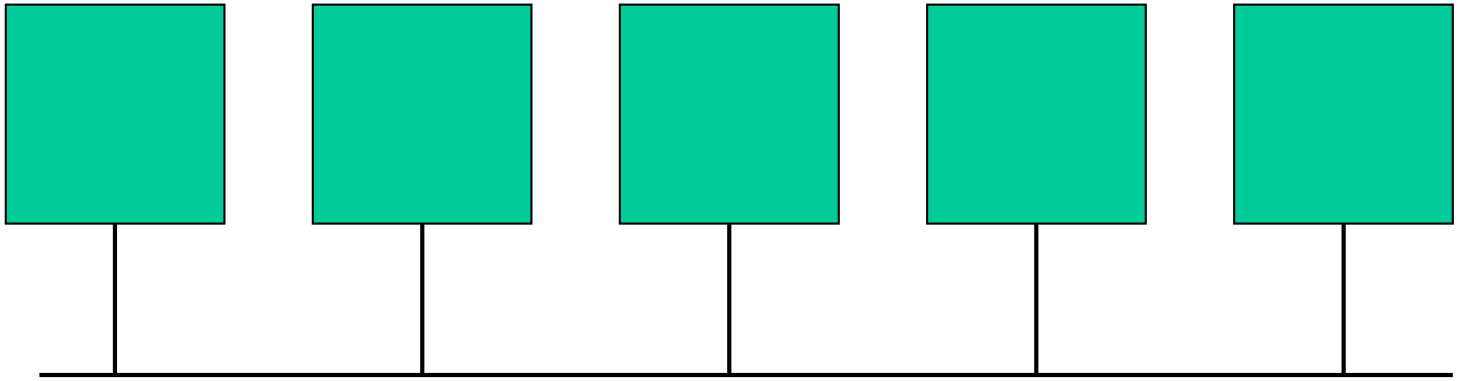
- Channel properties:
  - Depending on physical medium and the mode of usage
  - Point-to-point vs. broadcast
  - Propagation delay
  - Capacity
    - Depending on bandwidth and modulation/coding used
  - Transmission errors
    - Bit errors, error bursts
  - All parameters may be time-variant

© 2017 Burkhard Stiller

Point-to-point and Broadcast Links



Point-to-point
(Teilstreckennetz)

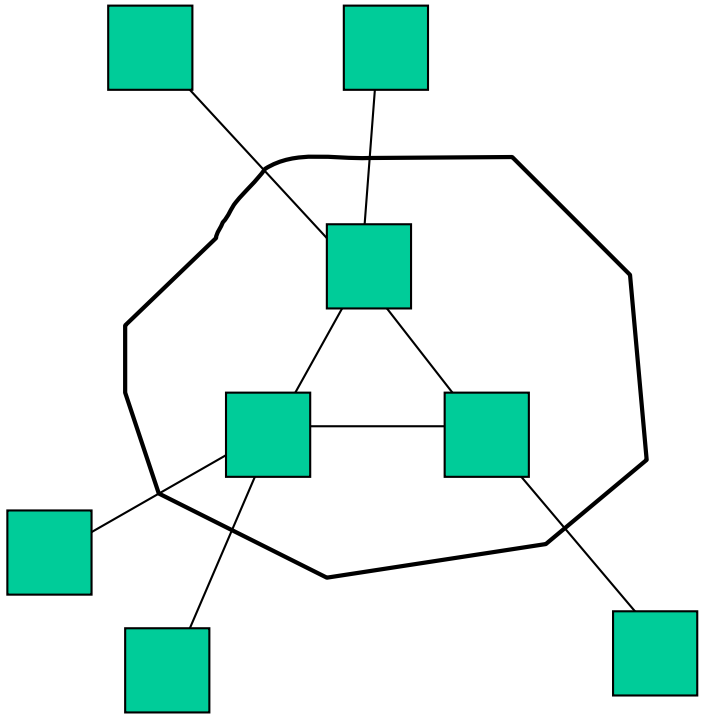


Broadcast/Shared Medium
(Diffusionsnetz)

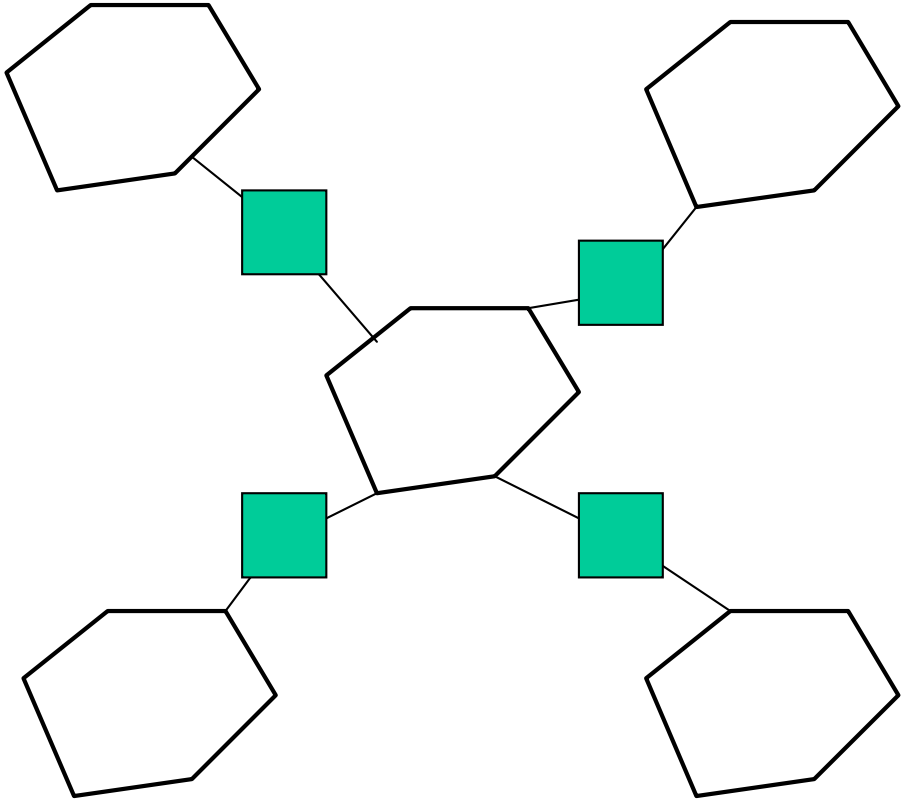
© 2017 Burkhard Stiller

Network: Recursive Definition

Nodes interconnected by links



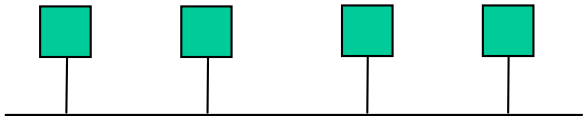
Networks interconnected by nodes



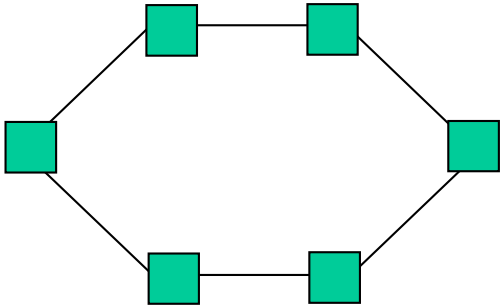
© 2017 Burkhard Stiller

Topologies

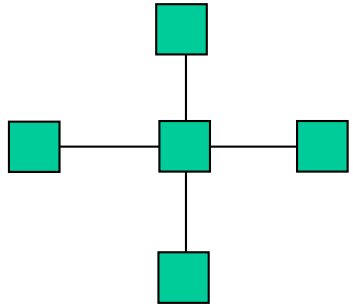
Bus



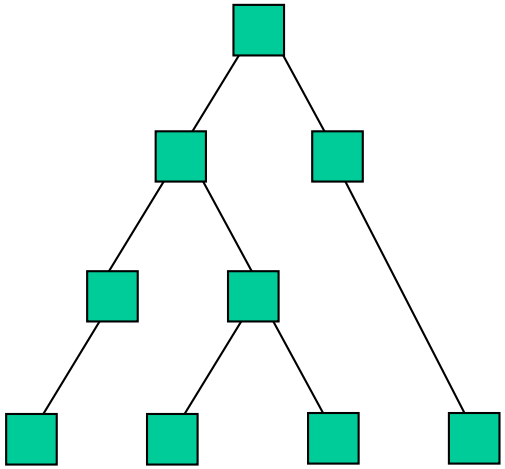
Ring



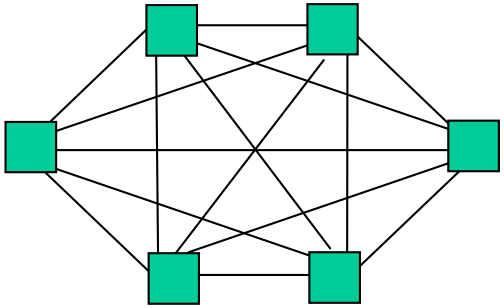
Star



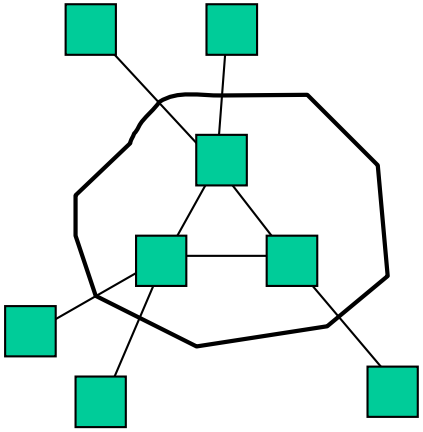
Tree



Full Mesh



Mesh



© 2017 Burkhard Stiller

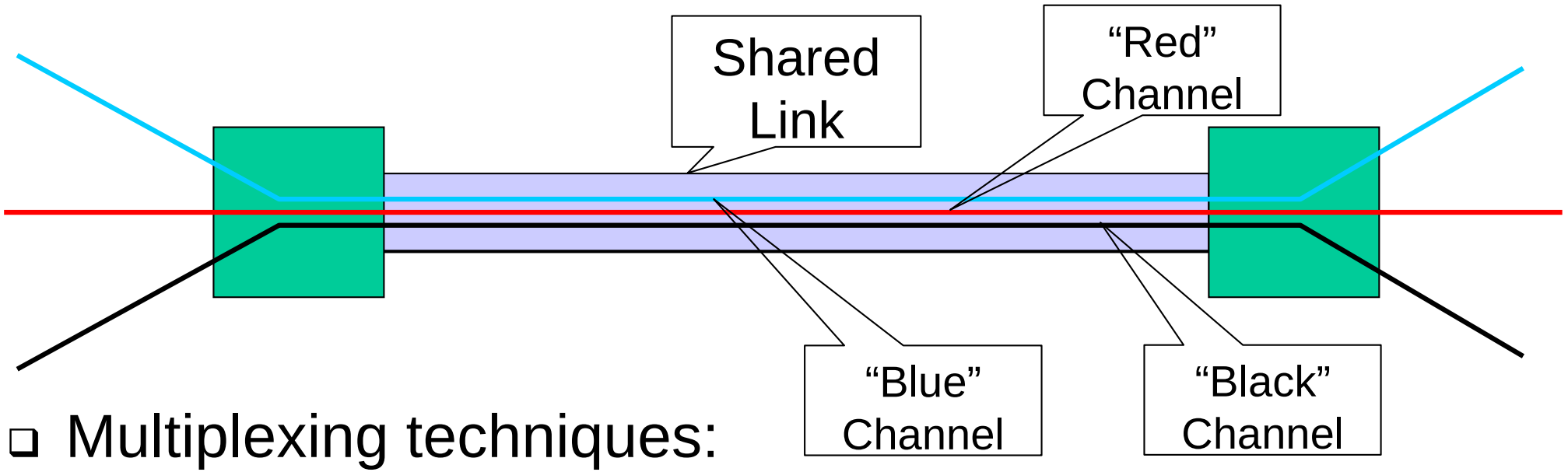
Important Aspects of Networks

- Resource sharing
- Inter-process communication
- Data transmission on a link
- Resilience (Ausfallsicherheit) against failures
- Error handling
- Co-ordination of access to shared resources
- Naming and addressing (identification of objects)
- Routing
- Fairness
- Charging and accounting

© 2017 Burkhard Stiller

Sharing …

Resource Sharing and Multiplexing



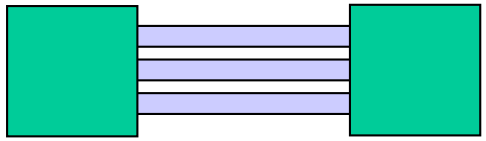
- Multiplexing techniques:
  Transmitting more than one signal over one path
  - Space-division multiplexing
  - Frequency-division multiplexing
  - Time-division multiplexing
  - Code-division multiplexing

© 2017 Burkhard Stiller

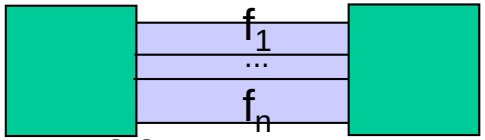
ifi

Multiplexing Techniques — Types

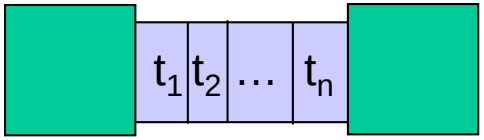
- Space-division multiplexing:
  - More than one physical path is grouped together



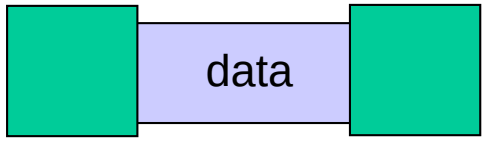
- Splitting a physical path and occupying the two-dimensional continuum of frequency and time:
  - Frequency-division multiplexing:
    - Quantity information transmitted using a certain range of frequency
    - Example: FM radio



  - Time-division multiplexing:
    - Quantity information transmitted using a certain range of time



- Splitting a channel by transmitting data encoded with multiple codes in an overlapping fashion:
  - Non-interfering (orthogonal) codes

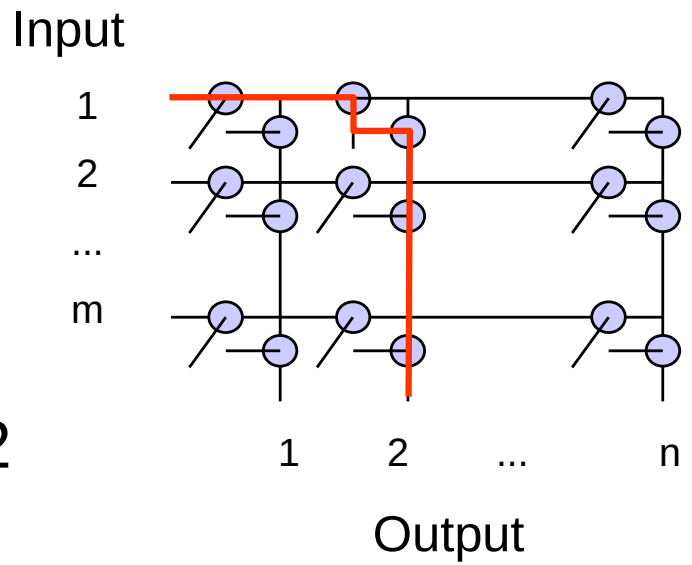


© 2017 Burkhard Stiller

Space-division Multiplexing

- Actions of (Raummultiplex, Raumvielfach)
  - Closing a coupler establishes connection
  - Analog transmissions are supported only by space-division multiplexing in coupling matrices (Koppelfelder)

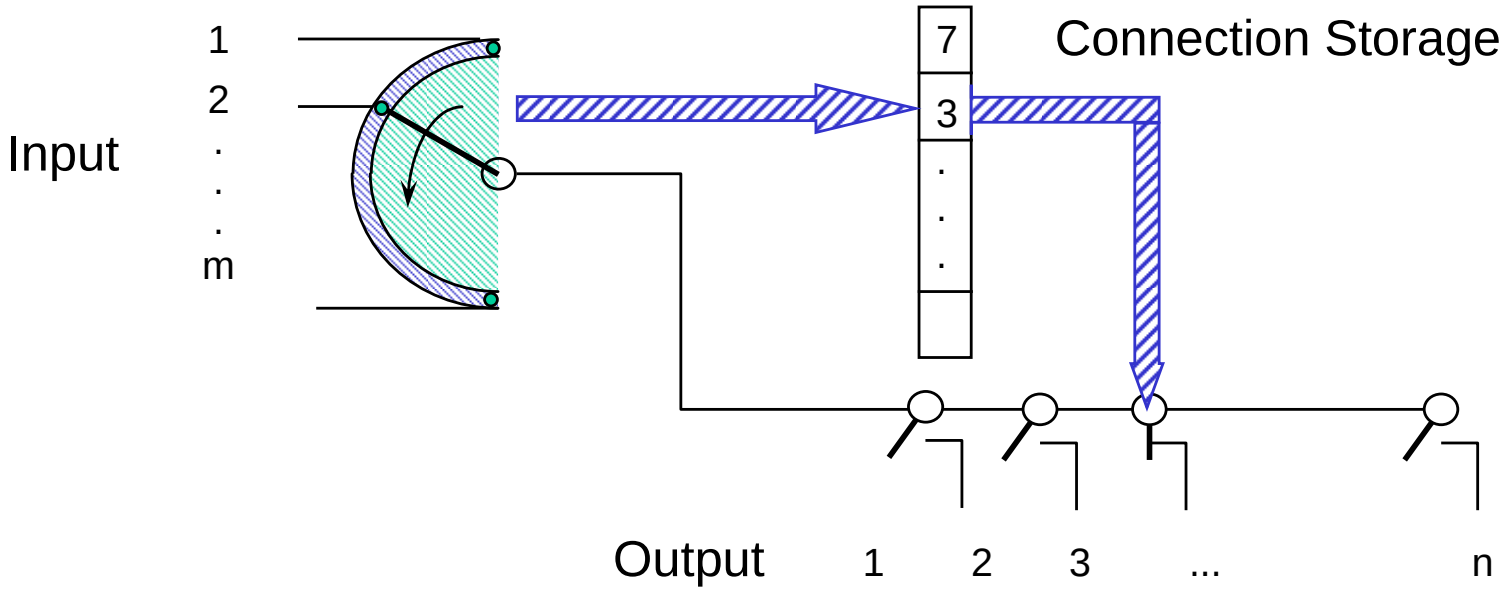
- In general:
  - m input links
  - n output links
- Example:
  - Input 1 interconnected to output 2



© 2017 Burkhard Stiller

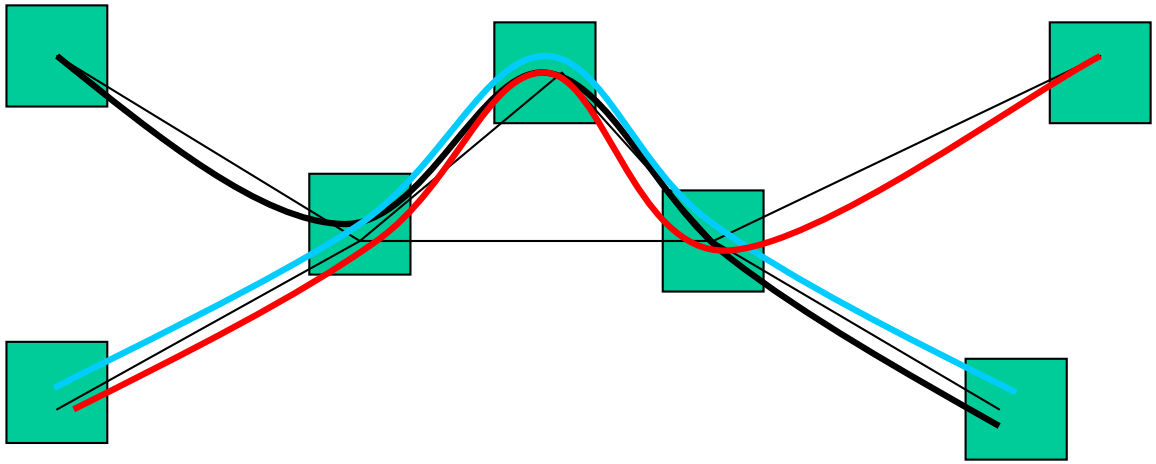
Time-division Multiplexing

- Actions of (Zeitmultiplex, Zeitvielfach)
  - Input is interconnected to output for a period of time
  - During defined sampling interval bus provides connectivity
  - Connection storage (context) maintains assignment of input link and output links for each sampling interval



Switching: Network-wide Resource Sharing

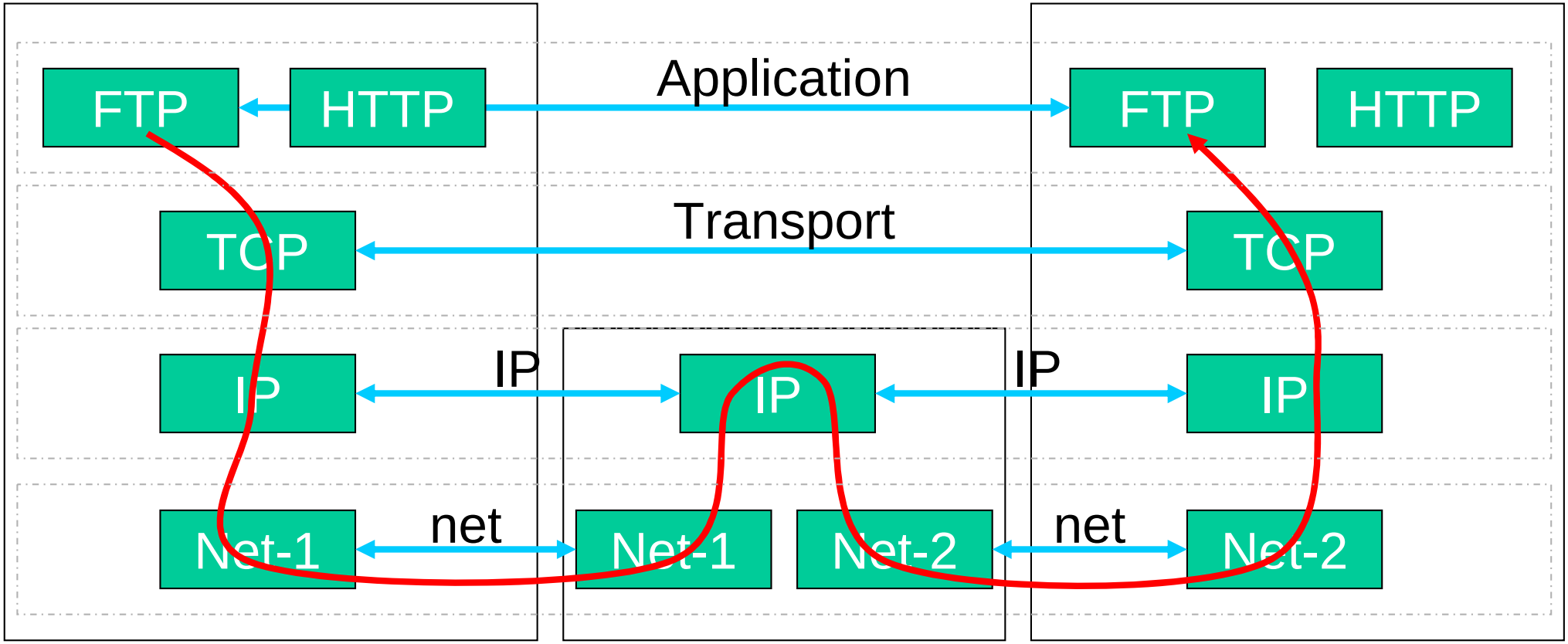
- Circuit switching: Management of end-to-end circuits with fixed resources
- Packet switching: Management of packets
  - Datagrams
  - Virtual circuits
- Message switching: Management of messages



© 2017 Burkhard Stiller

The Internet Architecture

Example: Packet-switching on the IP layer



FTP: File Transfer Protocol
HTTP: Hypertext Transfer Protocol

IP: Internet Protocol
TCP: Transmission Control Protocol

© 2017 Burkhard Stiller

Physical Media (1)

- Physical media (layer 1) technology choices:
  - (Unshielded) twisted pair (UTP) cables
  - Coaxial (coax) cables
  - Optical carrier (OC)
  - Wireless transmission, radio links (“radio“)

- Electromagnetic waves are used to carry information:
  - Signal propagation delay v:
    - Speed of light: $v = 3 * 10^8$ m/s
    - In copper cables: $v = 2.3 * 10^8$ m/s
    - In fiber optical carriers: $v = 2 * 10^8$ m/s

© 2017 Burkhard Stiller

Physical Media (2)

- Delay D of a signal between sender S and receiver R:
  - D = d/v, d is the distance between S and R in [m]
- Signal step T [s]
- Signal rate or baud rate B_S [baud]:
  - B_S = 1/T, number of signal value changes per second
- Data rate r [bit/s]:
  - r = B_S * ld(n), n number of different values of signal
    - Note: Only if n=2 (binary signals) B_S = r.

© 2017 Burkhard Stiller

Physical Media (3)

- Bit duration T_B [s]:
  - $T_B = 1/r$, time required to send a single bit
- Bit length L_B [m]:
  - $L_B = T_B * v = v/r$, length of media requited to carry a single bit
- Throughput or capacity C [bit/s]:
  - Value of C is determined by the useable physical medium
- Path capacity P_C [bit]:
  - $P_C = d/ L_B$, d is the distance between path end-points in [m]

- Encoding of information to be adapted to the medium:
  - Only a physical layer issue!

© 2017 Burkhard Stiller

Key Principles

- For any physical medium, frequency and length restrictions apply:

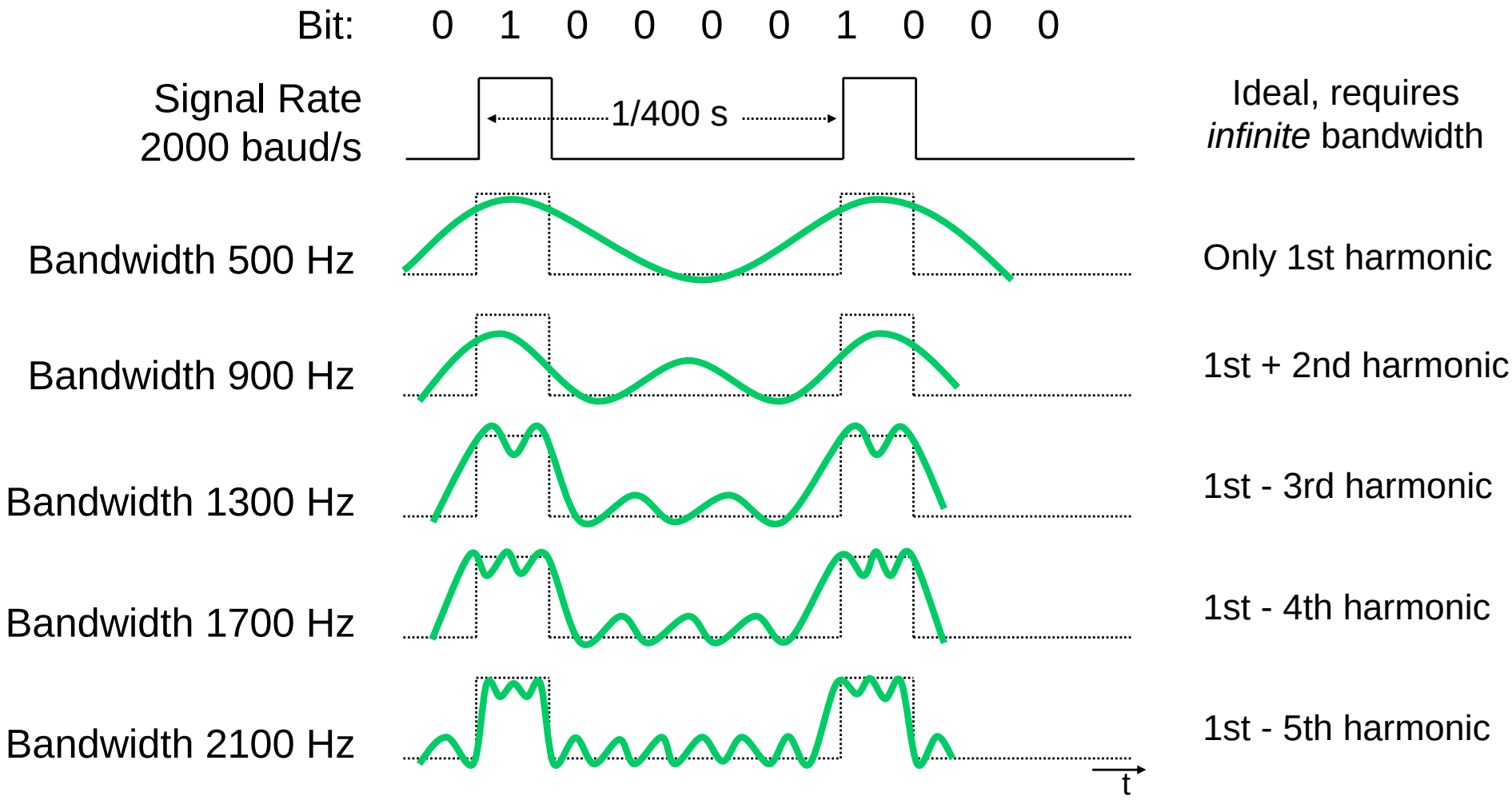
  Time · Bandwidth = const
  Length · Bandwidth = const

- *E.g.,* gradient mode fiber optic with 400 MHz characteristic per kilometer:

  400 MHz for 1 km or
  1 GHz for 400 m

© 2017 Burkhard Stiller

Bandwidth Effects on Digital Signals

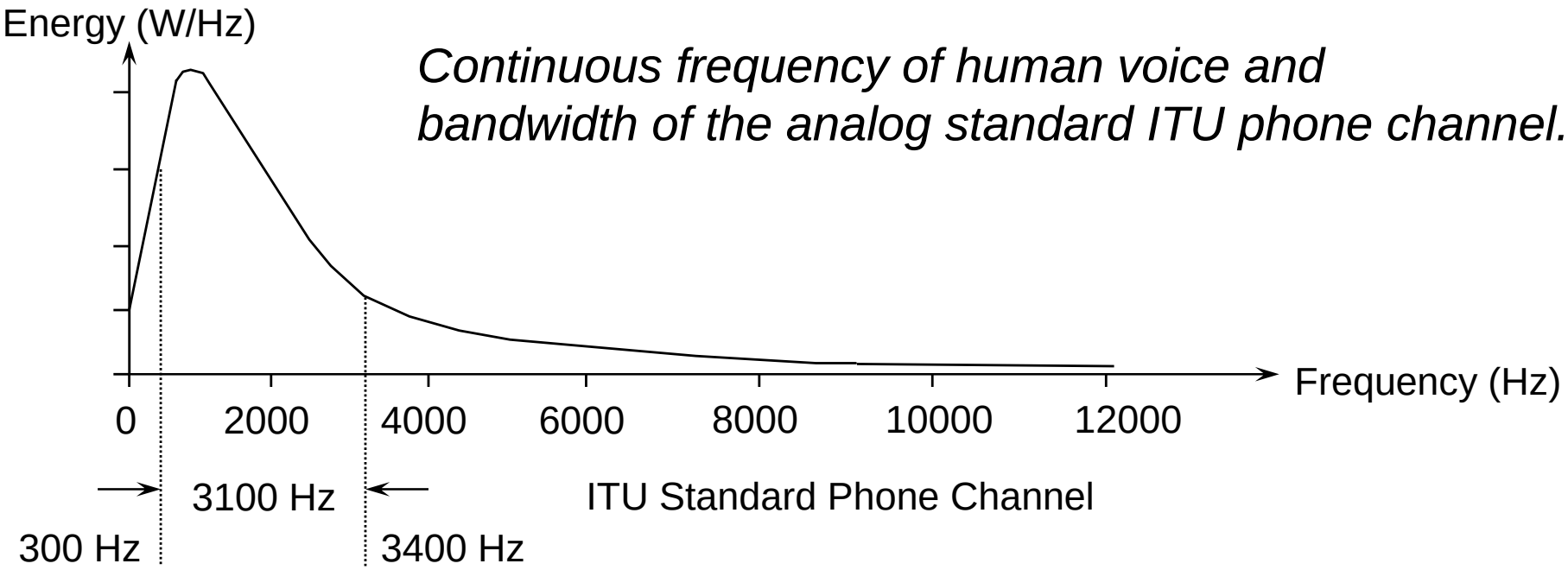


- Minimum bandwidth required for transmission of a sequence of bit of defined signal rate
- Exact determination done by Shannon/Nyquist theorem

© 2017 Burkhard Stiller

Frequencies

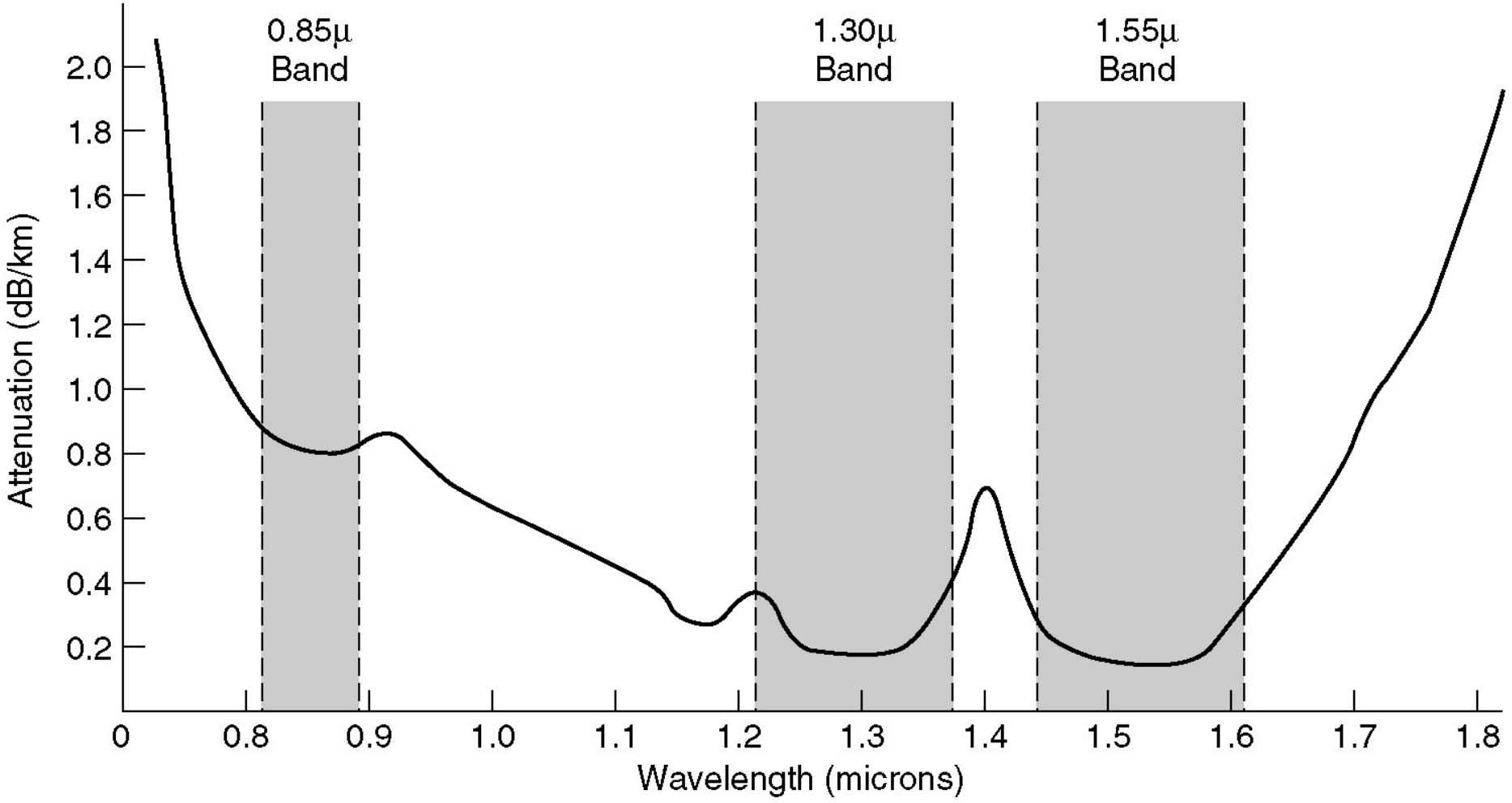
- Limited spectrum signal:
  - Natural limitations of continuous signals
  - Artificial limitations required



Continuous frequency of human voice and bandwidth of the analog standard ITU phone channel.

© 2017 Burkhard Stiller

Transmission of Light through Fiber



Wavelengths are high enough to enable high data rates and low enough to require relatively low power for long distance transmission.

© 2017 Burkhard Stiller

ifi

Properties of Typical Physical Media

Medium	*Transmission rate*	*Distance*
Cat 5/6/7 twisted pair	100-1,000 Mbit/s	100 m
Coaxial cable	10-1,000 Mbit/s	200-1000 m
Multi-mode fiber	100-10,000 Mbit/s	< 2 km
Single mode fiber	0.1-10 Gbit/s	10-100 km
DWDM	>128 * 10 Gbit/s	10-80 km
Wireless 2.4 GHz	11 … 600 Mbit/s	50-200 m
Multi-access Wireless 5.2 GHz,	11 … 600 Mbit/s	… 100 m
Multi-access Wireless point-to-p.	Various capacities	Various distances

(without a repeater)

© 2017 Burkhard Stiller

Examples — Gibagit Ethernet Media

Type	Cabling	Waves	Distance	Plugs
1000BASE-SX	62,5 µm Fiber Multimode	830 nm	2 – 260 m	Duplex SC
1000BASE-SX	50,0 µm Fiber Multimode	830 nm	2 – 550 m	Duplex SC
1000BASE-LX	62,5 µm Fiber Multimode	1270 nm	2 – 550 m	Duplex SC
1000BASE-LX	50,0 µm Fiber Multimode	1270 nm	2 – 550 m	Duplex SC
1000BASE-LX	10,0 µm Fiber Monomode	1270 nm	2 – 3000 m	Duplex SC
1000BASE-CX	STP Twinax		25 m	DB9 (Style 1)
1000BASE-CX	IEC 61076 Twinax		25 m	IEC (Style 2)
1000BASE-T	UTP, Cat 5		100 m	RJ45

© 2017 Burkhard Stiller

Cabling Examples (1)

- UTP Cat3



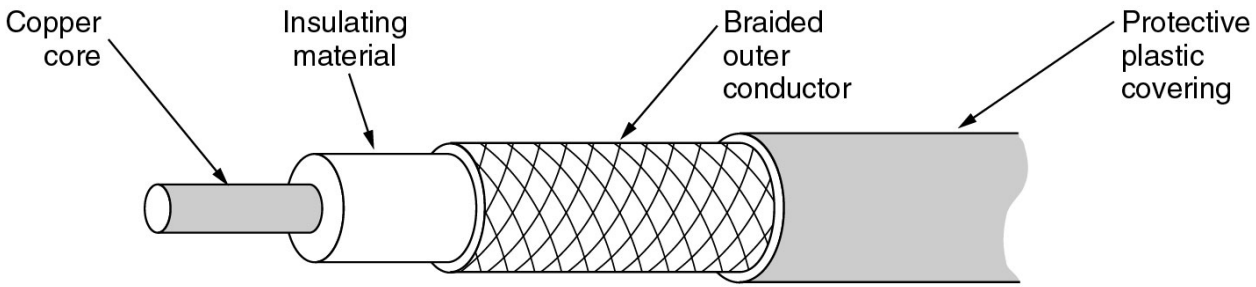
(a)

- UTP Cat5

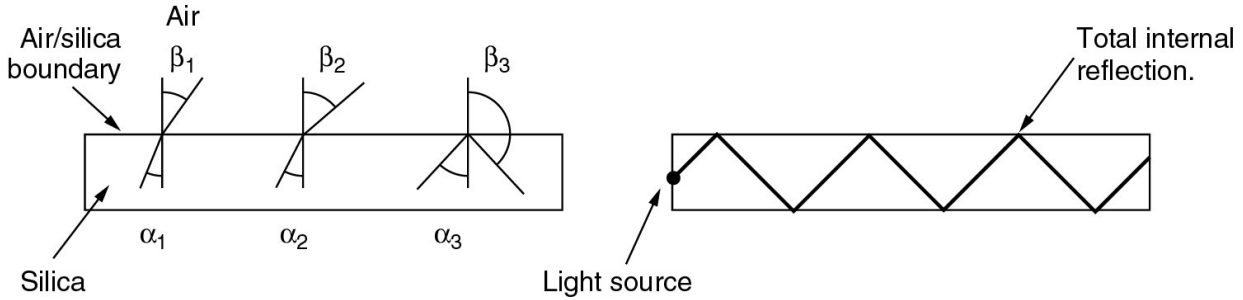


(b)

- Copper



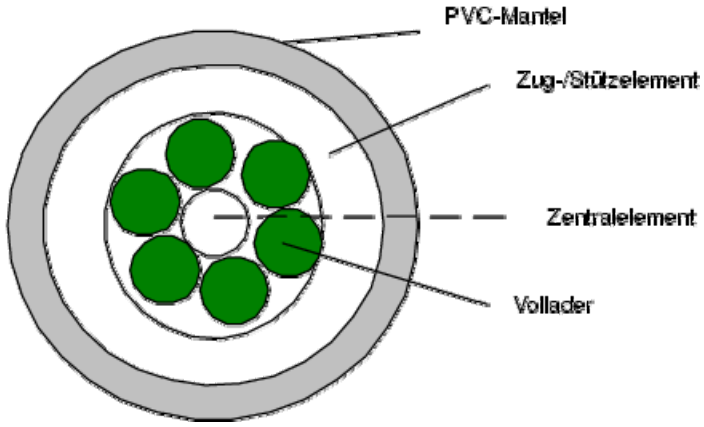
- Optical



(a) (b)

Cabling Examples (2)

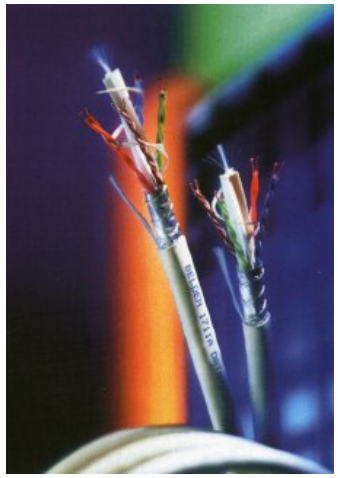
- Fiber optic:



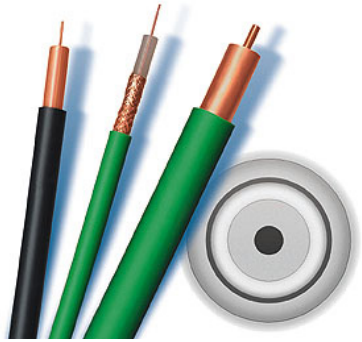


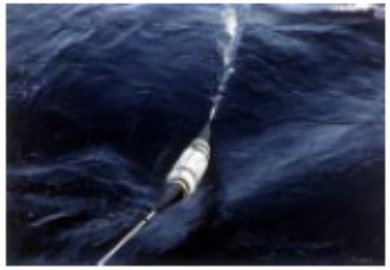
Kabelverlegegeschwindigkeit bis zu 15 km/h in Tiefen bis 7000 m.
Das Füllen eines Schiffes mit Kabel dauert i.a. mehrere Wochen.

- Twisted Pair:



- Coaxial:





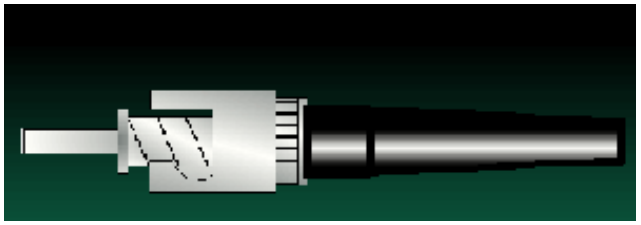


Unterwasserverstärker: 1-2 m Länge, 30-50 cm Breite, 300-500 kg, 40W, 0.5-1 Mio $, in bis zu 7000 m Tiefe.

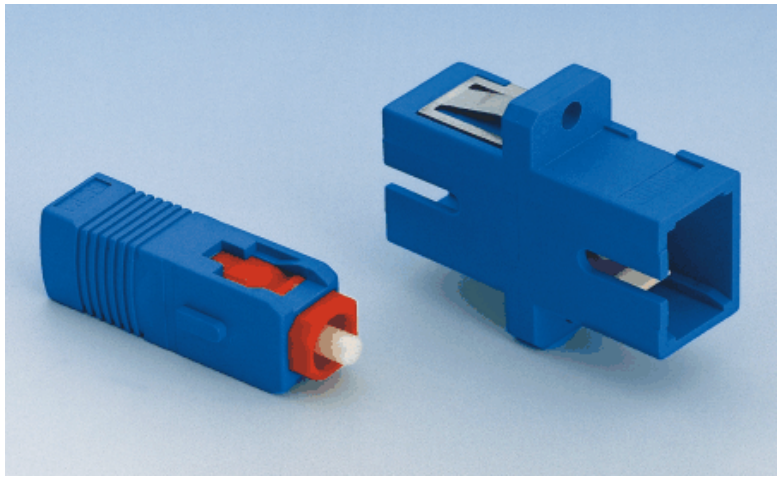
© 2017 Burkhard Stiller

Plug Examples

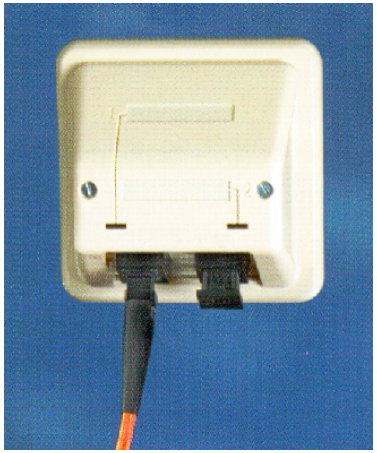
- ST plug:



- SC plug (fiber optic):



- MT-RJ plug:

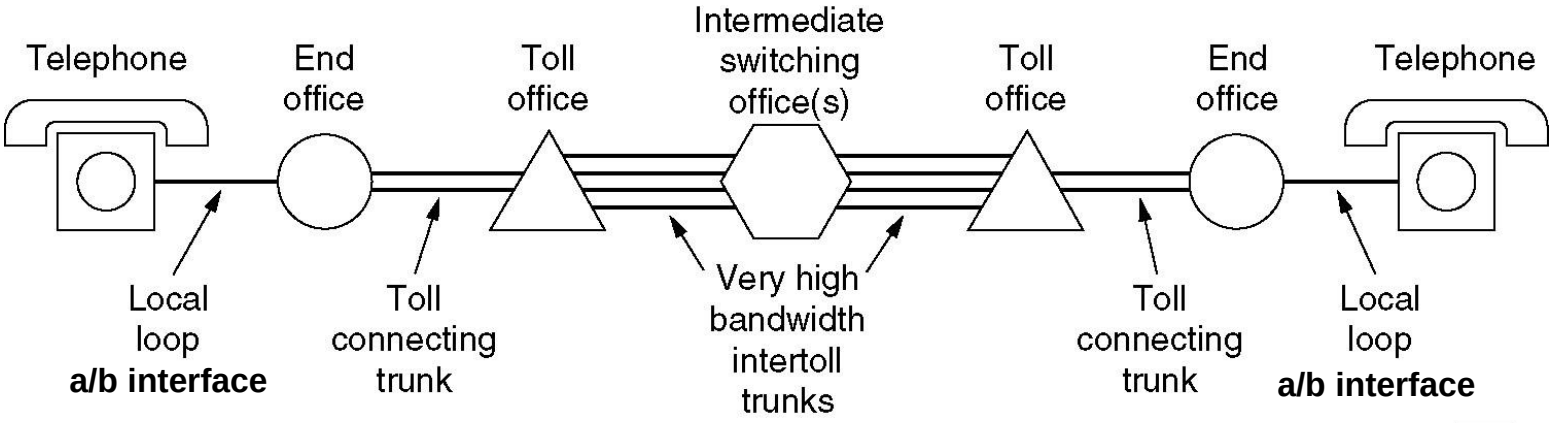


- SMA plug, LSA plug, PKI plug, Galaxy plug, Mini-MT plug, …

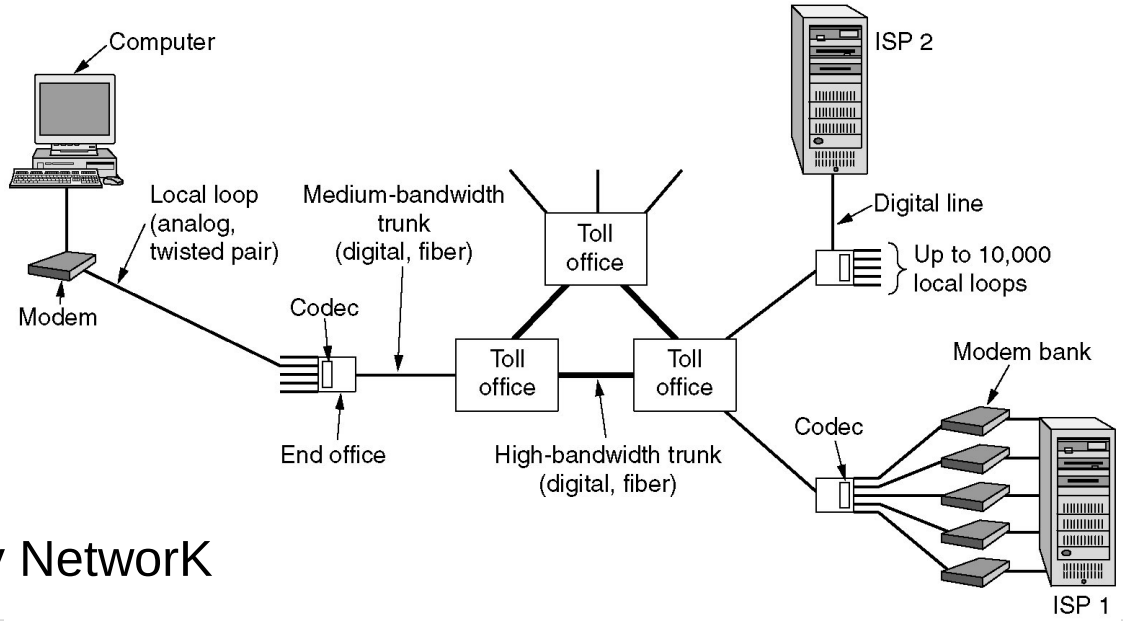
© 2017 Burkhard Stiller

The PSTN

- Traditional set-up:



- Modern Local Loop:



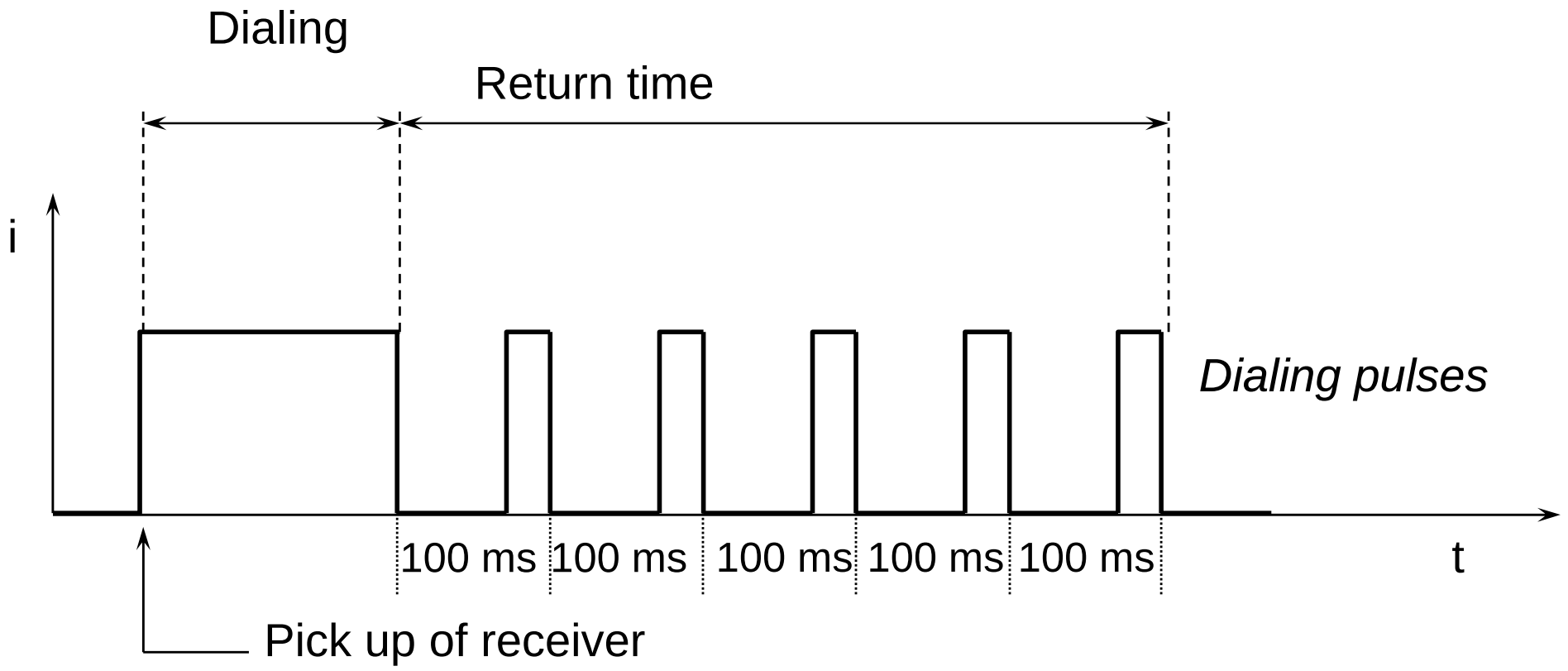
PSTN: Public Switched Telephony NetworK

Phone: a/b Interface

- Name a/b identifies two cables for the access to the phone network in terms of the local subscriber loop
- Power supply phone:
  - Pick up receiver: direct current from battery
  - Traditional analog phone current from local exchange
  - ISDN requires local power supply
- Signaling:
  - Dialing information (phone numbers)
  - Automated by acoustic signals
  - Pulse code signaling (Impulswahlverfahren)
  - In-band (same channel as voice data)

© 2017 Burkhard Stiller

Pulse Code Signaling (old)



Dialing of number 5 via a mechanical phone

© 2017 Burkhard Stiller

Multi Tone Dialing (today)

- Keyed dialing:
  - Standard for digital dialing for 12 key phones
  - Each key coded by two frequencies (redundancy)
  - Keys to be used for signaling *and* data exchanges

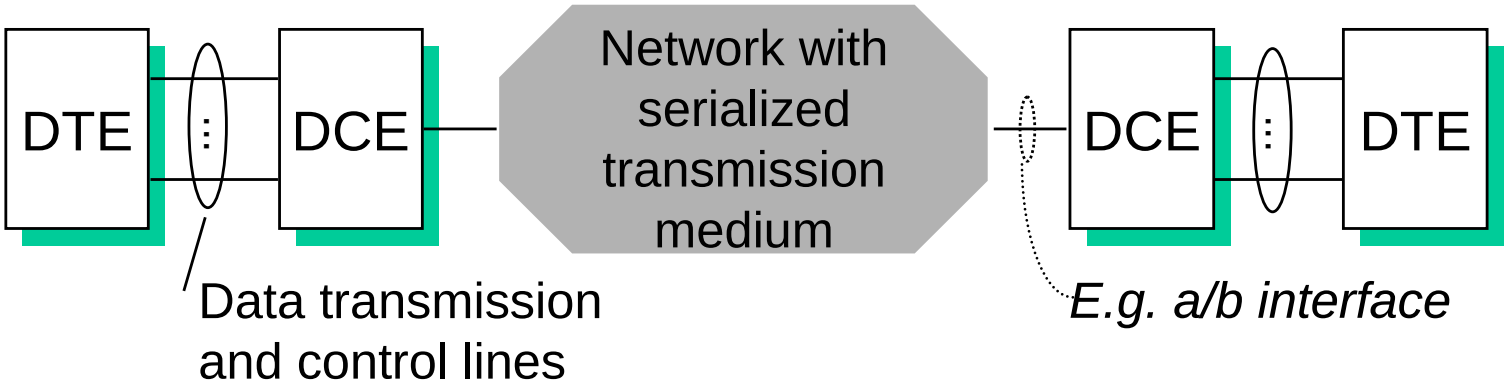
Frequency [Hz]	1209	1336	1447	1633
697	1	2	3	A
770	4	5	6	B
852	7	8	9	C
941	*	0	#	D

Key codes and frequency assignment for multi tone dialing of key phones with 16 keys

© 2017 Burkhard Stiller

Interface for Digital Data

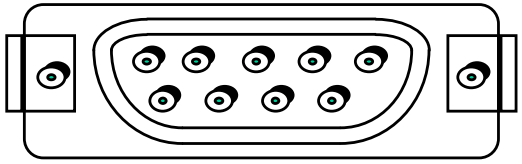
- A Data Circuit Terminating Equipment (DCE) (Datenübertragungseinrichtung) is located between Data Terminal Equipment (DTE) (Datenendeinrichtung) and the network
  - DCE offers functions for adapting the signal to the access link, e.g., for a modem (modulator/demodulator)
  - The DTE/DCE interface enables the user to interface to the network a number of different end-systems



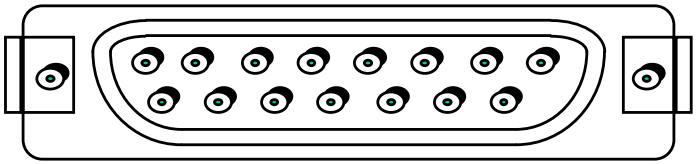
© 2017 Burkhard Stiller

Mechanical DTE/DCE Interfaces

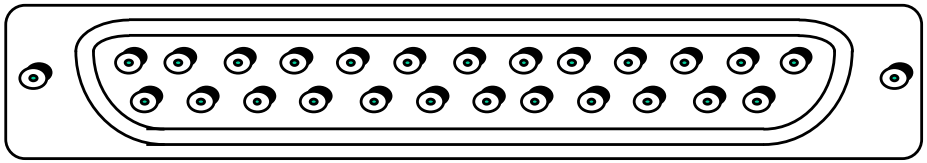
- ITU-T V and X interfaces offer a different number of lines
  - Standardization primarily by ISO, ITU-T adapted
- Plugs with 37 pins
- Plug with pins and cable attached to DTE
- Sleeve attached to DCE



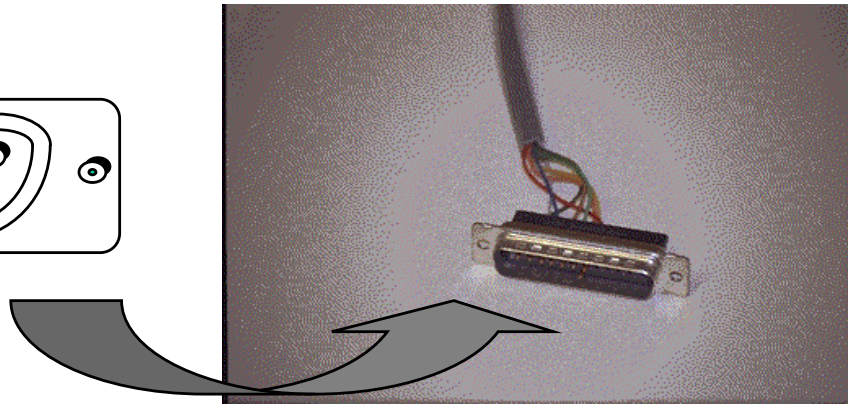
9 pin V interface



15 pin X interface

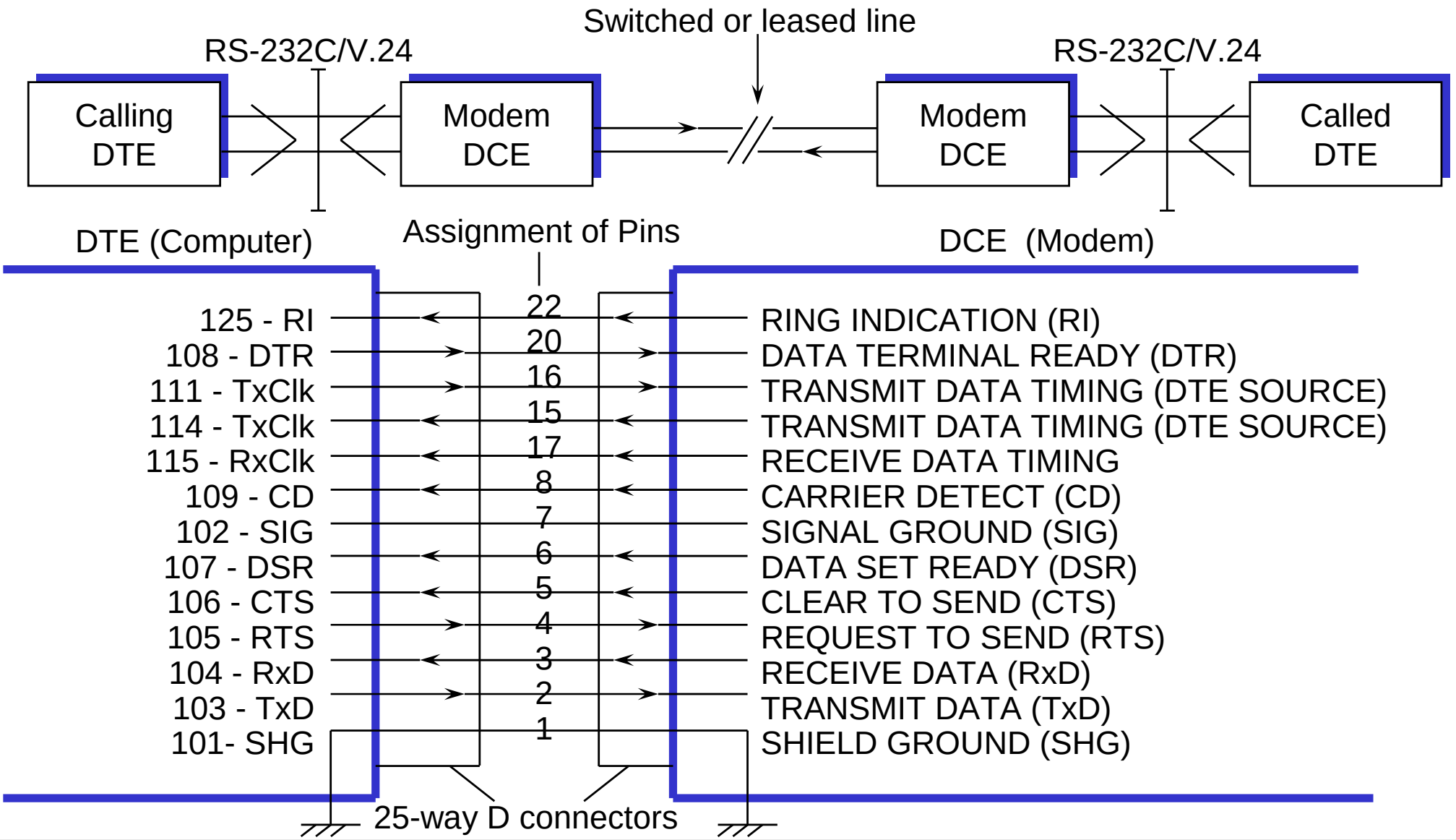


25 pin V interface



© 2017 Burkhard Stiller

Example: V.24 or RS-232C Interface



© 2017 Burkhard Stiller

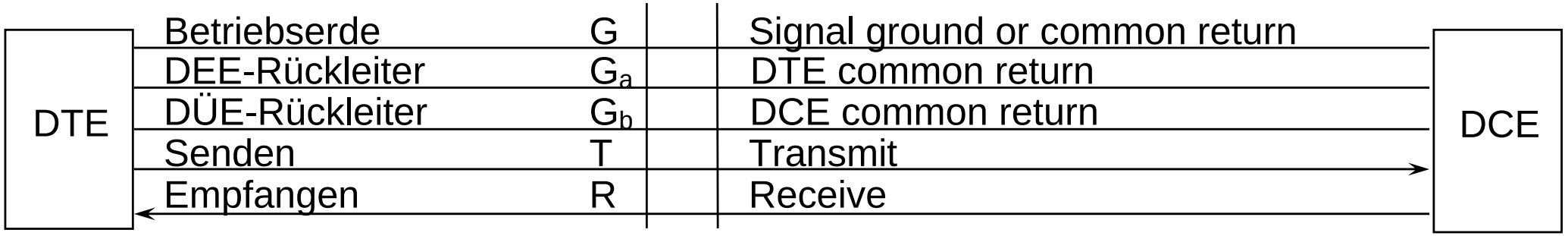
ifi

Comparison of X and V Interfaces

- ”V“ Philosophy: Function by wire
- “X“-Philosophy: Function by code
  - Control and switching functions moved into DCE or into the network, *e.g.*, synchronization and stroke
  - Coding of control and message data:
    - Control (Steuerung): DTE to DCE
    - Message (Meldung): DCE to DTE
  - Only 2 data, 2 control/message, 3 stroke, and 3 ground lines
  - X.20 recommendation:
    - DTE-DCE interface for start/stop transmission in public networks

Terminology based on DIN

Terminology based on ITU



© 2017 Burkhard Stiller

ifi

Modulation (1)

- Transformation of a source signal into a different signal representation

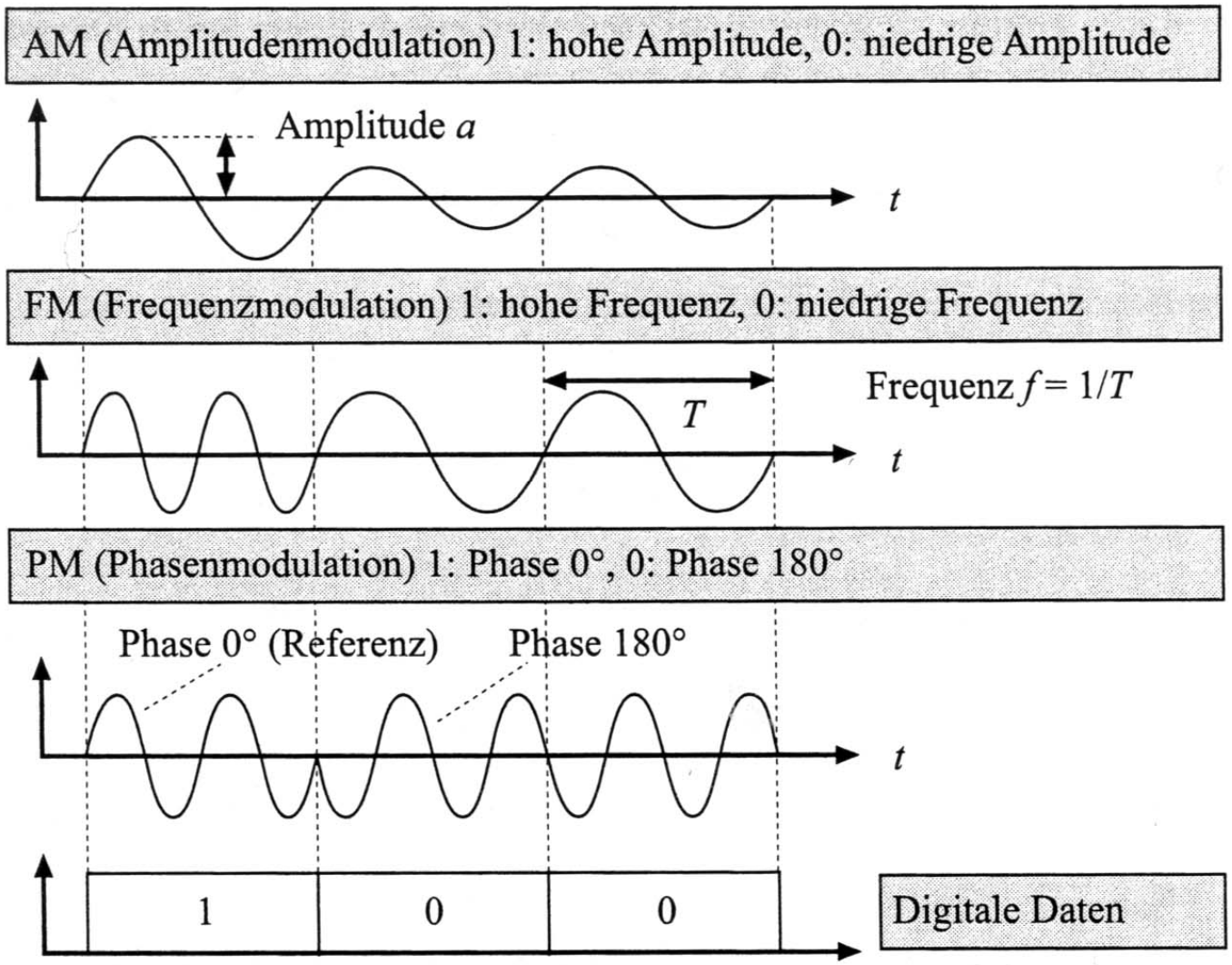
FDM: Frequency Division Multipleying
PCM: Pulse Code Modulations

Data	Transformation	Signal	Example
analog	Modulation	analog	FDM
analog	A/D transformation	digital	PCM
digital	D/A transformation	analog	Modem
digital	Recoding	digital	adaptation to other channels

© 2017 Burkhard Stiller

Modulation (2)

- Carrier is defined by a sinus oscillation (wave)
- Modulated signal may change:
  - Amplitude
  - Frequency
  - Phase



© 2017 Burkhard Stiller

Principles of a Modem (1)

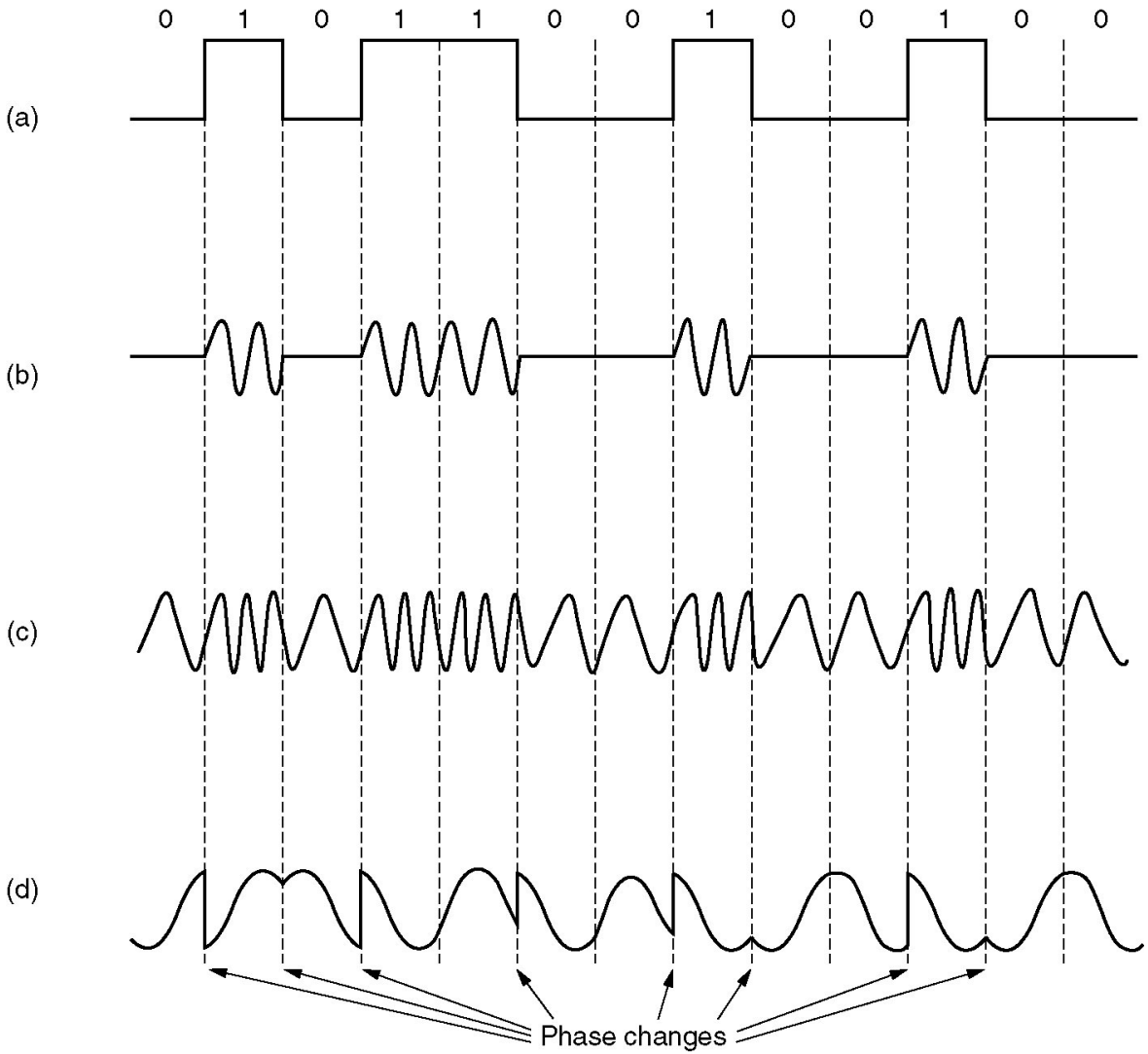
- Original signal modulated with different modulation schemes for transmission

(a) Binary signal

(b) Amplitude modulation

(c) Frequency modulation

(d) Phase modulation



© 2017 Burkhard Stiller

Principles of a Modem (2)

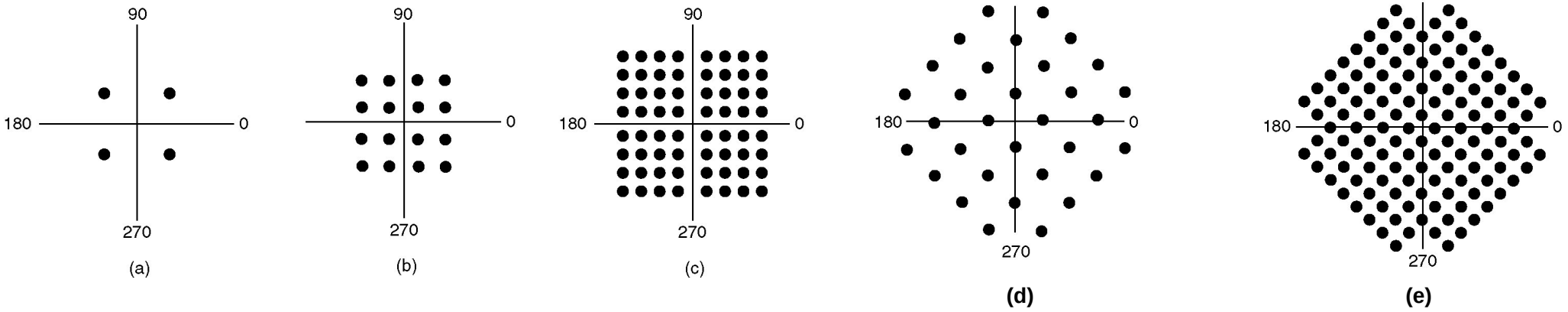
(a) QPSK

(b) QAM-16

(c) QAM-64

(d) V.32 for 9,600 bit/s

(e) V32 bis for 14,400 bit/s



QPSK: Quadrature Phase Shift Keying
QAM: Quadrature Amplitude Modulation

© 2017 Burkhard Stiller

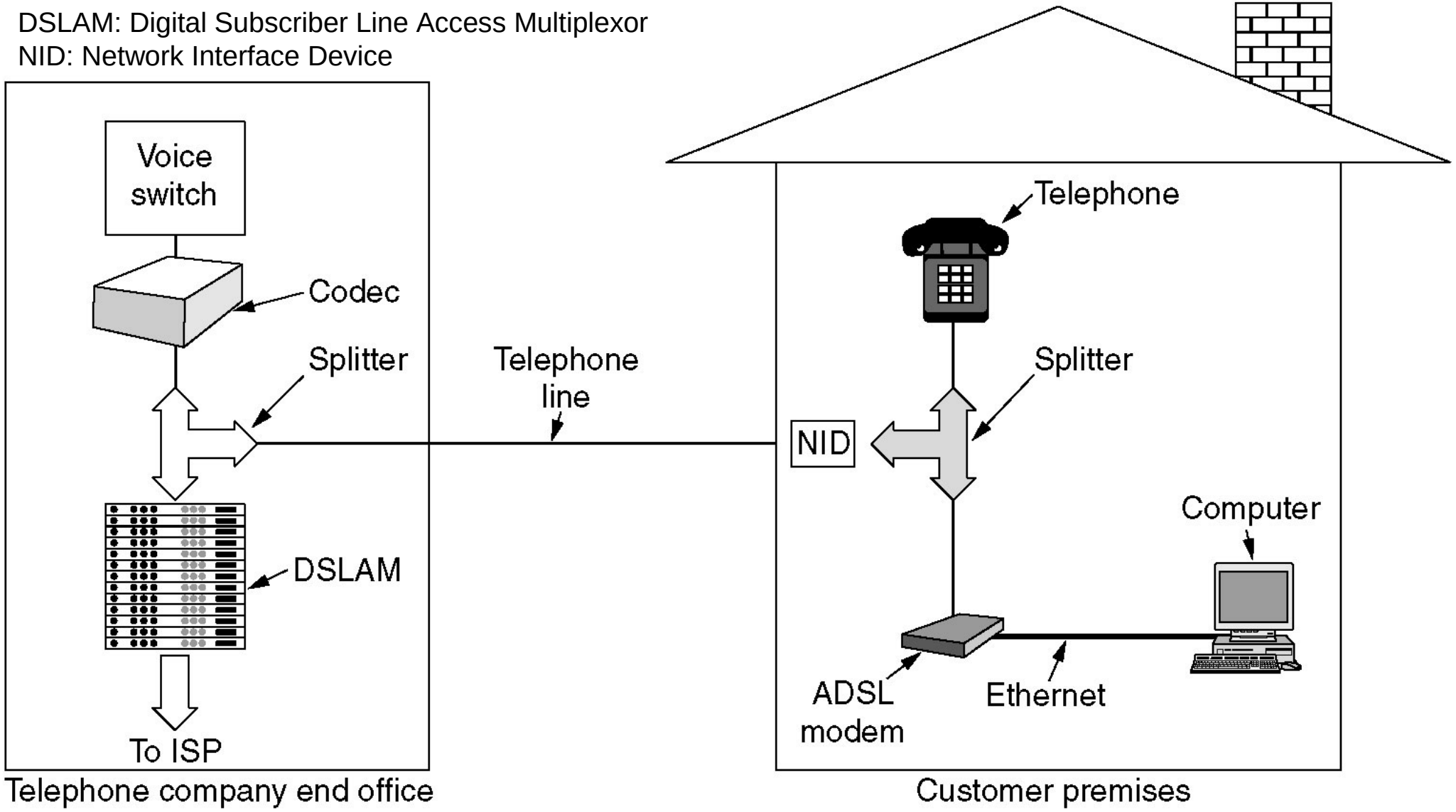
Modem Technologies

- Cable modems — via broadband cables:
  - Extension of frequencies on cable up to 860 MHz
  - Data rates (theoretically) up to 2 Gbit/s on a shared medium
- Powerline communications — via power network:
  - Coupling of high frequency carrier into power lines (16-148 kHz and 1-30 MHz)
  - Data rates (theoretically) up to 2 Gbit/s on a shared medium
- xDSL modems — via traditional phone cables:
  - Typical data rates at 6-8 Mbit/s up to < 100 Mbit/s

© 2017 Burkhard Stiller

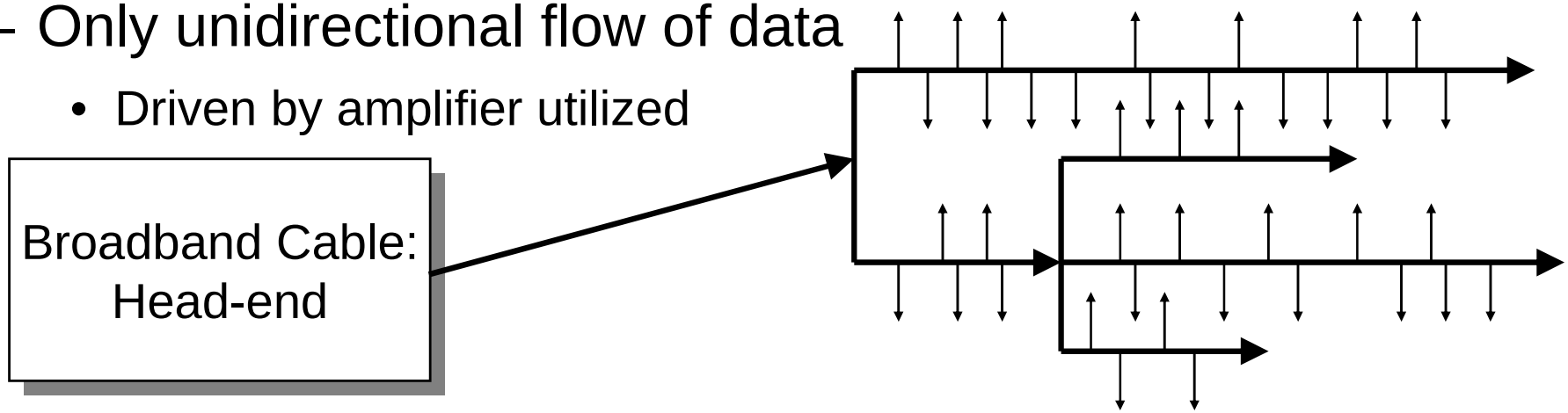
Asymmetric Digital Subscriber Line (ADSL)

DSLAM: Digital Subscriber Line Access Multiplexor
NID: Network Interface Device



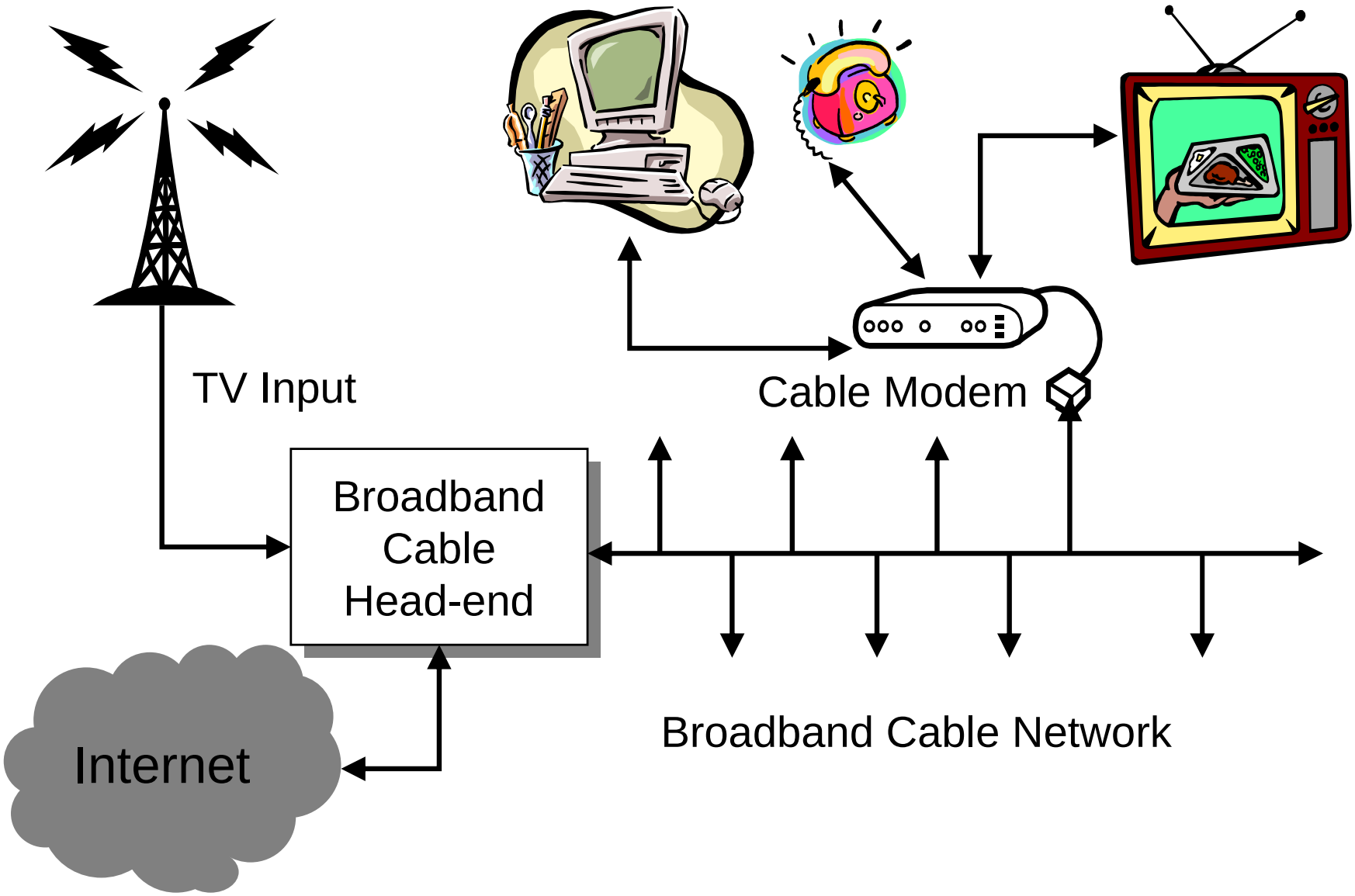
Traditional Broadband Cable Network

- Coax cable-based tree network:
  - All participants are provided by services from a head-end in a point-to-multi-point manner
  - Bandwidth: 606 MHz or 450 MHz
  - Frequency Division Multiplexing:
    - Every service will be assigned a dedicated frequency
    - Number of services is fixed by definition
  - Only unidirectional flow of data
    - Driven by amplifier utilized



© 2017 Burkhard Stiller

Modern “TV Cable” Networks



© 2017 Burkhard Stiller

Reliable …

Link Encoding – Layer 1 (1)

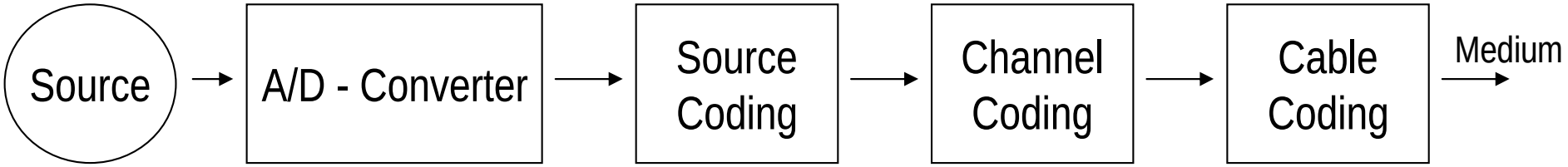
- Consideration of physical characteristics:
  - Digital transmission capacity
  - Energy level
  - Noise level
  - Error resilience (bit/burst)
- Consideration of timing characteristics:
  - Possibility of clock recovery at the receiver
- Consideration of transparency:
  - Will all transmitted bit patterns be received?
    - Consecutive 1's or 0's?
    - Problem of baseline (signal reference) wander

© 2017 Burkhard Stiller

Link Encoding – Layer 1 (2)

- Encoding principle:

A/D: Analog/digital



- Source coding: Reduction of redundancy (loss free or lossy)

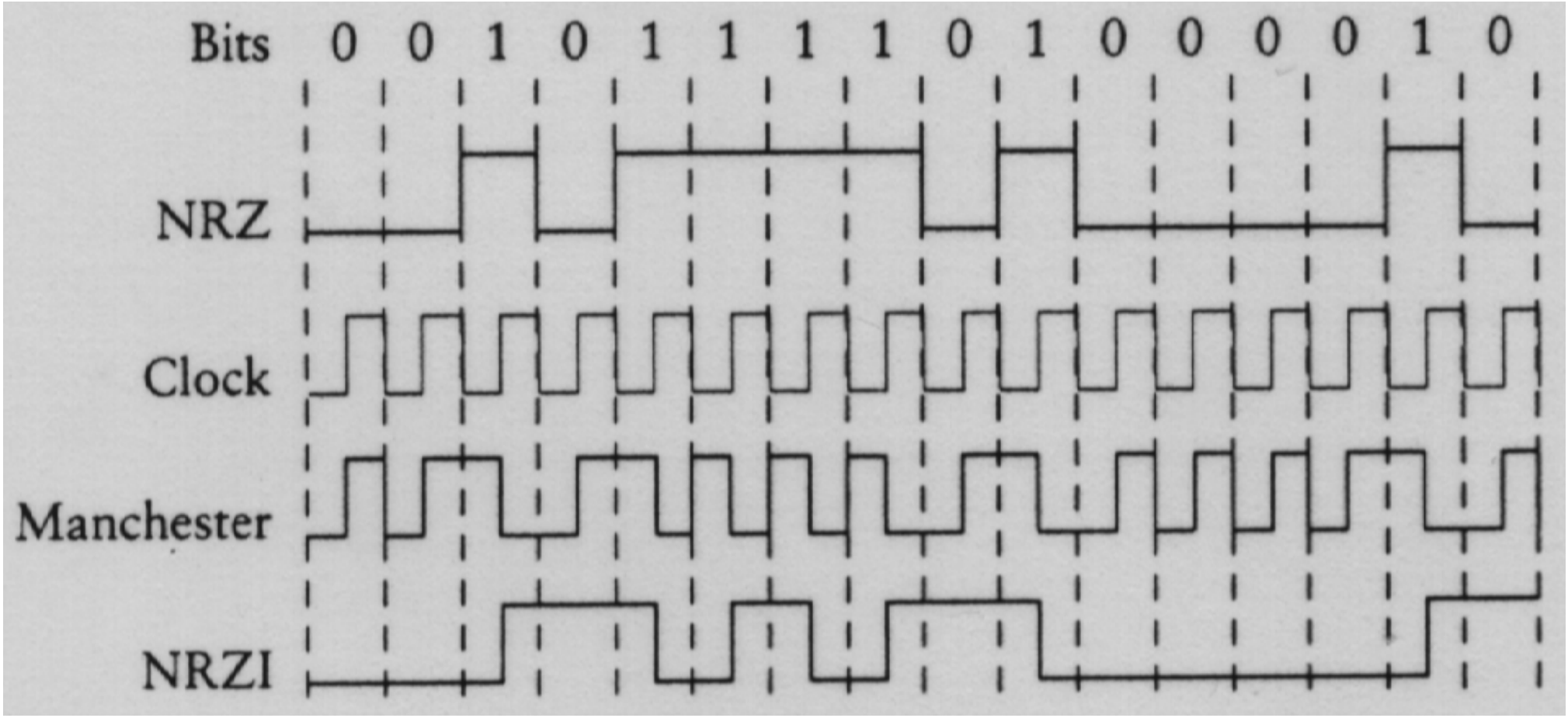
FFT: Fast Fourier Transformation, DCT: Discrete Cosine Transformation

  - Entropy coding (run length, Huffmann)
  - Source coding (prediction, transformation FFT, DCT)
  - Hybrid (JPEG, MPEG, H.263)
- Channel coding: Detection and correction of errors
- Cable/Link (en-)coding: Assignment of bits to signals

JP/MPEG: Joint Photographics/Motion Pictures Expert Group

© 2017 Burkhard Stiller

Link Encoding Methods in the Baseband

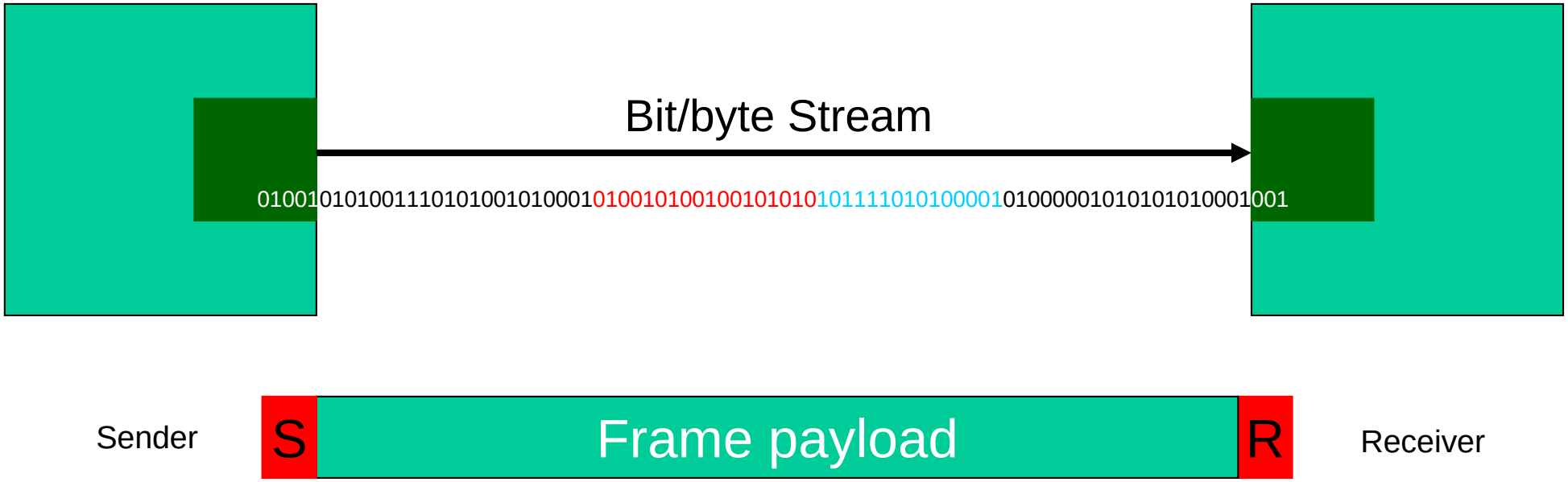


NRZ:Non-return to Zero, NRZI: Non-return to Zero Inverted

© 2017 Burkhard Stiller

Framing (Layer 2)

- To simplify storage management and operations on the bit stream:
  - Frame: Unit of processing at the link of sender/receiver
- In general: Frame <> packet



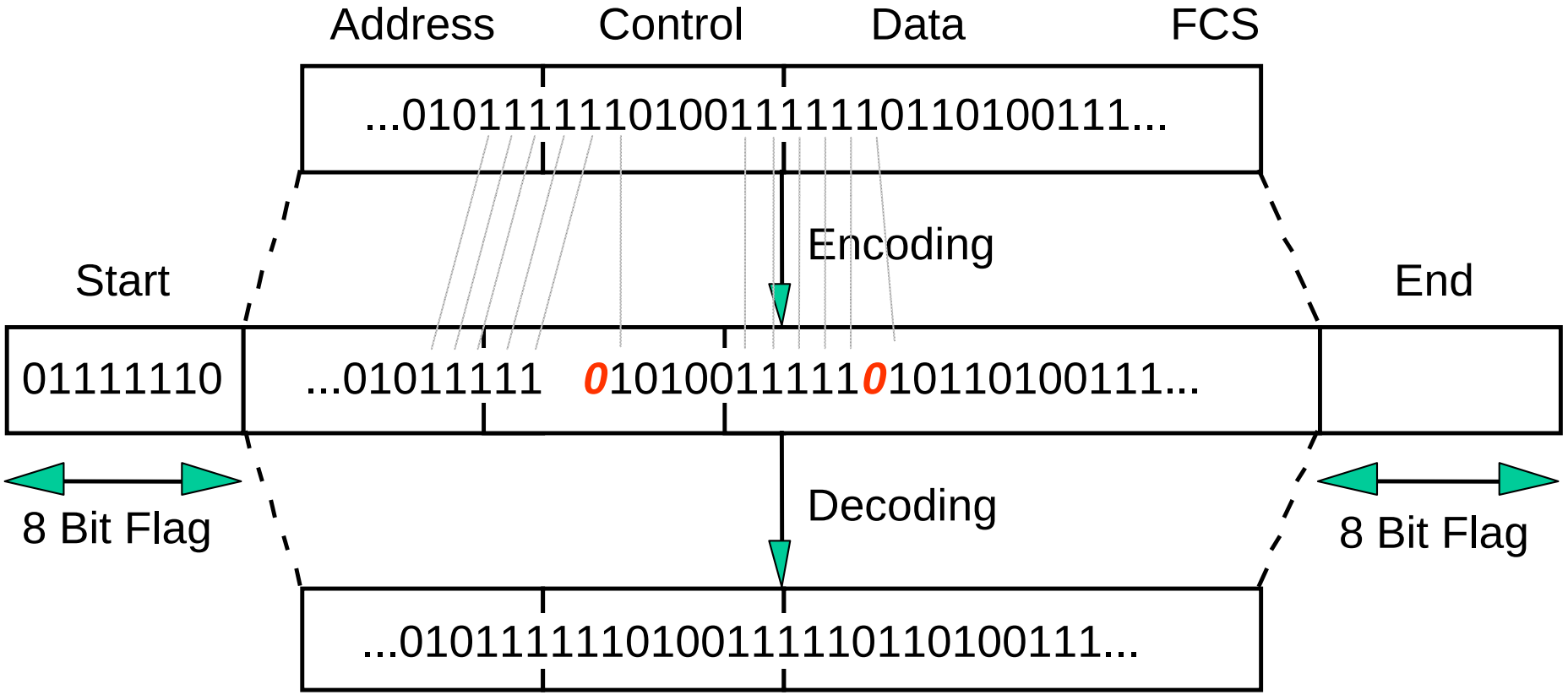
Framing Approaches (Layer 2)

- Sentinel-based (markierungsbasiert):
  - Use special pattern to delineate frames
- Counter-based:
  - Include payload length in header
- Clock-based:
  - Used in telephony environment
  - Each frame is 125 µs long
  - Used in plesiochronous/synchronous transmission networks:
    - *E.g.*, Plesiochronous Digital Hierarchy (PDH), Synchronous Network (SONET), Synchronous Digital Hierarchy (SDH)
- Use so-called Frame Check Sequence (FCS) to catch ill-recognized frames

© 2017 Burkhard Stiller

Reaching Transparency with Bit Stuffing

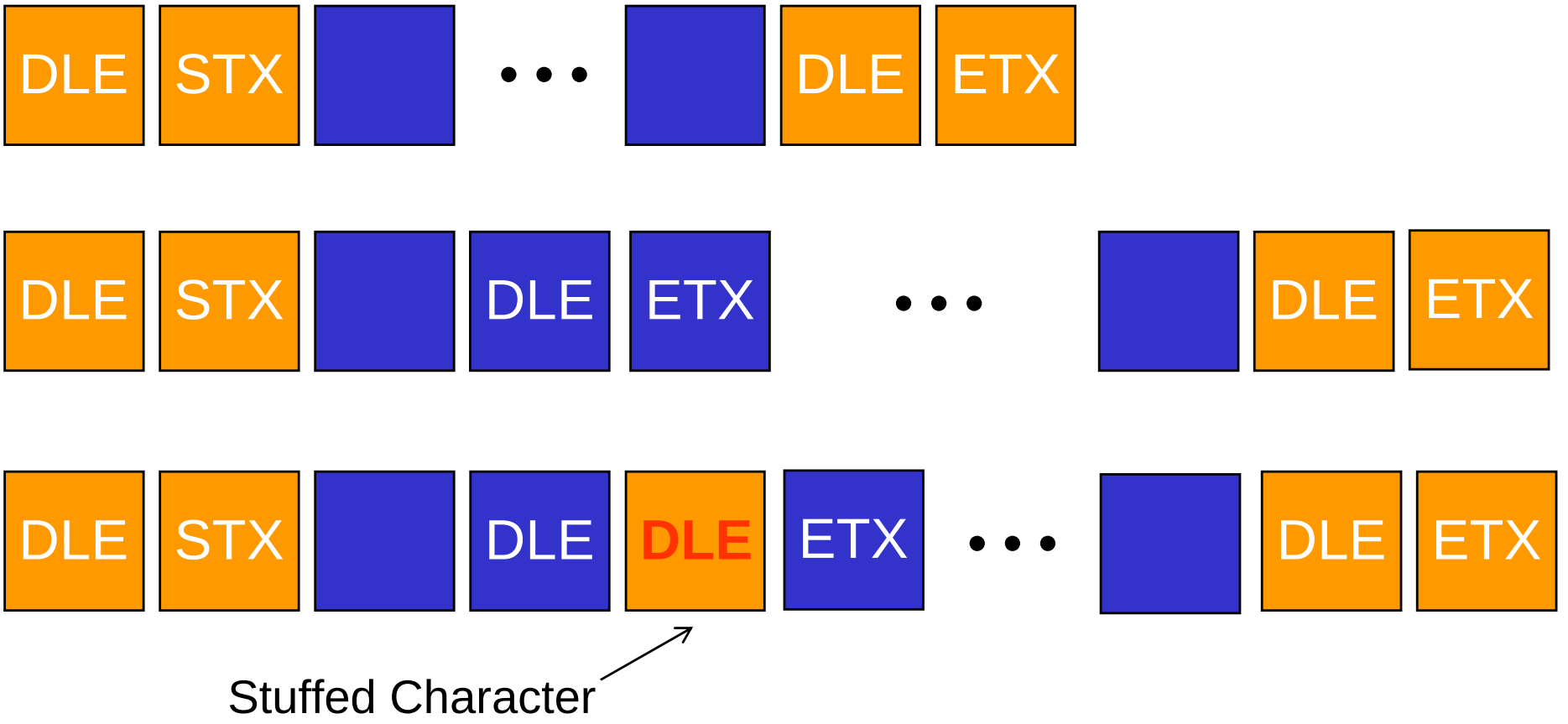
- Sentinel-based schemes need to avoid the mis-interpretation of data for a special start/end pattern.



E.g., High Level Data Link Control (HDLC)

© 2017 Burkhard Stiller

Character Stuffing



DLE: Data Link Escape, STX: Start of Text, ETX: End of Text

© 2017 Burkhard Stiller

ifi

Reliable Transmission on Unreliable Links

- The problem: Physical media are erroneous
- The goal: To guarantee a reliable transmission service
  - Error control:
    - Detect and fix single bit errors
    - Detect and fix burst errors
    - Detect and retransmit lost frames
    - Detect and eliminate duplicate frames
    - Detect and fix wrong sequences of frames
  - Flow control:
    - Adapt sending rate to receiver capabilities to avoid overflow situations
  - General concerns:
    - Sometimes error and flow control are coupled
    - Keep data in transit under control of some entity

© 2017 Burkhard Stiller

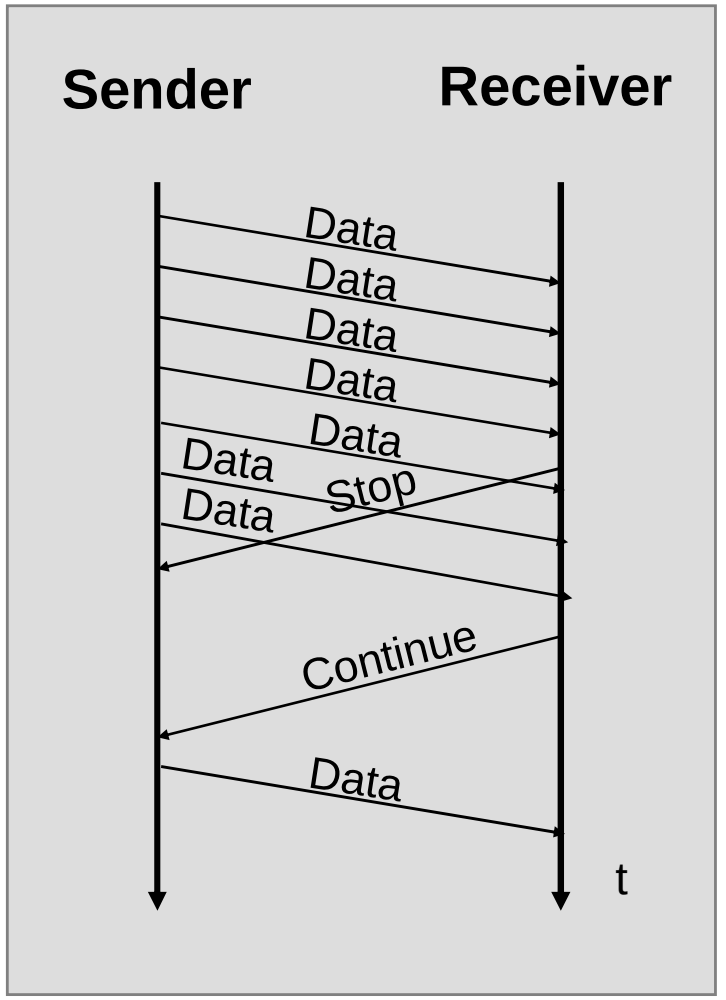
Flow Control

- Flow control:
  - Enabling continuous flow of data
  - Avoiding bursty data sending operations
  - Avoiding buffer overflows at receiver
- Credit- or window-based schemes:
  - Initial credit of N data units:
    - ACK i: Data units up to number i are acknowledged
    - CREDIT j: Sender is allowed to send up to number i+j data units
  - N=1: Stop-and-Go, Idle Repeat Request (IRQ)
  - N>1: Window-based
- Rate-based:
  - Data unit pacing (minimal inter-sending time)
  - Timer or counter

© 2017 Burkhard Stiller

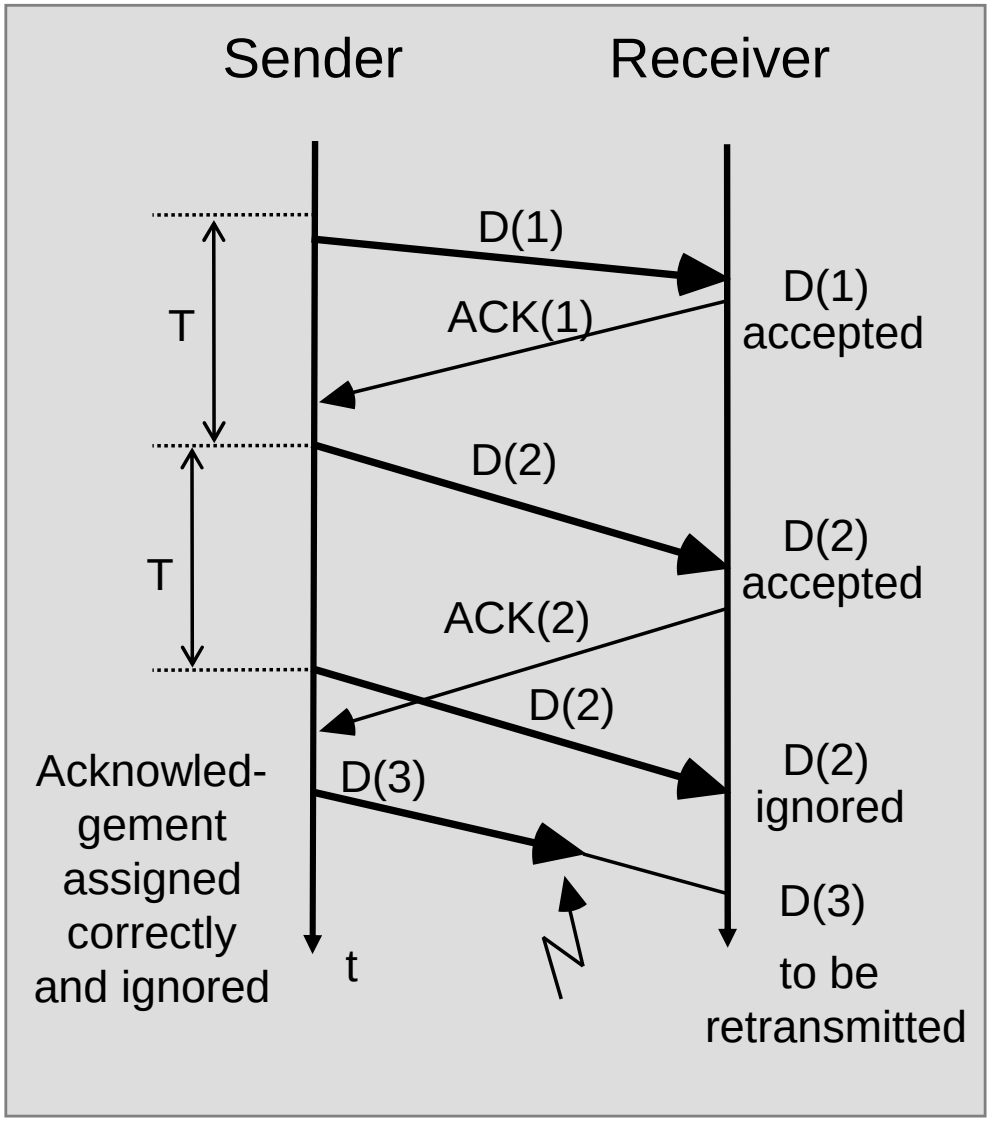
Simple Flow Control

- Simplest flow control method:
  - Sender and receiver exchange two control messages besides data:
    - Stop
      - Receiver is unable to cope with further incoming data
    - Continue
      - Receiver is able to accept further data
- Example: XON/XOFF protocol
    - ISO 7 Bit Alphabet
      - XON is DC_1 (Device Control 1)
      - XOFF is DC_3 (Device Control 3)
      - Suitable for full-duplex transmissions only



© 2017 Burkhard Stiller

Idle Repeat Request (IRQ)

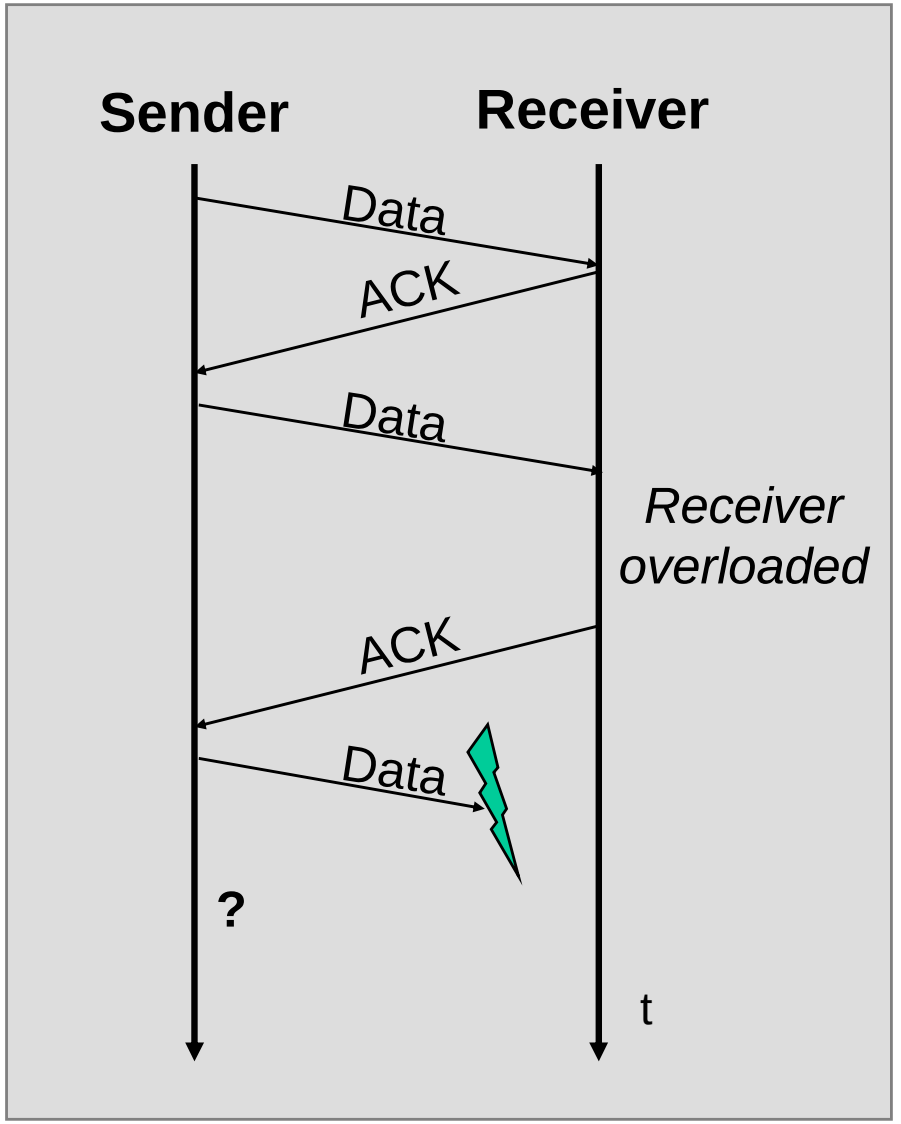


- Frame delineation and transparency
- Frame types:
  - Data D, acknowledgement ACK, and negative ACK
- Timeout T
  - At sender, receiver, or both
- Requires sequence numbers in data and acknowledgement frames

© 2017 Burkhard Stiller

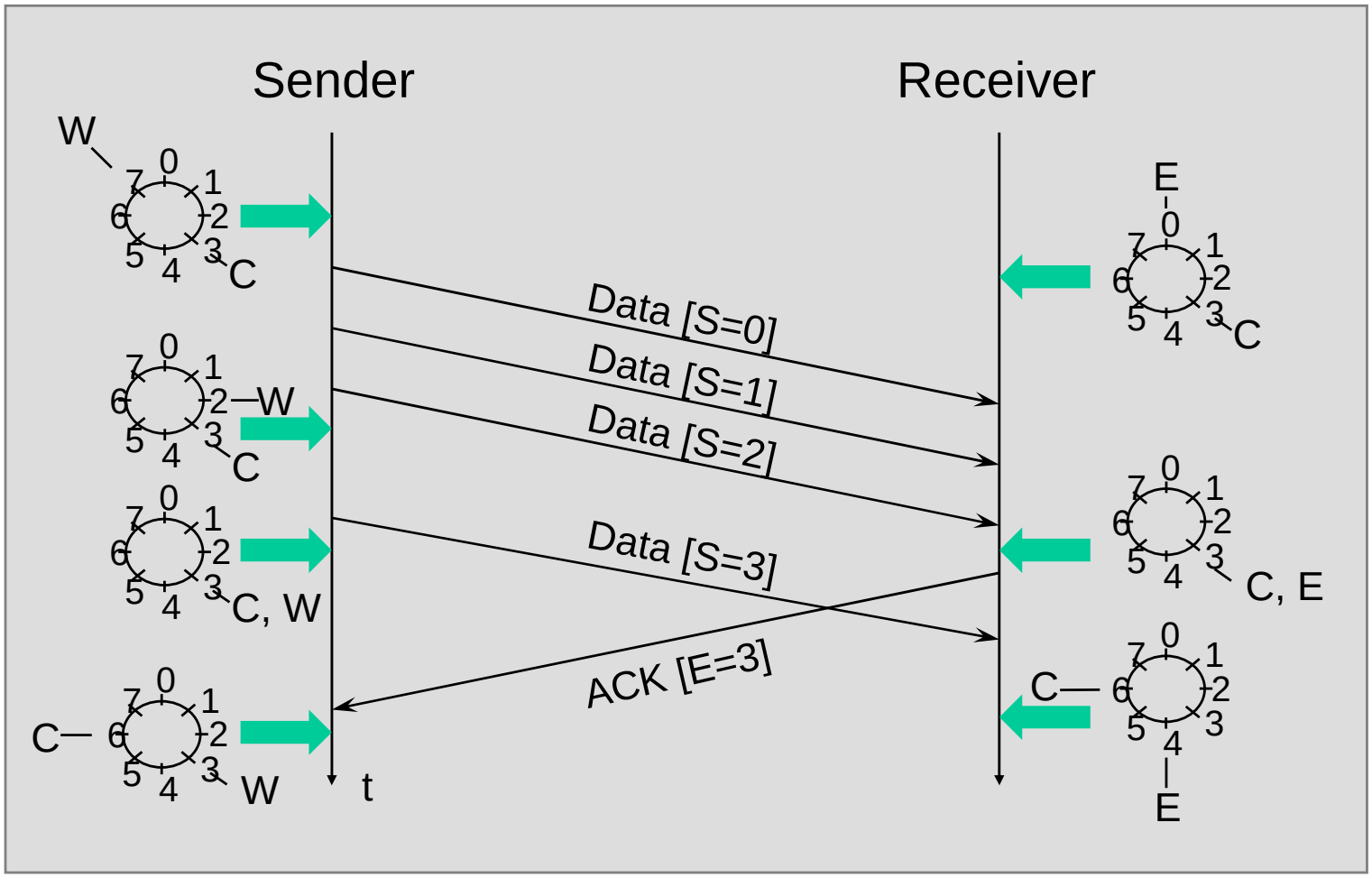
Implicit Flow Control

- Operation:
  - Withholding acknowledgement (ACK or NACK) may slow down the sender
  - Requires an error detection mechanism at the same time
- Problem: Sender is unable to distinguish cases:
  - (a) Packet sent is lost
  - (b) Acknowledgement withhold due to overload



© 2017 Burkhard Stiller

Example: Sliding Window



S: Sequence number of latest packet

C: Counter

E: Expected next sequence number = Acknowledgement up to number R-1

W: Upper window size (maximum possible sequence number) = 4

Drawback: Coupled flow and error control

Flow Control — Jumping Window



Window size: N=100

Window-based Flow Control Problems

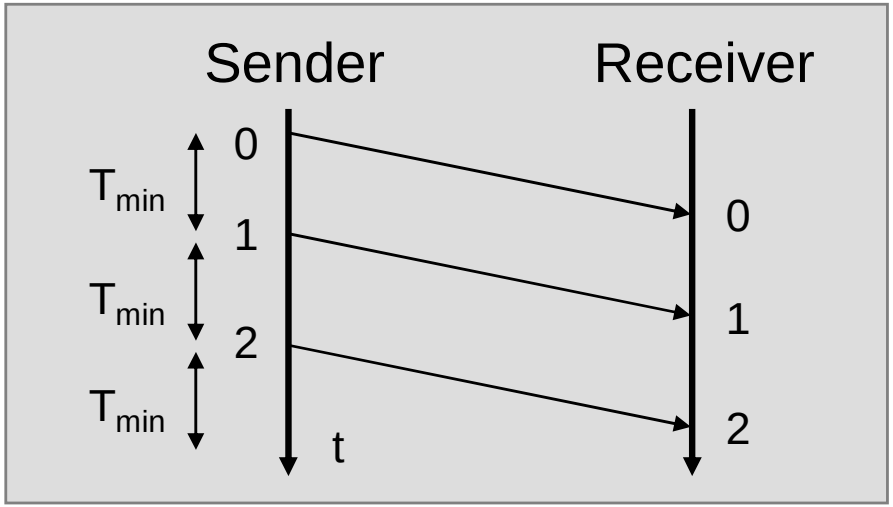
- Major problems:
  - Decision, when to offer a new credit:
    - Trade-off between protocol overhead and smooth sending
  - Long RTT, high path capacity, and small credits result in bursty nature of traffic:
    - Support high bandwidth-delay product/path capacity
  - Amount of data in transit is large
  - Once sent, data in transit cannot be influenced

- A solution is offered by rate-based schemes:
  - Avoiding bursty traffic nature by controlling the burstiness, continuous send operation, and regulating the send rate

© 2017 Burkhard Stiller

Rate-based Flow Control

- Data unit pacing:
  - Limitation of the maximum send rate due to a minimal inter-sending time
  - For every sent data unit a timer is started
  - Following data unit may be sent after time out at the earliest
    - Advantage: Continuous flow of data units
    - Drawback: High resolution timer required



© 2017 Burkhard Stiller

Error Control

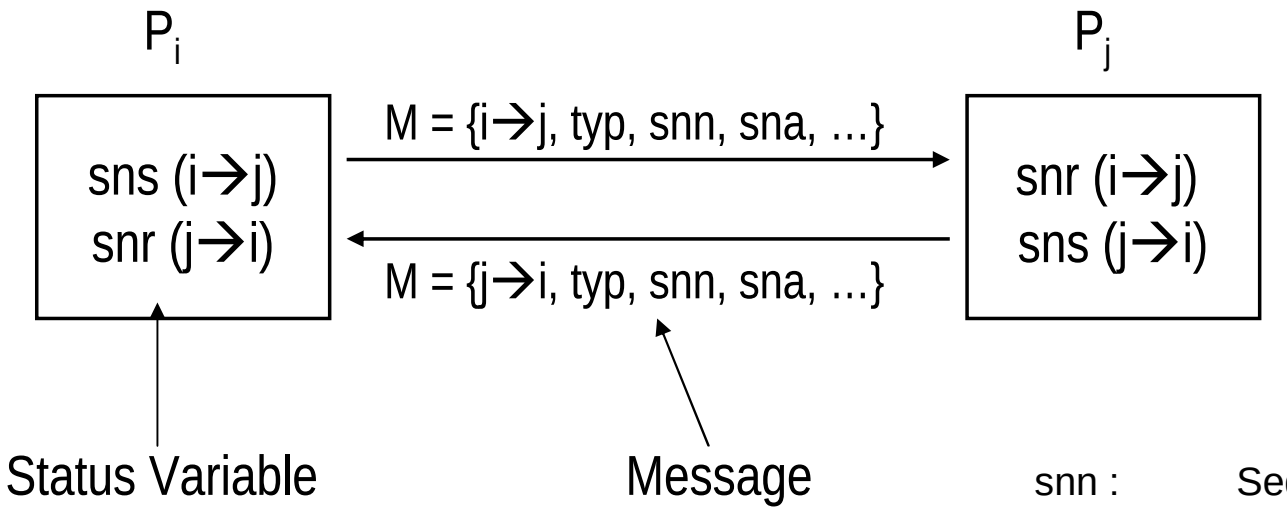
- Error control consists out of:
  - Error detection
  - Error correction

- General protocol functions for achieving reliability comprise:
  - Sequence numbering
  - Acknowledgement
  - Retransmission
  - Forward error correction
  - Checksum calculation

© 2017 Burkhard Stiller

Sequence Numbering (1)

- Goal: Solving the problem of packet loss and duplicate detection, sequencing, and flow control
- Solution: Every communication partner P numbers his messages M sequentially, dependent on the connection



snn : Sequence of a message
sna : No of next message
sns (i→j): No of next message to be sent
snr (j→i): No of next message to be received

Sequence Numbering (2)

M = { i→j, typ, snn, sna, … } defines

i→j : Connection and direction identifier
typ of { NM, ACK, NAK, SYN, CLS, ….}
snn : Sequence of a message
sna : No of next message
sns (i→j): No of next message to be sent
snr (j→i): No of next message to be received

NN normal M, positive ACK, NAK negative ACK,

SYN Connection set-up request, CLS connection release request

© 2017 Burkhard Stiller

Sequence Numbering (3)

Solution: M of P_j arrives at P_i

Case A: snn = snr (j→i): no message is missing

P_i : (1) snr (j→i) +: = 1

(2) Sent ACK to P_j with sna = snr (j→i)

(3) Process messages

Case B: snn > snr (j→i): messages are missing

P_i : (1) NACK to P_j with sna = snr (j→i)

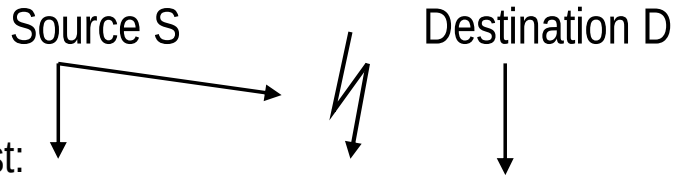
(2) delete messages

Note: Sender needs to store all messages until ACK arrives and not all error cases are considered.

Sequence Numbering (4)

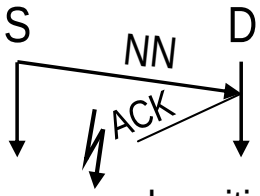
- Duplicates not detected
- Some losses not detected

Case a)



First message will be lost:
- Known sending behavior (Destination will complain)
- Retransmission (Sender complains, Timer)

Case b)



Positive ACK will be lost:
Retransmission of all messages and positive ACK for all correct messages

Case c)

Negative ACK will be lost:
Retransmission required

– Note: ACKs always required, NACK may be omitted

© 2017 Burkhard Stiller

Types of Acknowledgements

- Sender-dependent: Sender requests for ACKs from a receiver.
- Sender-independent: Receiver issues ACKs periodically or event-driven.
- Positive ACKs: Correct reception of data is acknowledged.
- Negative ACKs: Identification of incorrect received data.
- Selective ACKs: Naming of a single explicit protocol data unit or segment.
- Cumulative ACKs: Acknowledgements of one or many sequences of protocol data units or segments.

© 2017 Burkhard Stiller

Types of Retransmissions

- Selective: Retransmission of incorrect data only.
- Go-Back-N: Complete retransmission of data after an incorrectly received data unit arrived.
- Receiver-driven: Automatic Repeat Request (ARQ) as an explicit request.
- Sender-driven: Timer-based request due to missing acknowledgement.

- Sender stores data for retransmission purposes:
  - Buffer and memory requirements for high-speed multimedia data flows may be huge.

© 2017 Burkhard Stiller

Parity Check (1)

- Message consists out of a sequence of characters with m bit
- To transfer sent a sequence of characters and parity (BCC: Block Check Character):

$z_1 = b_{11}\ b_{12} \ldots b_{1m} \mid b_{1\ m+1}$ ← Parity (long)

$z_2 = b_{21}\ b_{22} \ldots b_{2m} \mid b_{2\ m+2}$

:

:

:

$z_n = b_{n1}\ b_{n2} \ldots b_{nm} \mid b_{n\ m+1}$

Parity (cross)

$BCC = z_{n+1} = b_{n+1,1} \ldots b_{n+1,\ m} \mid b_{n+1,\ m+1}$

Parity Check (2)

- XOR, Sum 0 = even parity, odd otherwise
- Example ASCII (7 bit code)

To transfer	even	odd
0000000	00000000	00000001
1111111	11111111	11111110

- Single and double errors detectable. Single errors correctable
- Additional error cases detectable in some cases only

© 2017 Burkhard Stiller

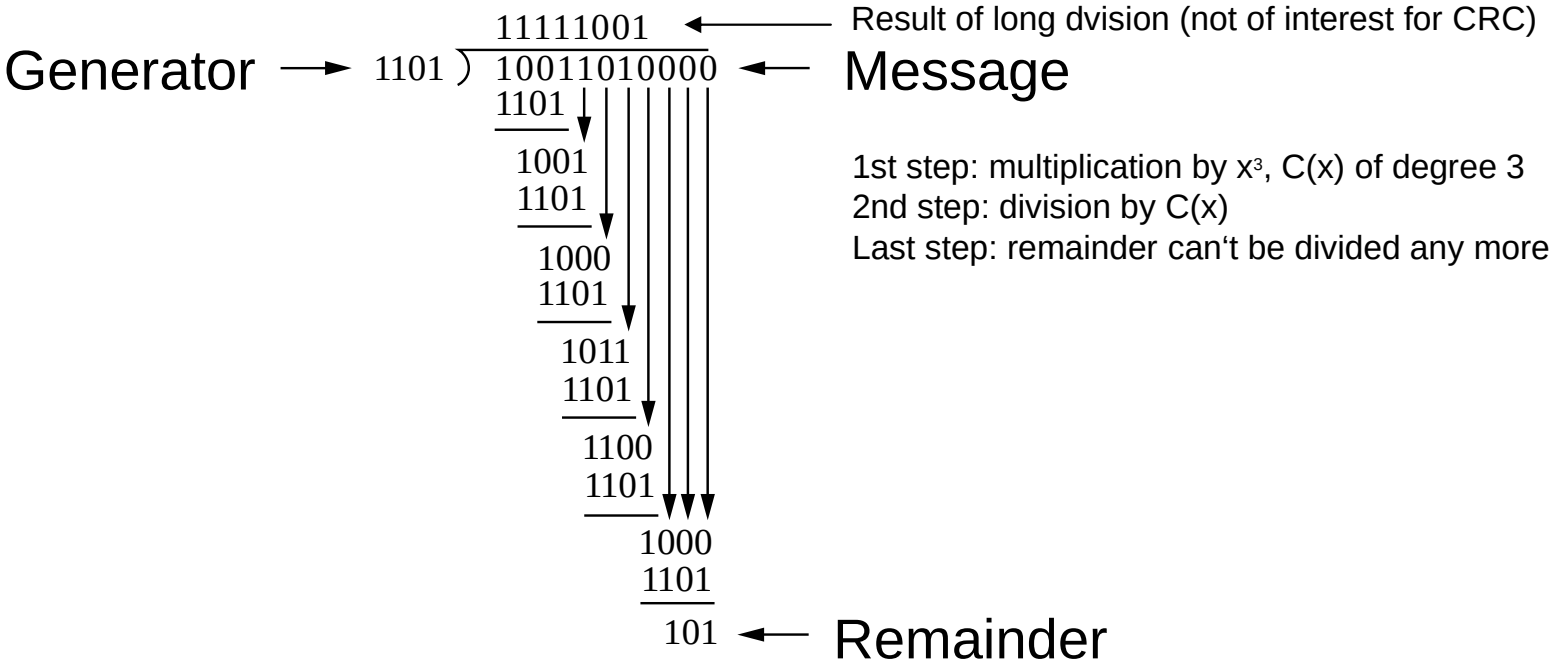
Cyclic Redundancy Check (CRC) (1)

- Add *k* bits of redundant data to an *n*-bit message.
  - Represent *n*-bit message as an *n-1* degree polynomial; *e.g.*, M="10011010" corresponds to $M(x) = x^7 + x^4 + x^3 + x^1$.
- Let *k* be the degree of a divisor polynomial *C(x)*; *e.g.*, $C(x) = x^3 + x^2 + 1$.
  - Transmit polynomial *P(x)* that is evenly divisible by *C(x)*, and receive polynomial *P(x)* + *E(x)*; *E(x)*=0 implies no errors.
- Recipient divides *(P(x) + E(x))* by *C(x)*
  - The remainder will be zero in only two cases:
    - *E(x)* was zero (i.e. there was no error), or
    - *E(x)* is exactly divisible by *C(x)*
    - Choose *C(x)* to make second case extremely rare

Courtesy of L. Petersen

Cyclic Redundancy Check (2)

- Example for a sender:
  - Multiply *M(x)* by *xk*: $x^{10} + x^7 + x^6 + x^4$ (“10011010000”)
  - Divide result by *C(x)* (1101)



  - Send “10011010000 - 101” = “10011010101”, since this must be exactly divisible by *C(x)*;

Minus with binary digits: XOR

Courtesy of L. Petersen

© 2017 Burkhard Stiller

Cyclic Redundancy Check (3)

- Detects all single-bit errors, as long as the *xk* and *x0* terms have non-zero coefficients.
- Detects all double-bit errors, as long as *C(x)* has a factor with at least three terms.
- Detects any odd number of errors, as long as *C(x)* contains the factor *(x + 1)*.
- Detects any “burst” error (i.e sequence of consecutive erroneous bits) for which the length of the burst is less than *k* bits.
- Detects most burst errors of larger than *k* bits can also be detected.

Courtesy of L. Petersen

Cyclic Redundancy Check (4)

- Common polynomials for *C(x)*:

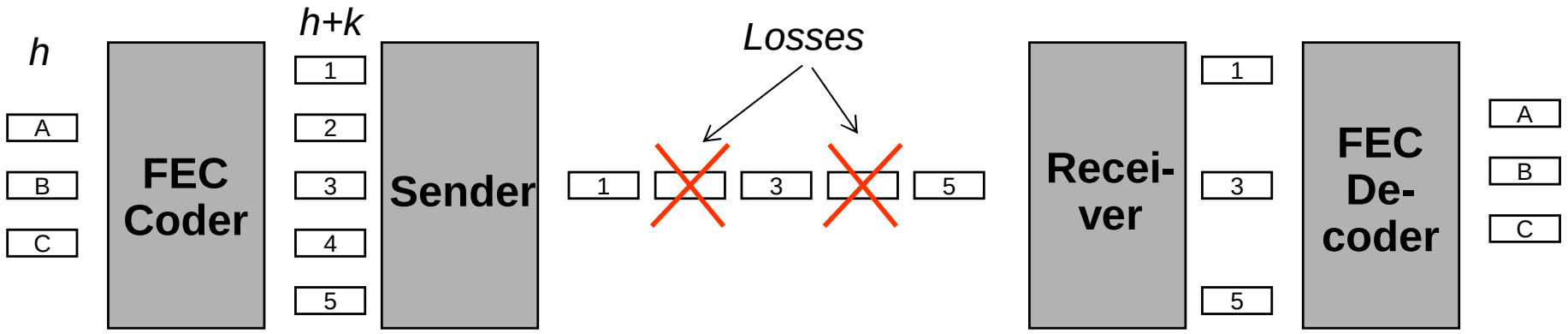
CRC	*C(x)*	Bit char	Bit CRC
CRC-8	$x^8+x^2+x^1+1$	6	6
CRC-10	$x^{10}+x^9+x^5+x^4+x^1+1$	8	8
CRC-12	$x^{12}+x^{11}+x^3+x^2+x^1+1$	8	16*
CRC-16	$x^{16}+x^{15}+x^2+1$		
CRC-CCITT	$x^{16}+x^{12}+x^5+1$		
CRC-32	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$		

* All single, double errors, all errors with odd number of bits, all burst errors <= 16 bit, 99.998% errors of bursts with 18 bit longe

Courtesy of L. Petersen

Forward Error Correction (FEC) (1)

- Error detection *and* correction scheme:
  - Generate out of *h* original data units *h+k* redundant ones applying a dedicated coding scheme
  - Transmit *h+k* data units
  - Receive h arbitrary data units to decode the original content
  - Advantage: Reduced delay in an error case
  - Drawback: Additional bandwidth required for redundant data units, however, small drawback, if *h>>k*



Forward Error Correction (2)

- Example:
  - Send 2 data units (A and B) of length 2 bit each
  - Define *redundant data units* C and D:
    - C = A + B
    - D = A + 10*B
  - A=10 and B=00 generates C=10 and D=10
    - Receive B and C: A = 00+10 = 10

Arrived	Calculation for A	Calculation for B
A, B	–	–
A, C	–	A+C
A, D	–	11*(A+D)
B, C	B+C	–
B, D	D+10*B	–
C, D	11*C+10*D	10*(C+D)

+	00	01	10	11
00	00	01	10	11
01	01	00	11	10
10	10	11	00	01
11	11	10	01	00

*	00	01	10	11
00	00	00	00	00
01	00	01	10	11
10	00	10	11	01
11	00	11	01	10

© 2017 Burkhard Stiller

Discussion — Error and Flow Control

- Go-Back-N advantages and drawbacks:
  - Simple algorithm and efficient implementation
  - In high-speed networks: Error indicating acknowledgements arrive one RTT later at the original sender
    - Loss of bandwidth as high as twice the RTT
    - No buffering at receiver side necessary
- Selective scheme's advantages and drawbacks:
  - Complex algorithm and high management efforts
  - Due to retransmission of incorrect data only, reduced (minimal) loss of bandwidth achievable
  - In high-speed networks: Sequenced delivery of data units requires huge buffers, since twice the RTT indicates the delay of incorrect data

© 2017 Burkhard Stiller

9. Shared Direct Links

Shared Media Access

Local Area Networks

Examples: Aloha, Ethernet, and Token Ring

© 2017 Burkhard Stiller

Shared …

Problem of Shared Media Access

- How to co-ordinate multiple independent senders and receivers using a shared medium for data exchange?

- Various solutions to be considered:
  - Distributed or centralized co-ordination
  - Pre-allocation of medium to each sender
    - Synchronous Time Division Multiple Access (TDMA)
  - Allocation of medium on demand
  - Constant frame lengths:
    - Cell-based approaches, *e.g.*, Asynchronous Transfer Mode (ATM)
  - Variable frame lengths:
    - Random access and contention-based, *e.g.*, Ethernet
    - Reservation-based, *e.g.*, Token Ring

© 2017 Burkhard Stiller

Typical LAN Properties

- Media access in Local Area Network (LAN) technology:
  - Distributed, on-demand, variable frame length
- Technical:
  - Spanning a room or a building (10 m to 2.5 km)
  - App. 10-150 computers connected to one LAN
  - Raw transmission rate (channel encoding) 10-1.000 Mbit/s
- Costs and operation:
  - Cheap adapters available (30 CHF and up)
  - Build and operate it yourself
  - New computers may be connected easily
  - Standardized equipment, products from different vendors are compatible (IEEE 802.x series of standards)

© 2017 Burkhard Stiller

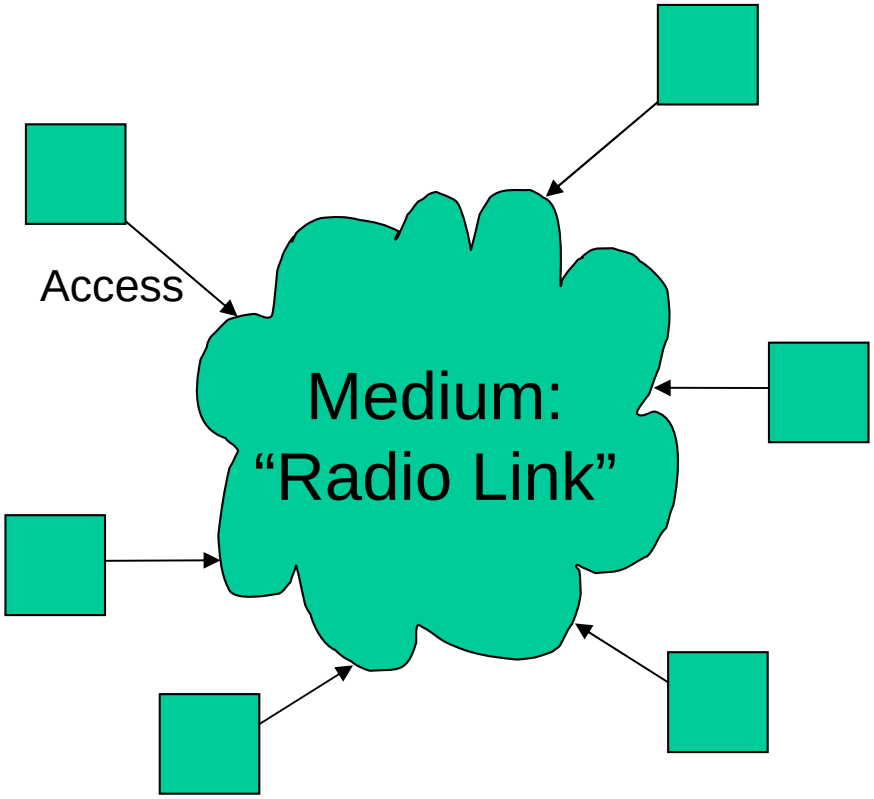
Contention-based Access Methods

- Frames to be transmitted in a time-based multiplexing scheme on a physical medium:
  - At time *t* only a single station must be allowed to sent
  - If a collision occurs, it must be detected
- Co-ordination of accessing the passive medium without any centralized control:
  - Media states: free, busy, contended
    - Collision (contention) is an interference during the sending operation
  - Sending time selected is based on local information only
  - Local observations cannot determine a media state, since electromagnetic waves travel with limited speed

© 2017 Burkhard Stiller

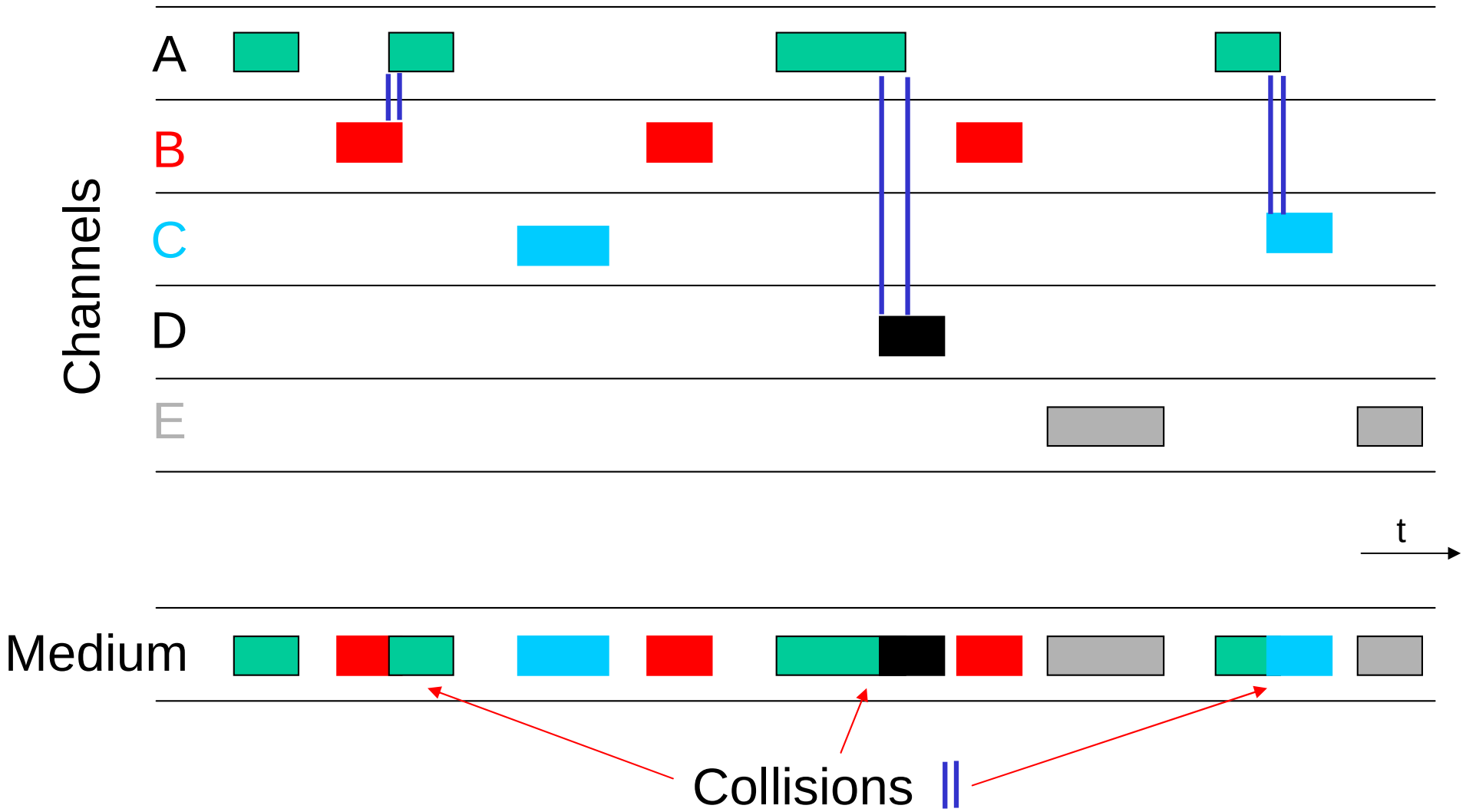
Aloha

- Simplest case:
  - No state of medium considered
  - No co-ordination
- Frame with a checksum:
  - Error correction possible
- Receiver acknowledges and ignores erroneous frames
- Sender retransmits if no ACK shows up in time t (t > RTT)
- AlohaNet: Experimental and productive radio net of University of Hawaii, 1970



© 2017 Burkhard Stiller

Aloha — Access Example

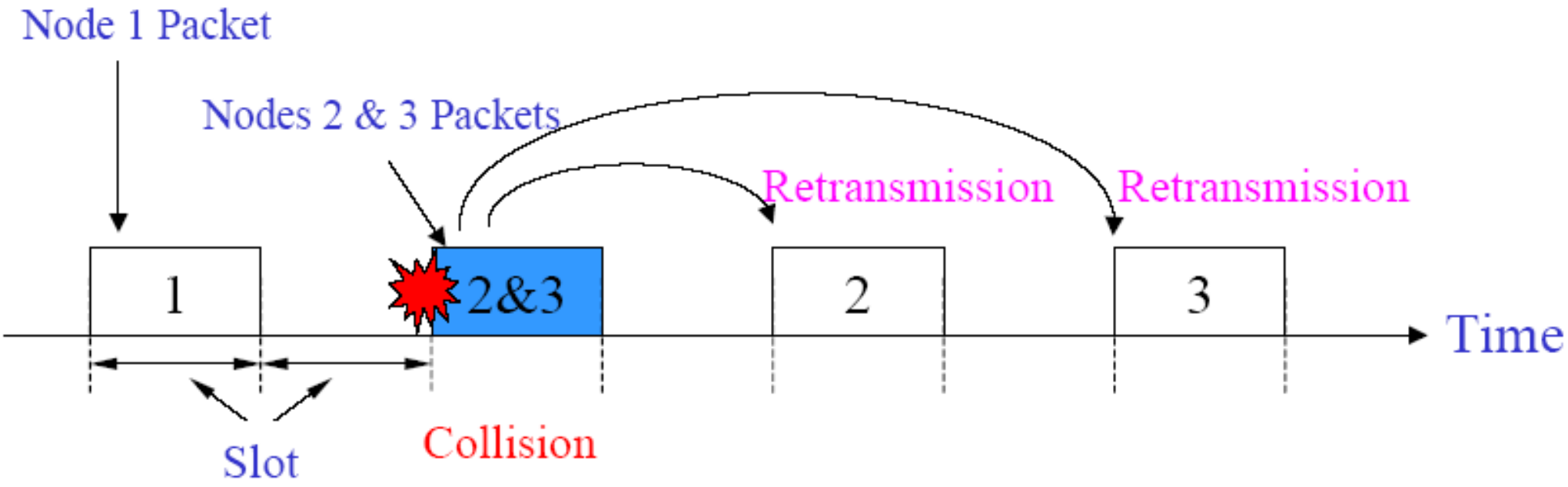


© 2017 Burkhard Stiller

ifi

Slotted Aloha

- Access scheme as within Pure Aloha, however, sending permissions only available for slot starts



© 2017 Burkhard Stiller

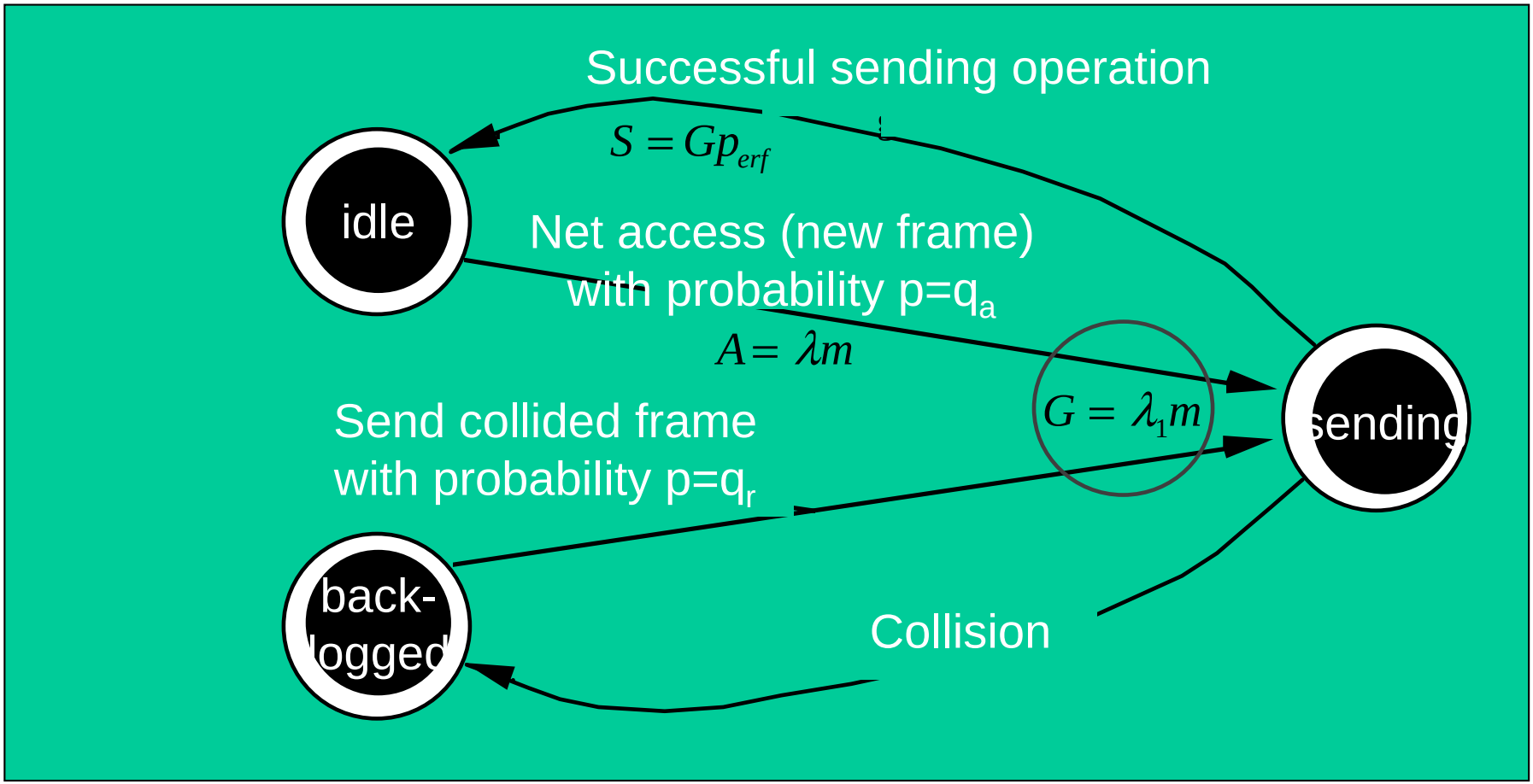
Important Issues and Access Algorithm

- Performance:
  - Function of ***f***(number of stations, traffic load, traffic characteristics, parameters of medium)
- Fairness:
  - Average/maximum time to access the medium
  - Depends on configuration and traffic
- Aloha medium access algorithm:

```
/* send one frame */
repeat
      send(data)
      r=receive()
      if r <> ack then
            wait_random_time
      end
until r=ack
```

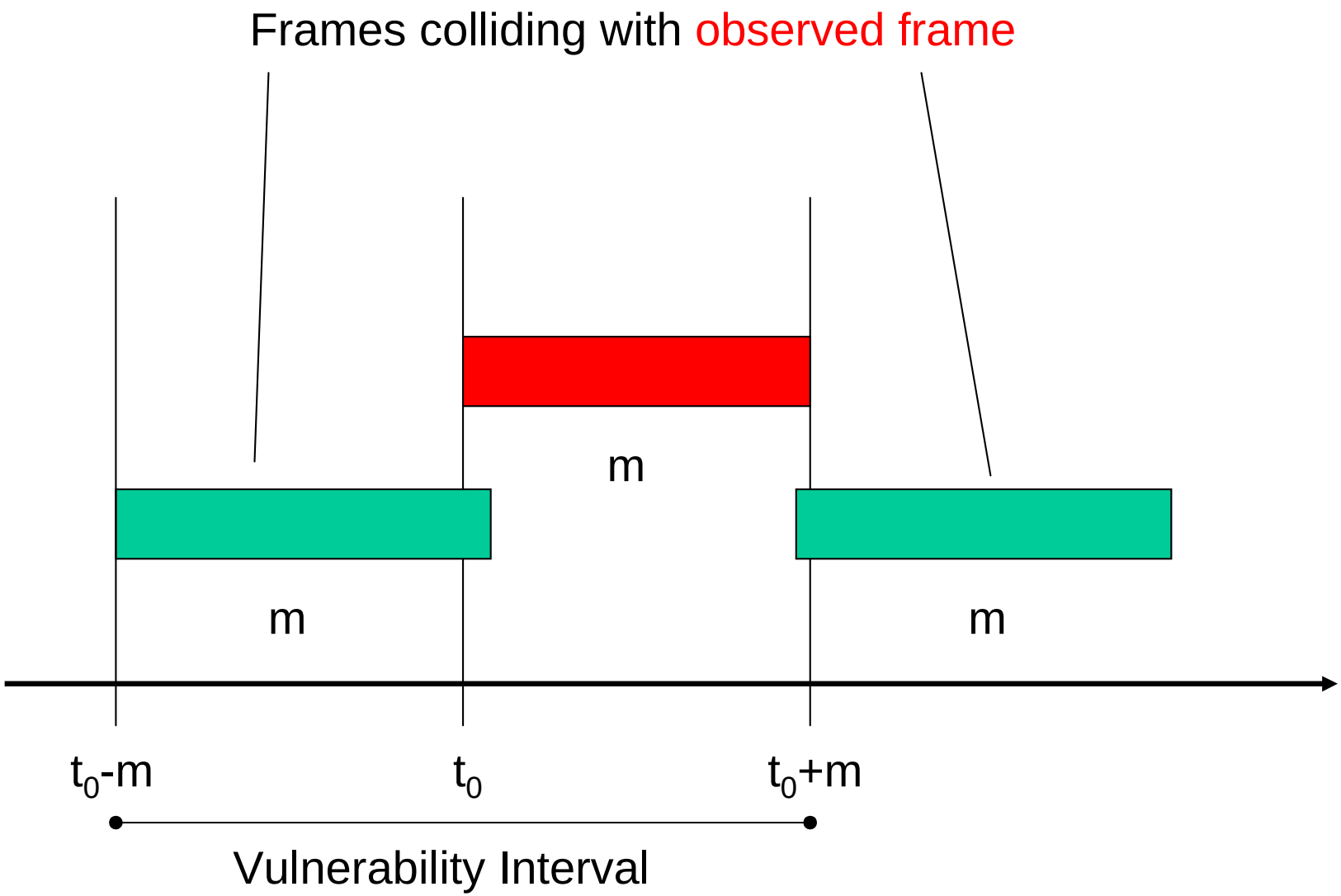
© 2017 Burkhard Stiller

State Machine for Access of Aloha Station



G: Attempts per packet time; S: Throughout per packet time; p_{erf}: successful transmission; m: Frame length

Vulnerability



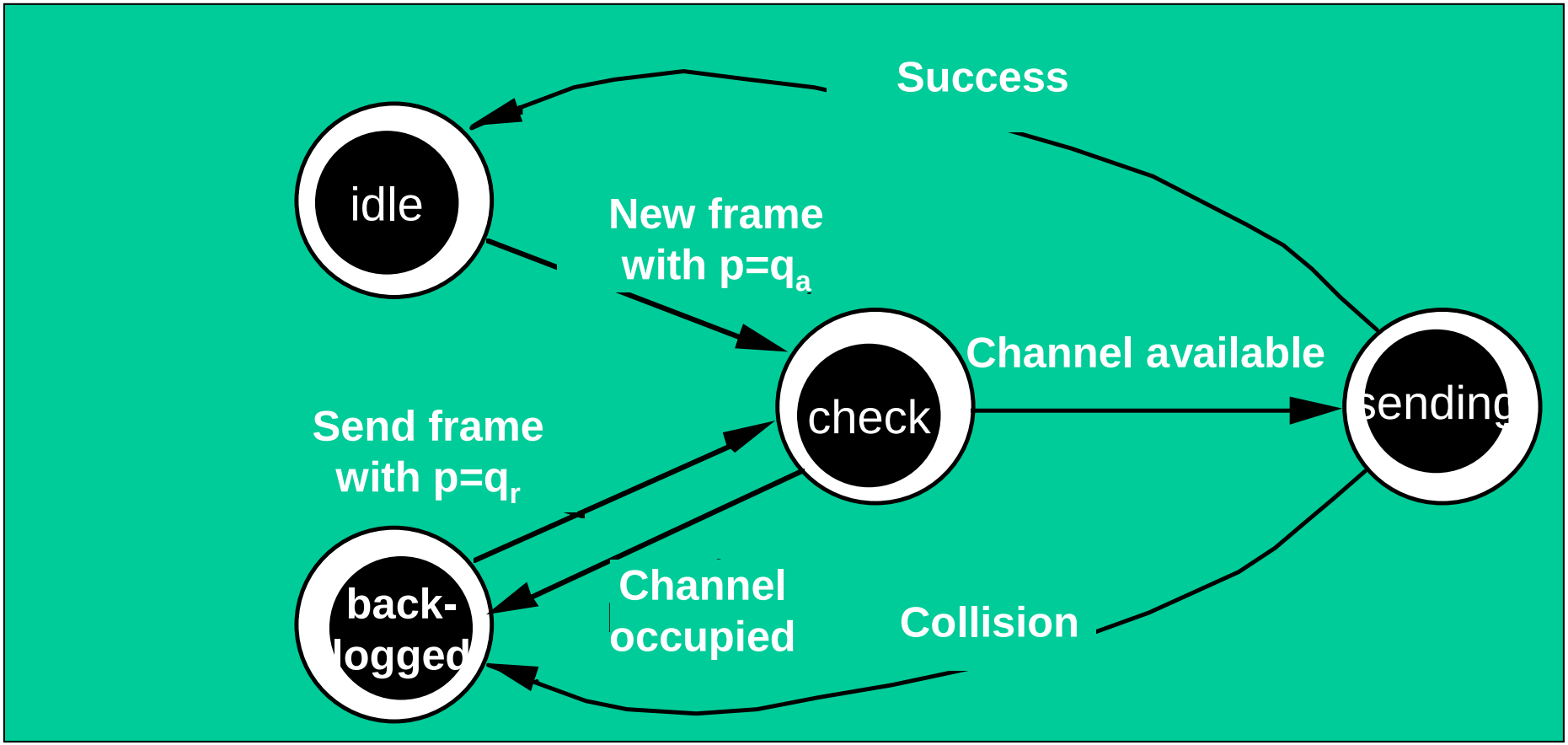
© 2017 Burkhard Stiller

Carrier Sense Multiple Access (CSMA)

- Solving the problem of medium access:
  - Check availability first before sending data

```
/* send one frame */
Repeat
    wait_channel_available()
    send(data)
    r=receive()
    if r <> ack then
        wait_random_time
    end
until r=ack
```

State Machine for CSMA

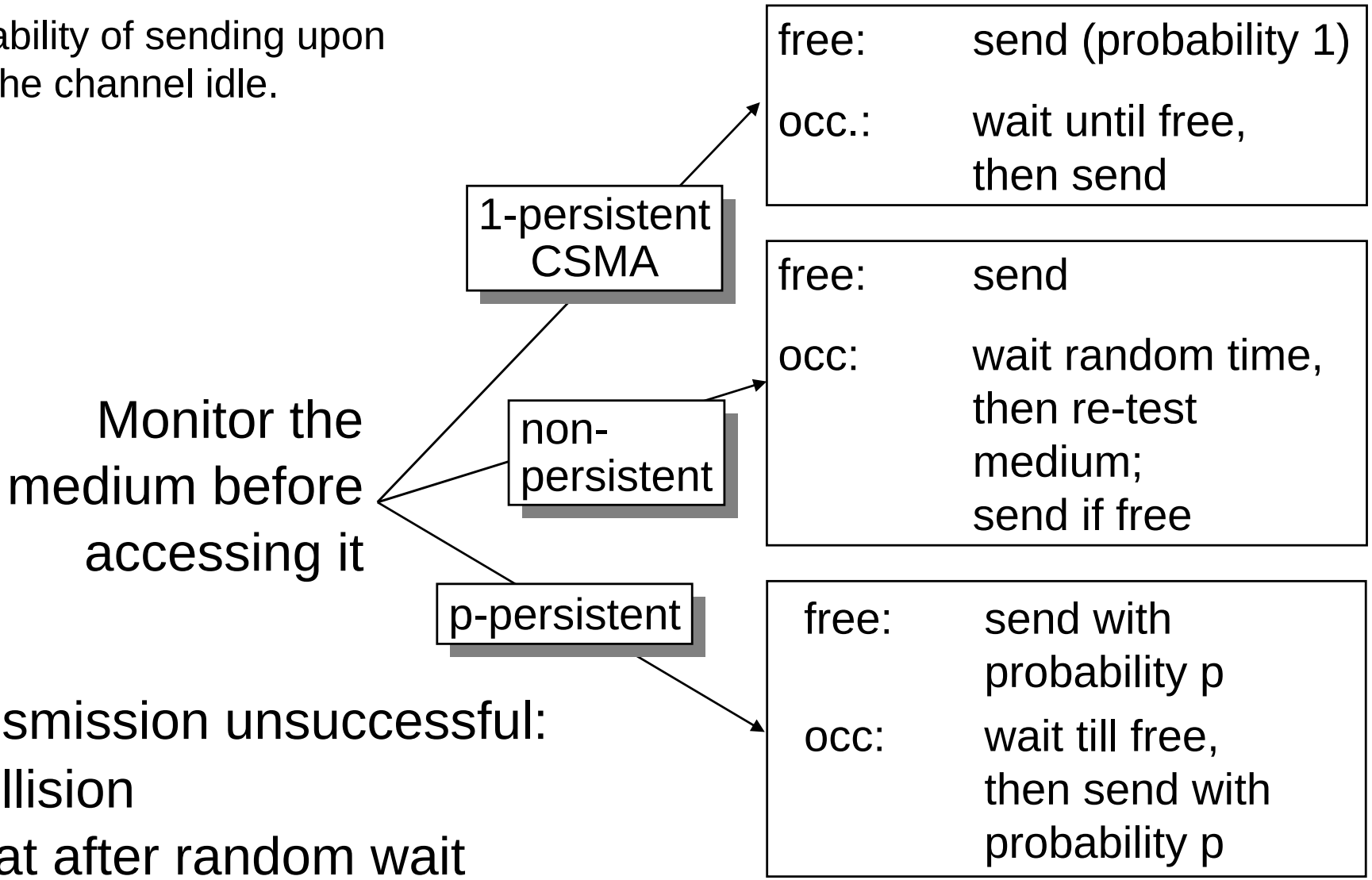


The “check” state determines the ”CS“ of the CSMA access scheme.

© 2017 Burkhard Stiller

Variants of CSMA

p: Probability of sending upon finding the channel idle.



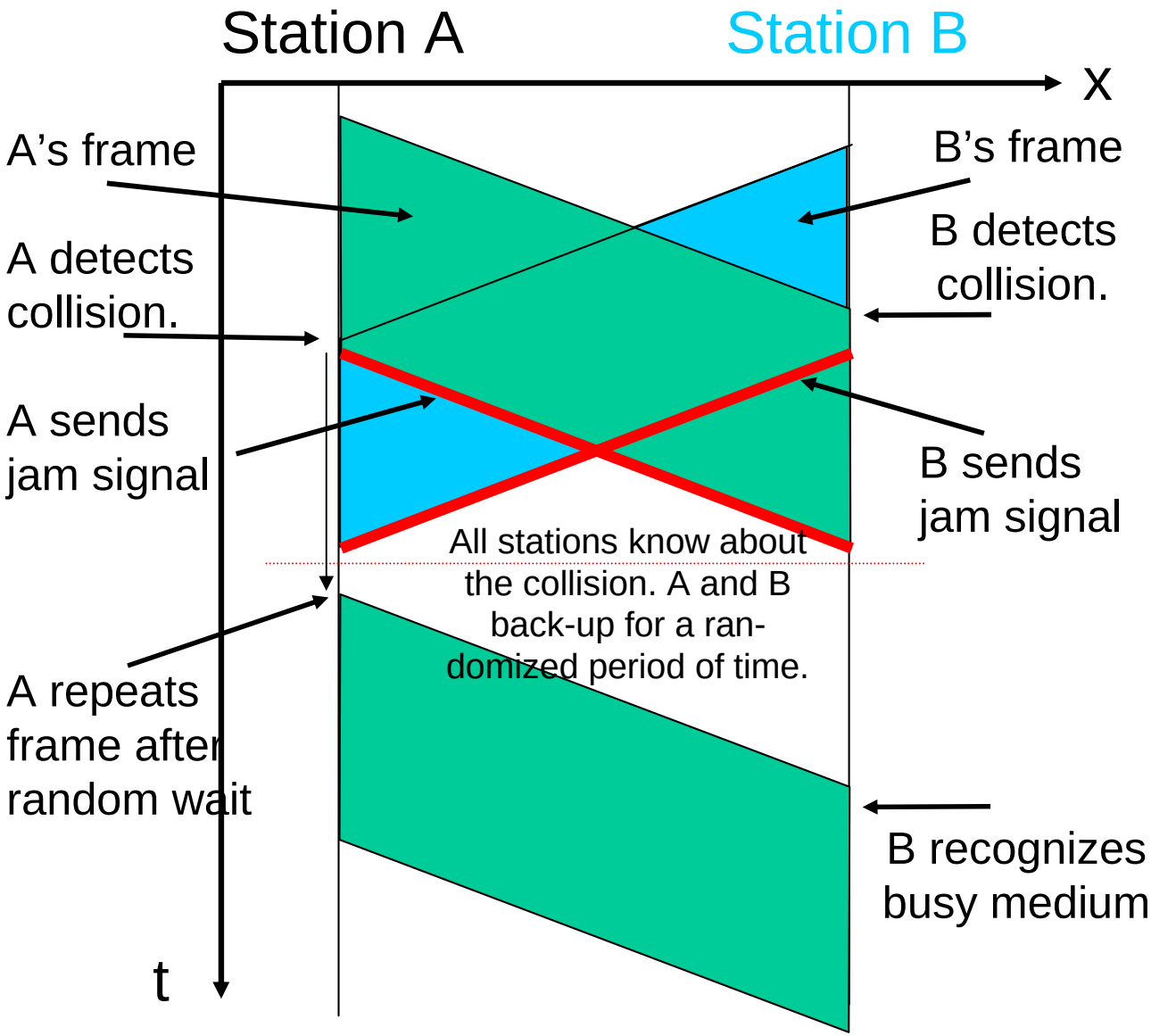
If transmission unsuccessful:

→ Collision

Repeat after random wait

CSMA with Collision Detection (CSMA/CD)

- Stations can detect collisions only reliably, if frame has minimum length
- Jamming signal makes sure that every station knows that the channel is in a collision state
- Random wait time increases exponentially with repeated collisions
  → *Binary Exponential Backoff*



© 2017 Burkhard Stiller

CSMA/CD Algorithm (1)

- If line is idle:
  - Send immediately
  - Upper bound message size of 1500 byte
  - Must wait 51.2 µs between back-to-back frames
    - Numbers: Ethernet
- If line is busy:
  - Wait until idle and transmit immediately
  - Called 1-persistent (special case of p-persistent)

```
/* send one frame */
t=1
Repeat
    wait_channel_available()
    r=monitor_while_sending(data)
    if r = collision then
        wait_random_time(t)
        t=2*t
    end
until r=success
```

© 2017 Burkhard Stiller

CSMA/CD Algorithm (2)

- Collision handling:
  - Jam for 512 bits, then stop transmitting frame
  - Minimum frame is of 64 byte length:
    - Header plus 46 byte of data

- Delay & try again in slot times (each of which 51.2 μs):
  - 1st time: uniformly distributed between 0 and 1 slots
  - 2nd time: uniformly distributed between 0 and 3 slots
  - 3rd time: uniformly distributed between 0 and 7 slots
  - Give up after several tries (usually maximum of 16)
  - Scheme termed “exponential back-off”

© 2017 Burkhard Stiller

Performance Tuning of CSMA/CD

- The maximum throughput of CSMA-based systems is roughly indirectly proportional to β:

  $\beta = \tau / m = (\tau * C) / L$

  τ: Propagation delay [s]

  m: Frame length [s]

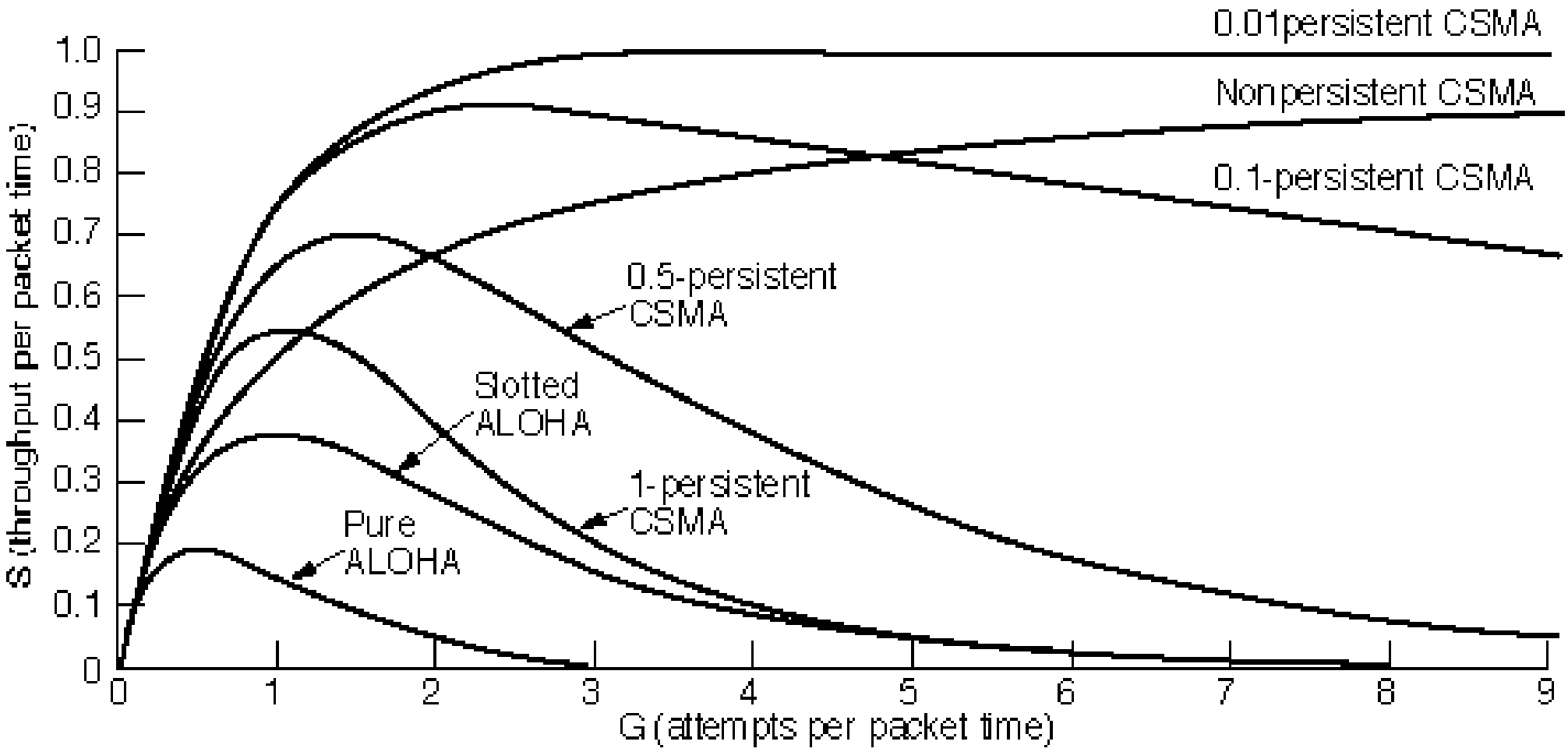
  L: Frame length [bit]

  C: Transmission rate [bit/s]

- For good performance, β should be <= 0.01.

© 2017 Burkhard Stiller

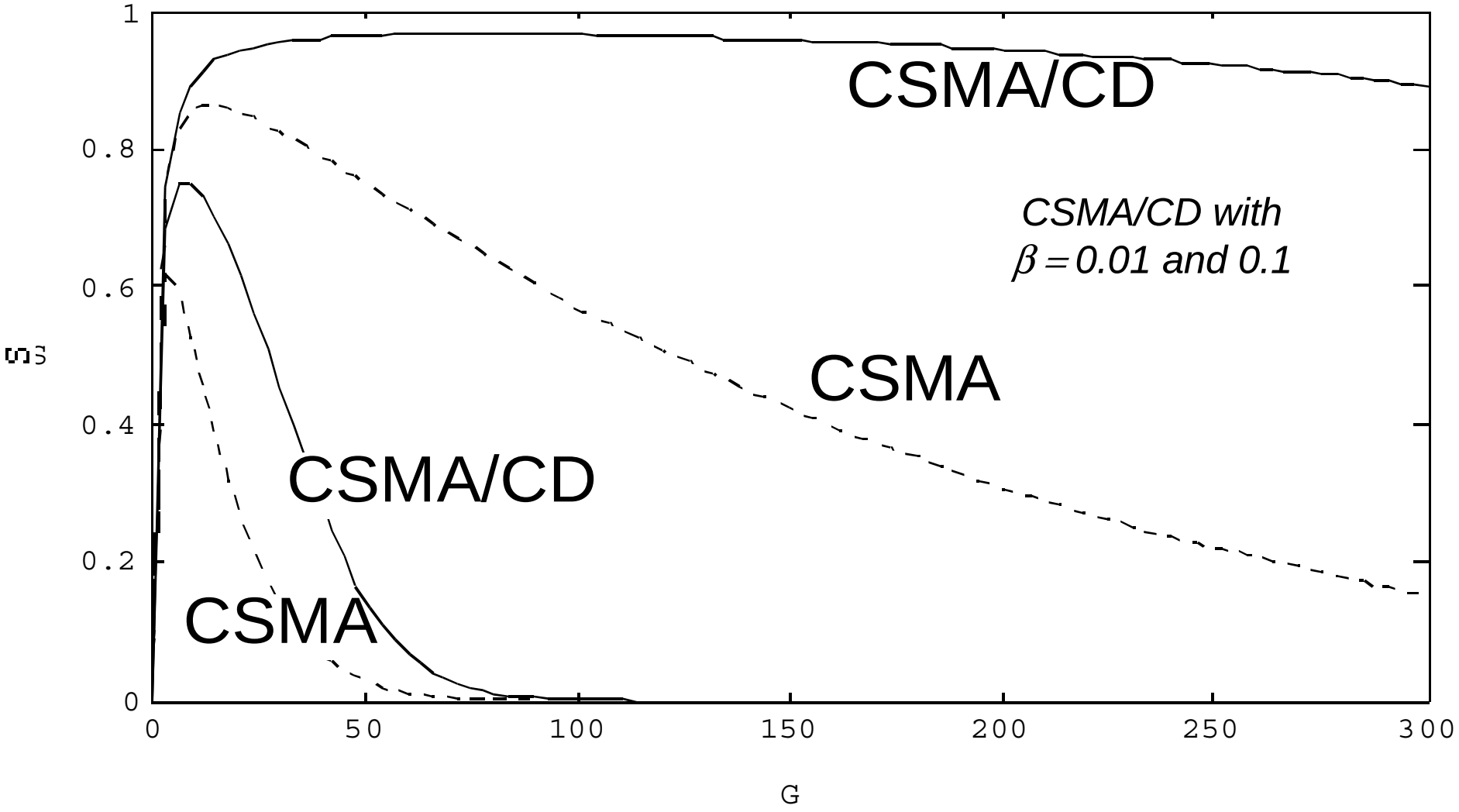
Achievable Throughput



© 2017 Burkhard Stiller

ifi

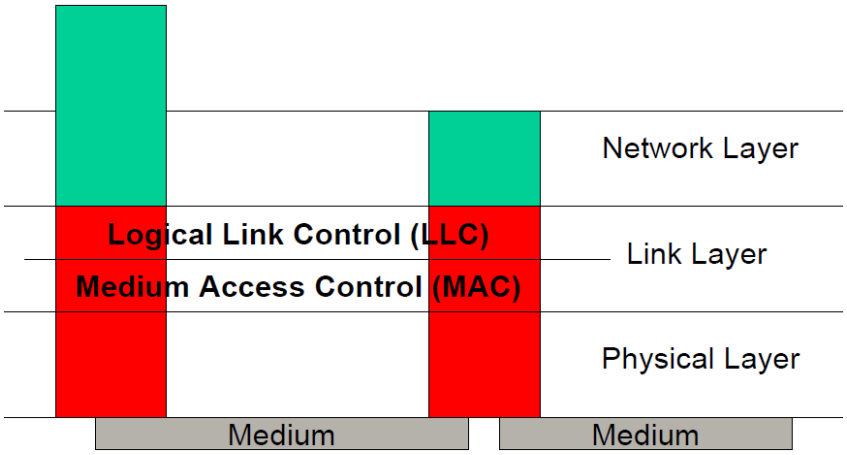
Throughput with Different β



© 2017 Burkhard Stiller

Ethernet History

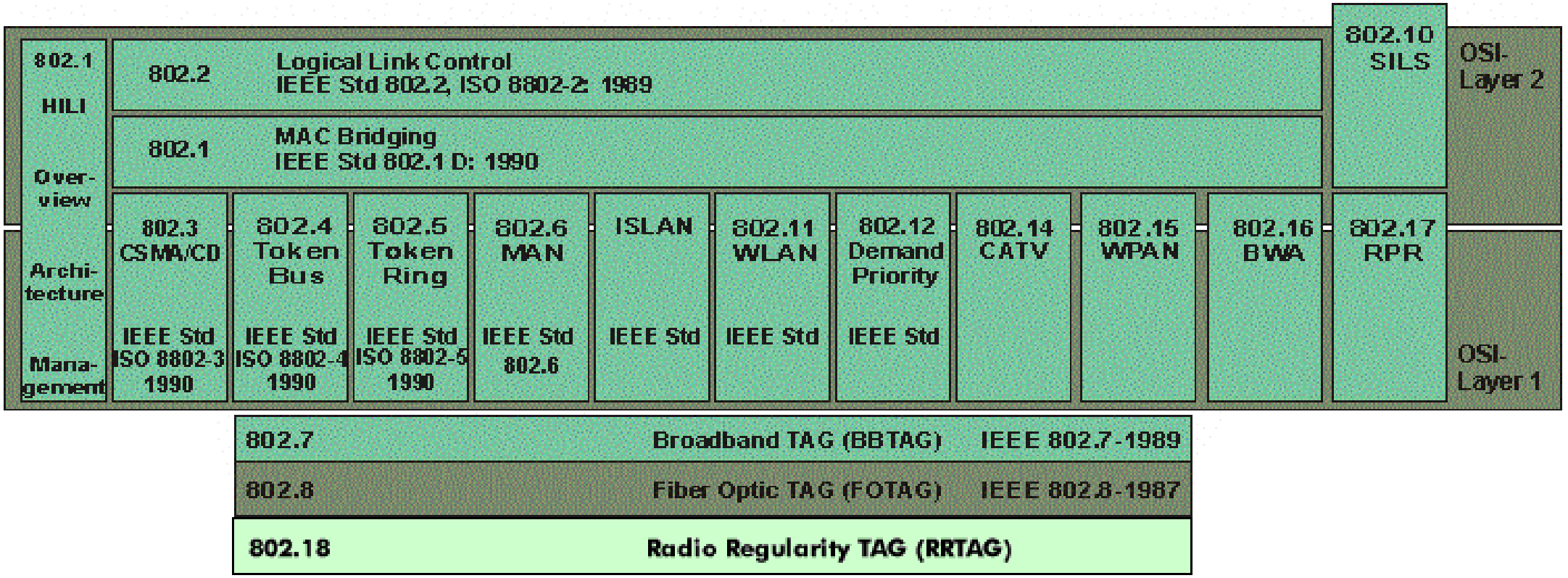
- Principles developed by Xerox Palo Alto Research Center (PARC)
  - Patent (http://www.delphion.com/details?pn=US0463220, page 1)
    - CSMA/CD medium access control algorithm
  - Industry standard set by DEC, Intel, Xerox (DIX), 1978
- Standardized by IEEE 802.3 and ISO 8802-3
- Variants:
  - 10, 100, 1000 Mbit/s
  - Thick/thin coax, STP/UTP, fiber
  - Segment lengths: 200, 500, 1000 m
  - Base-band or modulated carrier



© 2017 Burkhard Stiller

IEEE Standards Family 802

- IEEE defined for LANs access schemes dependent on the physical media and topology.



© 2017 Burkhard Stiller

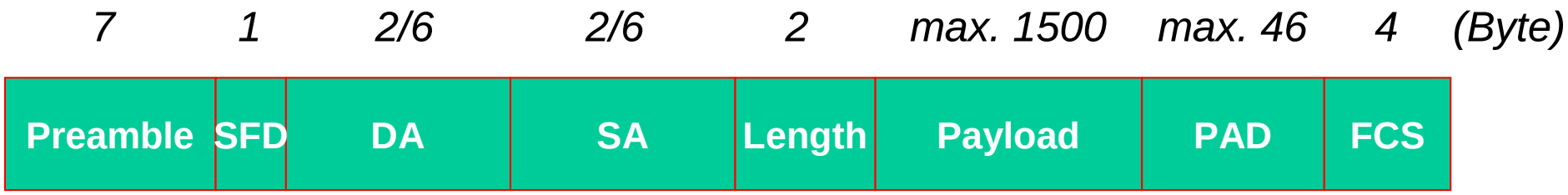
IEEE 802.3 Service Interfaces

- MAC sub-layer:
  - Connectionless, unreliable
  - Unicast, multicast, and broadcast addressing
  - FCS checked

- LLC sub-layer:
  - Type 1:
    Connection-less, unreliable, all addressing modes
  - Type 2:
    Connection-oriented, reliable, flow control
  - Type 3:
    Connection-less request, response type of service

© 2017 Burkhard Stiller

CSMA/CD Frame Format



Preamble: Bit synchronization
SFD: Byte synchronization (Start of Frame Delimiter)
DA: Destination address
SA: Source address
Length: Length of payload
Payload: Upper layer frame
PAD: To fill up a short frame (padding)
FCS: 32-Bit CRC for error detection (Frame Check Sequence)

© 2017 Burkhard Stiller

Ethernet MAC Addresses — Local Addresses

- Unique addresses:
  - 48-bit unicast address assigned to each physical adapter
    - Example: `8:0:2b:e4:b1:2`
  - Broadcast address:
    - All `1`s
  - Multicast address:
    - First bit is `1`
- Adapter receives all frames, accepts (passes to host):
    - Frames addressed to its own unicast address
    - Frames addressed to the broadcast address
    - Frames addressed to any multicast address it has been programmed to accept

© 2017 Burkhard Stiller

Current Ethernet Technology

- Shared medium:
  - Implementation as a hub with UTP (Unshielded Twisted Pair)
  - For higher bandwidths:
    - Point-to-point links
    - Half-duplex traffic
- Not more than 4 hubs between any station in network:
  - Network diameter max. 500 m
- Today:
  - Star-type topology with non-shared lines per node
- Available technology with bandwidth of:
  - 10 Mbit/s, 100 Mbit/s, 1000 Mbit/s, 10 Gbit/s

© 2017 Burkhard Stiller

Fast Ethernet — IEEE 802.3u

- 100 Mbit/s version of Ethernet
- Utilizing CSMA/CD algorithm, a later addition to IEEE 802.3:
  - 10 times faster than “normal” Ethernet
  - 10 times smaller (maximum of app. 200 m between stations)
  - Easy upgrade path from Ethernet:
    - Simply replace Ethernet hubs, adapters, and driver software!
  - Auto-sensing of physical media (10/100 Mbit/s)
- Works with several physical media

© 2017 Burkhard Stiller

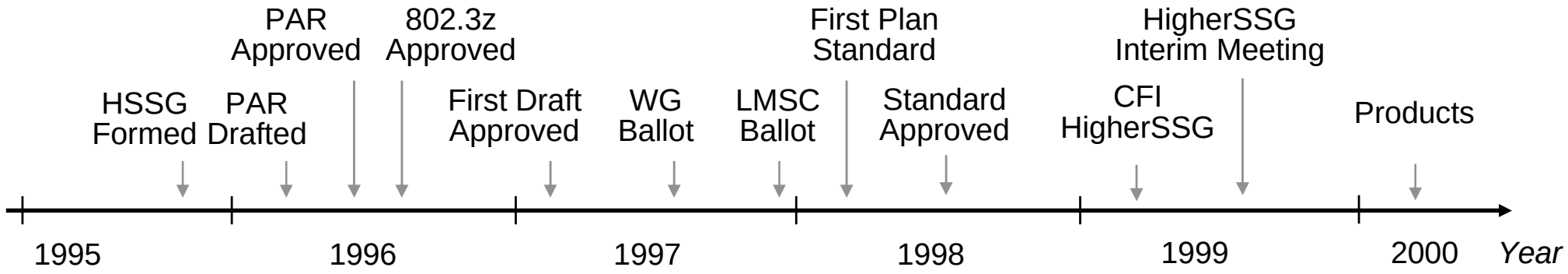
Gigabit Ethernet — IEEE 802.3z

- Marketing aspect:
  - Term Ethernet used to hint at easy/cheap upgrade, reliability.
- Theory is different:
  - CSMA/CD = Ethernet (historically)
  - If CSMA/CD is used on a shared medium, the allowable size of a Gigabit Ethernet segment will be rather small (roughly 20 m), shared Gigabit-Ethernet is not likely
  - If CSMA/CD is not used, is it still an Ethernet?
- Realistically, a Gigabit/s LAN need not be a CSMA/CD-based LAN to grant compatibility:
  - Important are cost, compatibility with existing cabling, and availability of good drivers for popular operating systems

© 2017 Burkhard Stiller

Gigabit Ethernet — Objectives/Standard

- IEEE 802.3z working group key objectives:
  - Half- and full-duplex operation at 1000 Mbit/s
  - Complying with IEEE 802.3 Ethernet frame format
  - Applying CSMA/CD access method
  - Allowing one repeater per physical collision domain
  - Providing address compatibility with Ethernet and Fast Ethernet technologies



CFI: Call for Interest, PAR: Project Authorization Request, WG: Working Group, HSSG: High-Speed Study Group

© 2017 Burkhard Stiller

ifi

Gigabit Ethernet — Physical Layer

- Symbols are used to code MAC data:
  - 8B/10B coding scheme (8 bit user data/10 bit physical data)
  - Code-inherent clock regeneration
  - Always min 4 and max 7 state changes per symbol
  - Baud rate: 1250 Mbaud and Bit rate: 1000 Mbit/s
  - Code group symbols (always different to data symbols):
    - Carrier Extension
    - Idle
    - Start-of-Packet
    - End-of-Packet
    - Configuration Marks
    - Violations

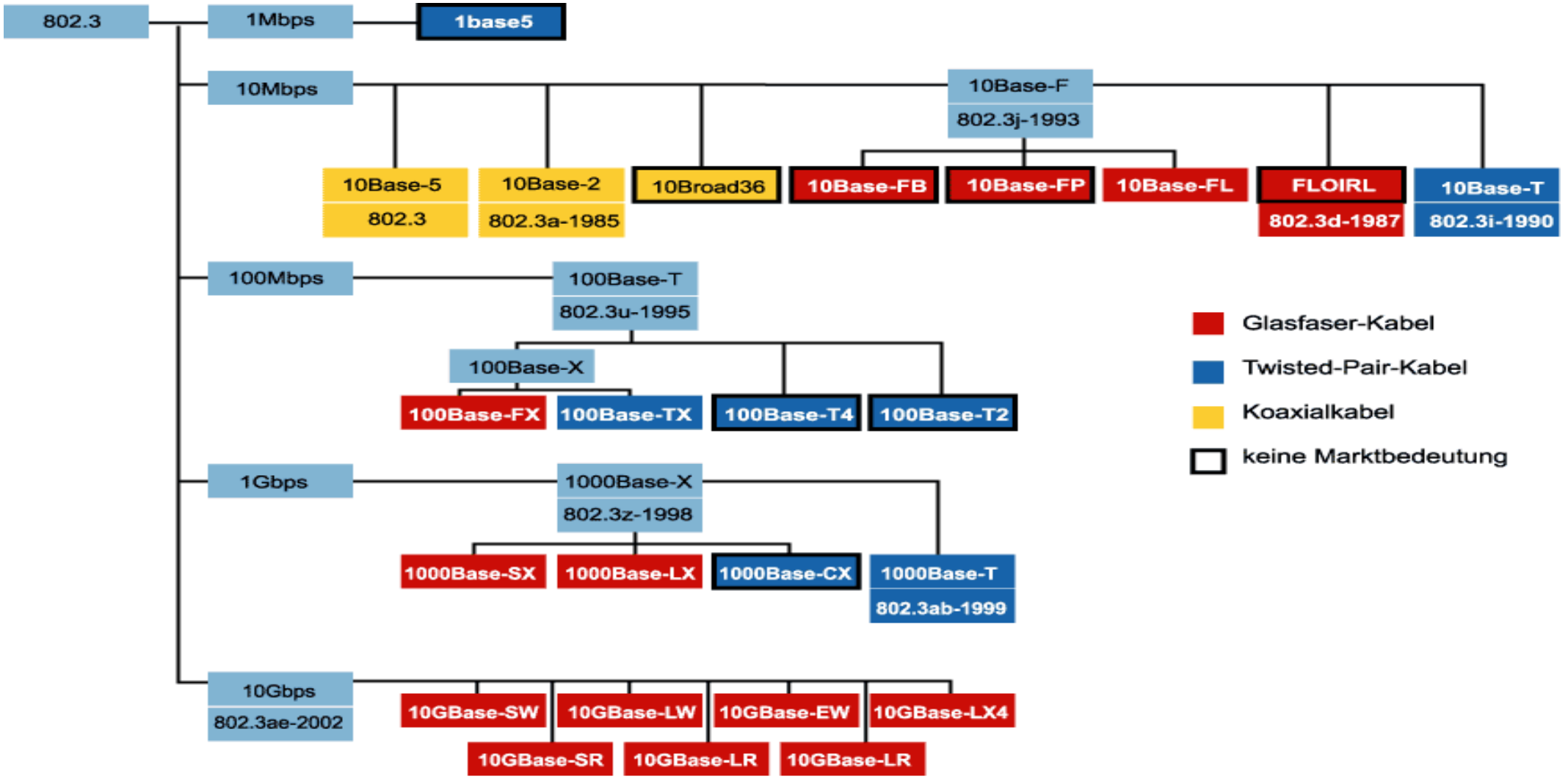
© 2017 Burkhard Stiller

10-Gigabit Ethernet — IEEE 802.3ae (1)

- Keeping 802.3- and Ethernet frame format (incl. minimal and maximal length)
- Keeping IEEE 802 functional requirements except for the Hamming distance
- Support of star topologies by point-to-point connections with a full duplex mode in accordance with IEEE 802.3x
- Full duplex shows that the CSMA/CD access scheme has been omitted — it‘s definitively not Ethernet anymore!
- Gigabit Ethernet used to be the sort-of last Ethernet technology

© 2017 Burkhard Stiller

10-Gigabit Ethernet — IEEE 802.3ae (2)



© tecChannel.de

© 2017 Burkhard Stiller

ifi

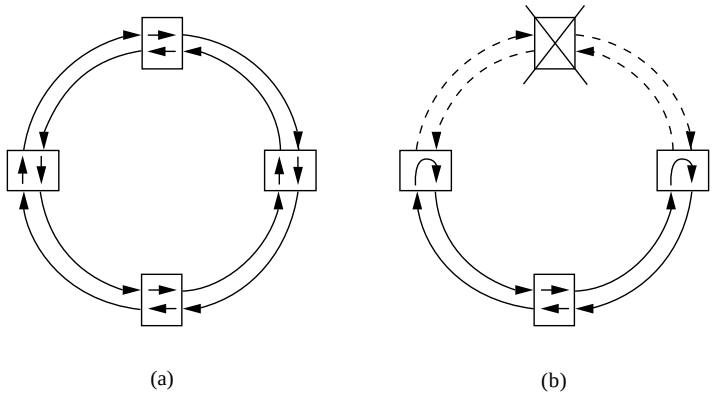
Token Ring Networks

- Sharing concept:
  - Frames flow in one direction: upstream to downstream
  - Special bit pattern (token) rotates around ring
  - Hosts must capture token before allowed to transmit
  - Hosts release token after done transmitting:
    - Immediate release
    - Delayed release
  - Remove host's frame when it comes back
  - Stations get a round-robin type of service
- Examples:
  - IBM: 4 Mbit/s Token Ring
  - 16 Mbit/s IEEE 802.5 (Token Ring)
  - 100 Mbit/s Fiber Distributed Data Interface (FDDI)

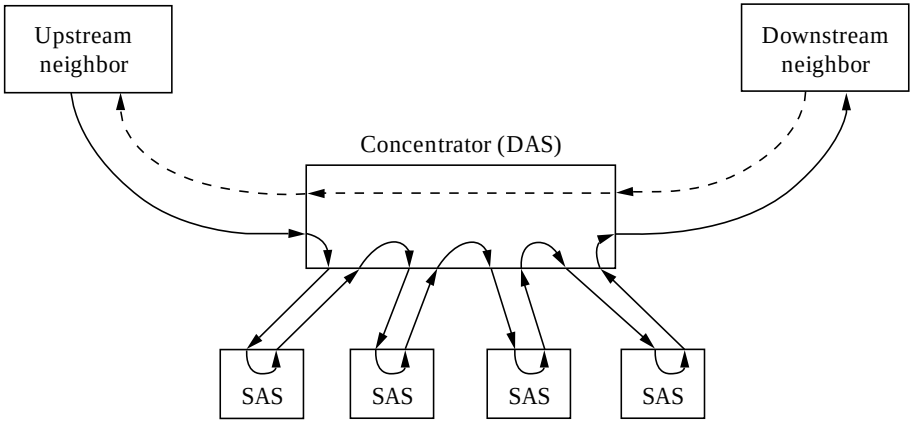
© 2017 Burkhard Stiller

FDDI

- Physical properties:
  - Dual rings
  - Single or dual attached hosts



- Each station imposes a delay (*e.g.*, 50 ns)
- Maximum of 500 stations
- Upper limit of 100 km
  - 200 km of fiber
- Uses 4B/5B encoding
- Can be implemented over copper (CDDI)



© 2017 Burkhard Stiller

FDDI Access Algorithm

- Timed Token principle:
  - Token Holding Time (THT) determines upper limit on how long a station can hold the token
  - Token Rotation Time (TRT) determines how long it takes the token to traverse the ring:

    TRT <= Active_Nodes * THT + Ring_Latency
  - Target Token Rotation Time (TTRT) determines the agreed-upon upper bound on TRT:
    - Each node measures TRT between successive arrivals of the token
    - If measured TRT > TTRT, then token is late so don't send data
    - If measured TRT < TTRT, then token is early so OK to send data
  - Worst case: 2x TTRT between seeing the token

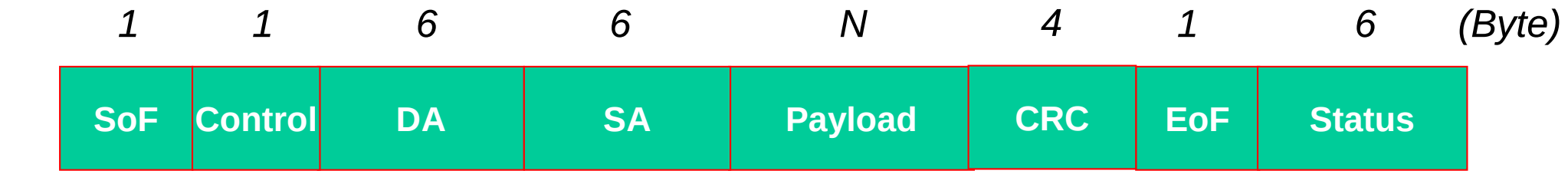
© 2017 Burkhard Stiller

Token Management

- Cases of lost tokens:
  - No token when initializing ring
  - Bit error corrupts token pattern
  - Node holding token crashes
- Generating a new token (and agreeing on TTRT):
  - Execute when joining the ring or suspecting a failure
  - Each node sends a special claim frame that includes the node's bid for the TTRT
  - Upon reception of a claim frame, update bid and forward
  - If the claim frame makes it all the way around the ring:
    - This bid was the lowest
    - Everyone knows TTRT
    - This host inserts new token

© 2017 Burkhard Stiller

FDDI Frame Format



- Control Field:
  - 1st bit: asynchronous (0) versus synchronous (1) data
  - 2nd bit: 16-bit (0) versus 48-bit (1) addresses
  - Last 6 bits: de-multiplexing key:
    - Includes reserved patterns for token and claim frame
- Status Field:
  - From receiver back to sender
  - Error in frame
  - Address recognized
  - Frame accepted (flow control)

© 2017 Burkhard Stiller

10. Packet Switching and Internetworking

Telecommunication Networks

Connecting LANs

Routers and IP

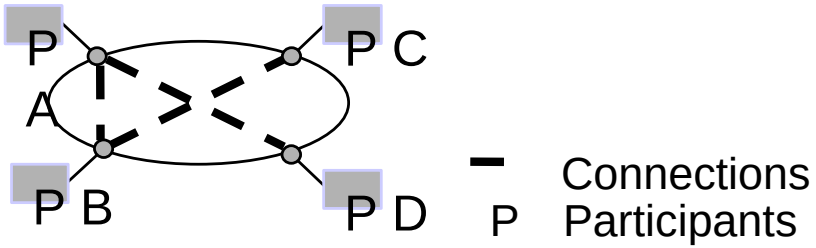
Routing

Inter-domain Aspects

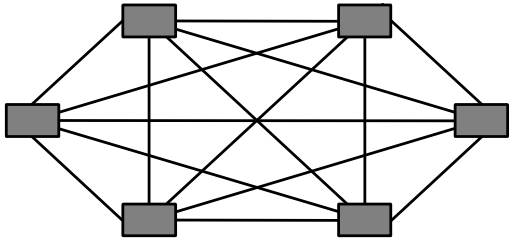
© 2017 Burkhard Stiller and Daniel Bauer

Telecommunications Network (TN)

- Participants use temporary paths to other participants
  - "Connection" used differently in TN compared to ISO/OSI
  - Context defines all details
    - Layer 3, MAC, and physical line



- Tasks to be performed
  - Establish a connection via routing/switching services
  - Network-internal supervision of connections via monitors
- A permanently connected path between any two participants leads to too many paths to be maintained:
  - Non-scalable
  - Fully meshed N participants would require a network of N * (N-1) / 2 paths



© 2017 Burkhard Stiller and Daniel Bauer

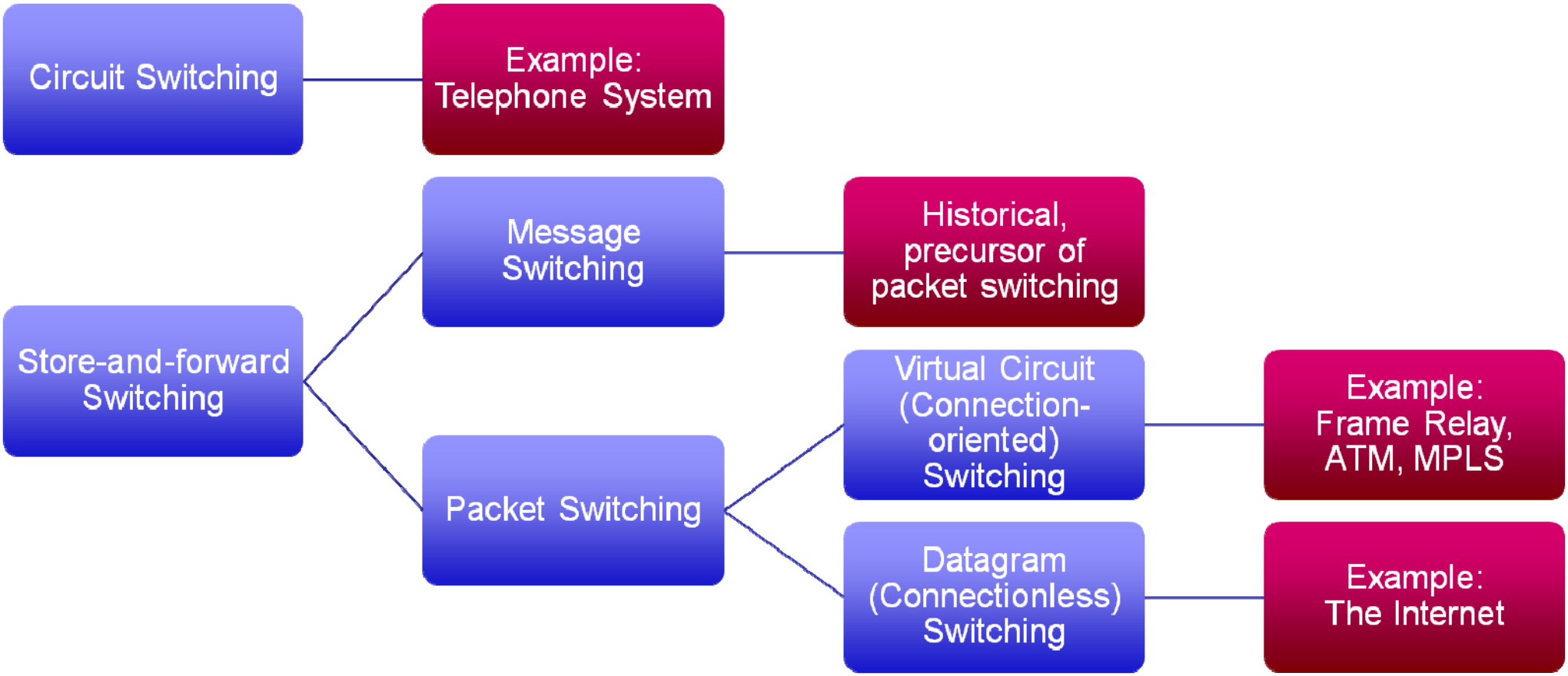
Internetworking – Switching and Signaling

- Switching (Vermittlung):
  - On-demand provisioning of a temporary path across network(s)
  - Operation performed by interconnecting switching centers
  - Switching techniques: connection-oriented/connectionless

- Signaling (Signalisierung):
  - Terminology defining the communication taking place to perform the switching between
    - Data Terminal Equipment (DTE) and network
    - Network elements internal to the network

© 2017 Burkhard Stiller and Daniel Bauer

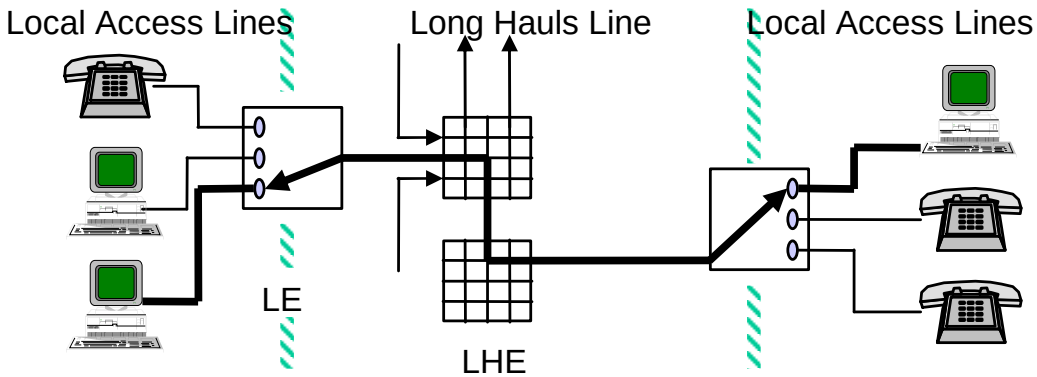
Transmission Methodologies



© 2017 Burkhard Stiller and Daniel Bauer

Circuit Switching

- Circuit Switching or Line Switching (Durchschaltvermittlung or Leitungsvermittlung)
  - Dominating switching technology for public telephone network
- Set-up of end-to-end, non-storing line/circuit between Ps
  - Transmission delay in circuit determined by physical transmission delays
- All bits are sent sequentially
  - Keeping sending sequence unchanged
    - Wire-like features
  - Classical technology (Heb-Dreh-Wähler)
  - Full digital switching



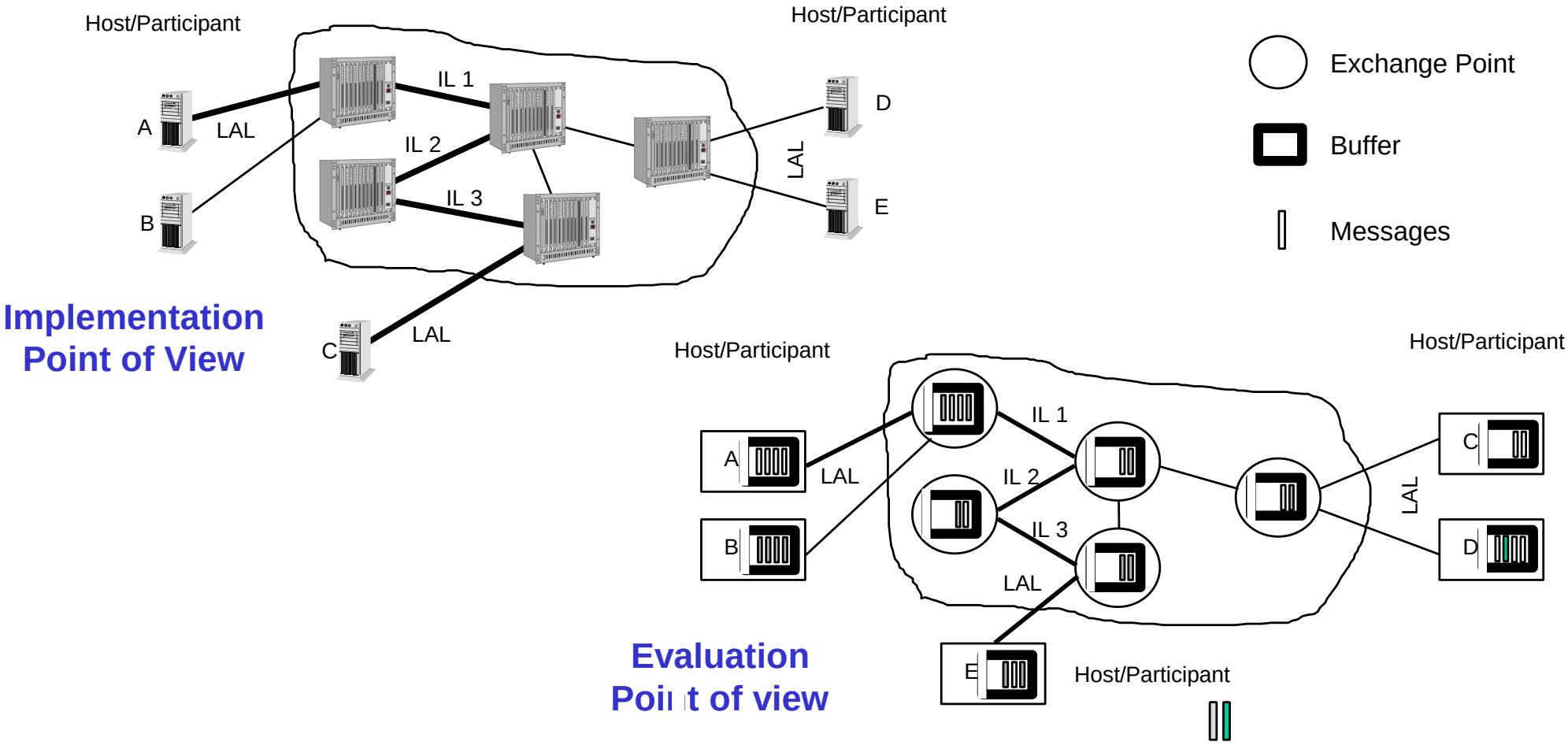
© 2017 Burkhard Stiller and Daniel Bauer

Store-and-forward Switching (1)

- Store-and-forward Switching (Speichervermittlung or Teilstreckenvermittlung):
  - Each and every network node contains memory
  - No busy signals, only waiting
  - Intermediately stored packets may wait for next hop/line
  - Loss of packet may happen due to memory overflow
  - Queued packets may be prioritized and sequentially changed
  - No timeliness between any two packets
  - Adaptation possible between differently performing devices

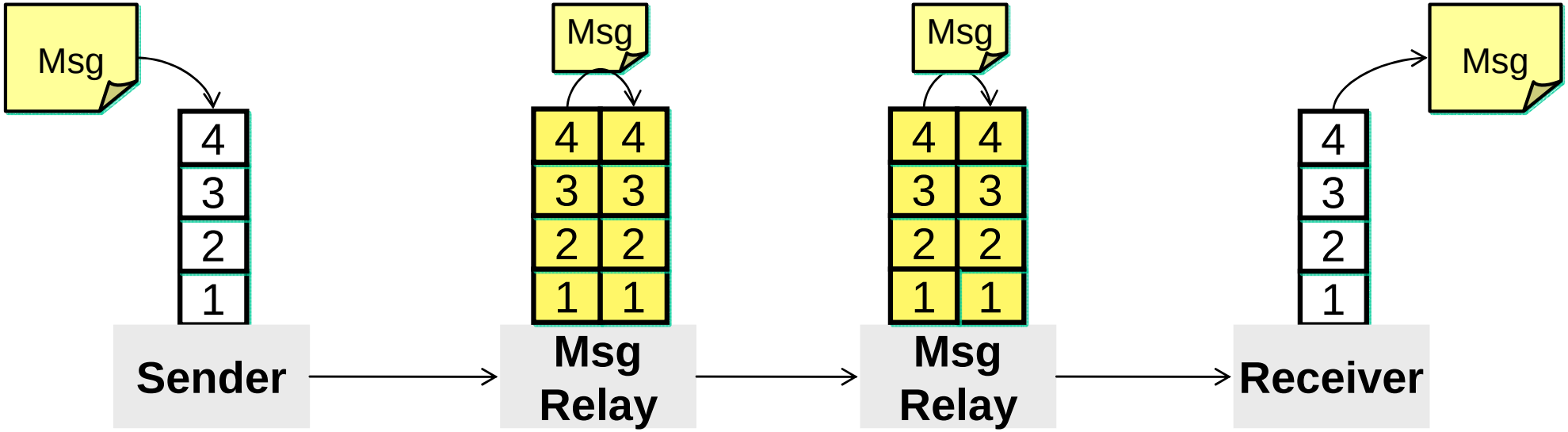
© 2017 Burkhard Stiller and Daniel Bauer

Store-and-forward Switching (2)



© 2017 Burkhard Stiller and Daniel Bauer

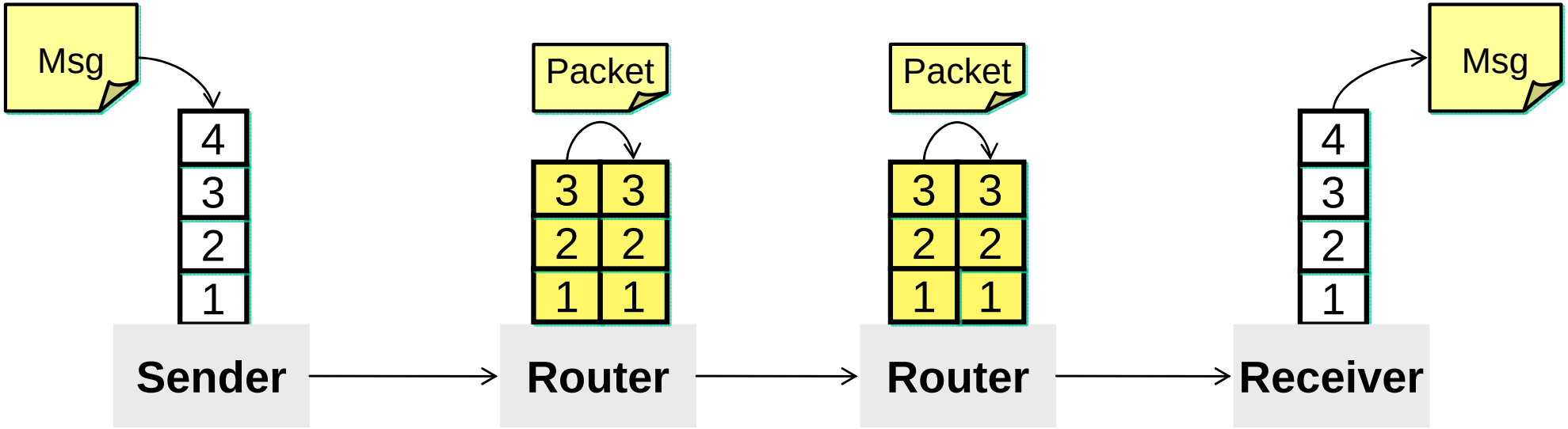
Message Switching (“Historical”)



- Message switching implemented on top of packet switching or circuit switching networks.
- Message are received completely before forwarded
- Message contents:
  - Header including source, destination and length information
  - Message text
- Examples: E-Mail, “Usenet” (UUCP)

© 2017 Burkhard Stiller and Daniel Bauer

Packet Switching

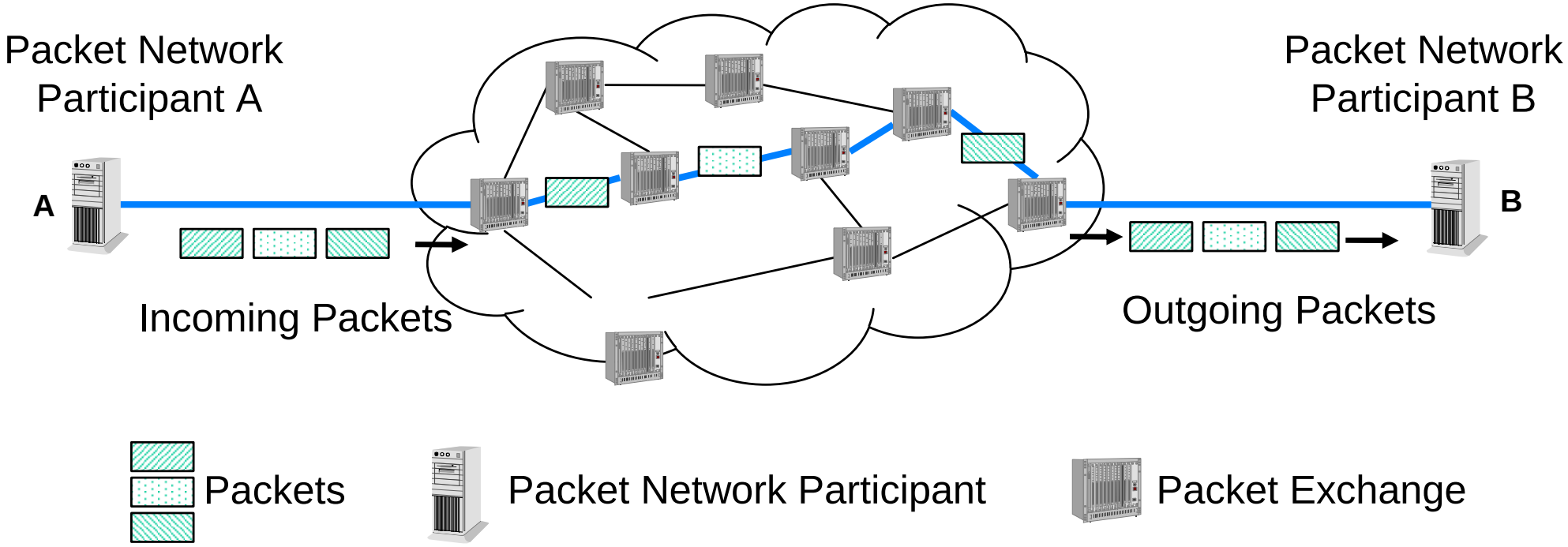


- Switching based on packets of limited but variable length.
- Packets are received completely before forwarded
- Packet contents:
  - Header including switching and length information
  - Switching information: Either source/destination address or virtual circuit ID
  - Packet content
- Example: The Internet Protocol's (IP) Protocol Data Units

© 2017 Burkhard Stiller and Daniel Bauer

Virtual Circuits

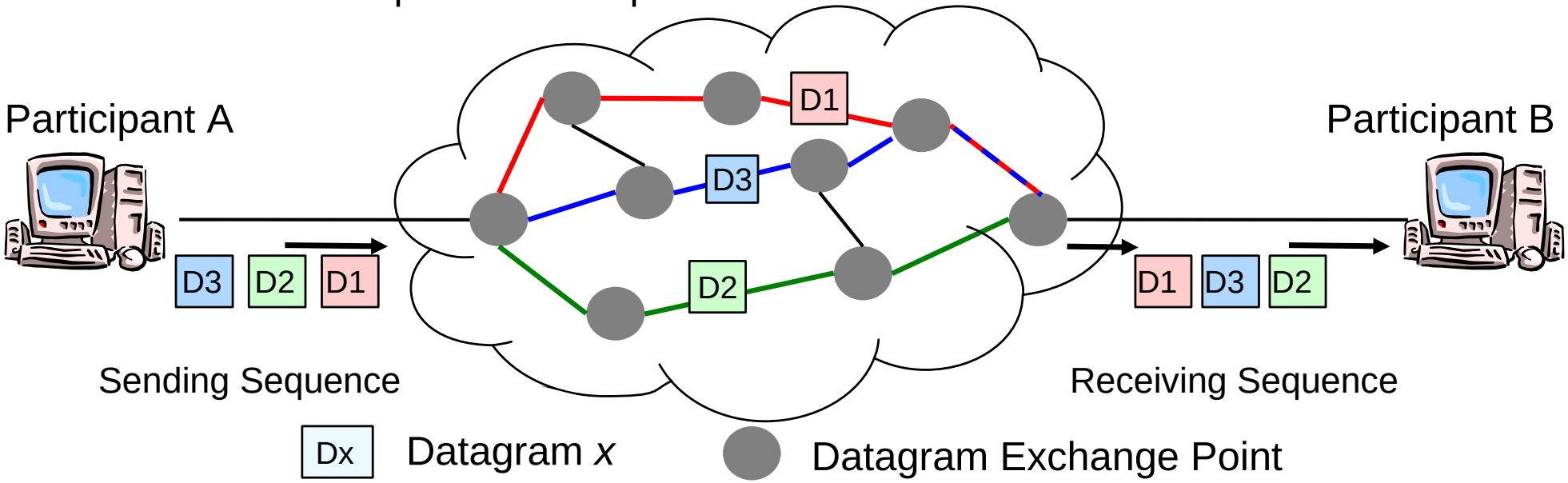
- Connection set-up of a virtual circuit (VC)
  - Virtual circuit is set up through a signaling protocol
  - All packets follow same path; packets contain *virtual circuit ID*
  - Path context available in all intermediate nodes



© 2017 Burkhard Stiller and Daniel Bauer

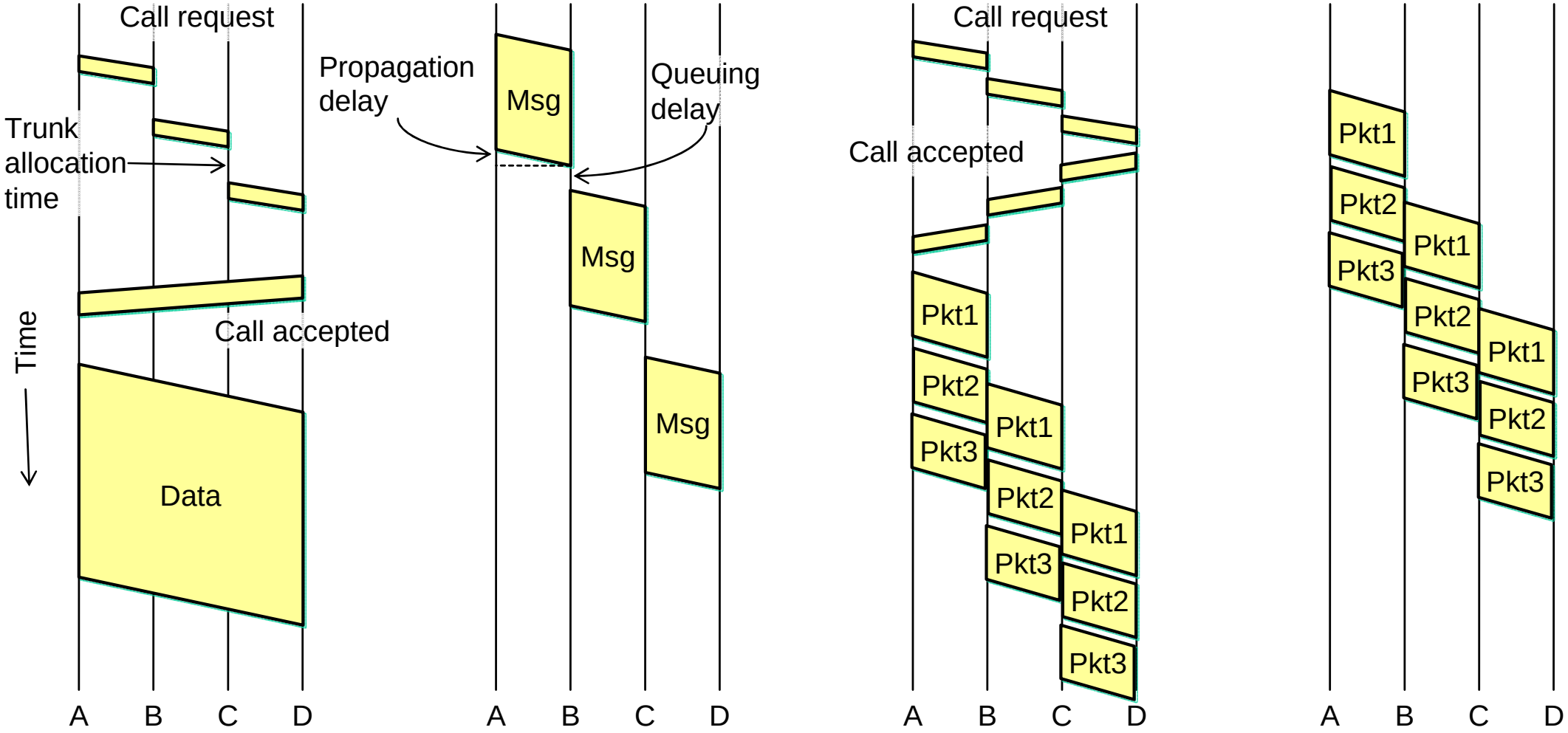
Datagram Switching (Internet)

- Datagram traffic in packet switching network:
  - Every datagram independent of others and uses ”his“ path
  - No set-up procedure required for any datagram
    - Typically, no reservation of resource in networks nodes
    - Datagram implies a best effort (unreliable!) service
    - Meshed networks may lead to datagrams delivered in arbitrary order due to multiple available paths



© 2017 Burkhard Stiller and Daniel Bauer

Comparison (1)



© 2017 Burkhard Stiller and Daniel Bauer

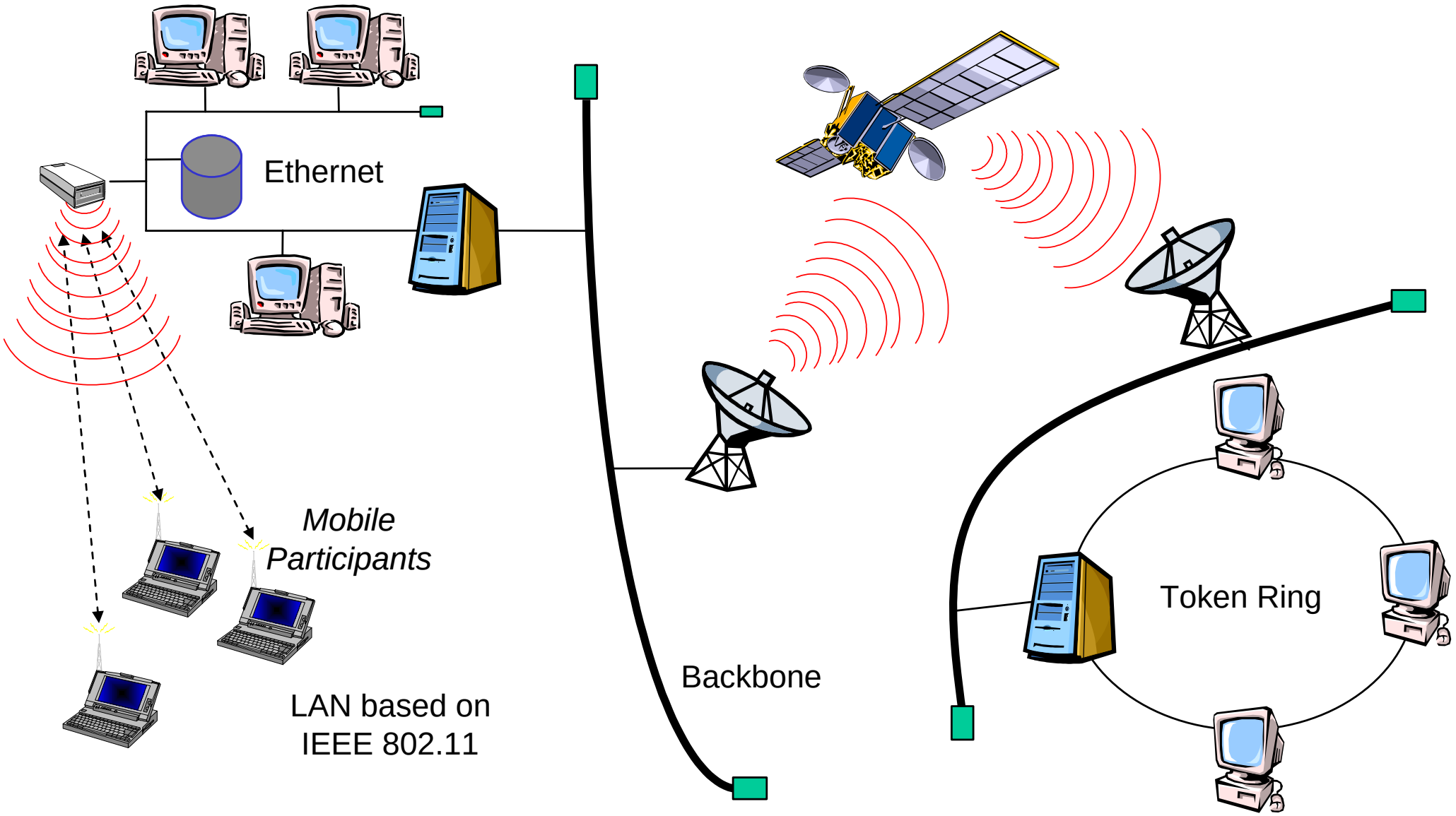
Comparison (2)

Functionality	Circuit-switched	Packet-switched	
		Virtual Circuit (VC)	Datagram
Call/Circuit setup	Required	Required	Not needed
Addressing	Addresses needed for setup	Addresses needed for setup, packets contain virtual circuit IDs	Datagrams contain source and destination addresses
Routing	During setup	During setup	Each Datagram routed individually
Data transmitted over the same route	Yes	Yes	No
Crash resiliency	Circuits of crashed switch are terminated	Virtual circuits of crashed router are terminated	Little effect due to re-routing
Store-and-forward	No	Yes	Yes
Efficiency of bandwidth usage	Low, circuits block bandwidth independent of usage	Medium to high, circuits share the same resources unless reserved	High, datagrams share the same resources
Congestion	Only during setup	On every packet – unless resources are reserved	On every packet
Guaranteed service quality	Yes	Possible when reserving resources during setup	Difficult, usually best effort

© 2017 Burkhard Stiller and Daniel Bauer

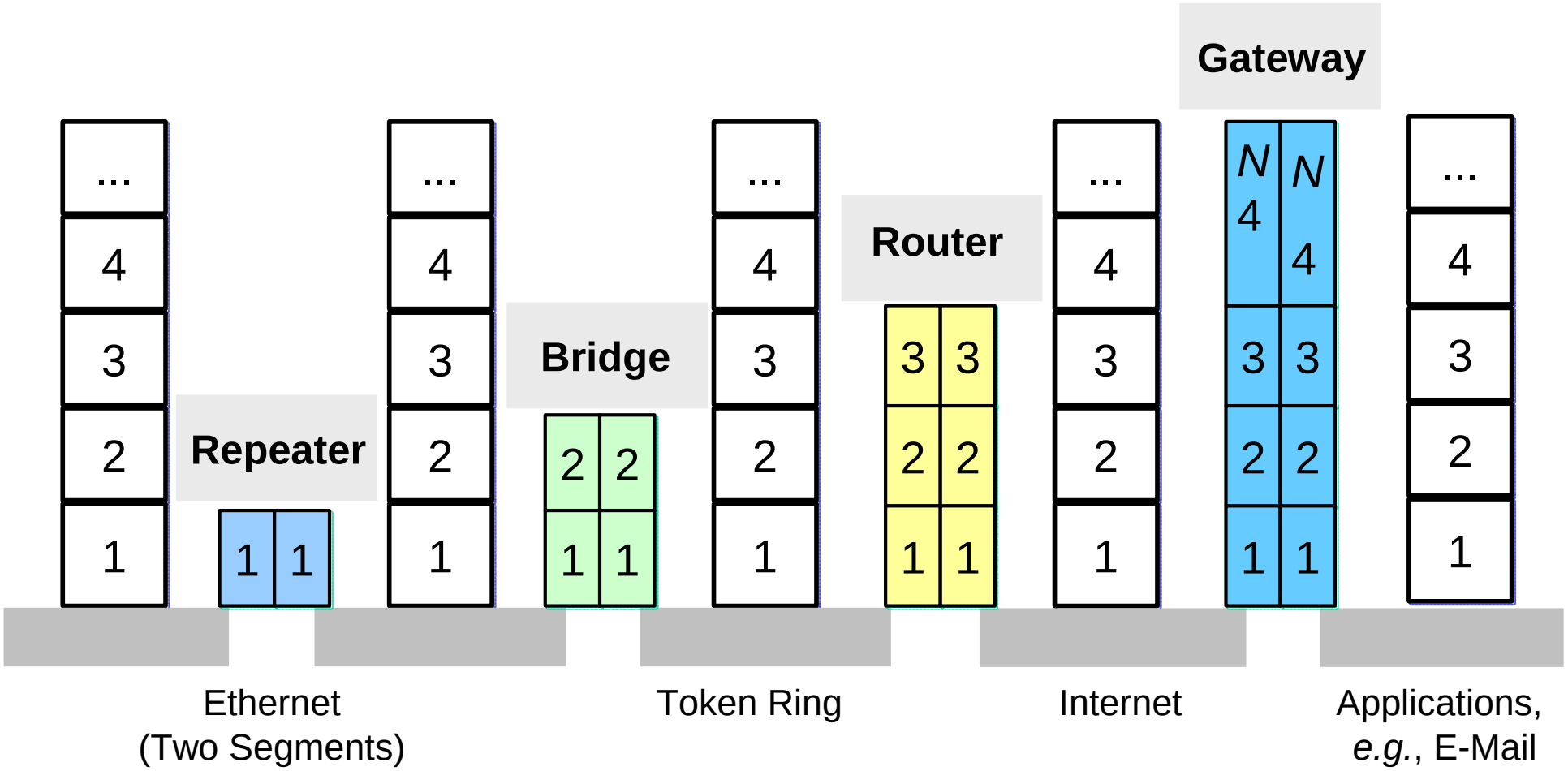
Connecting …

Internetworking – Internet (1)



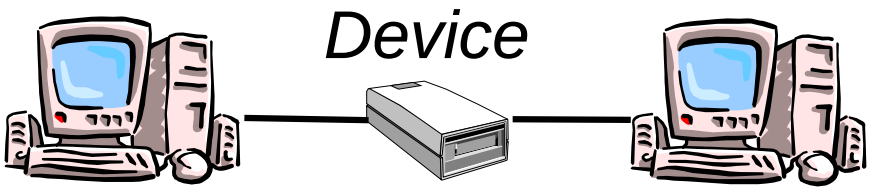
© 2017 Burkhard Stiller and Daniel Bauer

Internetworking – Internet (2)

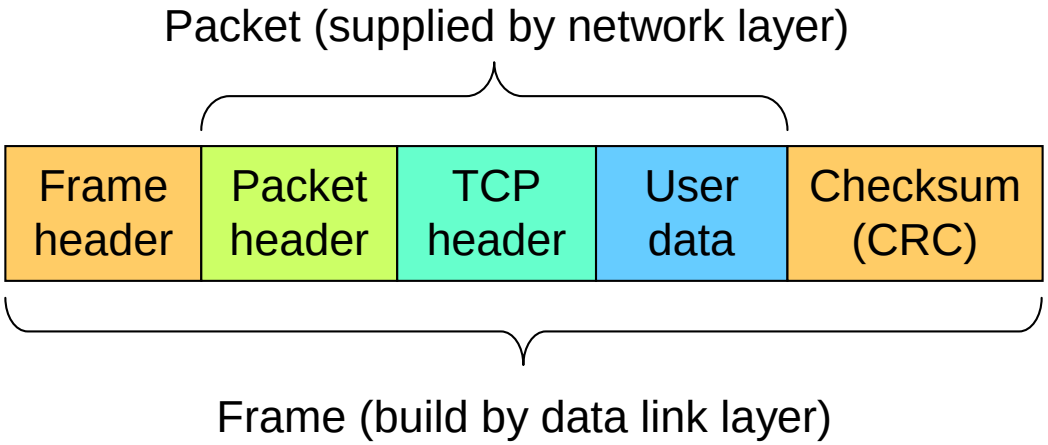


Devices and Messages – Overview

- Devices and layers



Layer	Device
Application	Application gateway, e.g., mail forwarder, firewall
Transport	Transport gateway, firewall
Network	Router
Data link	Bridge, switch
Physical	Repeater, hub

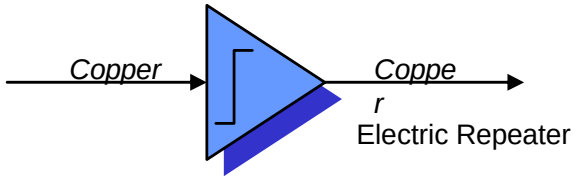


- Messages in terms of frames, packets, headers, trailers

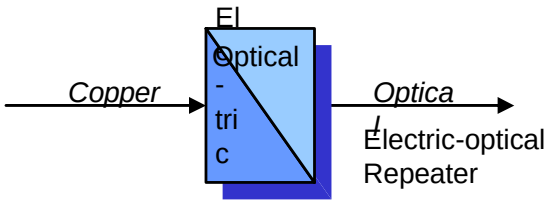
© 2017 Burkhard Stiller and Daniel Bauer

Repeater

- Coupling of physical media by signal regeneration/ amplification



  - Media may be different, e.g., twisted pair and fiber optic



  - Layer 2 protocol has to be identical
- No intermediate storage
- No handling of protocol data units or frames

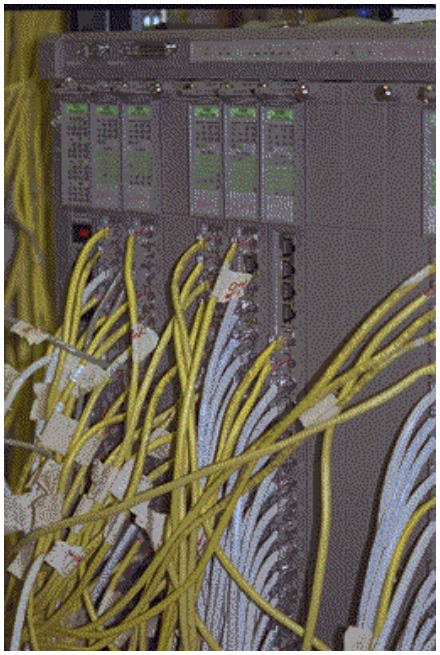


- Major characteristics
  - Extensions of the physical dimension of LANs
  - Change of physical transmission medium, *e.g.*, copper to fiber
- Advantages
  - Simple and cheap
  - No data processing
  - Regeneration of signal only
  - Extremely long haul connections possible (transatlantic cabling)
- Drawbacks
  - No filtering
    - All signals are forwarded
  - No enhancement of network capacity

© 2017 Burkhard Stiller and Daniel Bauer

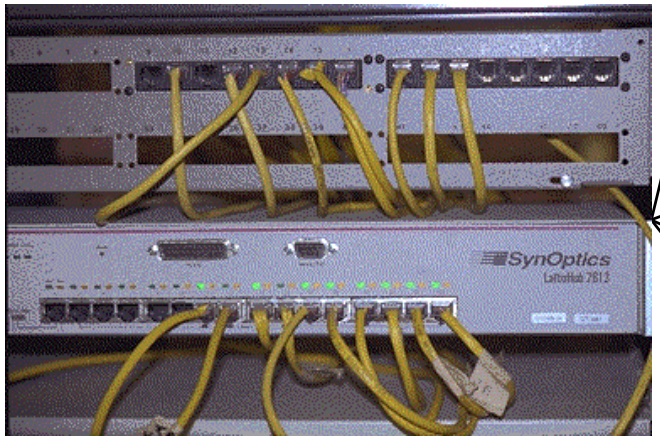
Hub

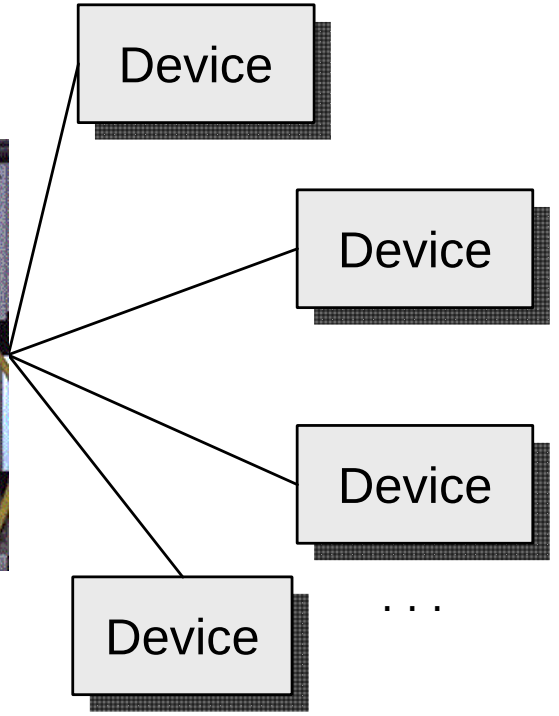
- Hub = Repeater with multiple ports:
  - Typical star topology with separate line from device to device (cascading)
  - Aggregate (concentrate) multiple connections
  - Total throughput does not increase (compared to a switch)



Hub

Hub





© 2017 Burkhard Stiller and Daniel Bauer

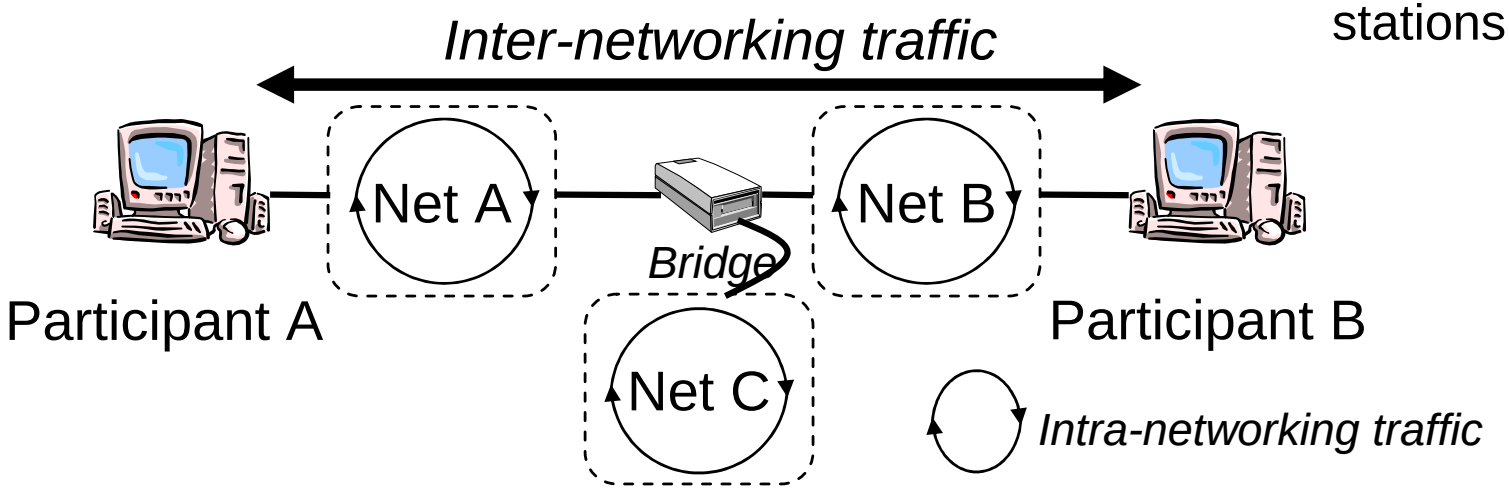
MAC Addresses

- Medium Access Control (MAC) addresses (“Hardware Address”, “Physical Address”)
  - Any port of a communicating device is equipped with a unique identifier assigned to all network interfaces
  - Valid within this network only
    - Examples: within an Ethernet 802.3 LAN or a Wireless 802.11 LAN
  - Often assigned by the manufacturer
- Following Institute of Electrical and Electronics Engineers (IEEE) number spaces
  - 48 bit (6 Byte) address space
  - 281,474,976,710,656 possible MAC addresses
  - Example: 00-21-CC-B6-62-7A (in hexadecimal representation)

© 2017 Burkhard Stiller and Daniel Bauer

Bridges – Operations

- Layer 2 Coupling of networks:
  - Requires MAC address per port
  - Networks may be of similar type:
    - *E.g.*, IEEE 802.x coupled with IEEE 802.x: *non-translating*
  - Networks of different type:
    - *E.g.*, IEEE 802.x coupled with IEEE 802.y (x≠y): *translating*
- Major tasks:
  - Accept and forward frames
    - Learning mechanism
  - No additional frame headers
- Advantages:
  - Enhancement of network capacity by partitioning:
    - Each separate partition may operate with full number of stations, length specifications



Naming synonyms: Bridges, layer 2 switches, LAN switches

MAC: Medium Access Control

© 2017 Burkhard Stiller and Daniel Bauer

Bridges — Types

Historic:
LAN-LAN coupling
LAN-WAN-LAN coupling



Local/Remote

2-Port/Multiport

Bridge

Translating/
Non-translating

Bridge interconnects
two or multiple networks

Coupling of LANs of
different/same MAC type

Transparent/Source Routing



Bridge invisible/visible to
end-systems

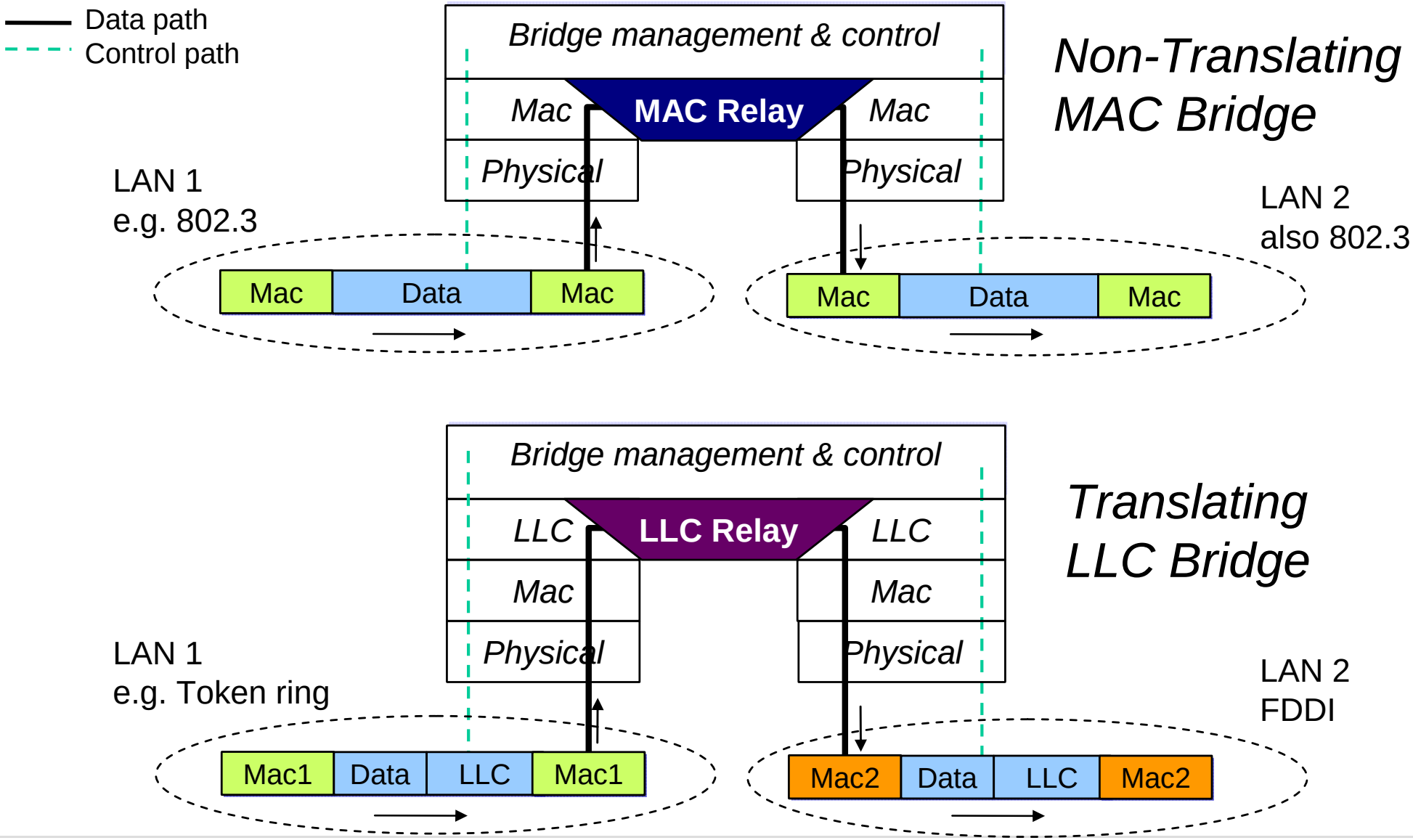
Switch

Transparent, Multiport Bridge

© 2017 Burkhard Stiller and Daniel Bauer

ifi

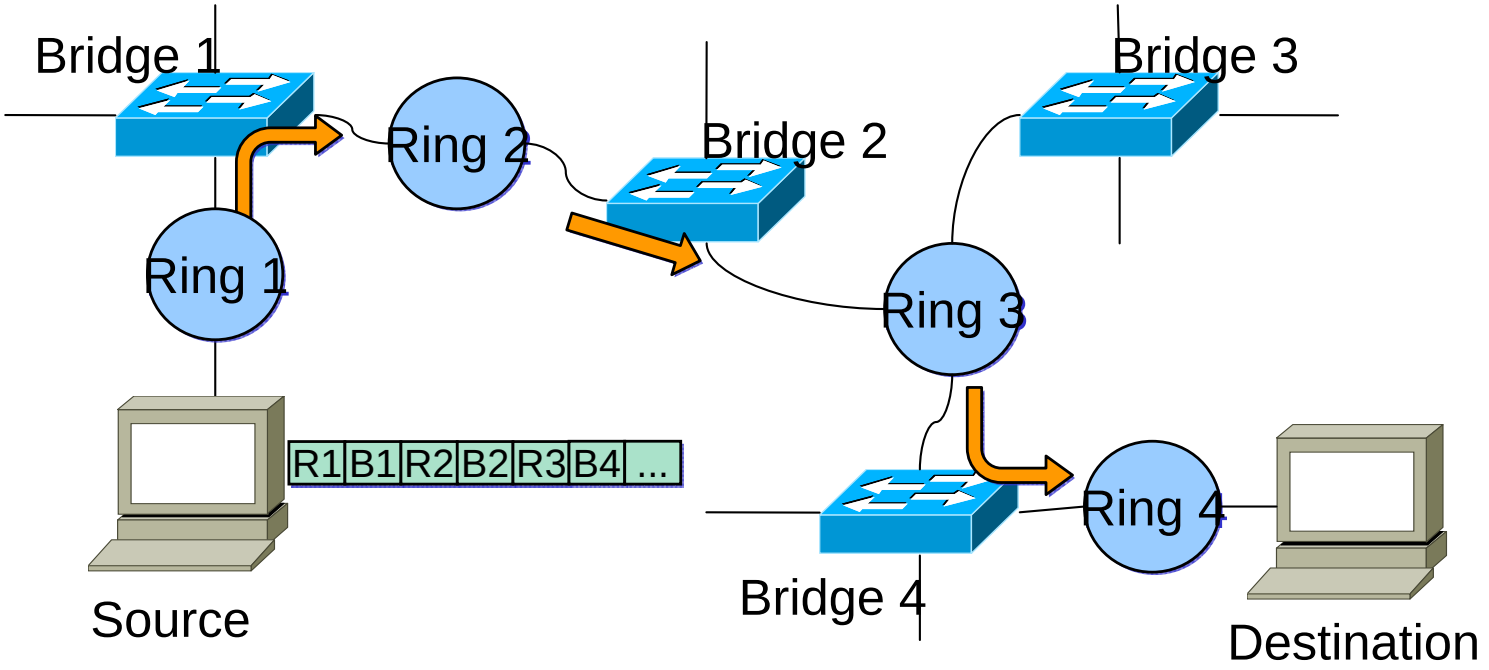
Non-translating vs. Translating Bridges



© 2017 Burkhard Stiller and Daniel Bauer

Source Routing Bridge

- Source routing bridges (IEEE 802.5):
  - Hosts store path information in the frames
  - Bridges forward frames based on path information
    - Frame contains sequence of ring- and bridge IDs on path from source to destination
  - Simple imple-menta-tion, but no practical use

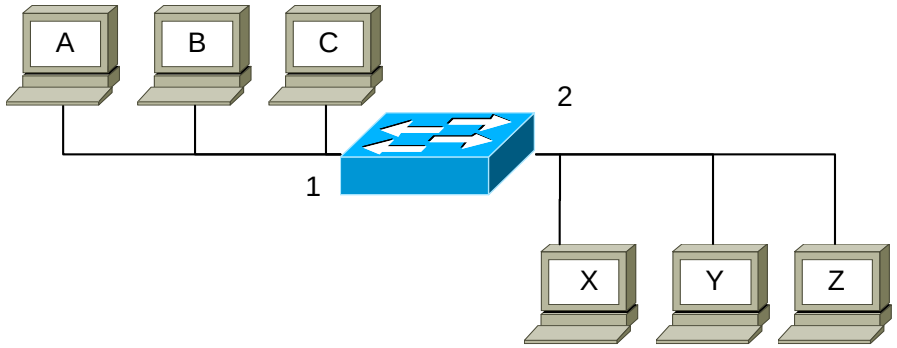


© 2017 Burkhard Stiller and Daniel Bauer

Transparent Bridge – Learning

- Transparent bridges (IEEE 802.1D)
  - Hosts do not need to know bridges
  - Forwarding decision taken in bridge (self-decisive)
    - Do not forward when unnecessary
    - Always forward broadcast frames
  - Bridge maintains a forwarding table (filter data base)
    - Learn location of hosts by examining frames
      - Learn table entries based on source address
    - Table is an optimization, does not need to be complete
  - Bridges are more complex

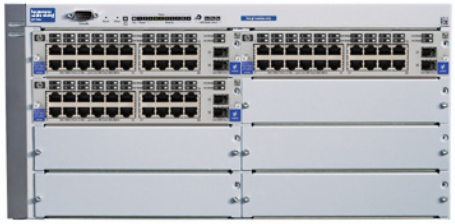
Host	Port
A	1
B	1
C	1
X	2
Y	2
Z	2



© 2017 Burkhard Stiller and Daniel Bauer

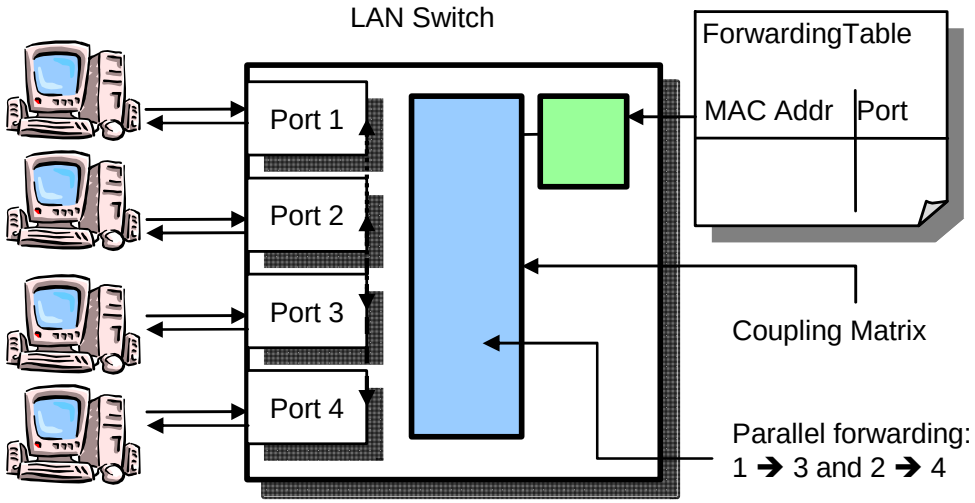
Switches

- *Switch* (“Schalter“):
  - Switch = powerful transparent bridge with high port density
  - Device or component in charge of frame forwarding





HP ProCurve Switches



- Switched LAN
  - Topology: Star, switch in center
  - Every host attached obtains full link capacity
    - Send/receive separated: full duplex (otherwise: half-duplex)
    - Parallel forwarding of multiple received PDUs

© 2017 Burkhard Stiller and Daniel Bauer

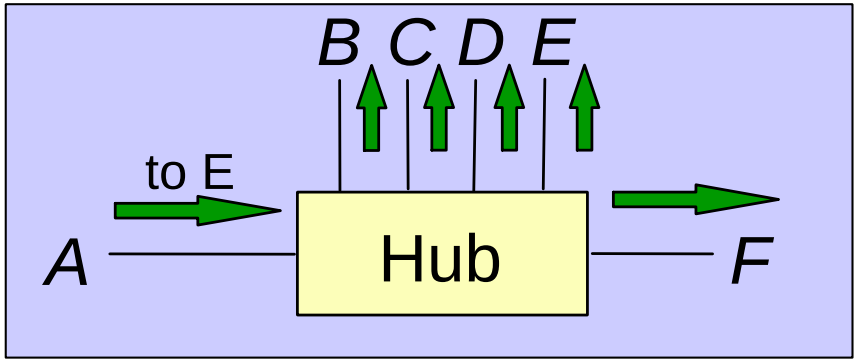
Switched LANs — Mechanisms

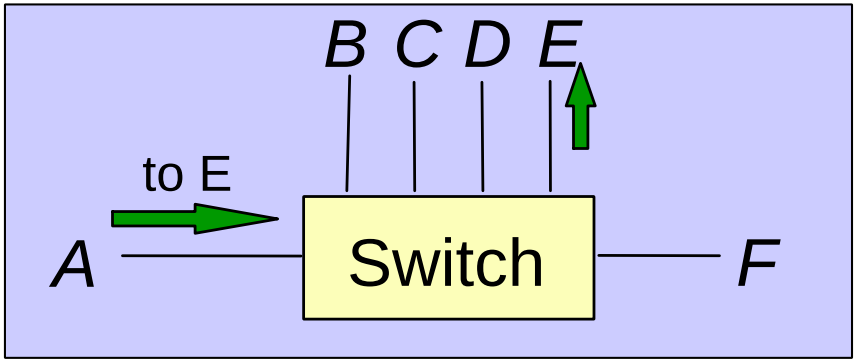
- Store-and-forward:
  - Data unit received in total and stored (analyze the “entire” packet)
  - Check of CRC and processing of filter operations
- Cut-through:
  - Address table access as soon as destination address received
  - Forwarding of packet as soon as path connected
  - Small latency
- Hybrid Switching:
  - Combination of Cut-through as well as Store-and-forward
  - Dedicated selection of mechanisms depends on error rate
- Predictive Switching:
  - Path connected before destination address received
  - Decision based on previously connected paths

© 2017 Burkhard Stiller and Daniel Bauer

Comparison: LAN Switches vs. Hubs

- At similar locations in networks
- Hubs repeat all packets while switches examine all of them
- Switches require address examination and forwarding
  - Store-and-forward or Cut-through:
  - Blocking vs. non-blocking architectures
  - Half-duplex and full-duplex modes
  - CSMA/CD no longer used for full-duplex point-to-point connections

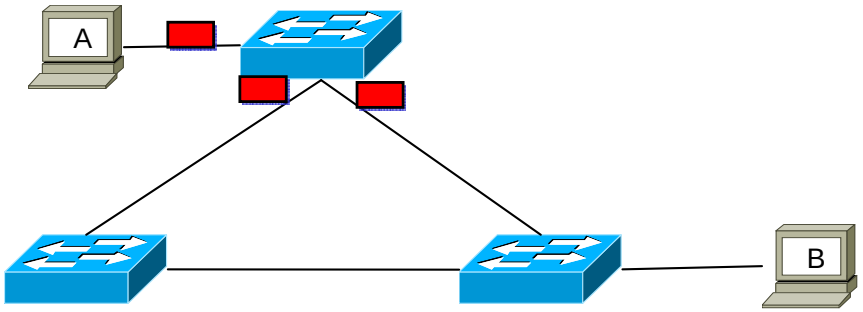




© 2017 Burkhard Stiller and Daniel Bauer

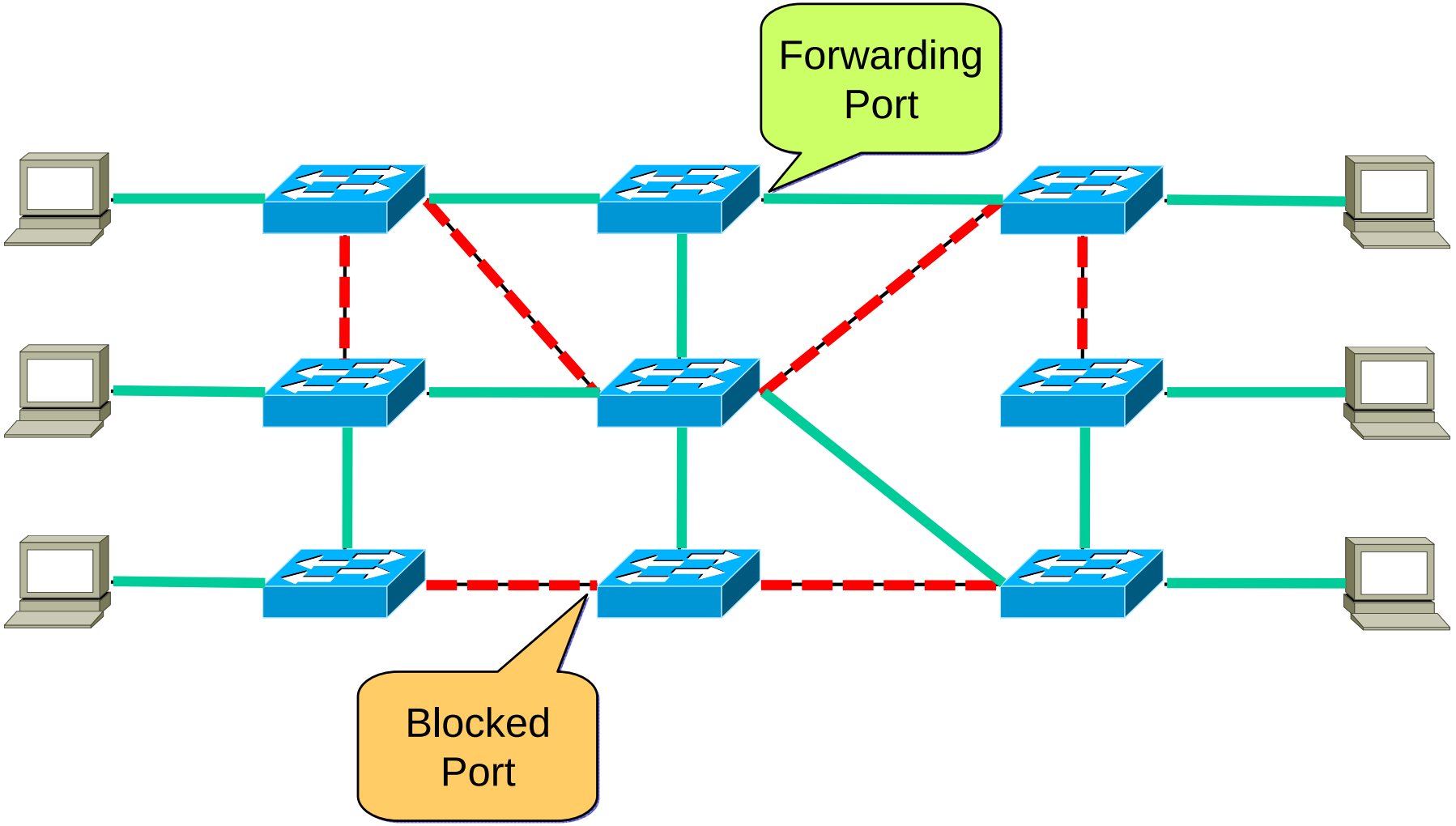
Bridges and Redundant Paths

- Bridge-coupled networks may show redundant paths
- Problem
  - Replication of many packets received (along different paths)
  - Endlessly circulating packets (looping), thus, redundant paths cause network breakdown due to “broadcast storms”
- Possible solutions
  - Do not allow for redundant paths or throw away packets
  - Make bridges smarter
    - ***Spanning Tree Algorithm*** (Radia Perlman, 1985, IEEE 802.1D)
    - Rapid Spanning Tree Protocol (IEEE 802.1D, revision 2004)
    - Shortest Path Bridging (IEEE 802.1aq, 2012)



© 2017 Burkhard Stiller and Daniel Bauer

Spanning Tree Algorithm — Idea



© 2017 Burkhard Stiller and Daniel Bauer

Spanning Tree Algorithm — Overview

- Each bridge has a unique ID (*e.g.*, B1, B2, B3)
  - Control frames can be sent to bridges
- Every port of a bridge has a weight (access cost)
  - Every port is uniquely identified per bridge

1. Select bridge with smallest ID as root
2. Select root port per bridge
3. Select bridge on each LAN that is “closest” to the root as that LAN’s designated bridge (use ID to break ties)
4. Each bridge forwards frames over each LAN for which it is the designated bridge

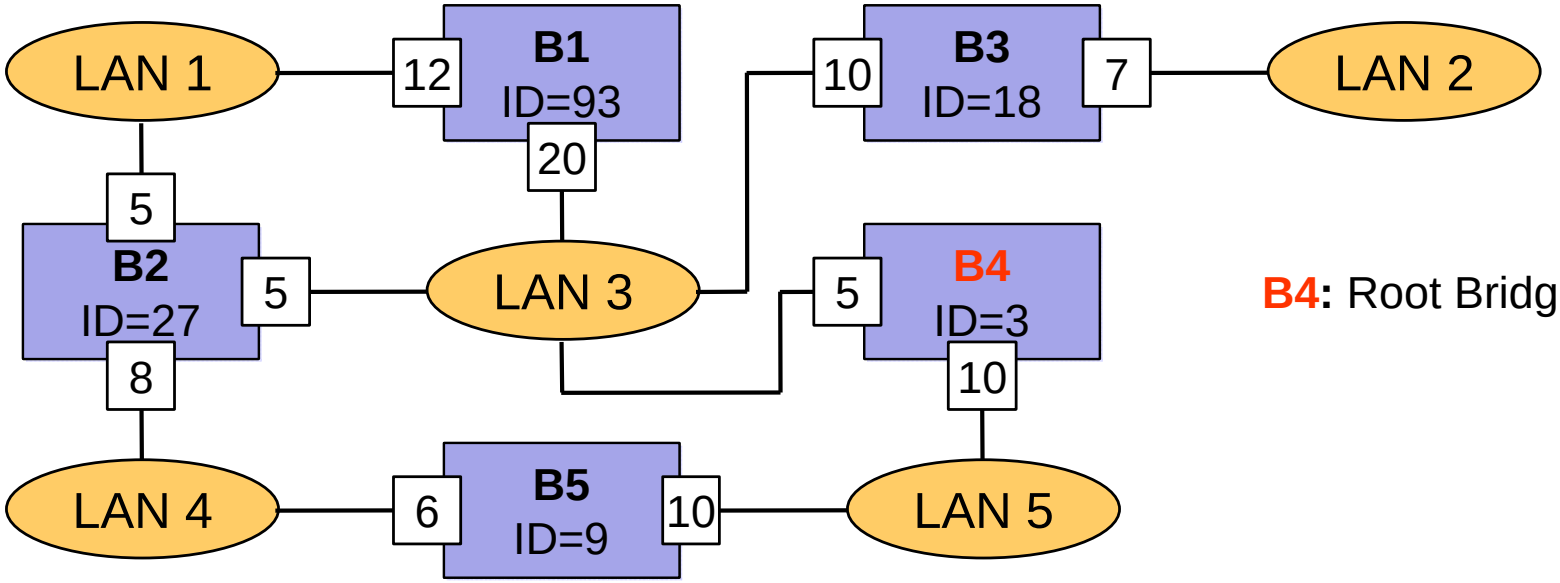
© 2017 Burkhard Stiller and Daniel Bauer

Spanning Tree Algorithm — Details

1. Select bridge with smallest ID as root of tree:
   - Every bridge assumes to be the root
   - Root bridges sent hello messages including their respective ID
   - Upon reception of a hello message with a smaller ID this bridge steps back and sends the hello message received via a broadcast message
2. Select a root port:
   - Root port of a bridge is determined by the cheapest weighted port in the direction of the root bridge
   - Minimize the sum of all access costs on a path to the root bridge
   - *E.g.*, link capacity may be used as a cost function
3. Select designated bridge:
   - Bridge with a cheapest root access will become designated bridge
   - Root bridge is a designated bridge for all networks attached to it
4. Forwarding of frames:
   - Incoming frames will be forwarded on ports for which the bridge is the designated bridge or if it’s the root port

© 2017 Burkhard Stiller and Daniel Bauer

Spanning Tree Algorithm: Root Paths



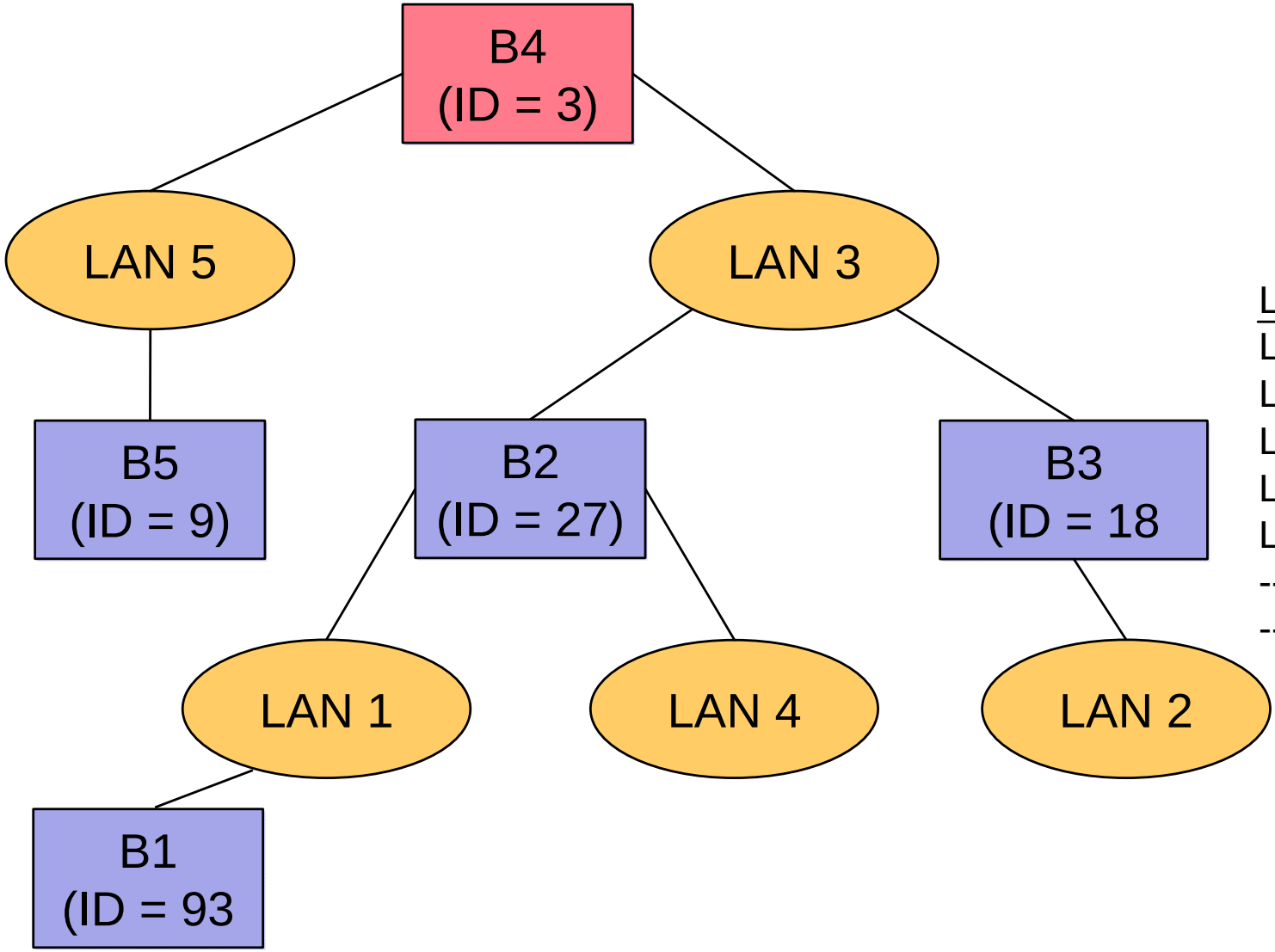
B4: Root Bridge (ID 3)

Bridge	Path costs to root bridge
B3	**10** (via LAN 3)
B1	20 (via LAN 3)
	17 = 12 + 5 (via LAN 1 & LAN 3)

Bridge	Path costs to root bridge
B2	**5** (via LAN 3)
	18 = 8 + 10 (via LAN 4 & LAN 5)
	25 = 5 + 20 (via LAN 1 & LAN 3)
B5	**10** (via LAN 5)
	11 = 6 + 5 (via LAN 4 und LAN 3)

© 2017 Burkhard Stiller and Daniel Bauer

Spanning Tree Algorithm: Result

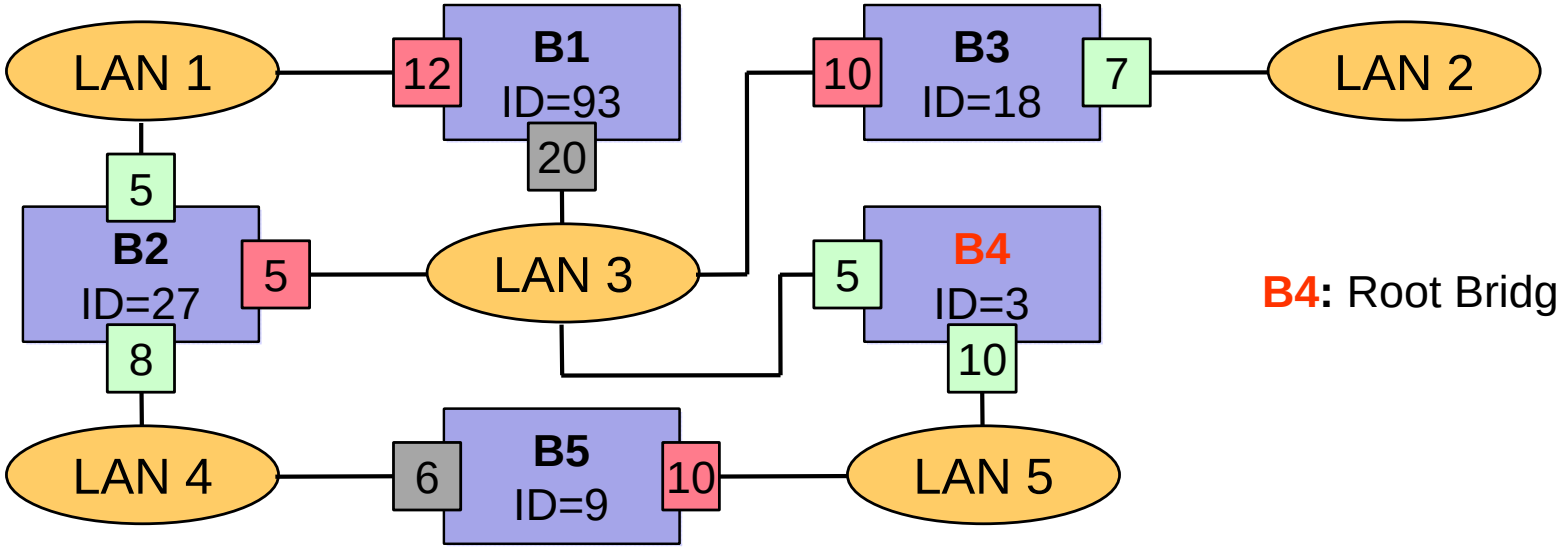


LAN	Designated Bridge
LAN 1	B2
LAN4	B2
LAN2	B3
LAN3	B4 (Root Bridge)
LAN5	B4 (Root Bridge)
---	B5
---	B1

© 2017 Burkhard Stiller and Daniel Bauer

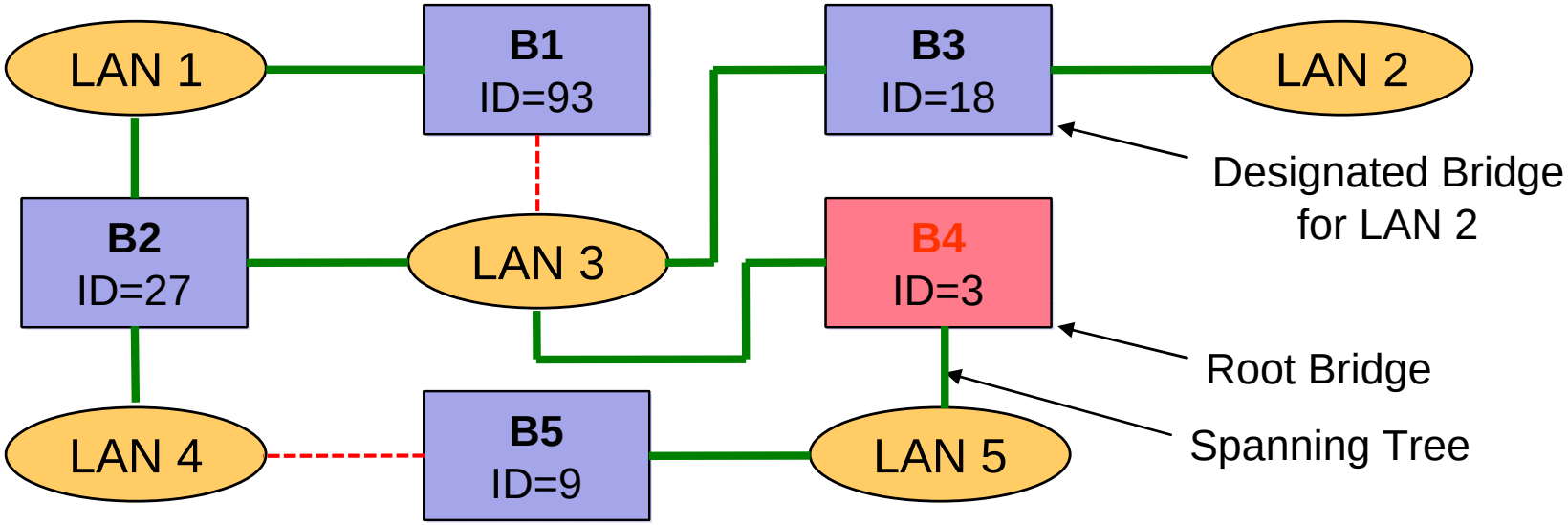
Spanning Tree Algorithm: Example

Initial state:



B4: Root Bridge (ID 3)

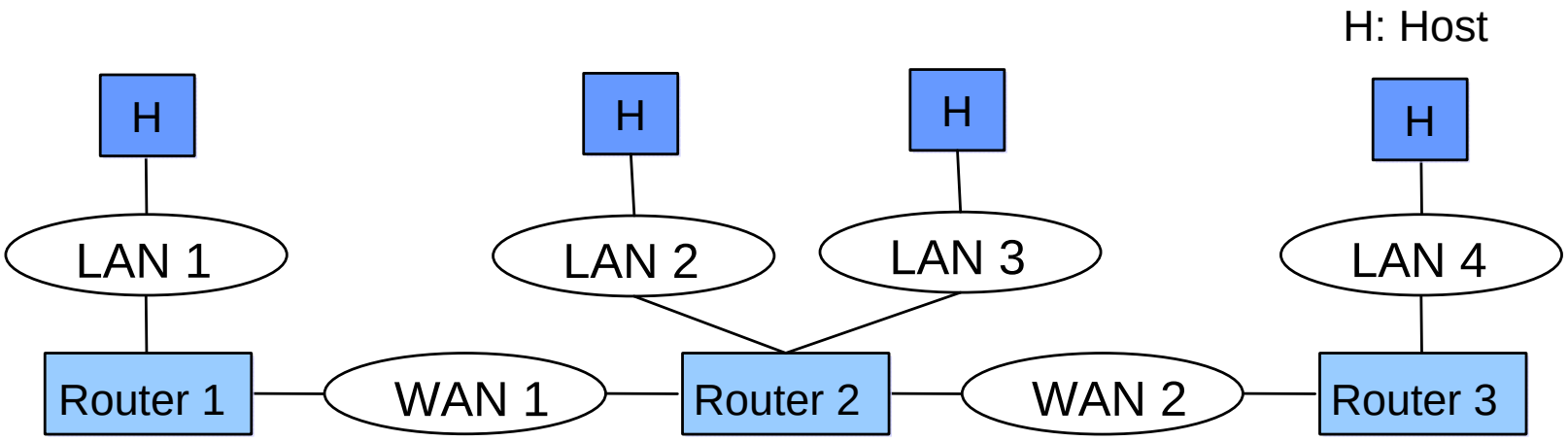
Result:

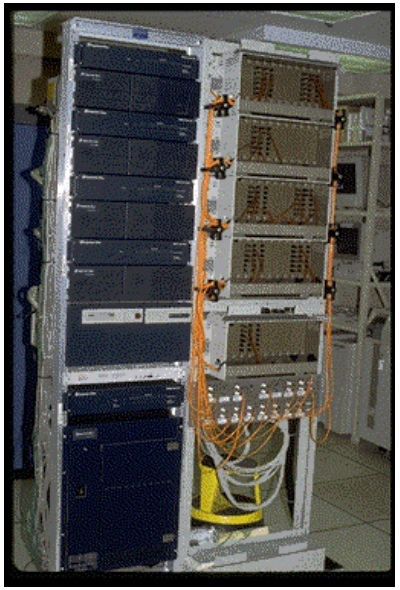


© 2017 Burkhard Stiller and Daniel Bauer

Router and IP

- Communication between hosts across networks
- Routing based on world-wide unique addresses, mainly hierarchical addresses, such as IP addresses
  - Segmentation and reassembly required
  - Security in network: access-based on address (firewall)
  - Limitation of layer 2 broadcast

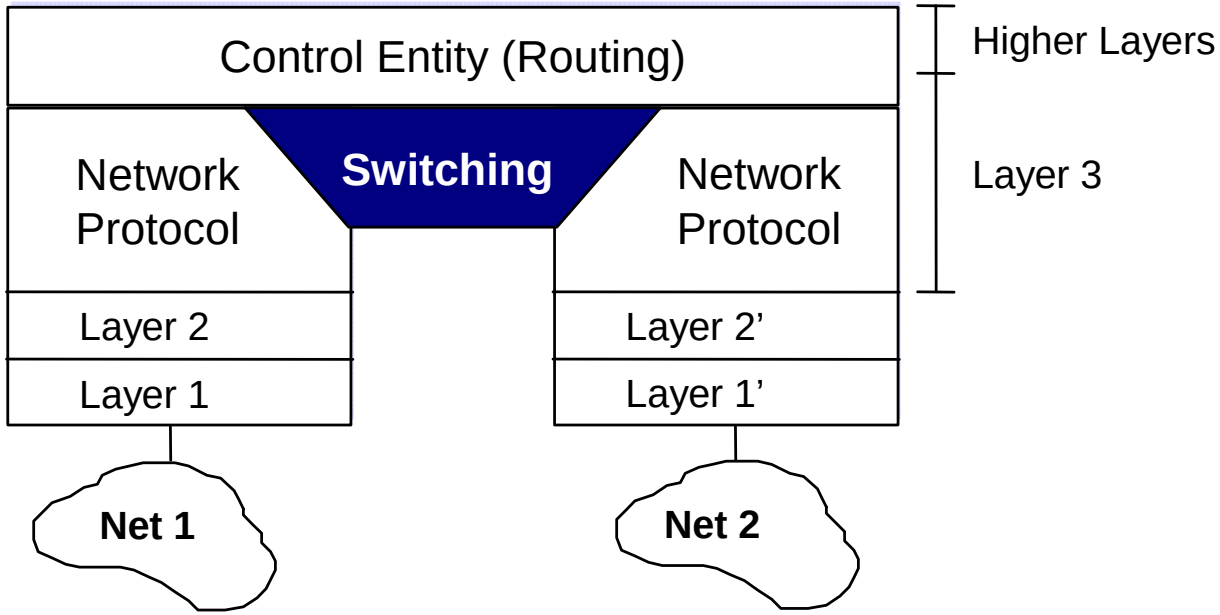




© 2017 Burkhard Stiller and Daniel Bauer

IP Router Architecture

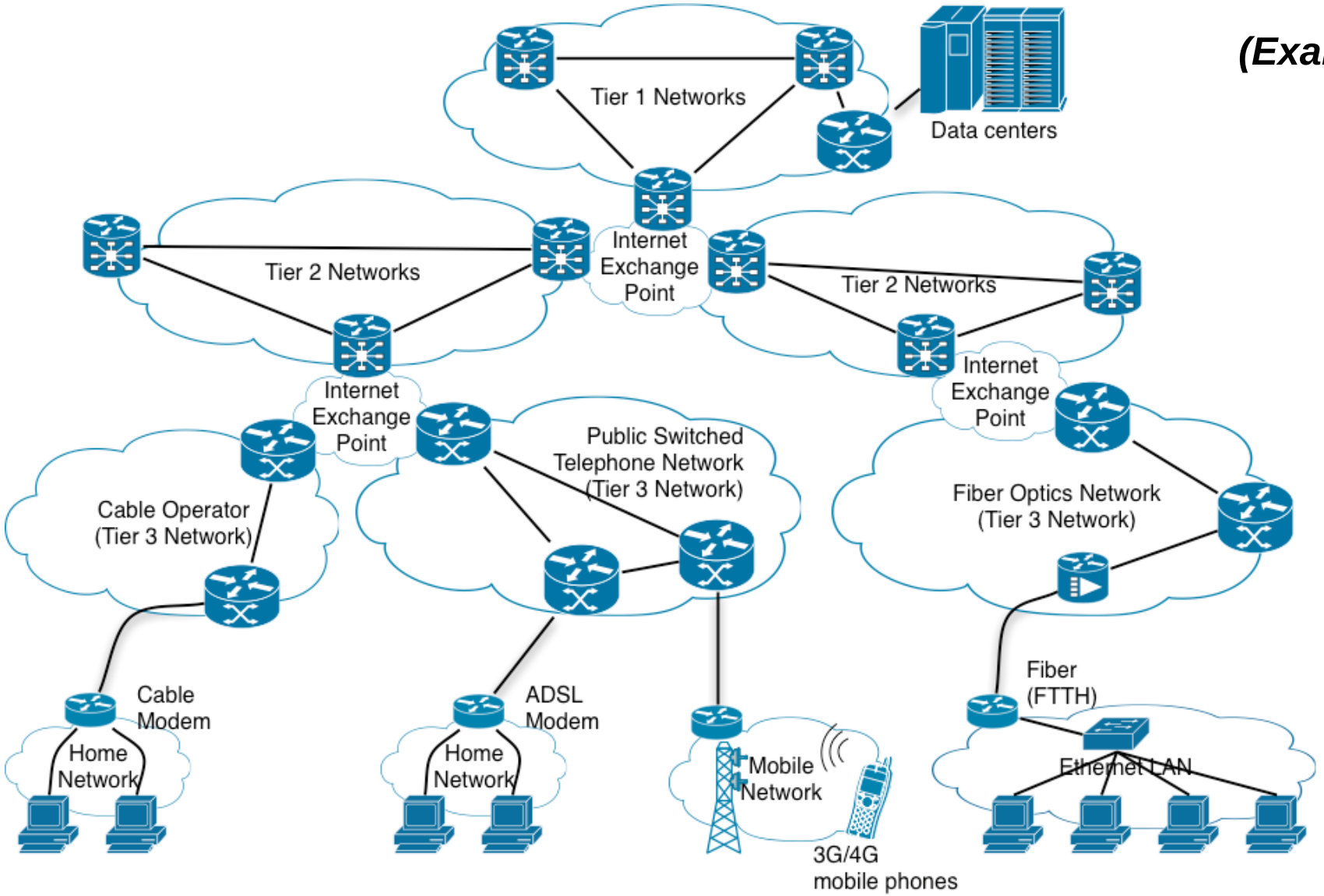
- Main characteristics:
  - Every network interface operates a distinct layer 1 & 2 entity
  - All networks use IP as common network protocol
  - Switching functions interconnects network protocol entities and implements the forwarding functionality



© 2017 Burkhard Stiller and Daniel Bauer

IP Networks and Subnetworks

(Example)



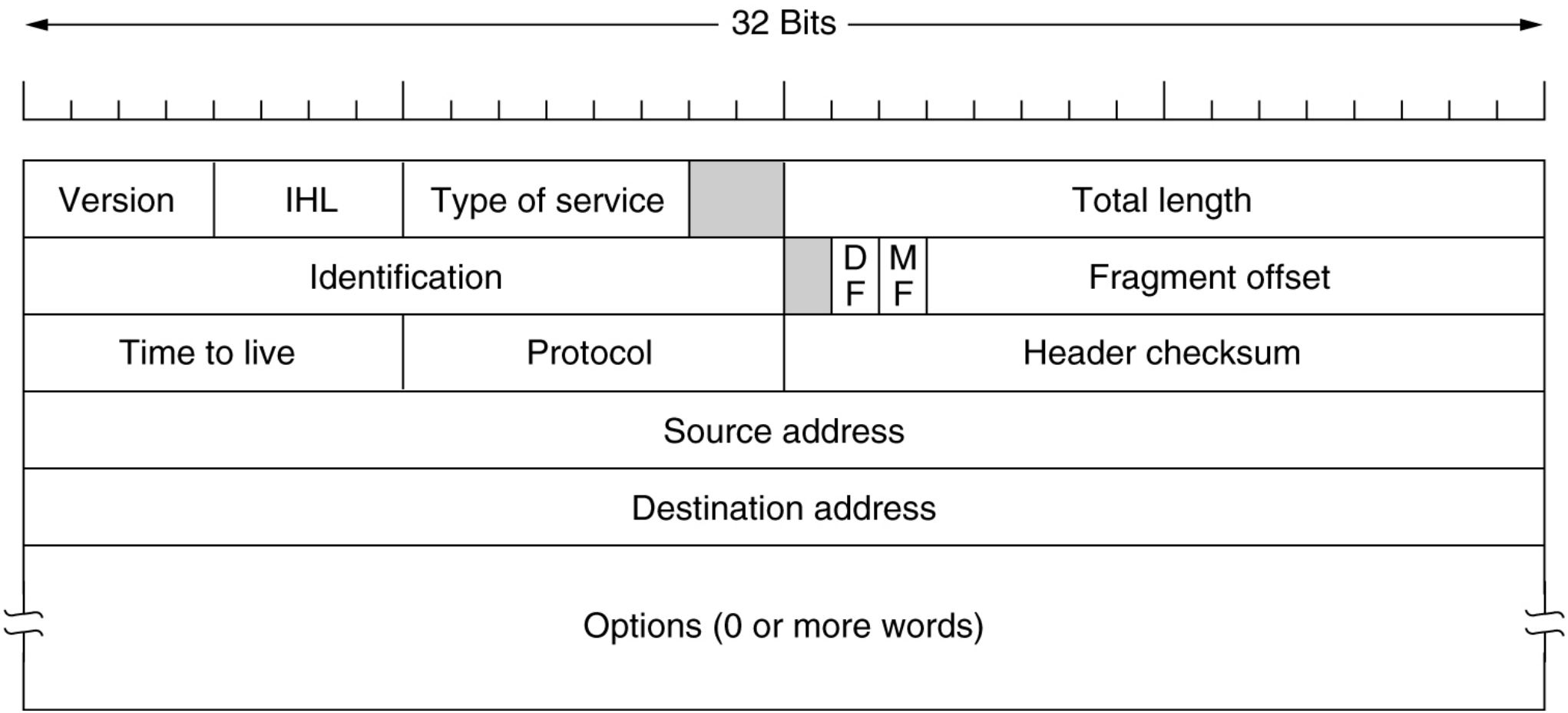
© 2017 Burkhard Stiller and Daniel Bauer

Internet Design Principles

- Separate network technology and services
  - Services are provided independent of the network
  - Service providers usually distinct from network providers
- End-to-end architecture, end-to-end principle
  - Network provides best-effort packet delivery
  - Intelligent processing happens at the edge in the hosts
- Scalability through distributed design and decentralized control
  - Network providers operate independently from each other
    ➔ Autonomous System (AS)
  - Delegation principle (*e.g.*, hierarchical routing or DNS)

© 2017 Burkhard Stiller and Daniel Bauer

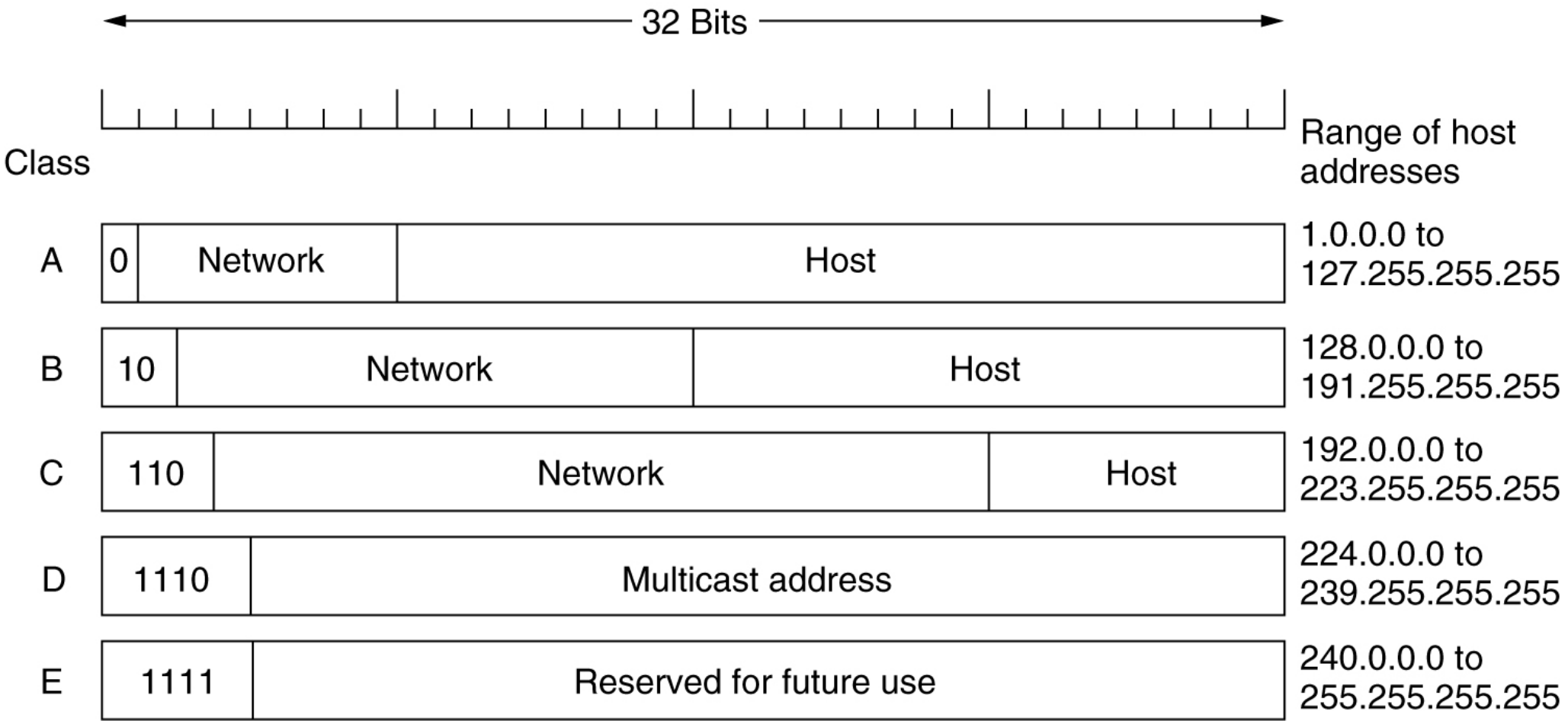
Internet Protocol (IPv4) Format



IHL: IP Header Length (in 32 bit)
DF: Don‘t fragment
MF: More fragments

© 2017 Burkhard Stiller and Daniel Bauer

IP Address Classes (Historical)



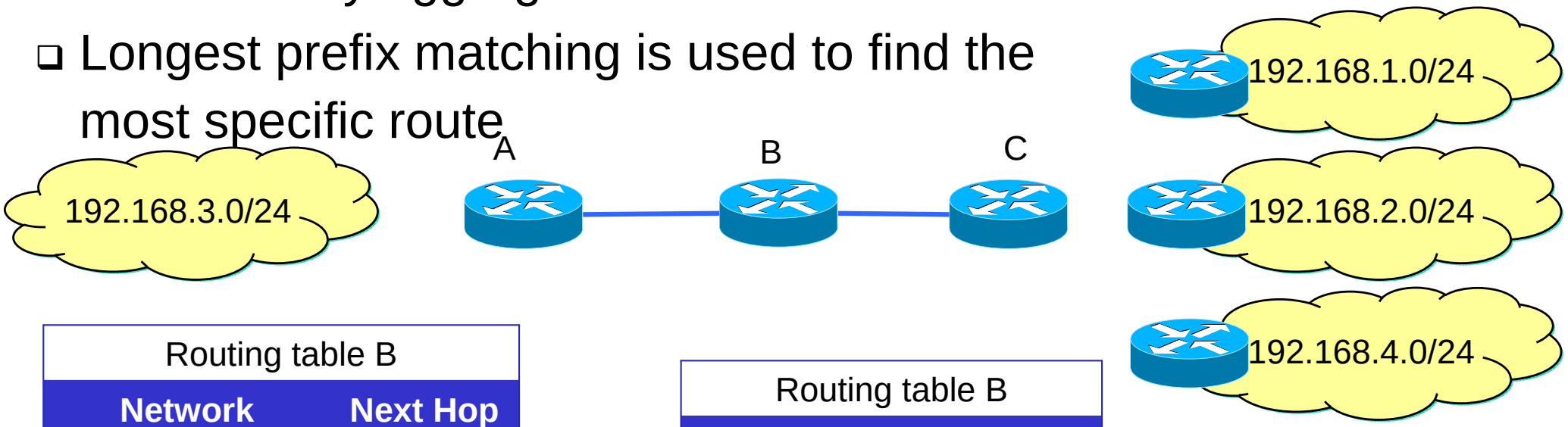
One major relevance of those address classes remains today: assignment of address blocks by

➔ ICANN (Internet Corporation for Assigned Names and Numbers), http://www.icann.org/

© 2017 Burkhard Stiller and Daniel Bauer

Classless Inter-Domain Routing (CIDR)

- CIDR was introduced 1993 and replaced the IP address classes
- Full IP Address space is divided into blocks at bit boundaries
- Routers may aggregate address blocks
- Longest prefix matching is used to find the most specific route



Routing table B

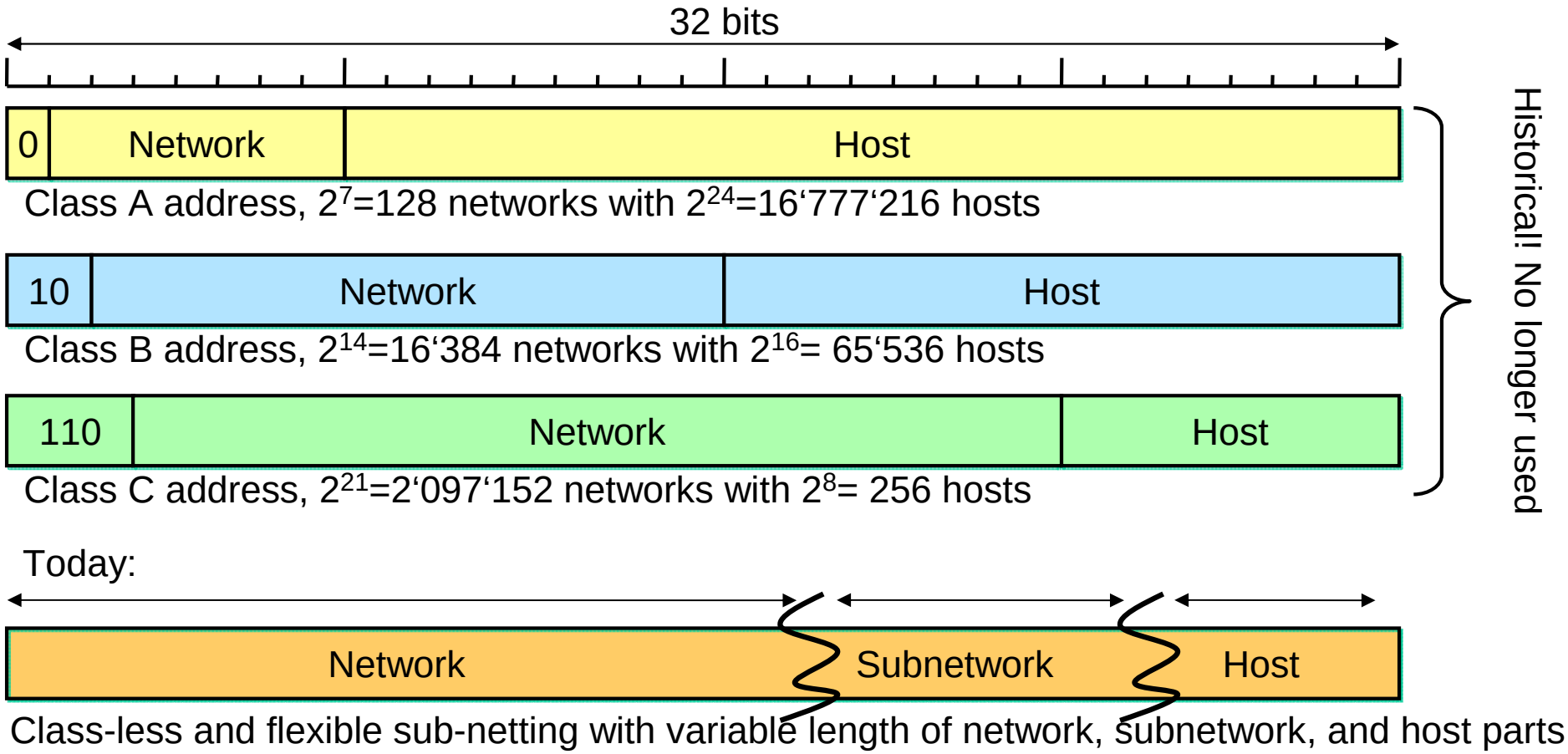
Network	Next Hop
192.168.1.0/24	C
192.168.2.0/24	C
192.168.3.0/24	A
192.168.4.0/24	C

Routing table B

Network	Next Hop
192.168.3.0/24	A
192.168.0.0/21	C

© 2017 Burkhard Stiller and Daniel Bauer

Efficient Use of IP Subnets

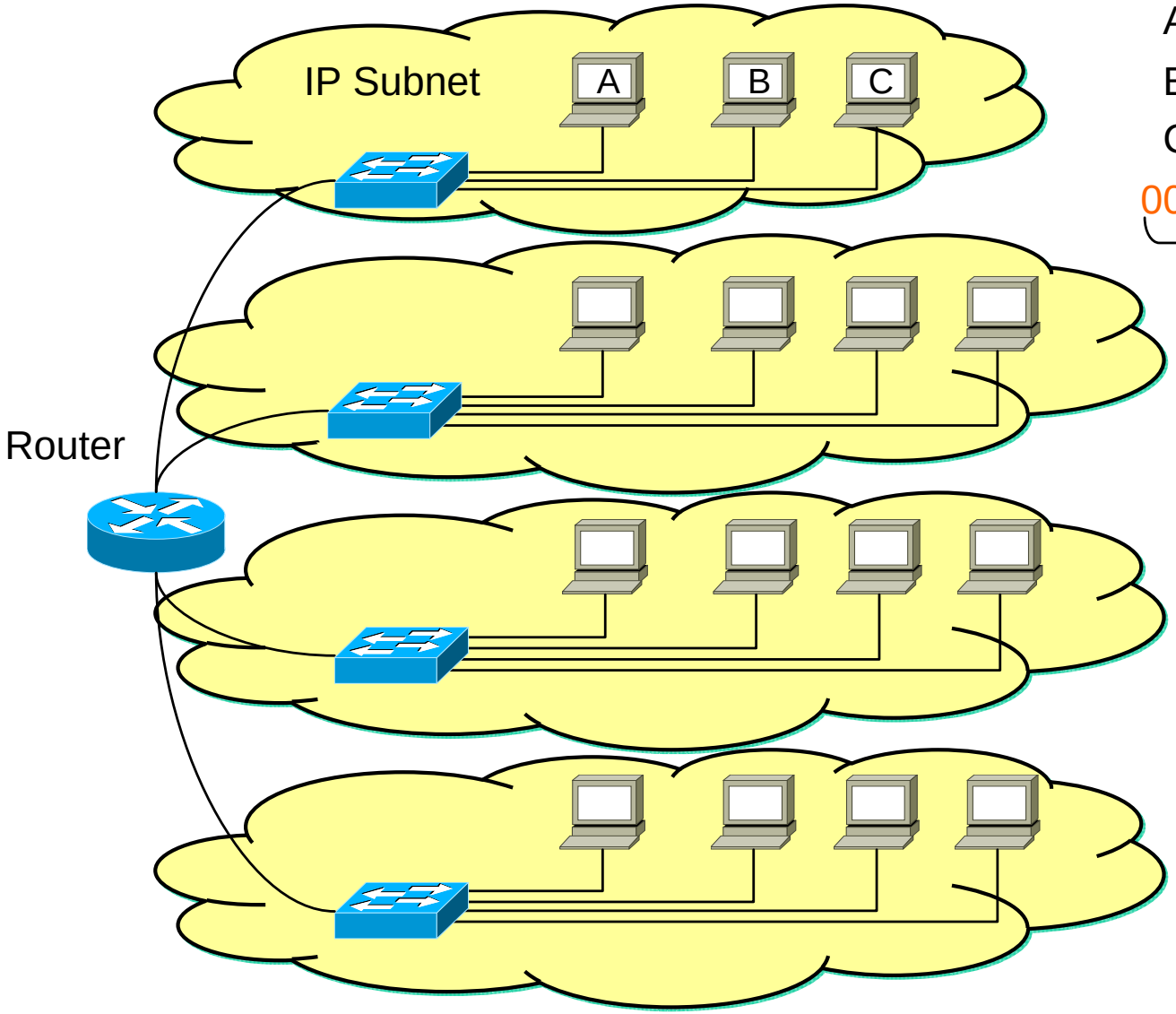


Example: Point-to-point connection between 2 routers:

- Requires 3 addresses (broadcast address, one for each router)
- Efficiency class-based addressing: One Class-C address: 3/256 = 1.17%
- Efficiency class-less sub-netting: 2-bits for host part: 3/4 = 75%

© 2017 Burkhard Stiller and Daniel Bauer

IP Address Structure (1)



A: 10.90.12.195

B: 10.90.12.205

C: 10.90.12.221

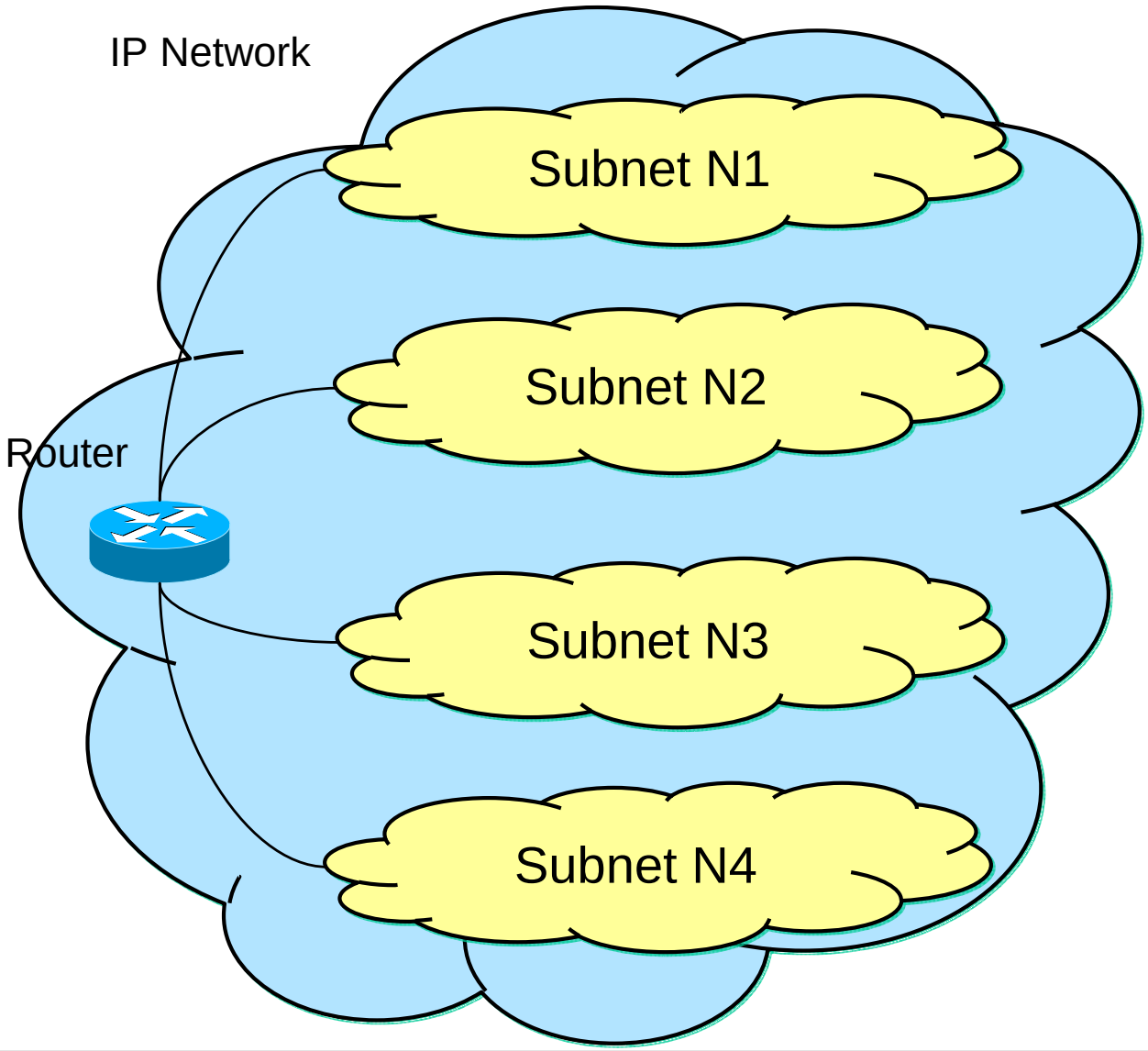
00001010.01011010.00001100.11011101

Network part | Host part

- All host interfaces in the same subnetwork share a common address prefix: the network part of the address
- Notation of network: 10.90.12.192/26
- Netmask: All bits in the network part set to 1: 255.255.255.192

© 2017 Burkhard Stiller and Daniel Bauer

IP Address Structure (2)



N1: 10.90.12.192/26
N2: 10.90.13.0/26
N3: 10.90.13.64/26
N4: 10.90.13.192/26

00001010.01011010.00001101.11000000

Network part | Subnet part

- All subnets in the same network share a common address prefix: the network part of the address
- Notation of network: 10.90.0.0/20
- Netmask: All bits in the network part set to 1: 255.255.240.0

© 2017 Burkhard Stiller and Daniel Bauer

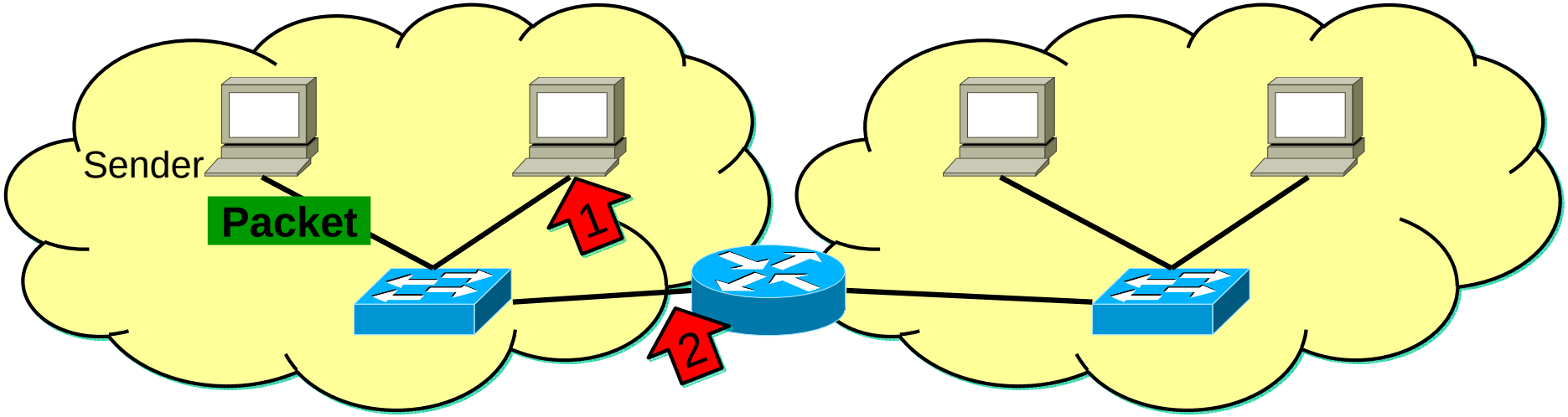
Special Addresses

Address	Name	Description
Host part all ‘0’	Network address	*E.g.*, 10.90.13.64 when using a /26 network
Host part all ‘1’	Broadcast address	*E.g.*, 10.90.12.127 for above network
0.0.0.0/8	“This” network	Used for broadcasting on the current network
127.0.0.0/8	Loopback network	Used for testing the IP stack
127.0.0.1	Localhost address	Specifies the local host
169.254.0.0/16	Link local network	Used for automatically configuring a network interface if DHCP is not available (See RFC 3927)
172.16.0.0/12	Private use	Used for building private IP networks
192.168.0.0/16	Private use	Used for building private IP networks
10.0.0.0./8	Private use	Used for building private IP networks
224.0.0.0/4	IP Multicast	IP Multicast group addresses
255.255.255.255	Limited broadcast	Broadcast within the IP subnet

➔ IANA (Internet Assigned Numbers Authority), http://www.iana.org/numbers

© 2017 Burkhard Stiller and Daniel Bauer

IP Forwarding – Hop-by-Hop



- Sending (forwarding) of an IP packet to an IP destination
  1. Destination in same subnetwork: Map IP destination address to MAC address and send directly
  2. Destination outside: Identify matching router's IP address, map to router's MAC address, and send to router
- Configuration required
  - Host IP address and network mask for each network interface
  - IP address of default router (and additional routers if required)

© 2017 Burkhard Stiller and Daniel Bauer

Forwarding/Routing Table

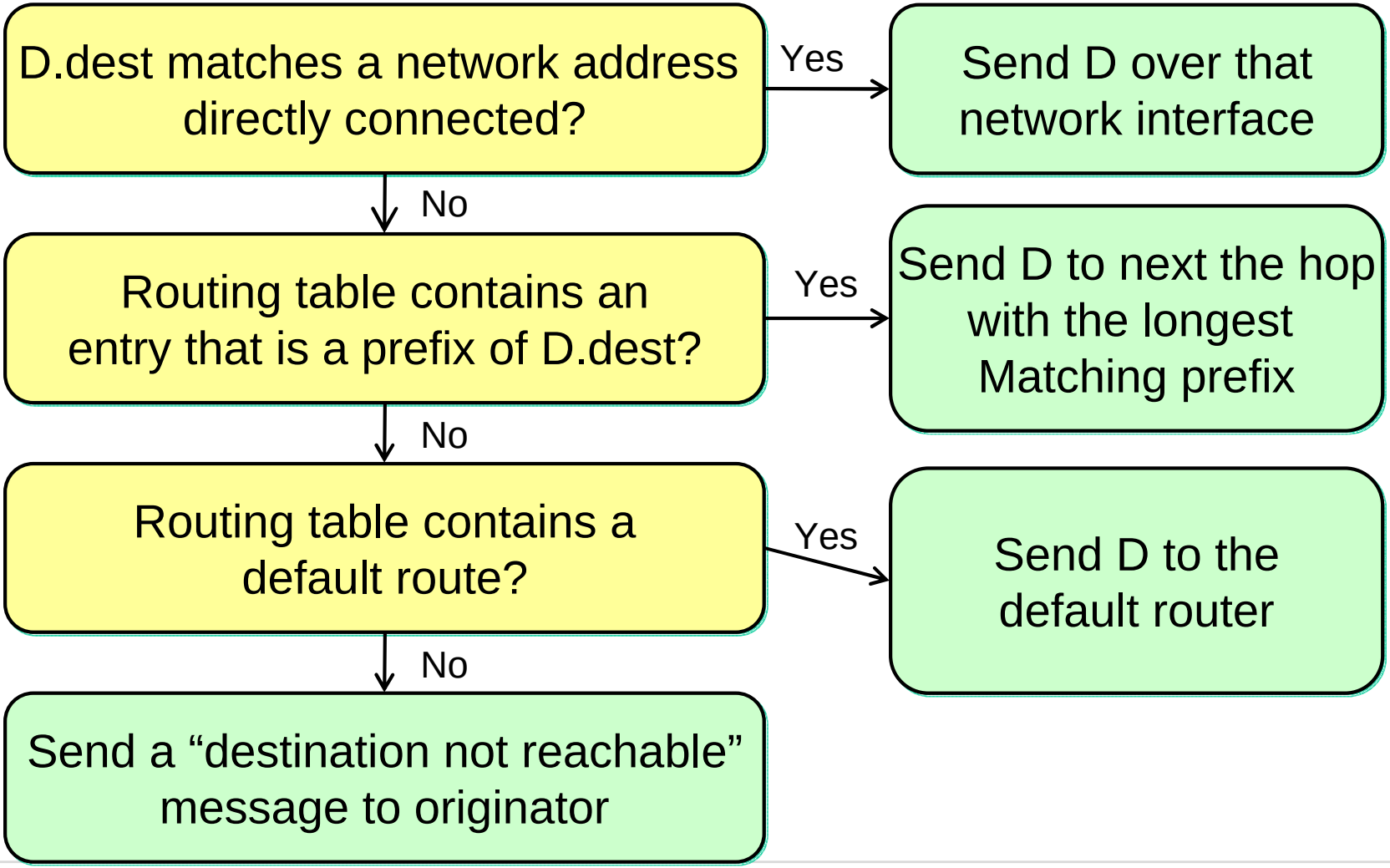
- Hosts and Routers maintain a single routing table
- Routing table identifies next IP address hop to be used
- Forwarding decision based on table entry with longest matching prefix of destination IP address
- Example:

Destination	Gateway	Netif
Default/0	10.5.16.254	en0
Default/0	10.5.222.254	en1
10.5.16/21	link#4	en0
10.5.222/23	link#5	en1
10.5.21.9/32	127.0.0.1	lo0
10.5.222.14/32	127.0.0.1	lo0
127/8	127.0.0.1	lo0
127.0.0.1/32	127.0.0.1	lo0
169.254/16	link#4	en0

© 2017 Burkhard Stiller and Daniel Bauer

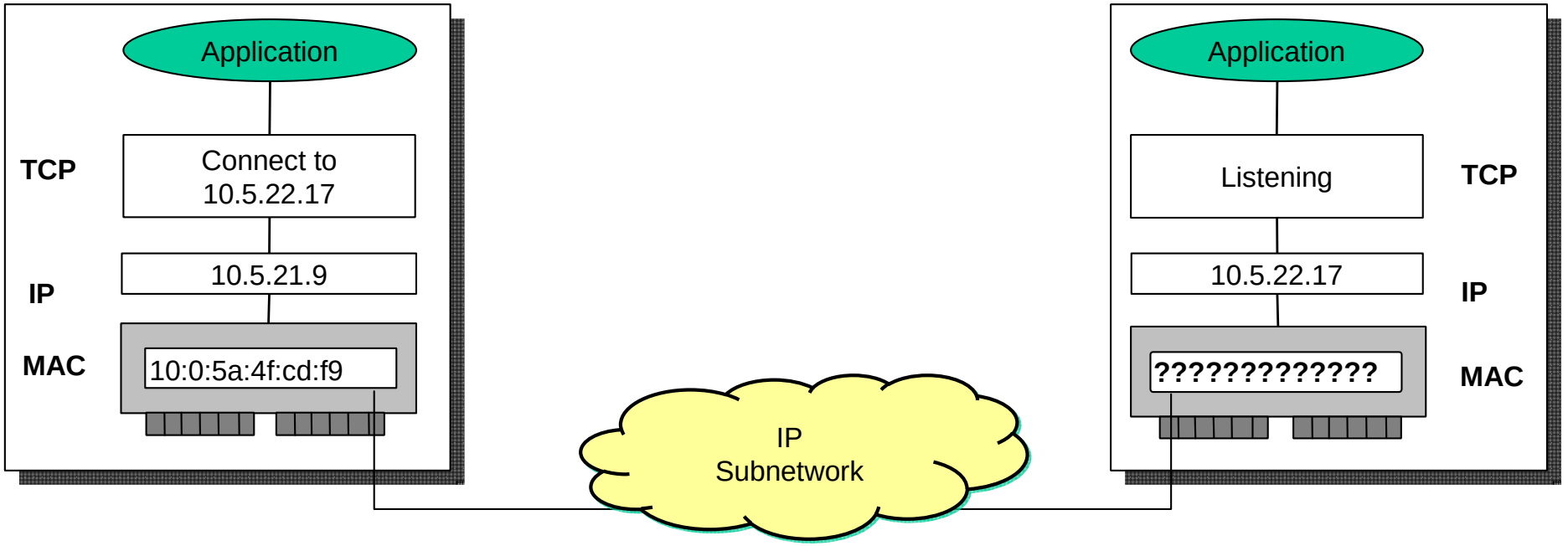
Forwarding Algorithm

- Forwarding of a Datagram D to destination D.dest:



© 2017 Burkhard Stiller and Daniel Bauer

Prerequisite: Address Mapping

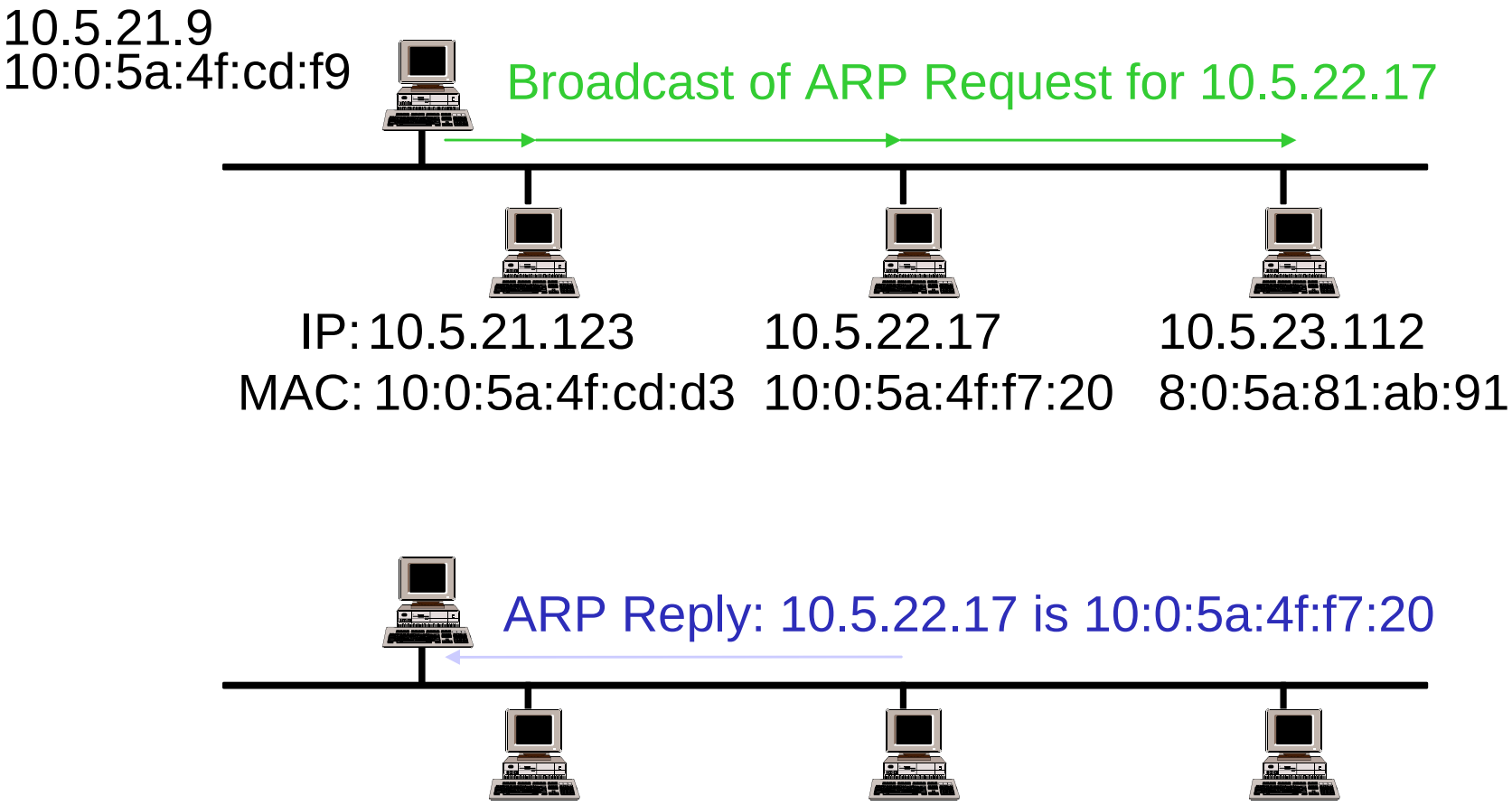


- Translation/mapping of IP address to MAC addresses
  - Caching of address mapping results for efficient operation
  - Determine (a) destination host or (b) next hop router
- Techniques:
  - Encode physical address in IP host part, use mapping table, or

© 2017 Burkhard Stiller and Daniel Bauer

Address Resolution Protocol (ARP)

- … create mapping table entries on-the-fly
  - Applying local network and broadcast capabilities



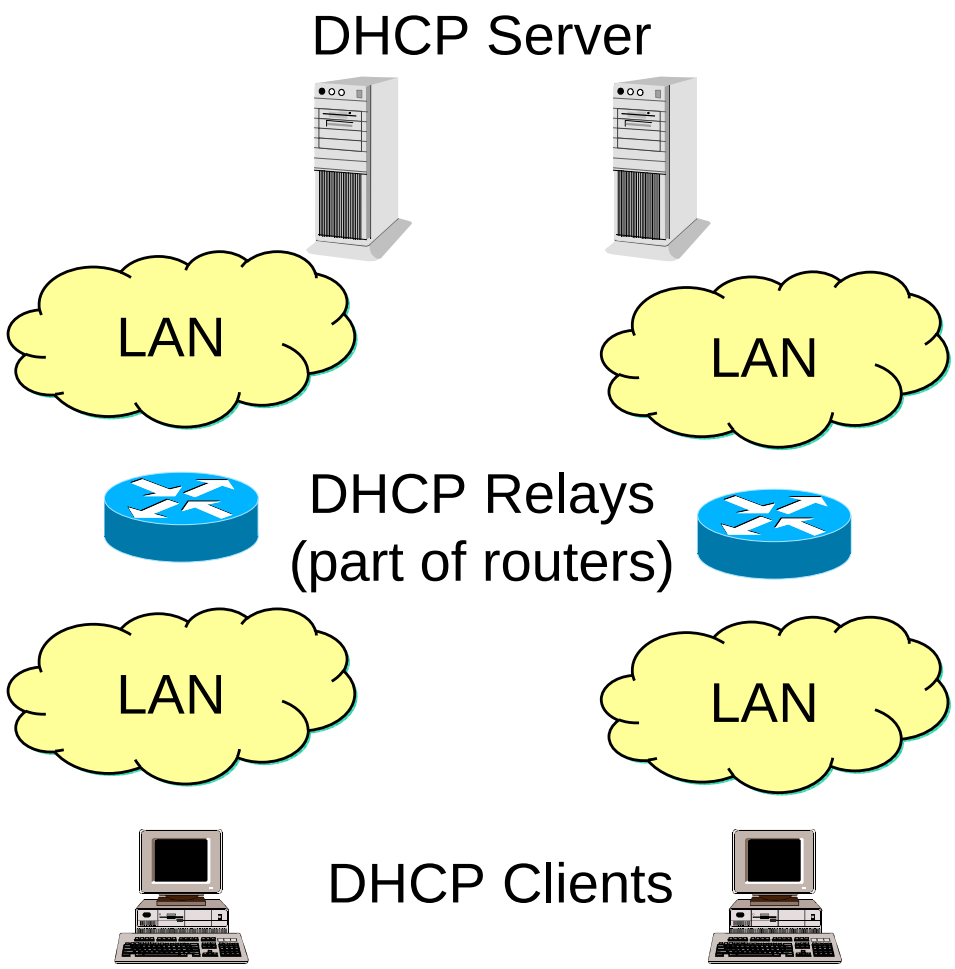
© 2017 Burkhard Stiller and Daniel Bauer

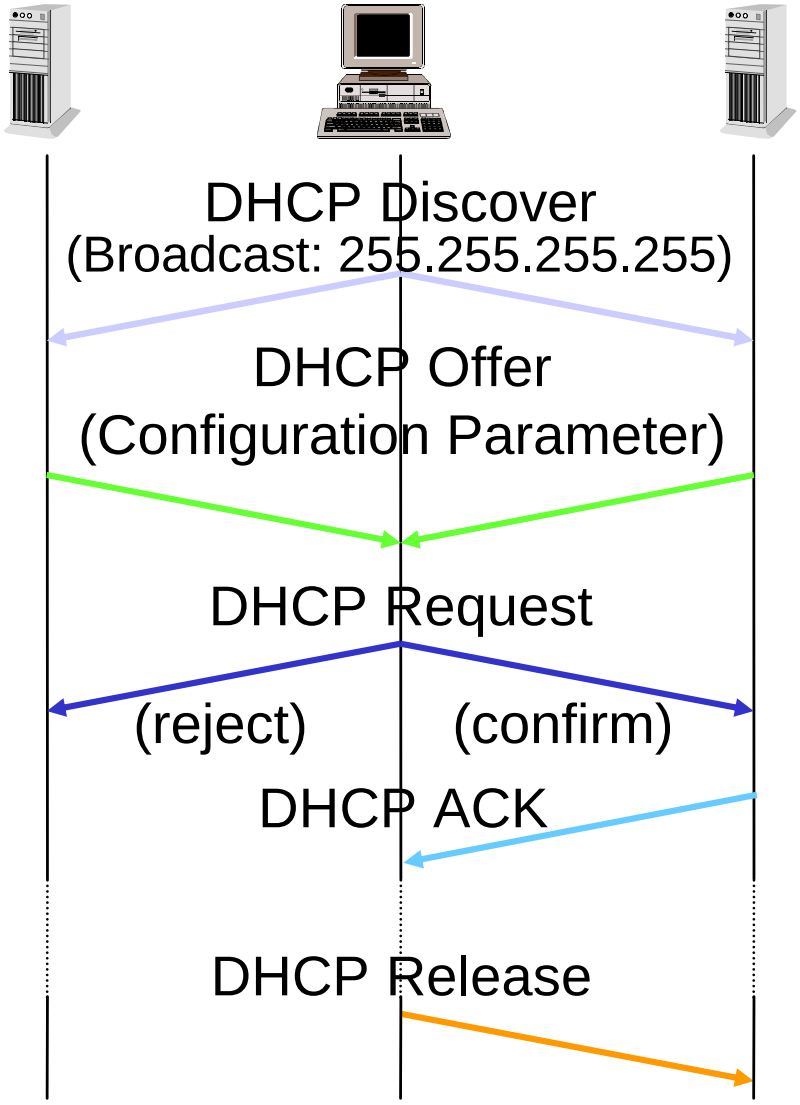
Dynamic Host Configuration Protocol

- Creating new IP address on-the-fly
  - Dynamic Host Configuration Protocol (DHCP)
- DHCP server allocates a dedicated IP address for specific duration (validity)
  - Renewal process of allocation with shorter process
- DHCP Configuration items:
  - Default router and network mask
  - NTP server (Network Time Protocol)
  - DNS server (Domain Name System)
- DHCP standardized by IETF (RFC 2131)

© 2017 Burkhard Stiller and Daniel Bauer

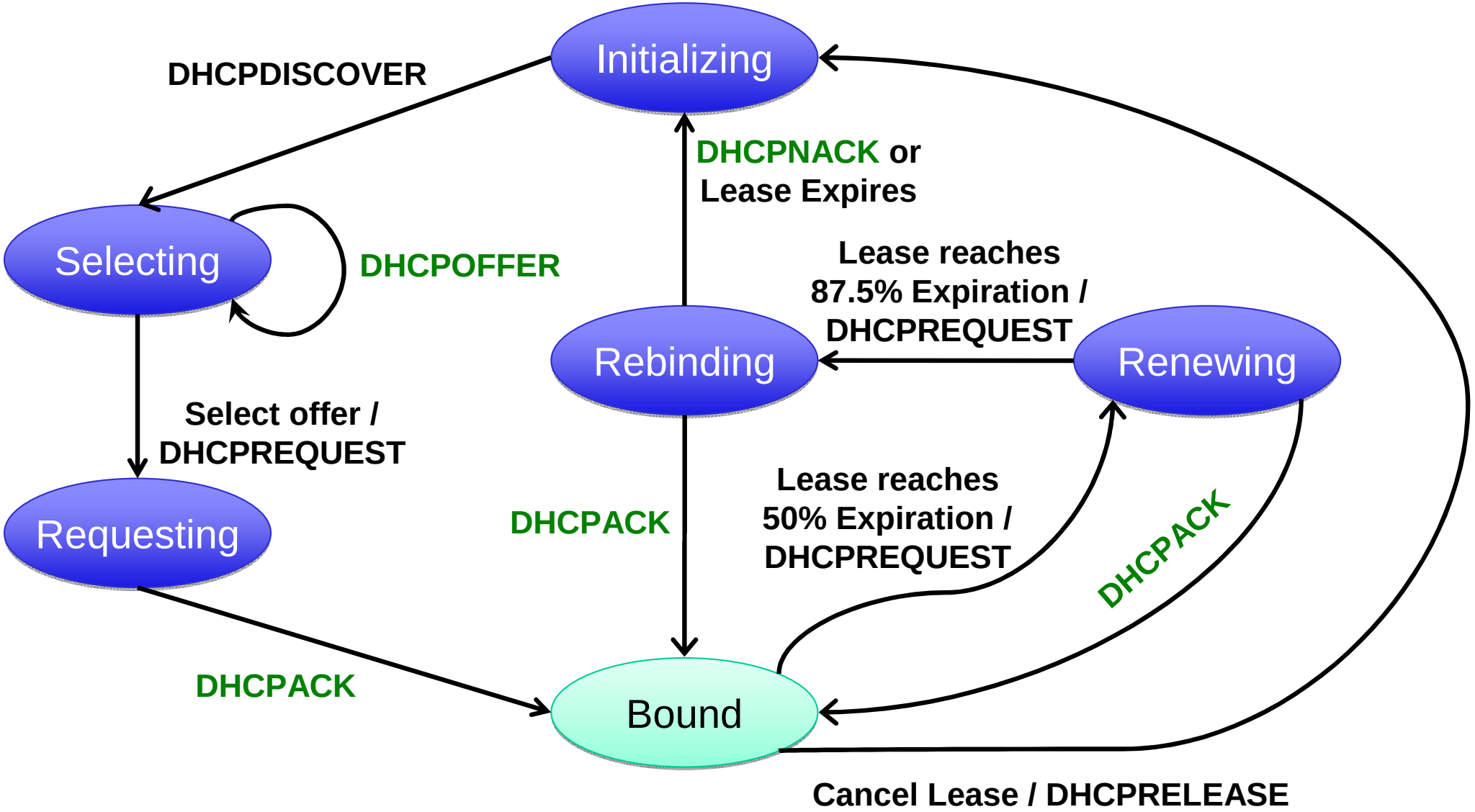
DHCP Operation





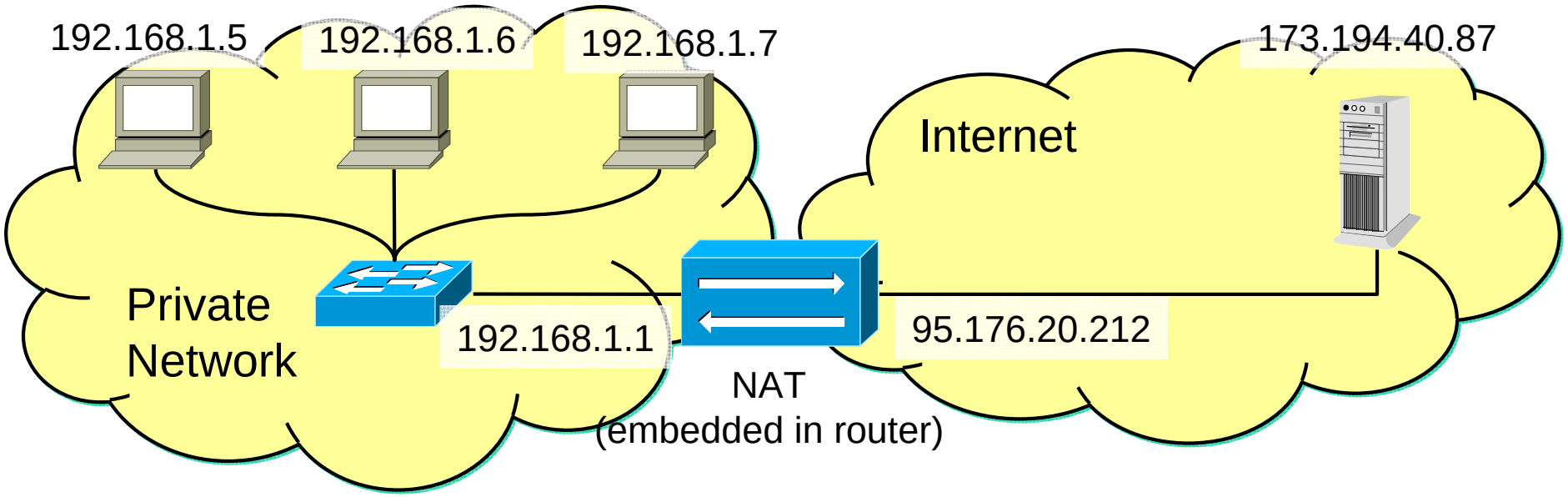
© 2017 Burkhard Stiller and Daniel Bauer

DHCP Client State Diagram



© 2017 Burkhard Stiller and Daniel Bauer

Network Address Translation (NAT)



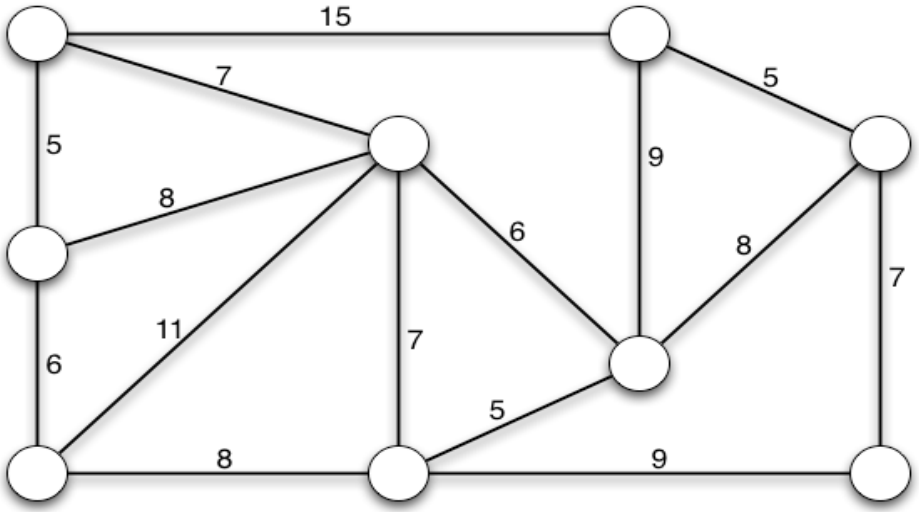
Source (IP Address and Port)	Destination (IP Address and Port)		Source (IP Address and Port)	Destination (IP Address and Port)
192.168.1.5:3457	173.194.40.87:80	⟷	95.176.20.212:123	173.194.40.87:80
192.168.1.5:8754	173.194.30.87:220	⟷	95.176.20.212:124	173.194.30.87:220
192.168.1.6:3457	173.194.40.87:80	⟷	95.176.20.212:125	173.194.40.87:80
192.168.1.7:7621	173.194.40.87:443	⟷	95.176.20.212:126	173.194.40.87:443

Translation

© 2017 Burkhard Stiller and Daniel Bauer

Routing and Forwarding

- Forwarding versus routing:
  - **Forwarding**: to select an output port based on destination address and routing table
  - **Routing**: process by which the routing table is built
- Network viewed as a graph
- Problem:
  Find the lowest cost path between any two nodes
- Basis for computation:
  - Topology graph (static or dynamic)
  - “Costs” for traversing a link (static or dynamic)



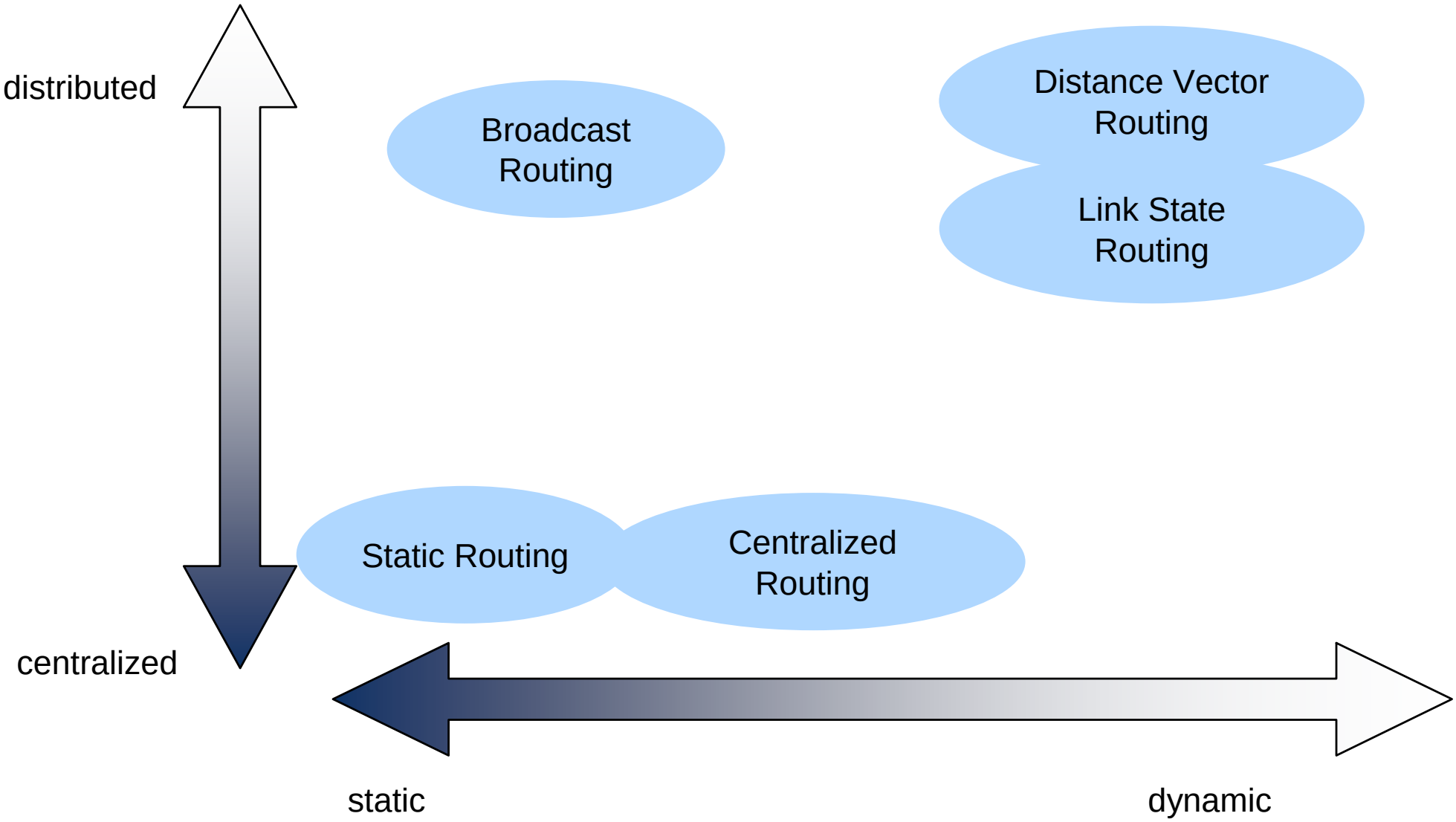
© 2017 Burkhard Stiller and Daniel Bauer

Classification of Routing Algorithms (1)

- Centralization:
  - Where does the routing algorithm operate?
    - Centralized (within a network management center)
    - Decentralized (distributed amongst routers)
- Dynamicity of routing algorithm:
  - Non-adaptive: Routing tables remain constant in routers
  - Adaptive: Routing decisions depend on network status, such as topology or load
- Conflict:
  - Nodes utilize old or non-complete information on network status
  - Effort to exchange routing information between routers

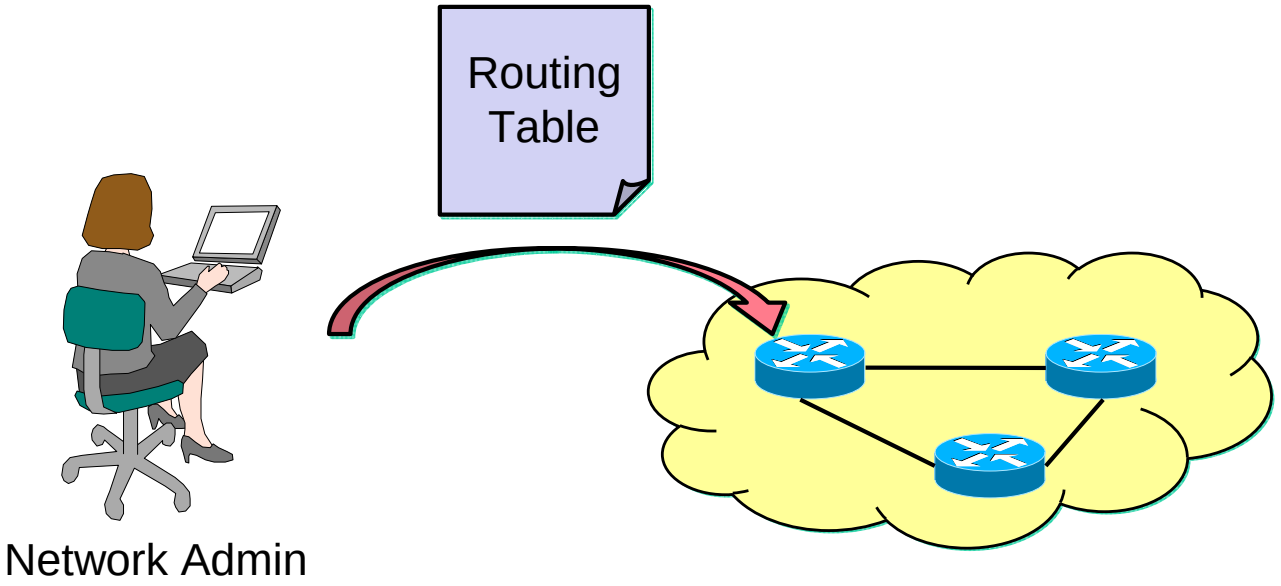
© 2017 Burkhard Stiller and Daniel Bauer

Classification of Routing Algorithms (2)



© 2017 Burkhard Stiller and Daniel Bauer

Static Routing

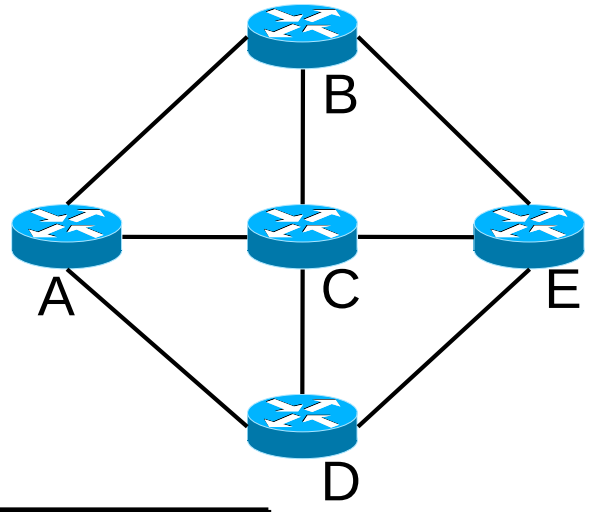


- Configured by network administrator
- Routers do not exchange topology information
- Limited fault tolerance
- Widely used for very small “stub” networks with default routes

© 2017 Burkhard Stiller and Daniel Bauer

Static Routing – Load Balancing

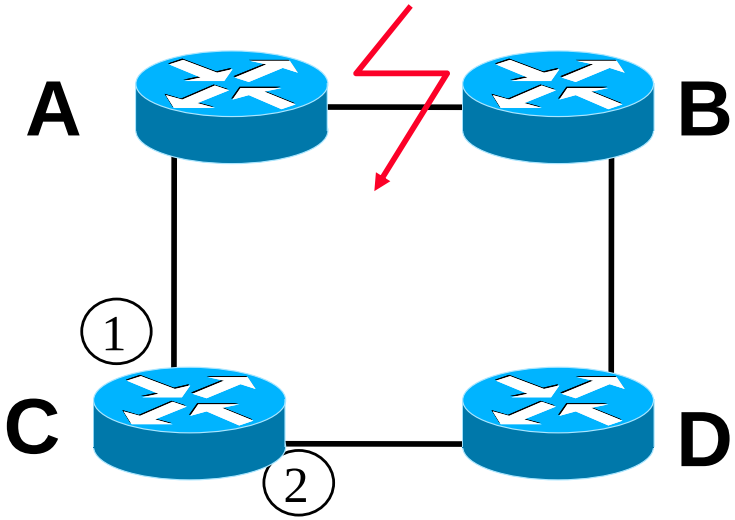
- Row includes N data elements, identifying the best, second best, third weighted route (outgoing link)
- Load balancing:
  - Draw of random number x, $0.0 \leq x \leq 1.0$
  - If $x < 0.6$ forward to B
  - If $0.6 \leq x < 0.9$ forward to C
  - Else forward to D

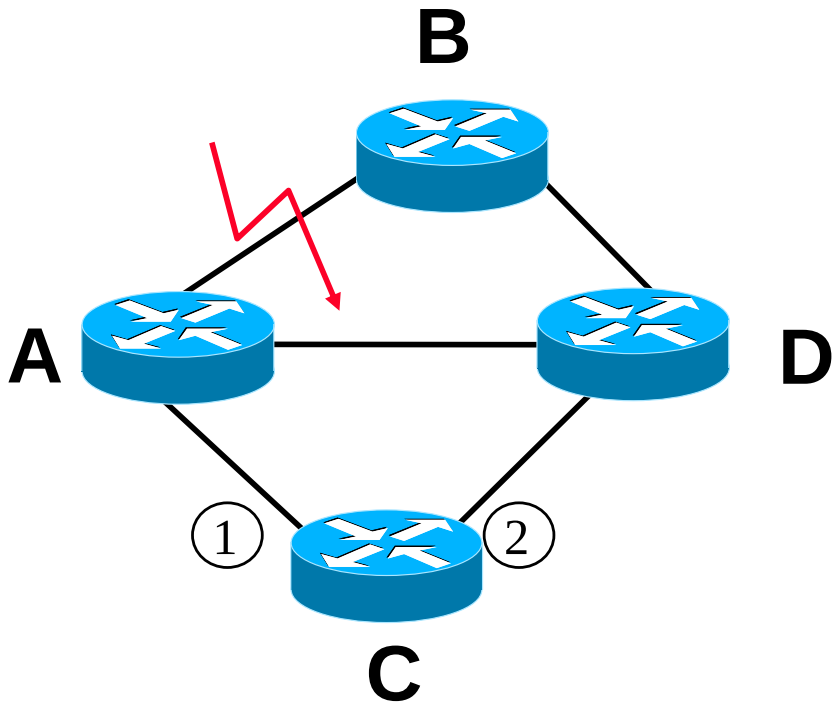


Dest.	1st Choice		2nd Choice		3rd Choice	
	No	Weight	No.	Weight	No	Weight
E	B	0.6	C	0.3	D	0.1
⋮						

© 2017 Burkhard Stiller and Daniel Bauer

Problems with Static Routing





Routing Table of Node C:

Dest.	Port	Cost
A	1	1
D	2	1
B	**1**	**2 => unreachable!**
B	2	2

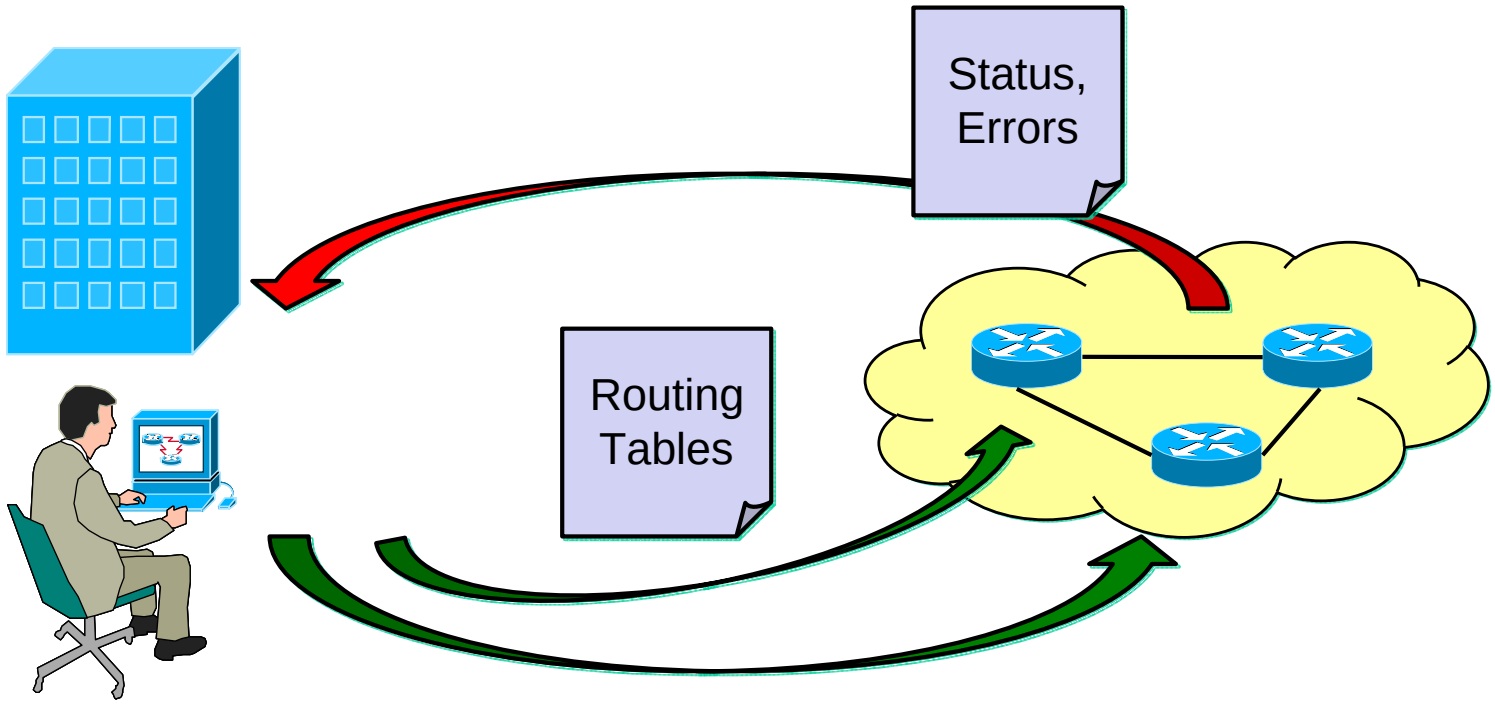
Routing Table of Node C:

Dest.	Port	Cost
A	1	1
D	2	1
B	**1**	**2 => rerouted over D**
B	2	2

© 2017 Burkhard Stiller and Daniel Bauer

Centralized Routing

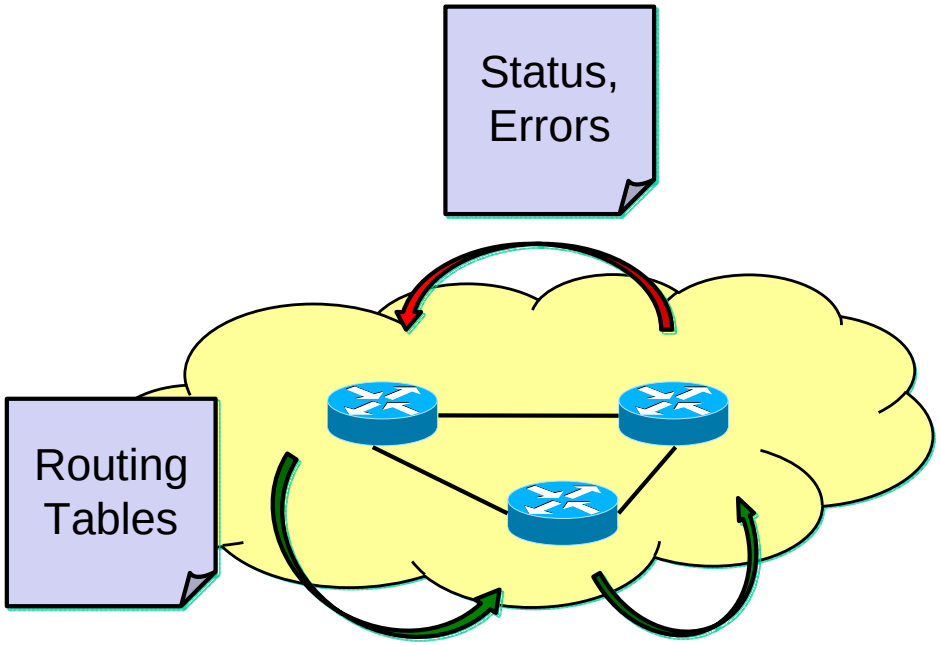
Network Management Center (occasionally also called Route Control Center)



- Configured by network management center
- Routers do not exchange topology information, but deliver status and error messages
- Seldomly used these days

© 2017 Burkhard Stiller and Daniel Bauer

Decentralized Routing

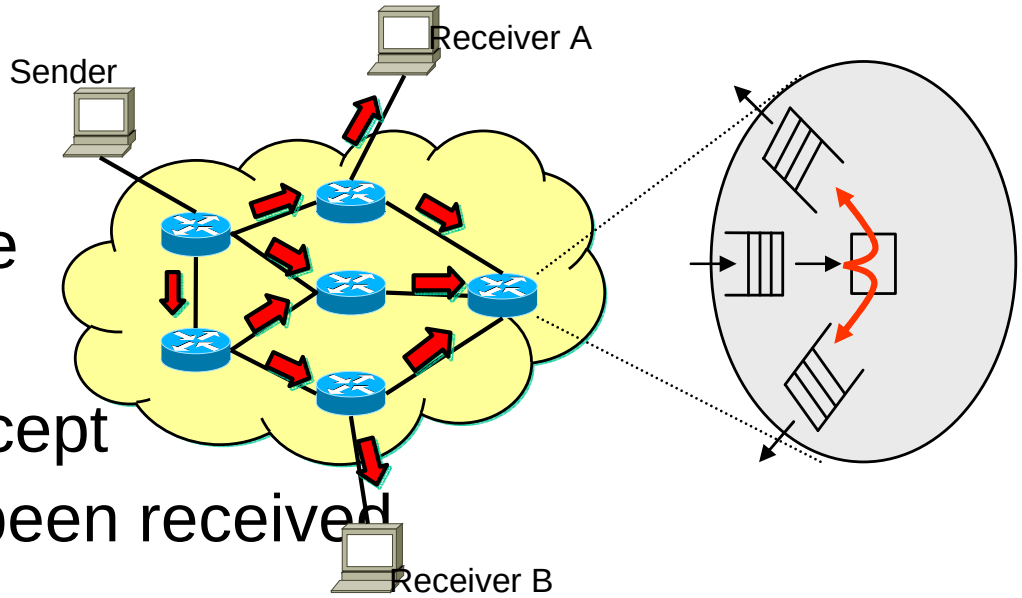


- Configured by interacting, autonomous routers
- Routers do exchange control and topology information
- High fault tolerance
- Widely used for smaller and larger networks

© 2017 Burkhard Stiller and Daniel Bauer

Flooding

- Simplest distributed algorithm, non-adaptive
- Does not use a routing table
- Every incoming packet is forwarded on every link, except on the link on which it had been received

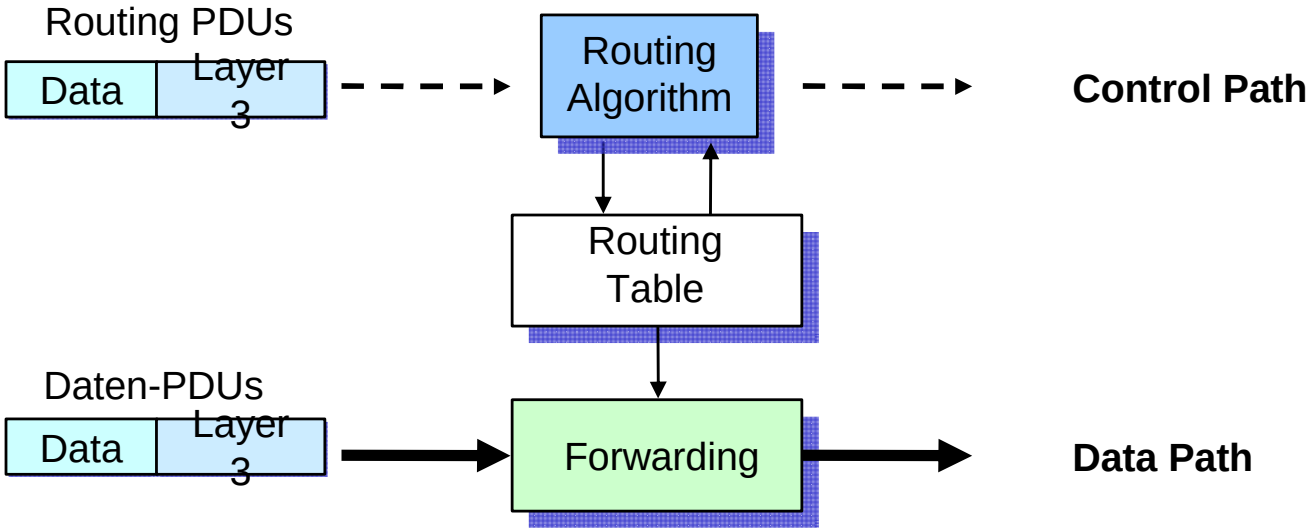


- Reduction of traffic
  - Duplicate detection by sequence numbers or message IDs
  - Time-to-live control via hops
  - Variants:
    - Selective flooding: Forwarding only on selected links
    - Random Walk: Randomized selection of link

© 2017 Burkhard Stiller and Daniel Bauer

Dynamic and Distributed Routing

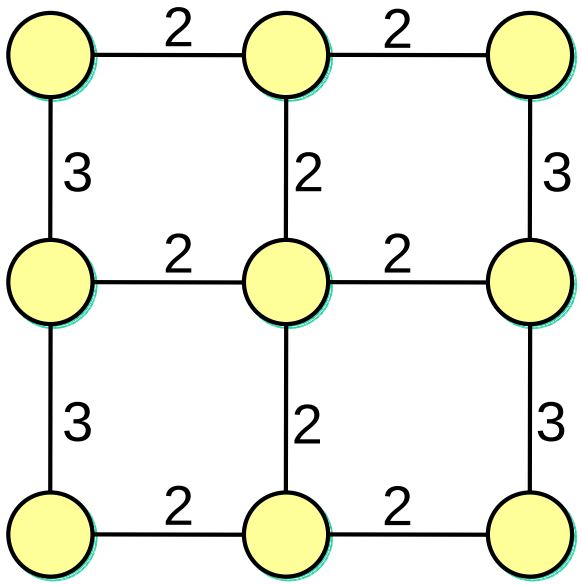
- Routers continuously exchange control information
- Control path on top: Routing PDU encapsulated into N-PDUs or T-PDUs)
  - Routing information obtained from routing protocol
  - Path computation maintains routing table



- How is the path computed?

© 2017 Burkhard Stiller and Daniel Bauer

Path Computation Algorithms

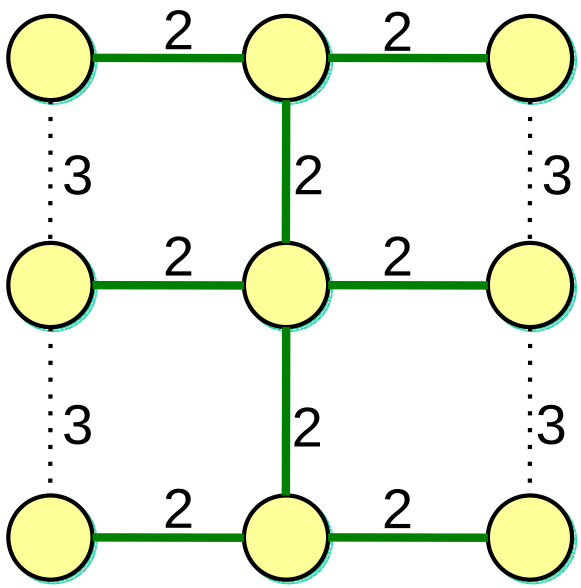


Network represented as undirected, connected graph

- Network is represented as a *connected graph*
- A *graph* consists of a set of *nodes* (*vertices*) being interconnected by *links (edges)*
- *Links (edges)* are associated with a traversing *cost* (link metric or link weight)
- Links may be directed or undirected, resulting in a directed or undirected graph
- Undirected links have symmetric traversal costs, directed links have costs per direction

© 2017 Burkhard Stiller and Daniel Bauer

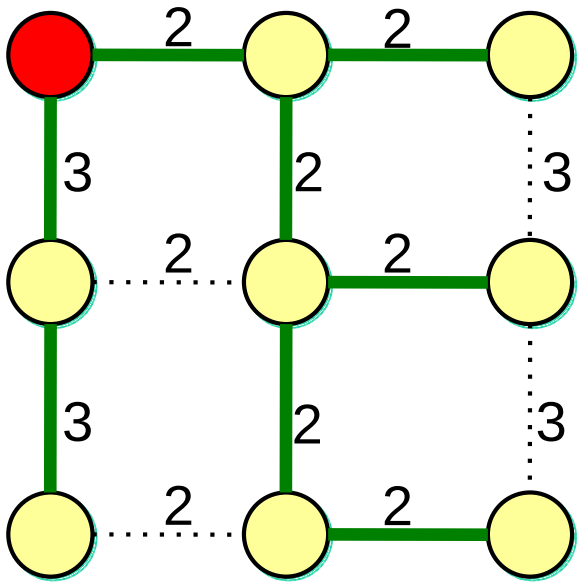
Minimum Spanning Tree (MST)



- Minimum Spanning Tree:
  - Set of links that connect all nodes in the graph such that the sum of the link costs is minimized
- For a given graph, more than one minimum spanning tree might exist

© 2017 Burkhard Stiller and Daniel Bauer

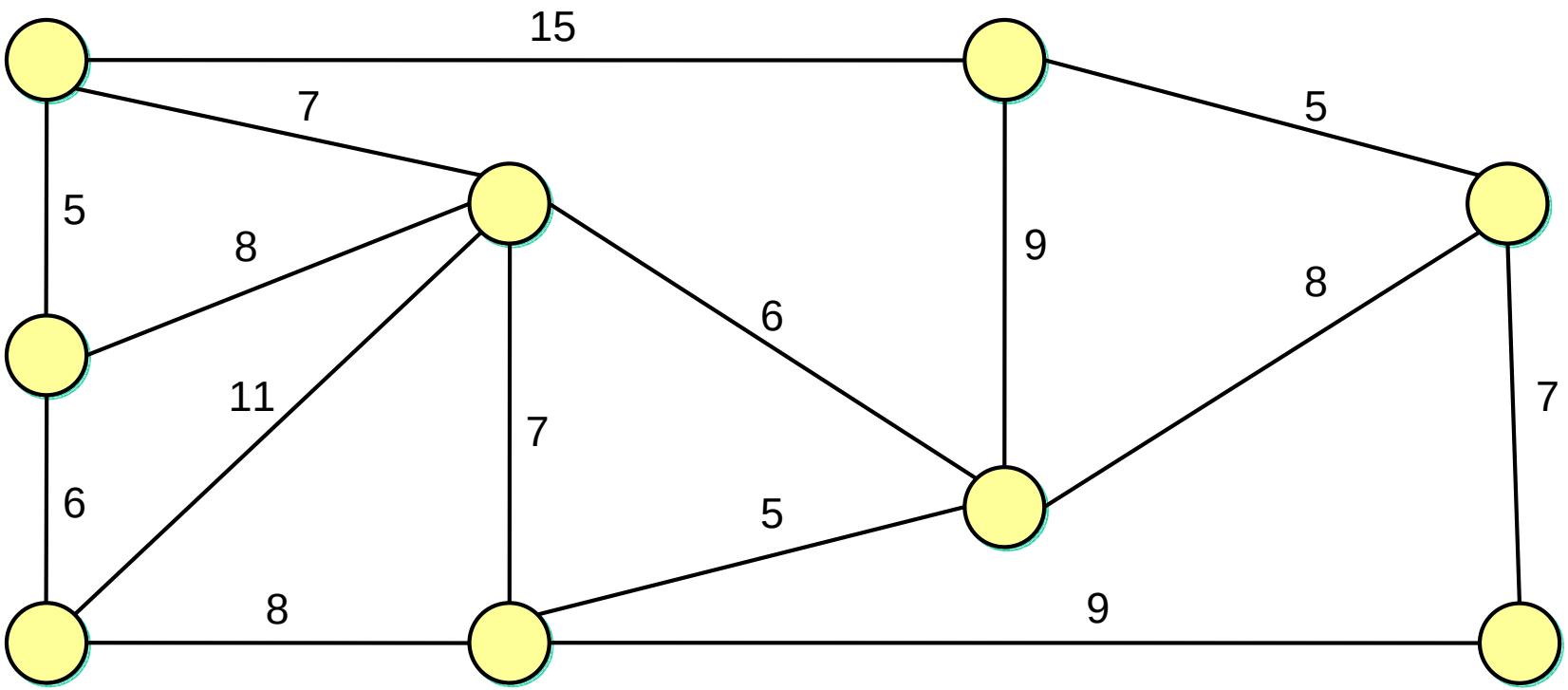
Shortest Path Tree



- *Path cost* is the sum of the costs of all links along a path
- *Shortest Path Tree*:
  - Spanning tree such that the path cost from the root of the tree to any node is the shortest path
  - Shortest path is the path with minimum cost
- Shortest Path Tree is not necessarily a minimum spanning tree!

© 2017 Burkhard Stiller and Daniel Bauer

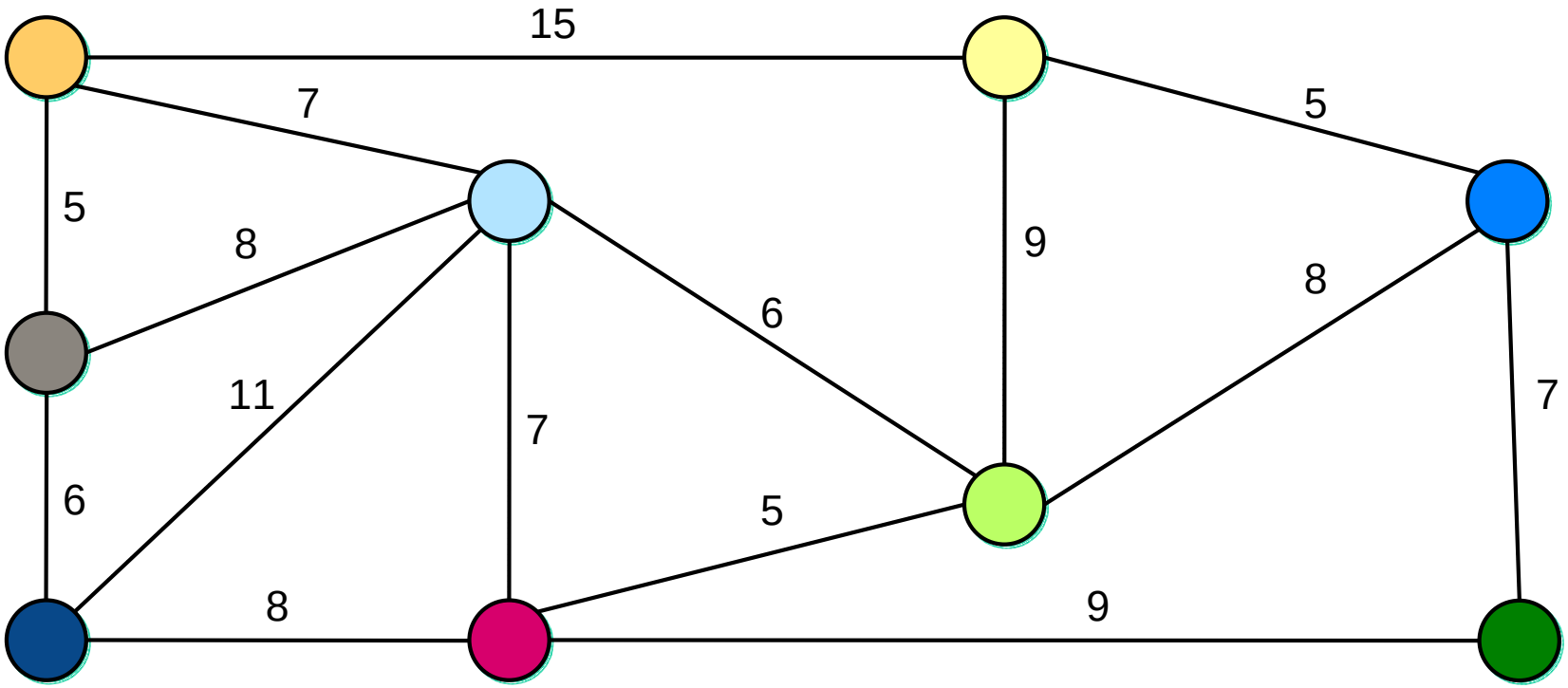
Kruskal‘s MST Algorithm



- Paint nodes in different colors, remove all links
- While not all nodes have the same color
  - Add a link with lowest cost, connecting two nodes of different color
  - Color the connected nodes with the same color

© 2017 Burkhard Stiller and Daniel Bauer

Kruskal‘s MST Algorithm

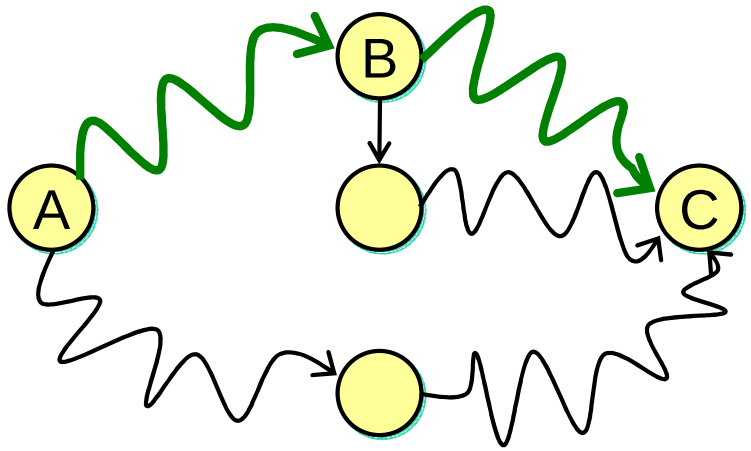


© 2017 Burkhard Stiller and Daniel Bauer

Bellman‘s Principle of Optimality

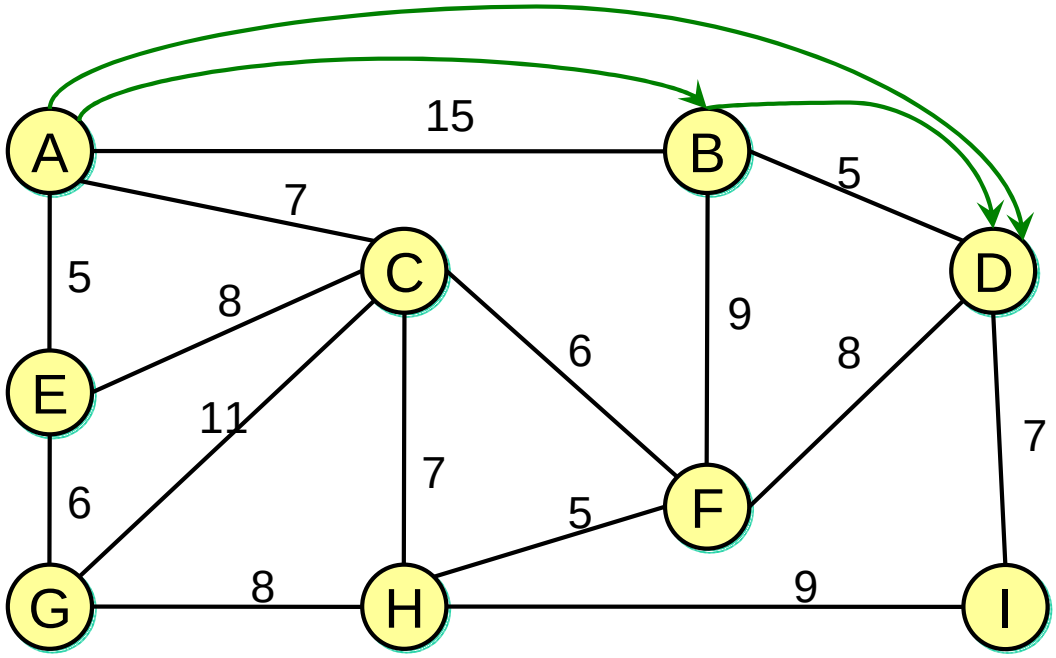
...applied to shortest path computation:

- If a shortest path from A to C contains node B, then the shortest path from B to C is a sub-path of the shortest paths from A to C.
- Proof sketch:
  - Assume the contrary. This results in a contradiction, since a shorter path from A to C can be found.



© 2017 Burkhard Stiller and Daniel Bauer

Floyd-Warshall Shortest Path Algorithm (1)



	A	B	C	D	E	F	G	H	I
A	0	15	7	∞	5	∞	∞	∞	∞
B	15	0	∞	5	∞	9	∞	∞	∞
C	7	∞	0	∞	8	6	11	7	∞
D	∞	5	∞	0	∞	8	∞	∞	7
E	5	∞	8	∞	0	∞	6	∞	∞
F	∞	9	6	8	∞	0	∞	5	∞
G	∞	∞	11	∞	6	∞	0	8	∞
H	∞	∞	7	∞	∞	5	8	0	9
I	∞	∞	∞	7	∞	∞	∞	9	0

- Computes the set of shortest paths from every node to every other node
  - Based on Bellman's Principle of Optimality
- Starts with direct links and inspects longer paths with every iteration
- Uses cost matrix as its main data structure

© 2017 Burkhard Stiller and Daniel Bauer

Floyd-Warshall Shortest Path Algorithm (2)

	A	B	C	D	E	F	G	H	I
A	0	15	7	∞	5	∞	∞	∞	∞
B	15	0	22	5	20	9	∞	∞	∞

	A	B	C	D	E	F	G	H	I
A	0	15	7	20	5	24	∞	∞	∞
B	15	0	22	5	20	9	∞	∞	∞

	A	B	C	D	E	F	G	H	I
A	0	15	7	20	5	13	11	14	23
B	15	0	15	5	20	9	22	14	12
C	7	15	0	14	8	6	11	7	16
D	20	5	14	0	22	8	21	13	7
E	5	20	8	22	0	14	6	14	23
F	13	9	6	8	14	0	13	5	14
G	11	22	11	21	6	13	0	8	17
H	14	14	7	13	14	5	8	0	9
I	23	12	16	7	23	14	17	9	0

```
For k := 1 ... N do
    For u := 1 ... N do
        for v := 1 ... N do
            if(cost[u,k] + cost[k,v] < cost[u,v]) then
                cost[u,v] := cost[u,k] + cost[k,v];
                NextHop[u,v] := NextHop[u,k];
```

© 2017 Burkhard Stiller and Daniel Bauer

Bellman-Ford Algorithm

Initial state for shortest path tree from src == A:

Dest	Cost	Pred.
A	0	-
B	∞	-
C	∞	-
D	∞	-
E	∞	-
F	∞	-
G	∞	-
H	∞	-
I	∞	-

cost[u] = current estimate of cost of shortest path from src to u.

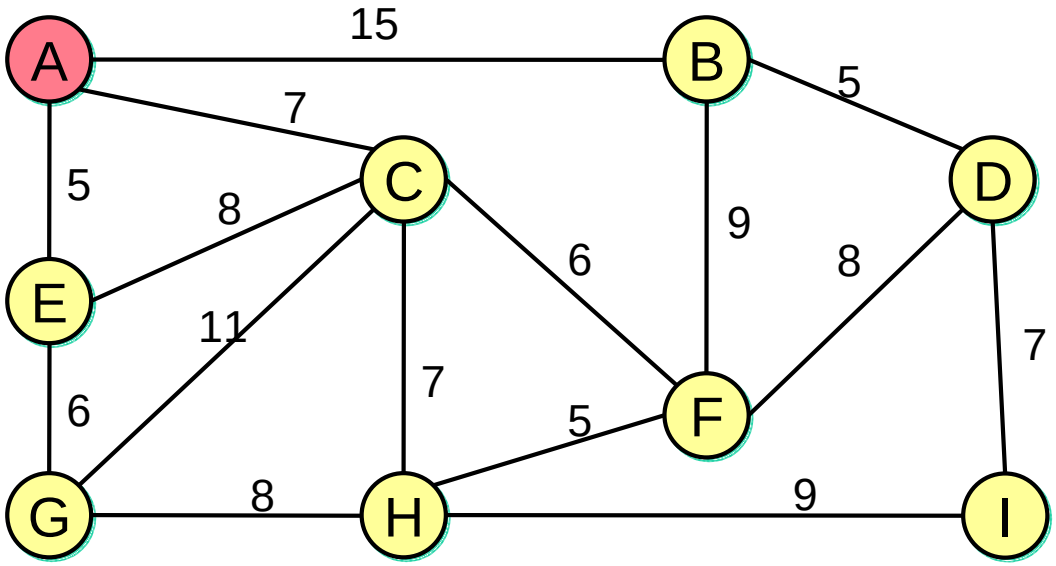
```
cost[] = ∞; cost[src] = 0;
pred[] = NULL;

For k := 1 ... N-1 do
      For all links (u,v) with cost w do
            if(cost[u] + w < cost[v]) then
                  cost[v] := cost[u] + w;
                  pred[v] := u;
```

© 2017 Burkhard Stiller and Daniel Bauer

Bellman-Ford Example Run

	k=0		k=1		k=2		k>=3	
	C	P	C	P	C	P	C	P
A	0	-	0	-	0	-	0	-
B	∞	-	15	A	15	A	15	A
C	∞	-	7	A	7	A	7	A
D	∞	-	∞	-	20	B	20	B
E	∞	-	5	A	5	A	5	A
F	∞	-	∞	-	13	C	13	C
G	∞	-	∞	-	11	E	11	E
H	∞	-	∞	-	14	C	14	C
I	∞	-	∞	-	∞	-	23	H



© 2017 Burkhard Stiller and Daniel Bauer

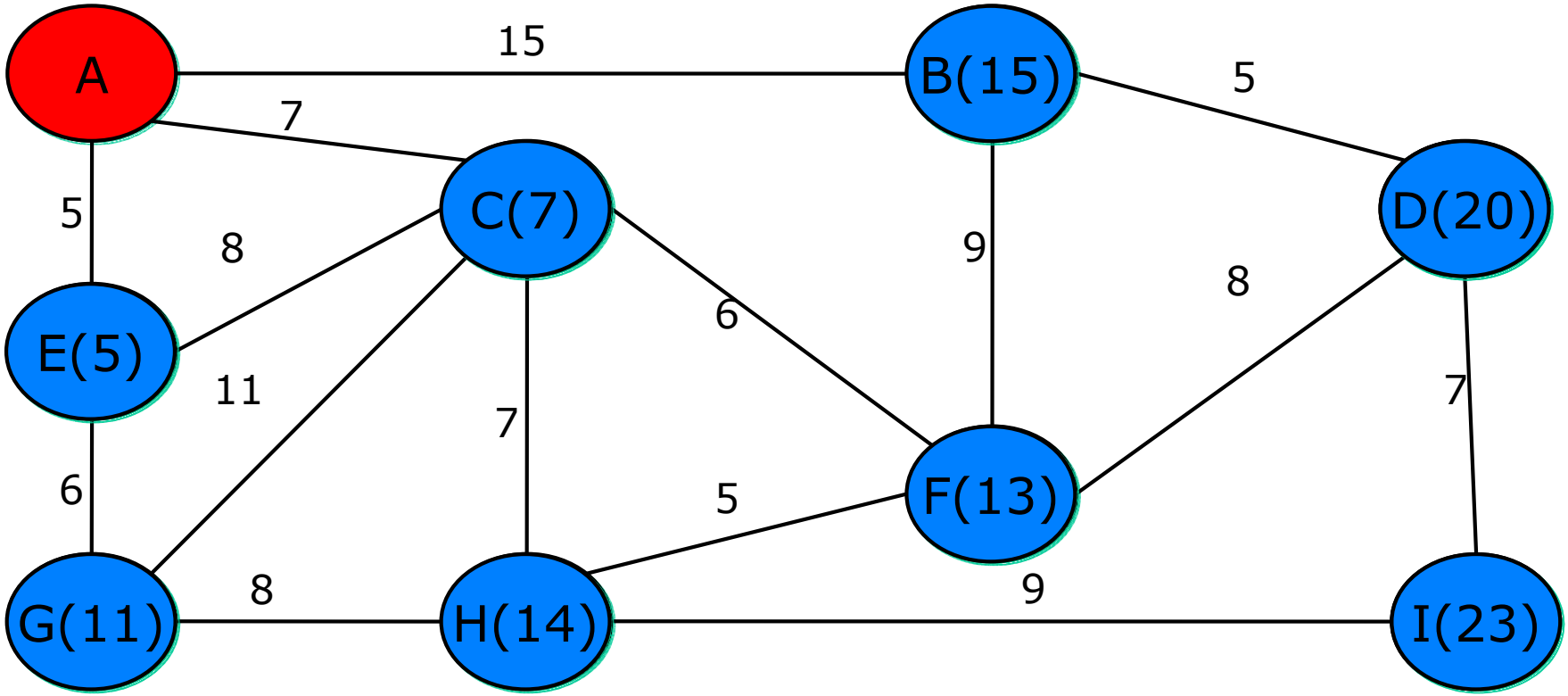
Dijkstra's Shortest-Path Algorithm

- „Greedy" algorithm that explores the network node-by-node.
- Data structures: cost from the source, set of unvisited nodes, previous node

```
cost[] = ∞; cost[source] = 0;
Q := set of all Nodes (that are not yet visited);
While (Q is not empty) do
  u := node in Q with minimal cost;
  remove node u from Q;
  For each link(u,v) that starts in u with cost w do
            if(cost[u] + w < cost[v]) then
                  cost[v] := cost[u] + w;
                  pred[v] := u;
```

© 2017 Burkhard Stiller and Daniel Bauer

Dijkstra‘s – Example Run



How are these computations now embedded into routing protocols?

© 2017 Burkhard Stiller and Daniel Bauer

Distance Vector Routing (1)

- Each router maintains a set of triples:

  `(Destination, Cost, NextHop)`

- Each node sends updates to (and receives updates from) its directly connected neighbors:
  - Periodically on the order of several seconds
  - Whenever its table changes (called triggered update)
- Each update is a distance vector:

  `(Destination, Cost)`

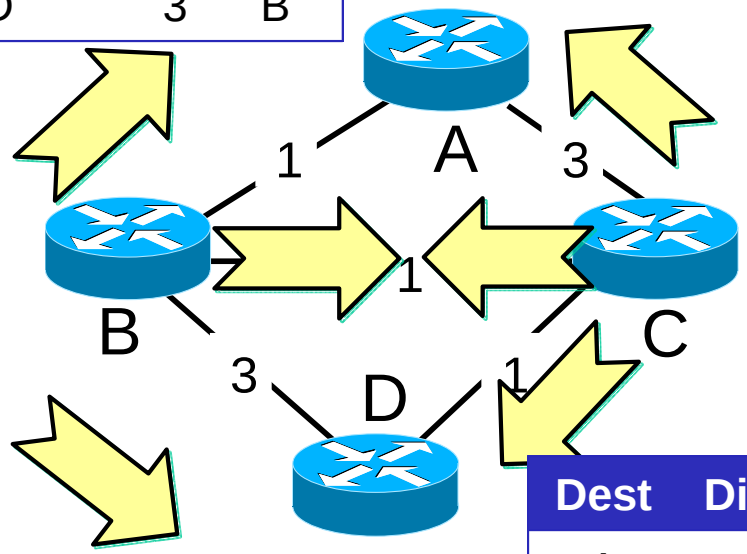
- Update local table if a “better” route is received:
  - Smaller cost or coming from next hop
- Refresh existing routes, delete if they time out

© 2017 Burkhard Stiller and Daniel Bauer

Distance Vector Routing (2)

Dest	Dist	Port
A	0	-
B	1	B
C	2	B
D	3	B

Dest	Dist	Port
A	1	A
B	0	-
C	1	C
D	2	C



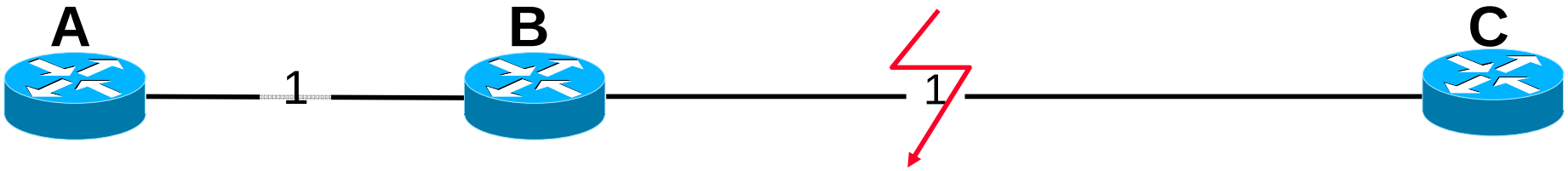
Dest	Dist	Port
A	2	B
B	1	B
C	0	-
D	1	D

Dest	Dist	Port
A	3	C
B	2	C
C	1	C
D	0	-

© 2017 Burkhard Stiller and Daniel Bauer

Problems with Distance Vector

- Long convergence time in case of errors
  "count to infinity problem"



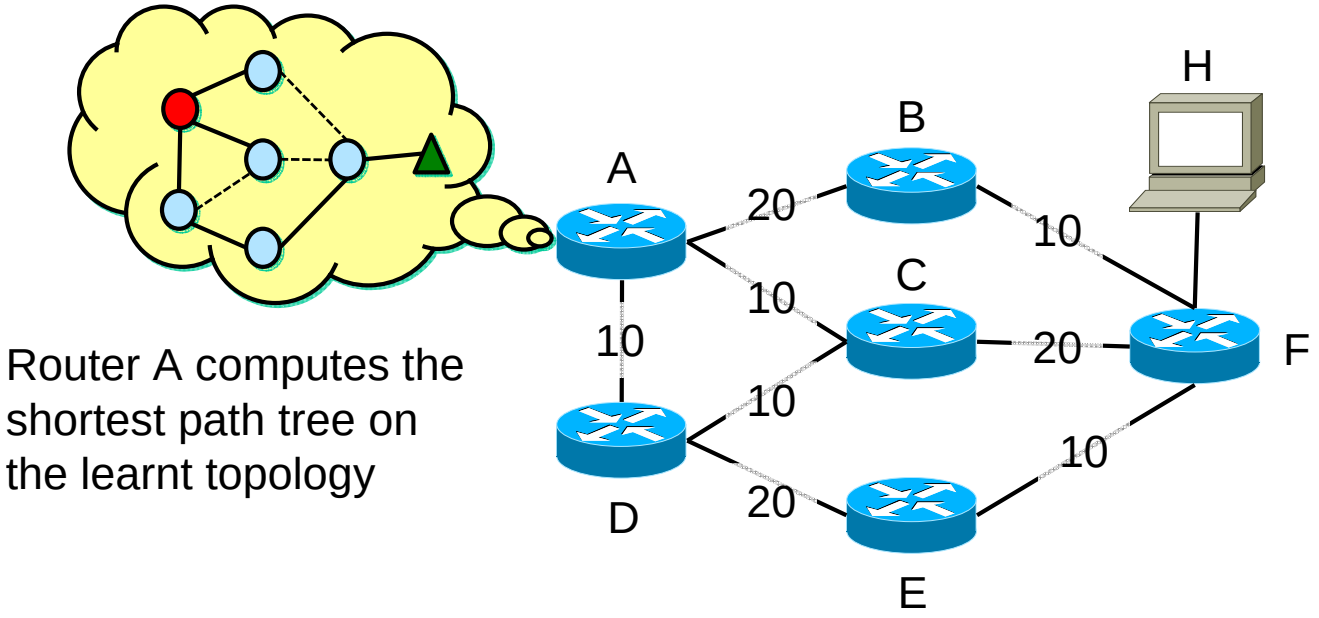
Dest.	Port/Dist	Dest.	via	via	Port/Dist
B	B/1	A	A/1	-	A/1
C	**B/6**	C	**A/5**	-	**A/5**

C declared unreachable, if a threshold ("infinity") is reached.
The larger the threshold, the slower the convergence.
The smaller the threshold, the smaller the network diameter.

© 2017 Burkhard Stiller and Daniel Bauer

Link State Routing (1)

- Routers distribute their local view (the “link-state”) to all other routers
- Each router maintains a complete view of the topology



Router A computes the shortest path tree on the learnt topology

Link state packet of F that is flooded through the network

Link-state from F	
Neighbor	**Cost**
B	10
C	20
E	10
Reachable	**Cost**
H	5

© 2017 Burkhard Stiller and Daniel Bauer

Link State Routing (2)

- Algorithm:
  - Detect neighbors: broadcast HELLO packets
  - Measurement of latency or costs to each neighbor: ECHO packet RTT (Round Trip Time)
  - Generation of a “Link State Packet” including:
    - Sender, list of neighbors and latencies, timestamp
    - Periodic or even-driven (*e.g.*, new neighbor or failure) generation
  - Distribution of “Link State Packet” to all neighbors:
    - Flooding with refinements such as discarding of duplicates or discarding of information that is too old
  - Failure detection: No more HELLOs from neighbor
  - Calculation of shortest path to all routers according to Dijkstra’s algorithm

© 2017 Burkhard Stiller and Daniel Bauer

Link State Routing – Discussion

- ✓ Fast convergence in case of changes in the network
  - A single link-state update is sufficient
- ✓ Doesn't suffer from the count-to-infinity problem
- ✓ High stability since every router has a complete view of the network

- ✘ Topology database consumes more memory than a simple distance vector
- ✘ More complex to implement and to configure than distance vector

© 2017 Burkhard Stiller and Daniel Bauer

Link Metrics

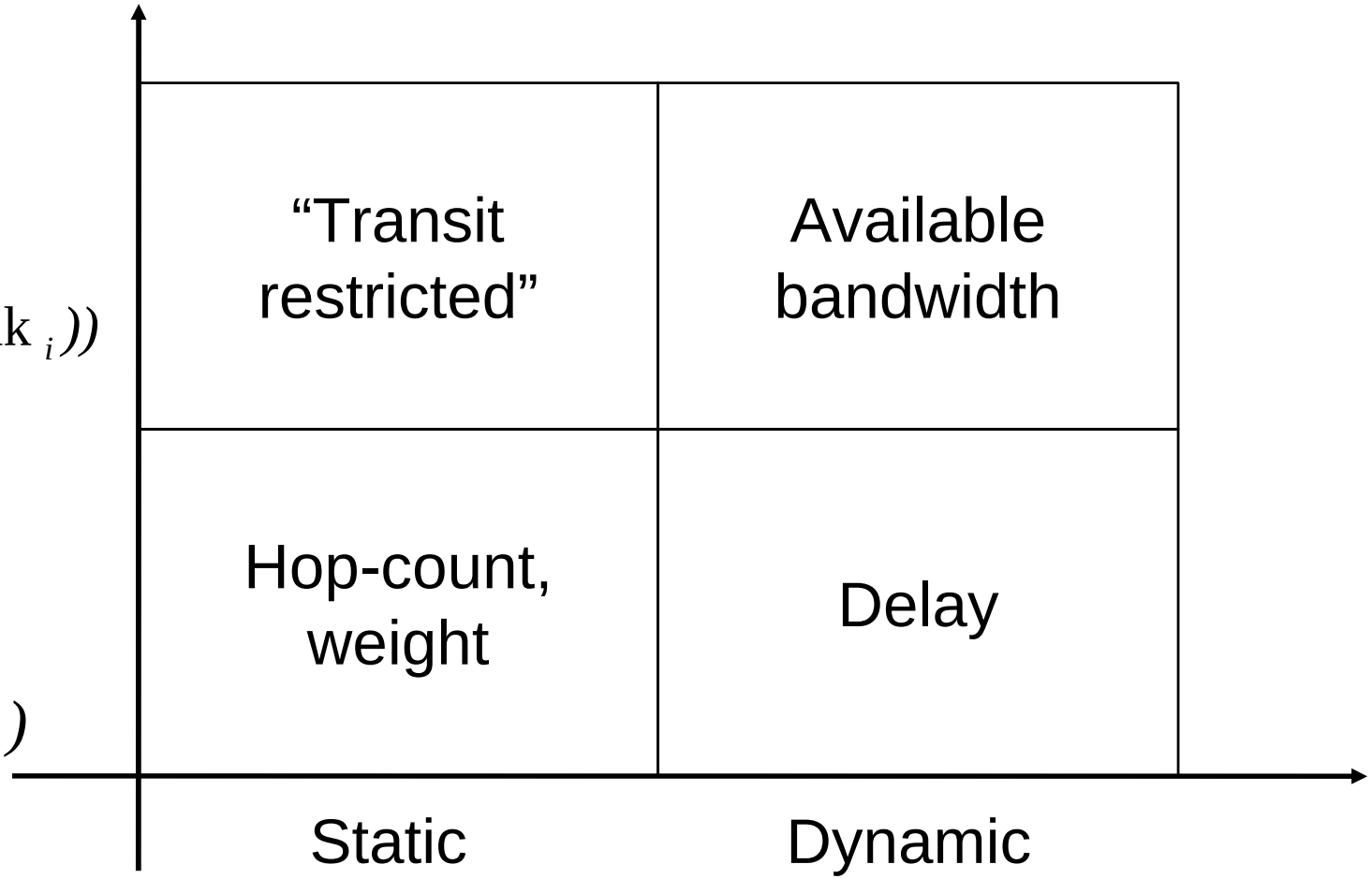
- Paths are computed based on “metrics”:

Restrictive:

$$\text{metric}(\text{path}) = Min(\text{metric}(\text{link}_i))$$

Additive:

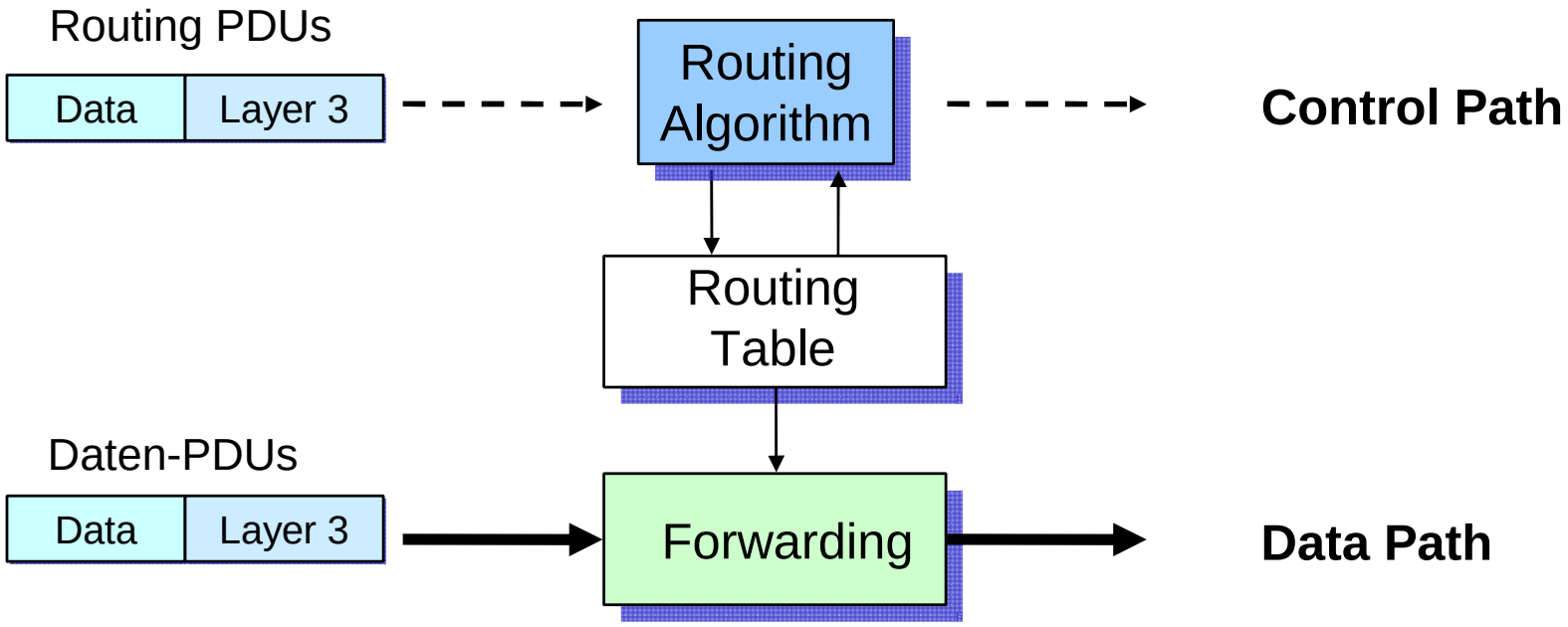
$$\text{metric}(\text{path}) = \sum \text{metric}(\text{link}_i)$$



© 2017 Burkhard Stiller and Daniel Bauer

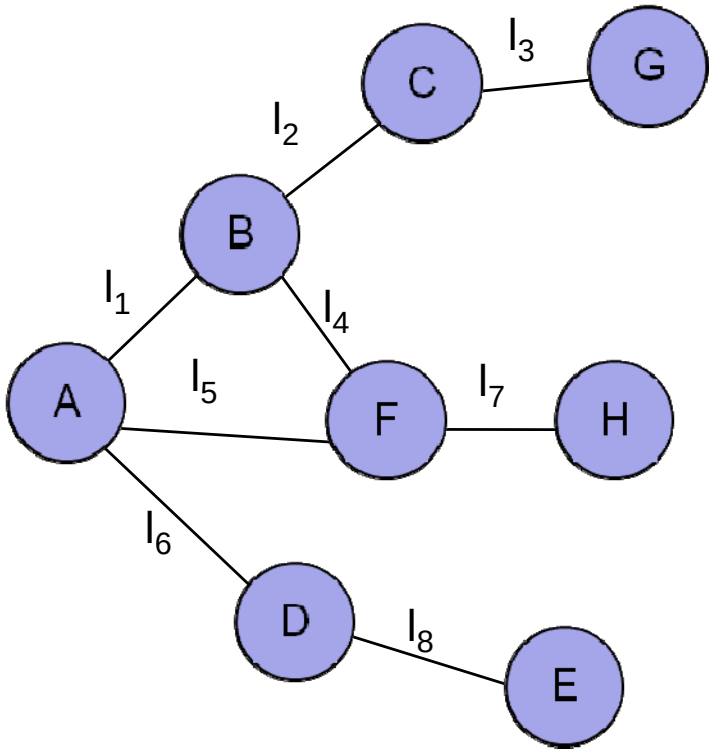
Routing Table and Forwarding

- Routing Algorithm re-computes the routing table whenever there is a change (for example, a link failure)
- Forwarding process consults routing table to decide where a datagram should be forwarded



© 2017 Burkhard Stiller and Daniel Bauer

Routing Table



Routing table in Router B

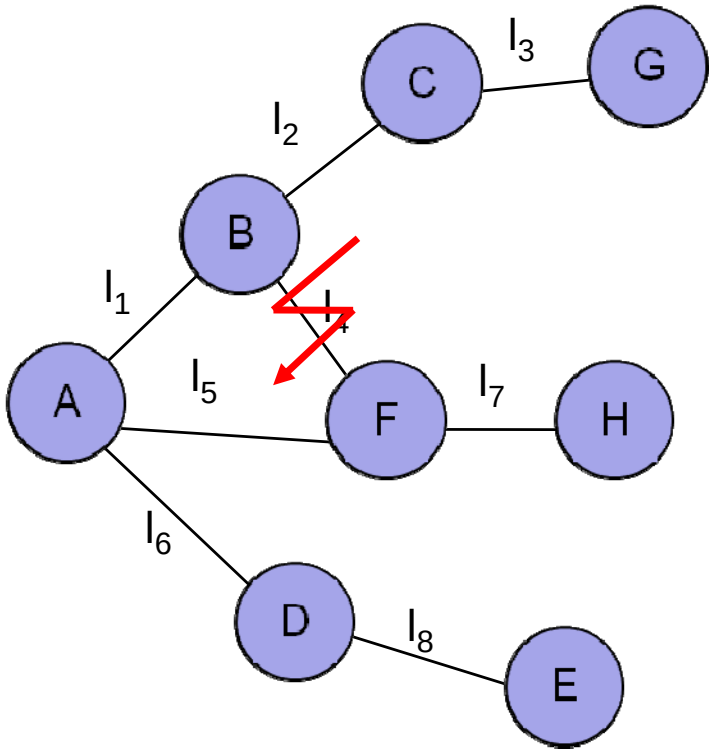
to	via
A	- (l_1)
C	- (l_2)
D	A (l_1)
E	A (l_1)
F	- (l_4)
G	C (l_2)
H	F (l_4)

A-H: Router

l_1-l_8: Incoming/outgoing links

© 2017 Burkhard Stiller and Daniel Bauer

Routing Table — Link Failure



A-H: Router

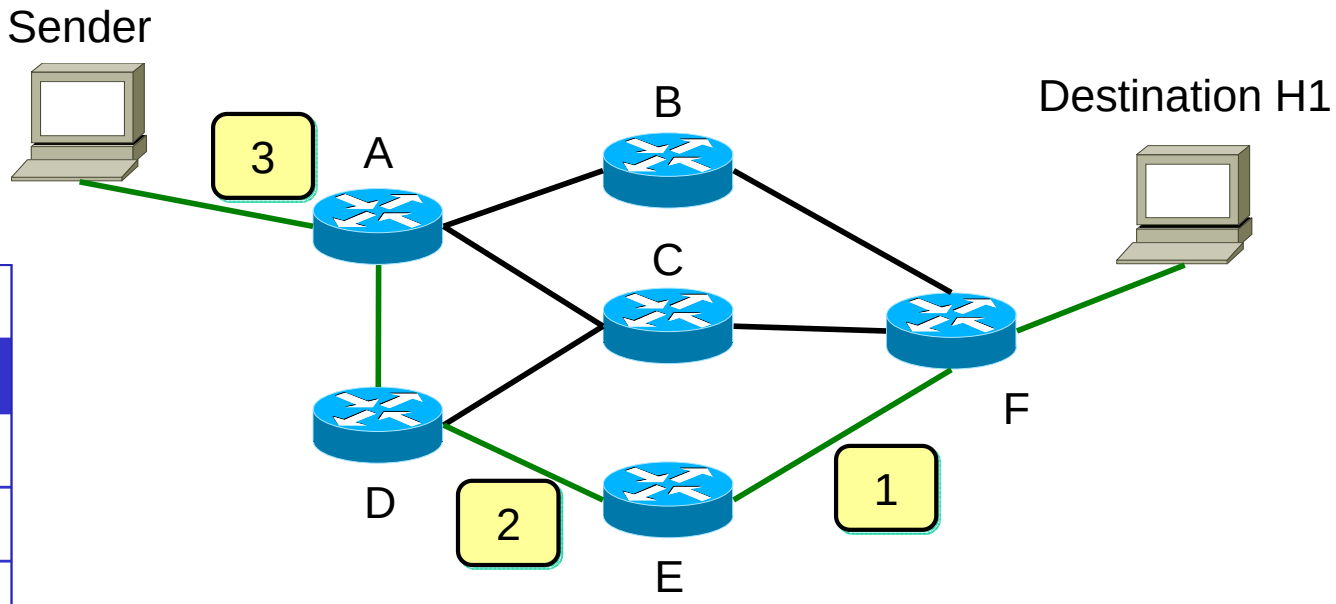
l_1-l_8: Incoming/outgoing links

Routing table in Router B

to	via
A	- (l_1)
C	- (l_2)
D	A (l_1)
E	A (l_1)
F	A (l_1)
G	C (l_2)
H	A (l_1)

© 2017 Burkhard Stiller and Daniel Bauer

Routing for a Datagram Service

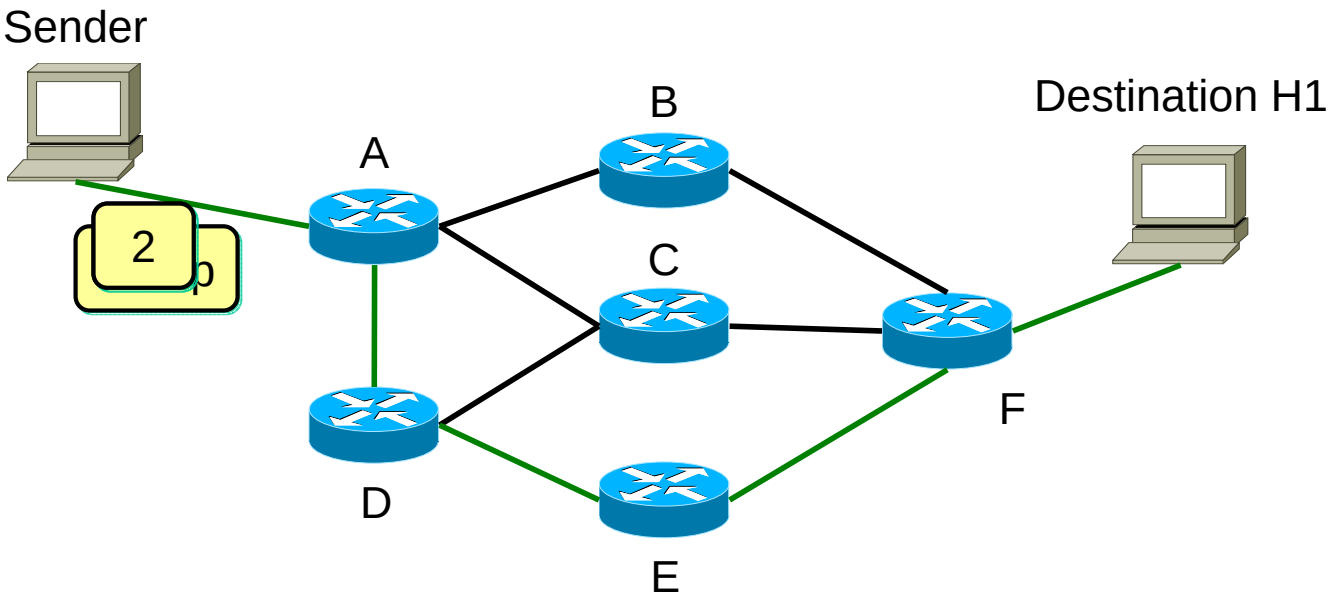


Routing table A	
Neighbor	**Next Hop**
B	B
C	C
D	D
E	D
F	B
H	B

© 2017 Burkhard Stiller and Daniel Bauer

Routing for a Virtual Connection Service

Routing table A	
Neighbor	**Next Hop**
B	B
C	C
D	D
E	D
F	B
H	B



VC Table A	
VCI	**Next**
1	D

VC Table D	
VCI	**Next**
1	E

VC Table E	
VCI	**Next**
1	F

VC Table F	
VCI	**Next**
1	-

VC: Virtual Connection
VCI: VC Identifyer

© 2017 Burkhard Stiller and Daniel Bauer

Inter-domain …

Scalability Issues in the Internet

- Scalability in the number of participants major concern of Internet design
- Scalability in the number of sub-networks integrated is another major concerns of Internet's routing protocols

- Design of IP and the Internet evolved to tackle scalability
  - Efficient use of IP addresses with flexible sub-netting
    - Class-less Inter-Domain Routing (as seen before)
  - Hierarchical and autonomous routing domains with decentralized control
  - Introduction of IPv6 that enlarges the address space

© 2017 Burkhard Stiller and Daniel Bauer

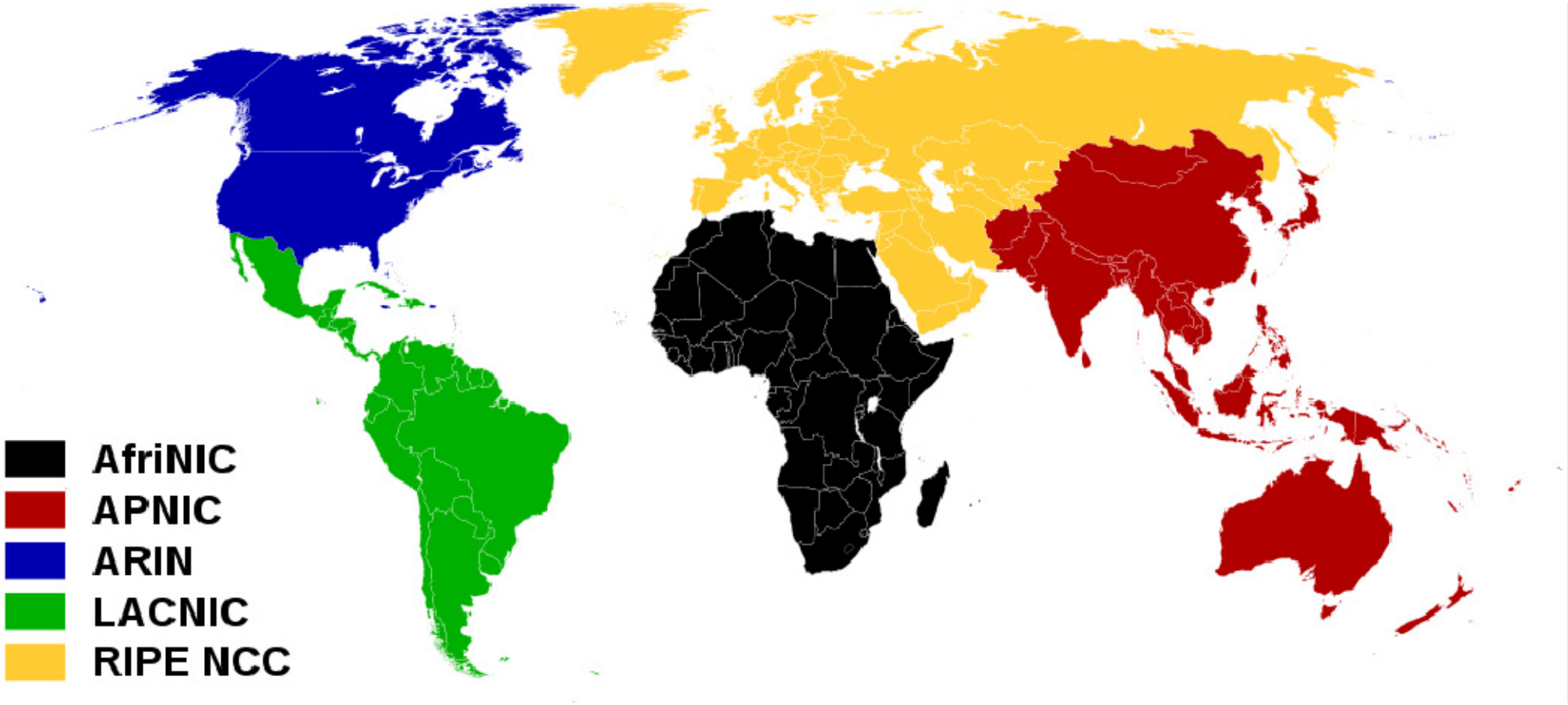
World Internet Usage – June 30, 2017

World Regions	Population (2017 Est.)	Population % of World	Internet Users 30 June 2017	Penetration Rate (% Pop.)	Growth 2000-2017	Internet Users %
Africa	1,246,504,865	16.6 %	388,376,491	31.2 %	8,503.1%	10.0 %
Asia	4,148,177,672	55.2 %	1,938,075,631	46.7 %	1,595.5%	49.7 %
Europe	822,710,362	10.9 %	659,634,487	80.2 %	527.6%	17.0 %
Latin America / Caribbean	647,604,645	8.6 %	404,269,163	62.4 %	2,137.4%	10.4 %
Middle East	250,327,574	3.3 %	146,972,123	58.7 %	4,374.3%	3.8 %
North America	363,224,006	4.8 %	320,059,368	88.1 %	196.1%	8.2 %
Oceania / Australia	40,479,846	0.5 %	28,180,356	69.6 %	269.8%	0.7 %
WORLD TOTAL	7,519,028,970	100.0 %	3,885,567,619	51.7 %	976.4%	100.0 %

http://internetworldstats.com/stats.htm

© 2017 Burkhard Stiller and Daniel Bauer

Regional Internet Registries



NIC: Network Information Center or Domain Name Registry

wikipedia.org

© 2017 Burkhard Stiller and Daniel Bauer

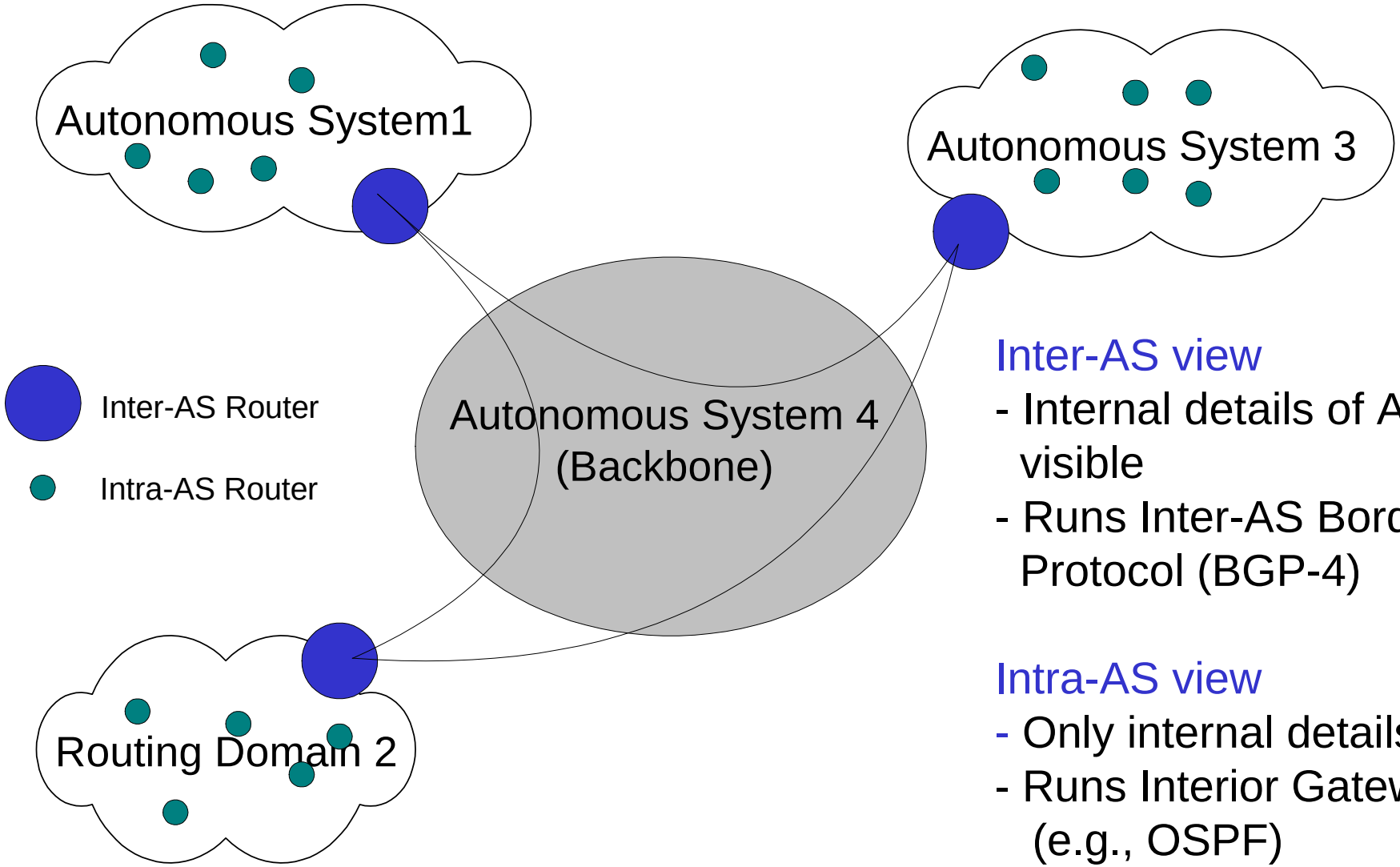
ifi

Autonomous System (AS)

- An AS is a routing domain and corresponds to an administrative domain showing a unique number (ID)
  - Examples: University, company, backbone network
  - Classification of AS's into three types:
    - Stub AS:
      - AS that has only a single connection to one other AS
      - Such an AS will only carry local traffic
    - Multi-homed AS:
      - AS connected to more than one other AS
      - AS refuses to carry transit traffic
    - Transit AS (routing backbone):
      - AS that has connections to more than one other AS
      - Designed to carry both transit and local traffic

Example:
AS559 SWITCH ,CH
http://bgp.potaroo.net/cidr/autnums.html

Hierarchical Routing



Inter-AS view
- Internal details of AS‘s are not visible
- Runs Inter-AS Border Gateway Protocol (BGP-4)

Intra-AS view
- Only internal details are visible.
- Runs Interior Gateway Protocols (e.g., OSPF)

© 2017 Burkhard Stiller and Daniel Bauer

BGP-4 (RFC 1771)

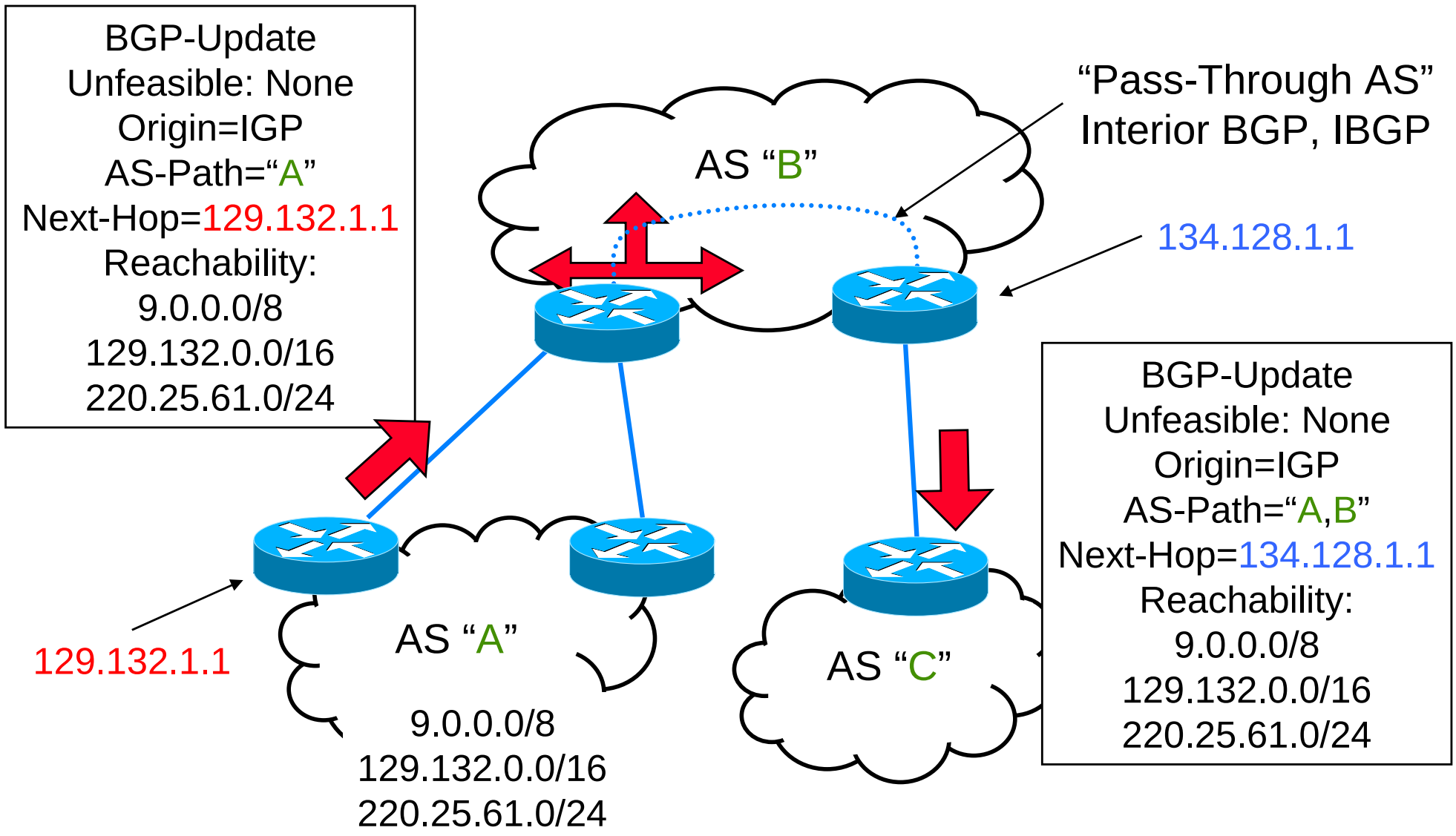
- BGP is the *recommended* inter-AS routing protocol.
- BGP is a *path vector* protocol defined by the IETF.
  - Path vector is a *distance-vector* protocol with loop protection
  - It distributes reachability information including the complete path of traversed autonomous systems.
  - Instead of a single metric, a set of attributes is used:

    AS path, origin of route, next hop towards destination,

    multi-exit discrimination in case of parallel paths into an AS, …
  - After the initial database exchange, only updates are sent.
  - BGP uses TCP for reliable transport of routing updates.

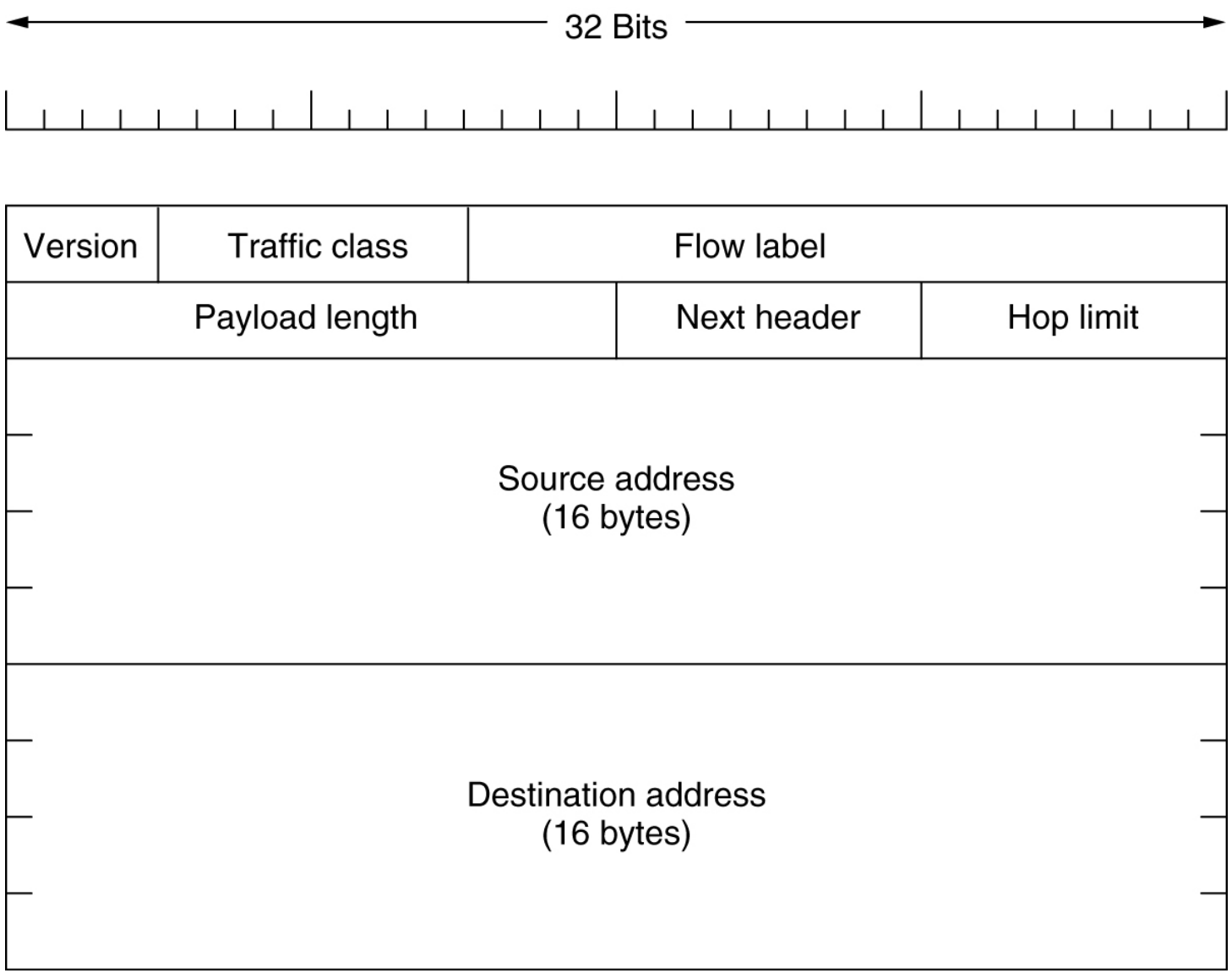
© 2017 Burkhard Stiller and Daniel Bauer

BGP-4 Example



© 2017 Burkhard Stiller and Daniel Bauer

IPv6



© 2017 Burkhard Stiller and Daniel Bauer

11. End-to-end Protocols

End-to-end Characteristics

UDP and TCP

Congestion Control

Implementations and Sockets

© 2017 Burkhard Stiller

End … Hop-by-hop vs. End-to-end Services

- Underlying best-effort network:
  - Drops messages
  - Re-orders messages
  - Delivers duplicate copies of a given message
  - Limits messages to some finite size
  - Delivers messages after an arbitrarily long delay
- Common end-to-end services:
  - Guaranteed message delivery
  - Deliver messages in the same order they are sent
  - Deliver at most one copy of each message
  - Support arbitrarily large messages
  - Allow the receiver to apply flow control to the sender
  - Support multiple application processes on each host

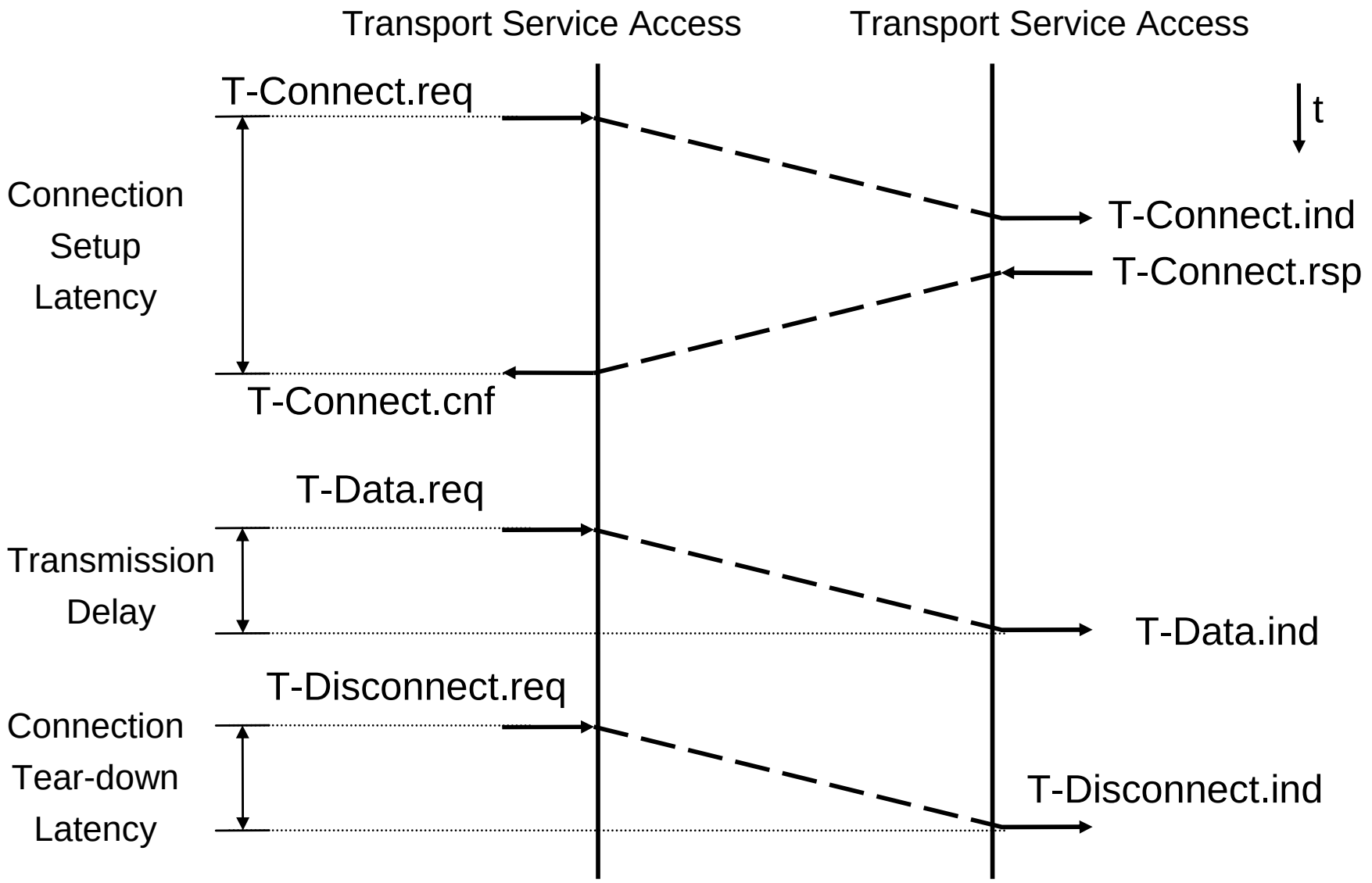
© 2017 Burkhard Stiller

Transport vs. Link Layer Protocols

Link Layer Protocols	Transport Protocols
Entities are connected via a single point-to-point link	Entities do not reside in neighbor systems
Mostly fixed and small Round Trip Times (RTT)	Mainly variable and much larger RTTs, leading to a larger bandwidth-delay product
No change of packet sequence	Ordered delivery not guaranteed
Link limits sending rate	Possible bottleneck may exist on any intermediate link

© 2017 Burkhard Stiller

Transport Layer Latencies

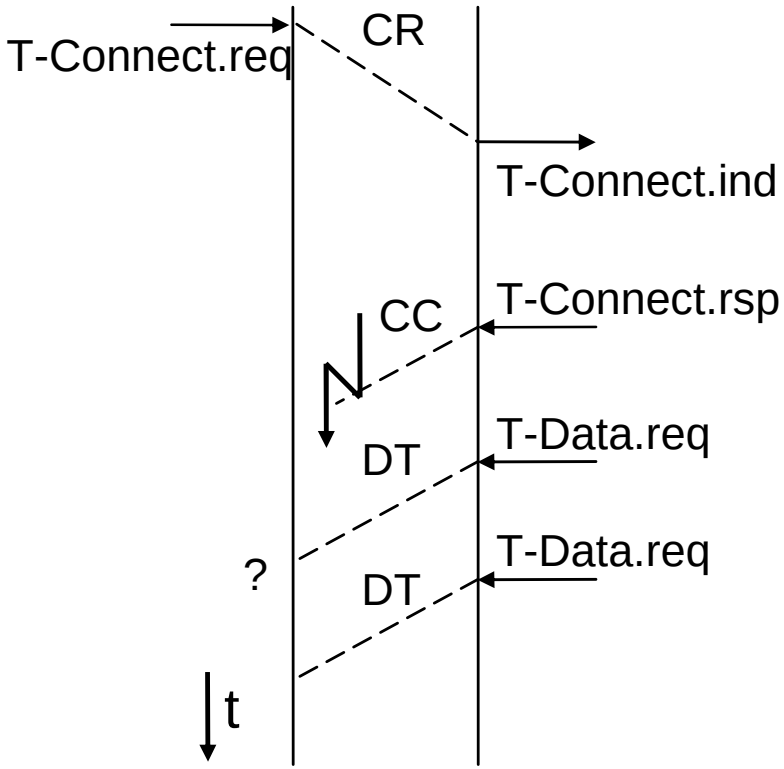


© 2017 Burkhard Stiller

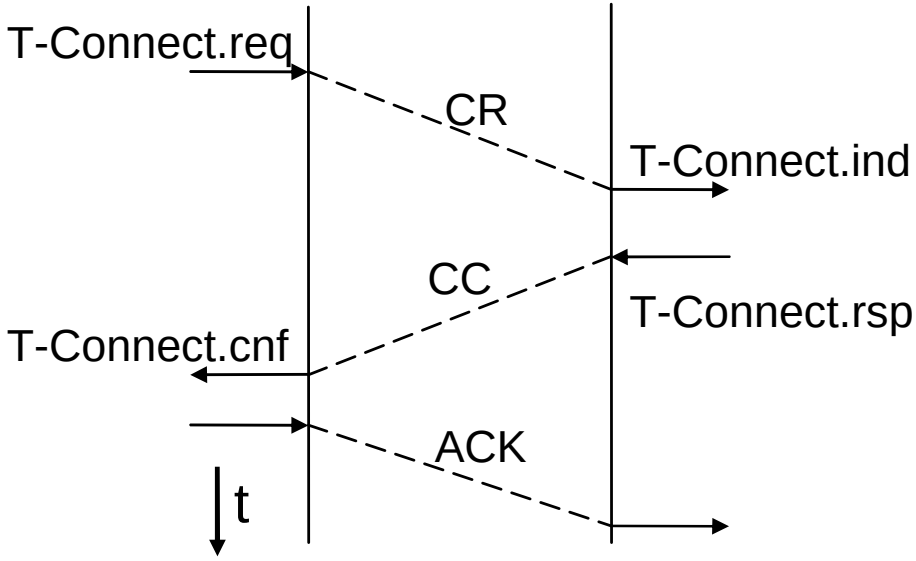
ifi

Transport Layer Connection Reliability

- Loss of a connect request:



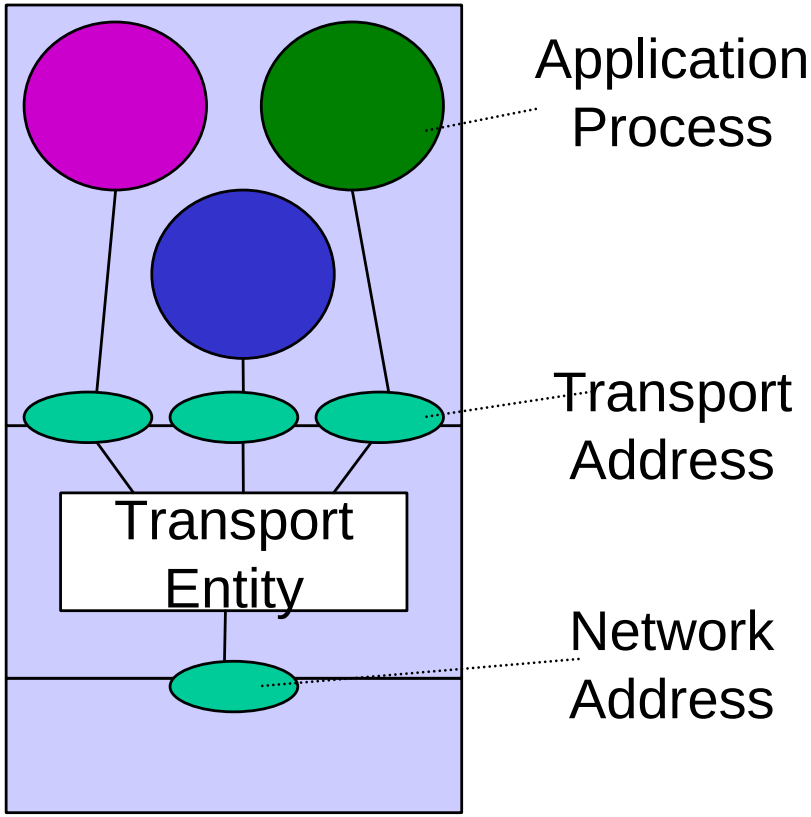
- 3-way-handshake:
  - Connection established as soon as CR and CC have been acknowledged
  - Requires an additional message



© 2017 Burkhard Stiller

Internet Transport Layer Addressing (1)

- Goal: Communication between application processes
- Process ID of operating system can not be used for unique identification of processes



- Full **transport address** equals:
  - IP address ***and*** port number
  - Globally unique
- Port equals an end-point of a communication (OSI CEP)

© 2017 Burkhard Stiller

Internet Transport Layer Addressing (2)

- Identification of transport layer services, in particular the TCP protocol, by ports (cf. SAPs/CEP)
- Reserved port numbers for 1024 frequently used services (well-known ports): http://www.iana.org/assignments/port-numbers
  - 13: NTP (Network Time Protocol)
  - 20: FTP (File Transfer Protocol) Data
  - 21: FTP Control data
  - 25: SMTP (Simple Mail Transfer Protocol)
  - 53: DNS (Domain Name Service)
  - 80: HTTP (Hyper Text Transfer Protocol)
  - 119: NNTP (Network News Transfer Protocol)

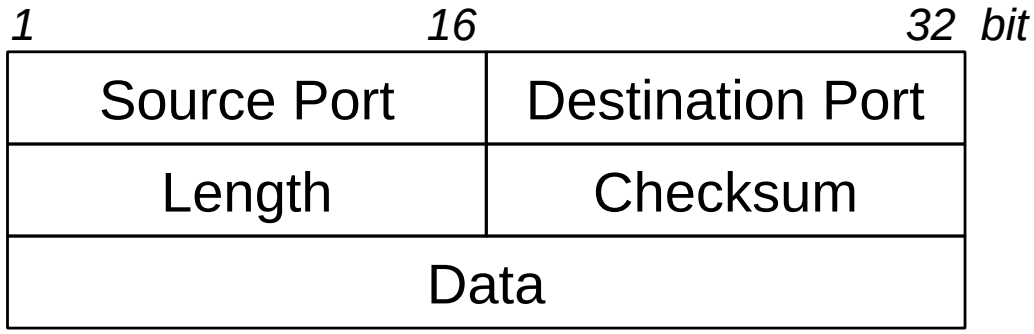
```
> telnet walapai 13
Trying 130.90.1.121...
Connected to host123.
Escape character is '^]'.
Mon Aug  5 16:57:19 2002
Connection closed by foreign
host
```

© 2017 Burkhard Stiller

User Datagram Protocol (UDP)

- Connection-less
  - Unreliable and unordered datagram service
- Message-based, including multicasts
  - Adds simple multiplexing
  - No flow control
- Endpoints identified by ports:
  - Servers have *well-known* ports
  - See `/etc/services` on Unix
- Optional checksum:
  - Error control
    - Utilizes IPv4 checksum
  - No error correction

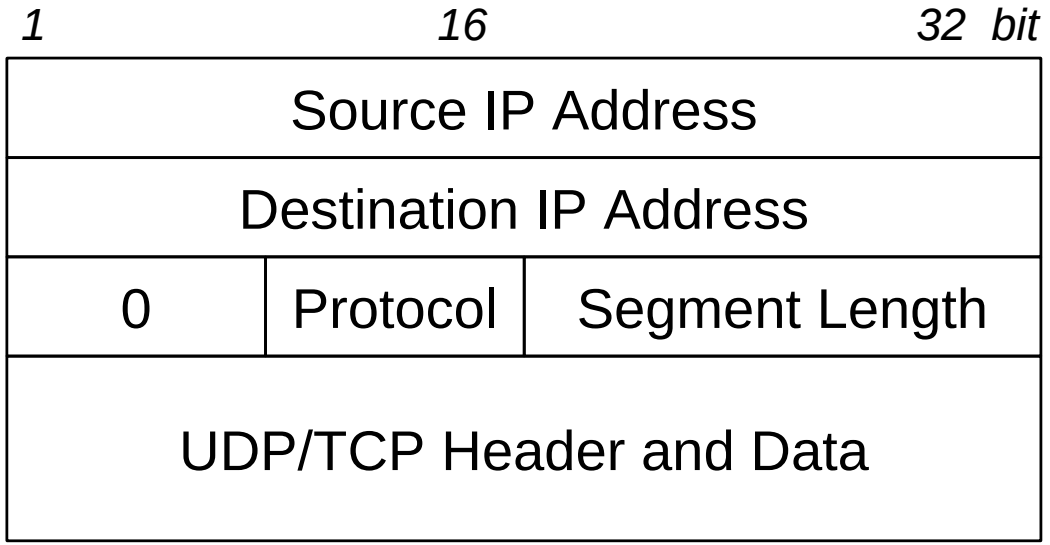
Header format



© 2017 Burkhard Stiller

UDP Checksumming

- Definition of a pseudo header (ph), here IPv4:



Protocol:
UDP with ID 17

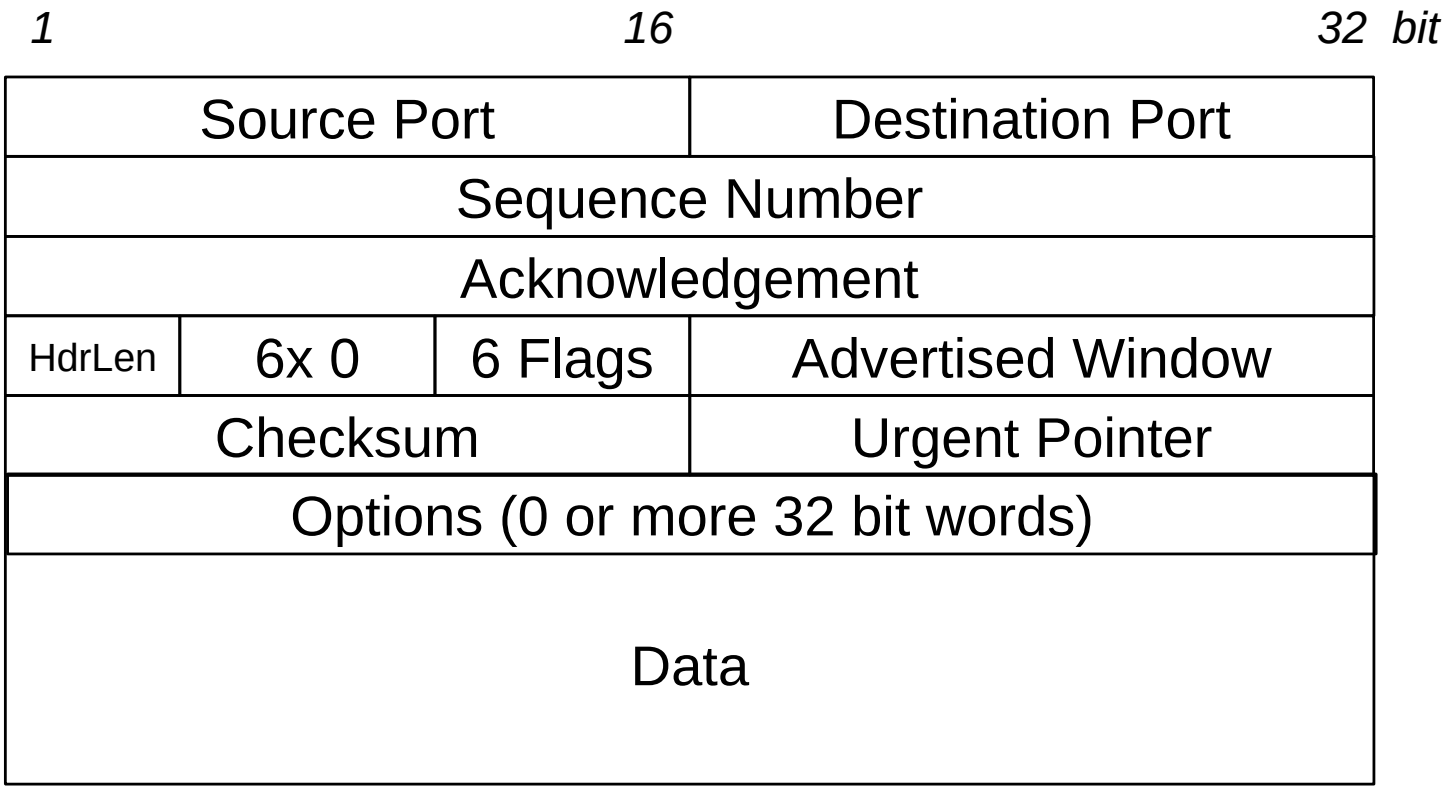
- Calculation of checksum only across:
  - Pseudo header and real UDP header as well as data
- Pseudo header is NOT transmitted, just for calc. only!
- Reliably securing UDP protocol data and detecting wrong destination delivery (due to Destination IP in ph)

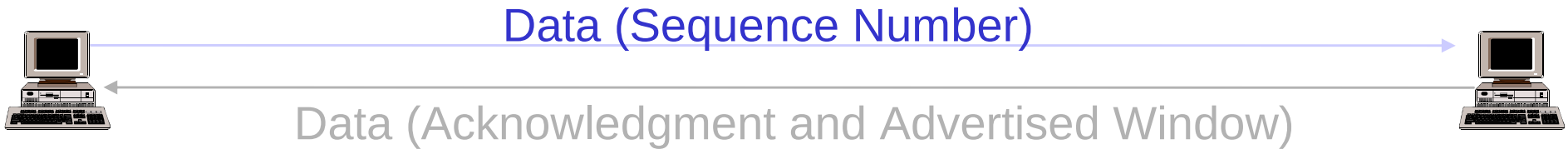
© 2017 Burkhard Stiller

Transmission Control Protocol (TCP)

- Connection-oriented and reliable
  - Between two sockets (connection) in full duplex mode
  - Reliable: three way handshake, ordered delivery
  - Error control: Sequence numbering, checksum, acknowledgements and numbers, retransmissions
  - Flow control: keep sender from overrunning receiver
  - Congestion control: keep sender from overrunning network
- Byte-stream driven data transfer
  - Sending process writes some number of bytes
  - TCP breaks into *segments* (TCP packet) and sends via IP
    - Segment size determined according to MTU (Maximum Transmission Unit) of local link
  - Receiving process reads some number of bytes

TCP Header (Segment Format) (1)





© 2017 Burkhard Stiller

TCP Header (Segment Format) (2)

- Source and destination port:
  - Part of the transport layer address
- Sequence number and acknowledgement:
  - Sending and receiving sequence number
- TCP header length (HdrLen):
  - Number of 32 bit words in header
- Advertised window:
  - Sending window of flow control
- Urgent pointer:
  - Pointer to important data
- Flags:
  - URG:
    - Set to 1 in case the Urgent Pointer is used
  - SYN:
    - Used at connection set-up
  - ACK:
    - Signals the validity of the current acknowledgement field in header
  - FIN:
    - Information that sender will not sent further data
  - RST:
    - Restart of a connection
  - EOM or PSH:
    - Information that the end of the message has been reached (End of Message or Push Flag)

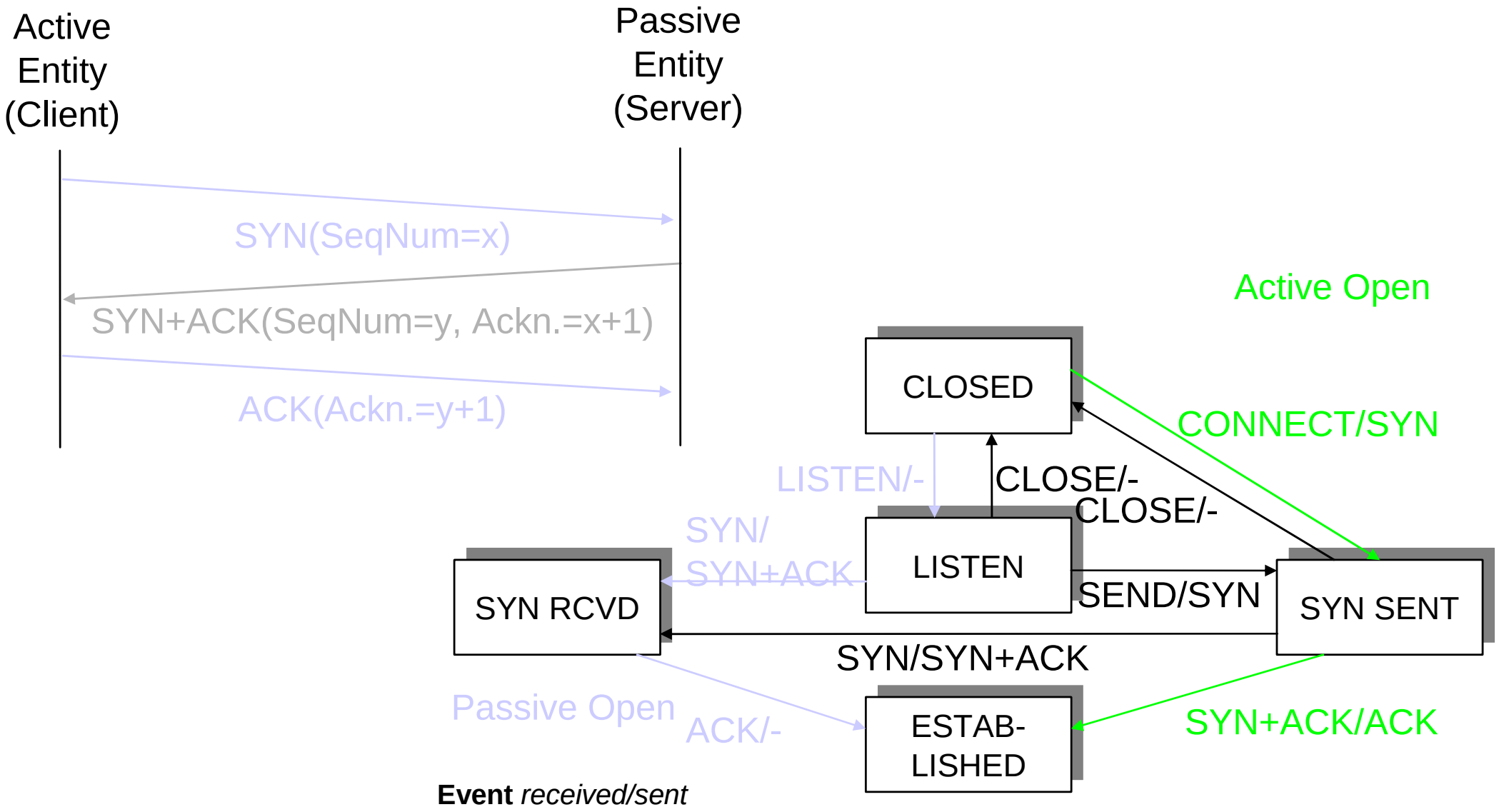
© 2017 Burkhard Stiller

TCP Connection Setup (1)

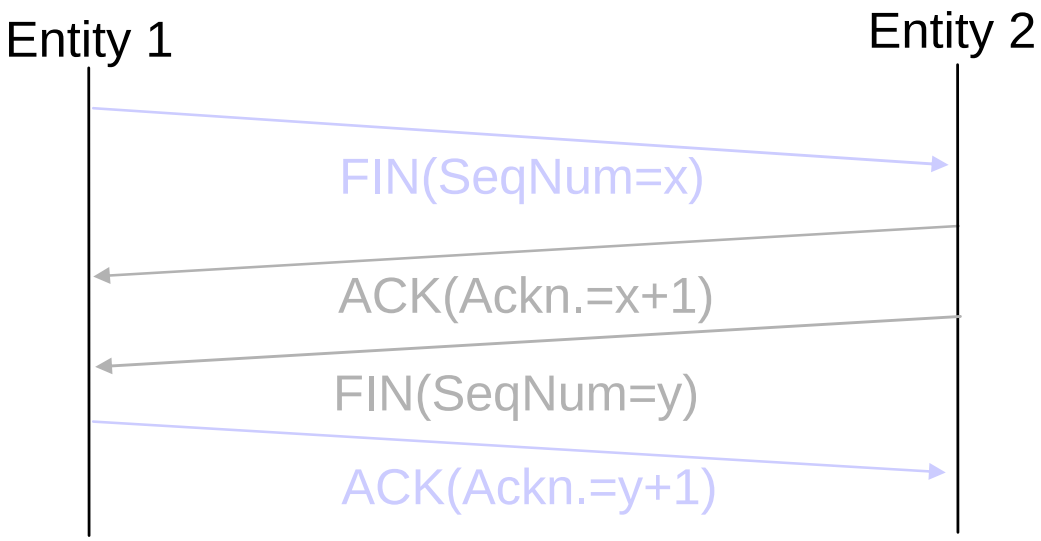
- Opening of a socket:
  - Active mode:
    - Request of TCP connection from a specific socket (connect)
  - Passive mode:
    - User informs TCP being able to accept an incoming connection (listen/accept)
      - Option (accept):
        - Well-known socket to accept all incoming connections
- Functional view:
  - 3-way-handshake
  - Exchange of initial sequence numbers:
    - Avoids conflicts with ”old“ connections, having used the same port, and remaining ”old“ packets within the network

© 2017 Burkhard Stiller

TCP Connection Setup (2)

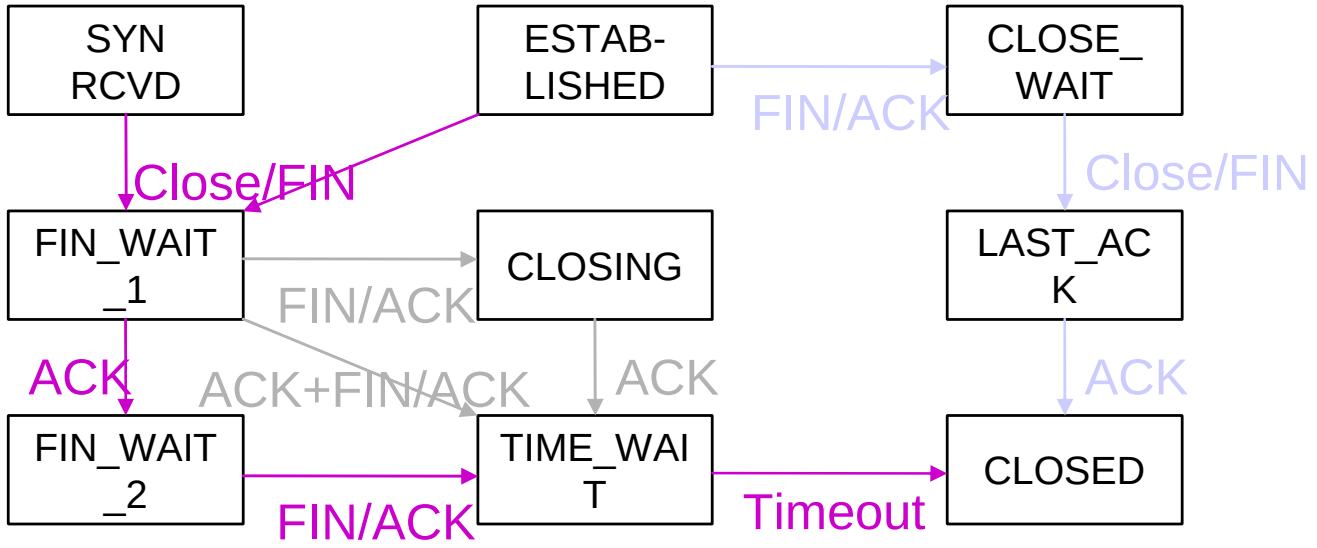


TCP Connection Teardown



- Situations:
  - Receiver closes connection first
  - Sender closes connection first
  - Sender and receiver close connection at the same time

- Both entities close the connection
- All data being sent will reach the receiver



MSL: Maximum Segment Lifetime

© 2017 Burkhard Stiller

ifi

TCP Error Detection and Correction

- Piggybacking:
  - 16 bit window size: transmission volume of 2^{16} = 65536 Byte
  - Variable “Receiver Window“: High speed requires 2^{32} Byte
- Checksumming:
  - Reliably securing TCP header, payload (user data), and TCP pseudo header:
    - IP source and destination address, IP protocol field (6), and TCP segment size
- Acknowledgement:
  - As soon as possible upon receipt of segment or delayed acknowledgement
  - Cumulative or selective (optional)

© 2017 Burkhard Stiller

TCP Retransmission (1)

- Retransmission schemes: Lacking acknowledgement
  - Standard: Go-Back-N:
    - Acknowledgement number *n* acks the receipt of all bytes up to sequence number *n-1*
    - Retransmission driven by time out of retransmission timer
    - Negotiation of scheme during connection setup:
      - Selective Repeat (RFC 1106)
        - NAK requests erroneous segment explicitly
  - Optional: Selective ACKs:
      - Selective Acknowledgment (1996, RFC 2018)
        - SACK acknowledges single, well-received segments
  - Use of a retransmission timer:
    - Started upon sending of a segment, retransmission started, if timer timed out and no ACK has been received

© 2017 Burkhard Stiller

TCP Retransmission (2)

- Jacobson/Karels algorithm:
  - New calculation for average RTT:

    Diff = Sample_RTT - Estimated_RTT

    Estimated_RTT = Estimated_RTT +
    (δ x Deviation = Deviation + δ(|Diff|- Deviation)

    where δ is a fraction between 0 and 1

  - Consider variance when setting timeout value:
    - Time_Out = μ x Estimated_RTT + ϕ x Deviation
      where μ = 1 and ϕ = 4

  - Remarks:
    - Algorithm only as good as granularity of clock (500 ms on Unix)
    - Accurate timeout mechanism important for congestion control

© 2017 Burkhard Stiller

TCP Flow Control

- Cf. slide M08-50 and following
- Flow control maintains the flow of user data
- TCP utilizes the sliding window mechanism:
  - ACK flag acknowledges receipt of all bytes numbered with a smaller number than the sequence number received
  - Advertised Window determines the number of additional bytes, the receiver is able to accept
  - Receiver allows sender to sent data up to the amount of:
    - Acknowledgment + Advertised Window

© 2017 Burkhard Stiller

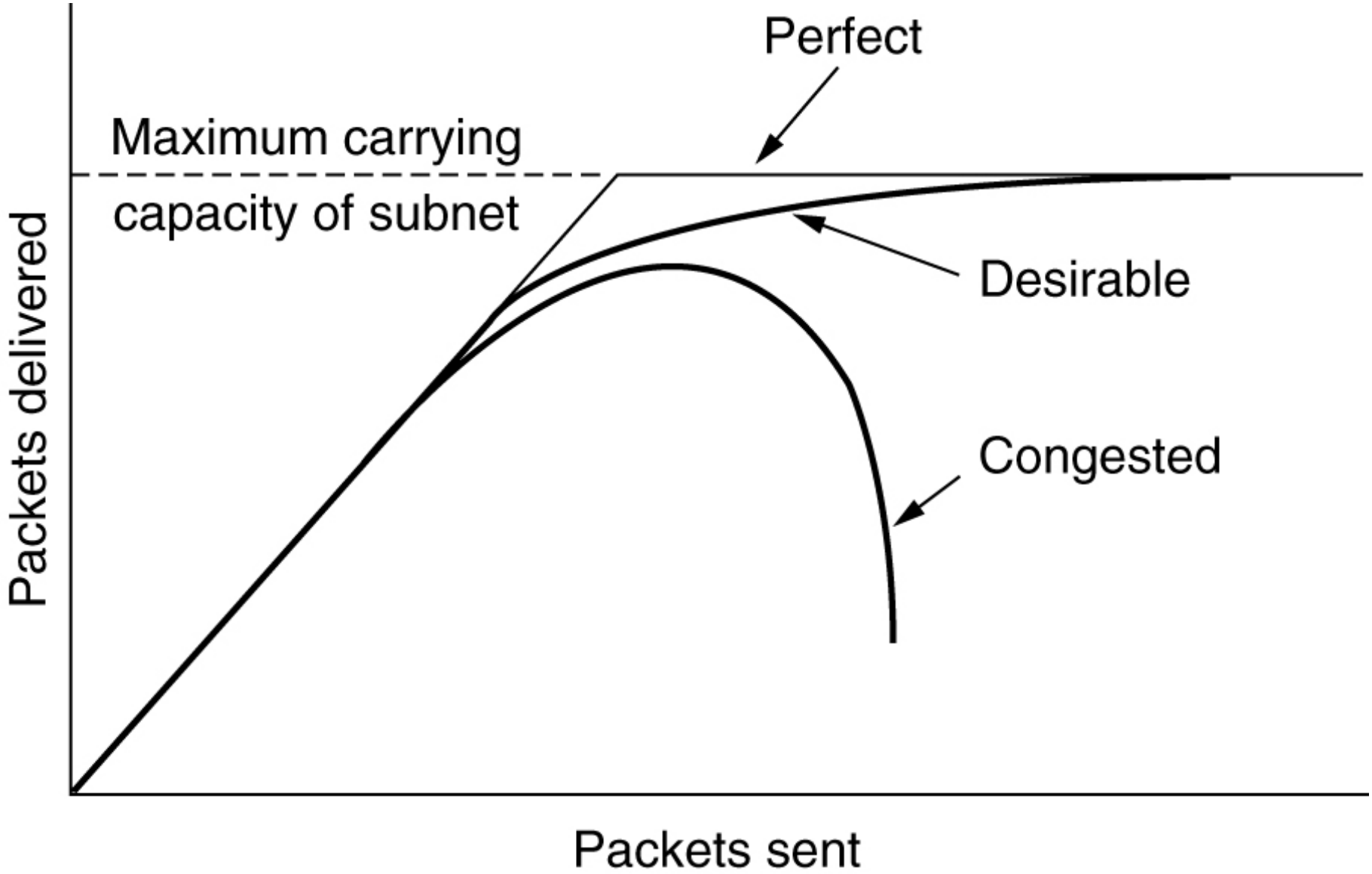
Congestion …

Internet Traffic Problem

- Internet USA 1987:
  - Congestion Collapse!
  - Network was fully loaded, throughput = 0
  - Only retransmissions observed
- Van Jacobson, LBL Labs, 1988:
  - TCP Congestion Control mechanism
    - Initial version: TCP Tahoe
- Traffic characteristics in the Internet?
  - Extremely difficult to describe mathematically
    - Many Internet flows are TCP-based
- Question: What is the gain, if the performance of a congestion control mechanism is enhanced by 1%?

© 2017 Burkhard Stiller

Congestion Effects



© 2017 Burkhard Stiller

Congestion Control Principles and Prevention

- Monitor the system
  - Detect when and where congestion occurs
- Pass information to where action can be taken
- Adjust system operation to correct the problem.

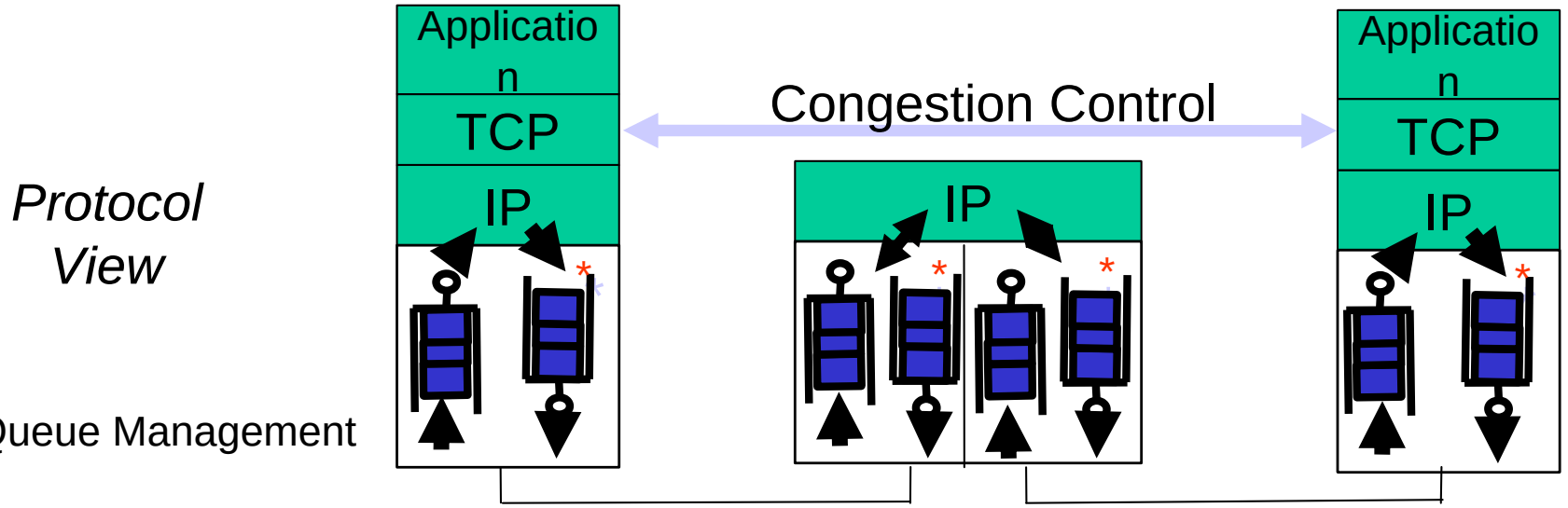
- Prevention approches

Layer	Policies
Transport	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy • Timeout determination
Network	• Virtual circuits versus datagram inside the subnet • Packet queueing and service policy • Packet discard policy • Routing algorithm • Packet lifetime management
Data link	• Retransmission policy • Out-of-order caching policy • Acknowledgement policy • Flow control policy

© 2017 Burkhard Stiller

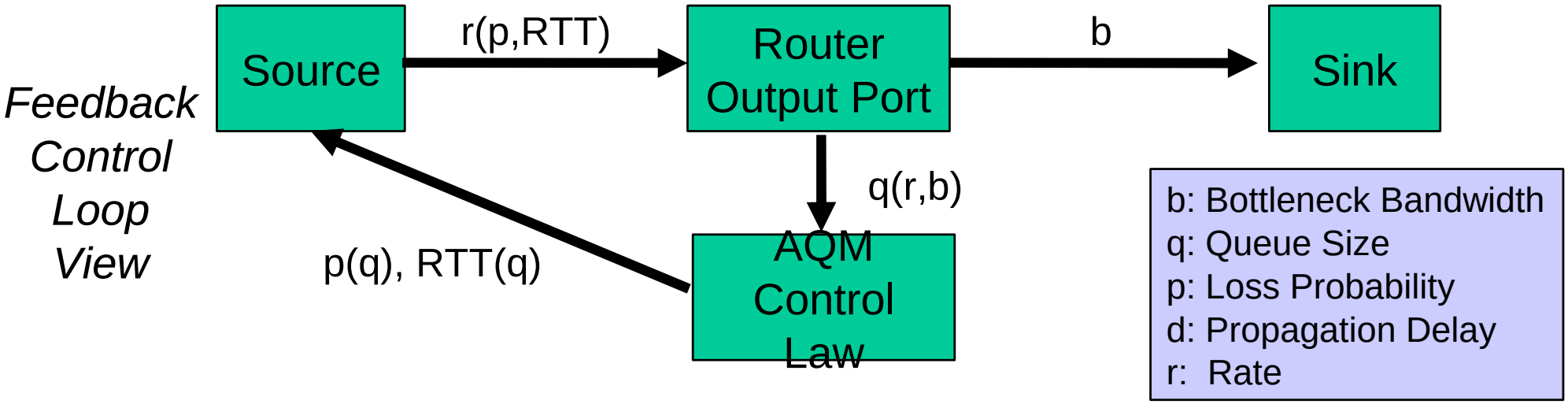
TCP Congestion Control (1)

- Congestion control:
  - Addresses the queuing situation in routers
  - Congestion leads to retransmissions in the transport layer
  - This leads to an increase of the congestion
  - Congestion control needs to adapt sending rate(s)
  - Queue management within routers need to drop packets



© 2017 Burkhard Stiller

TCP Congestion Control (2)



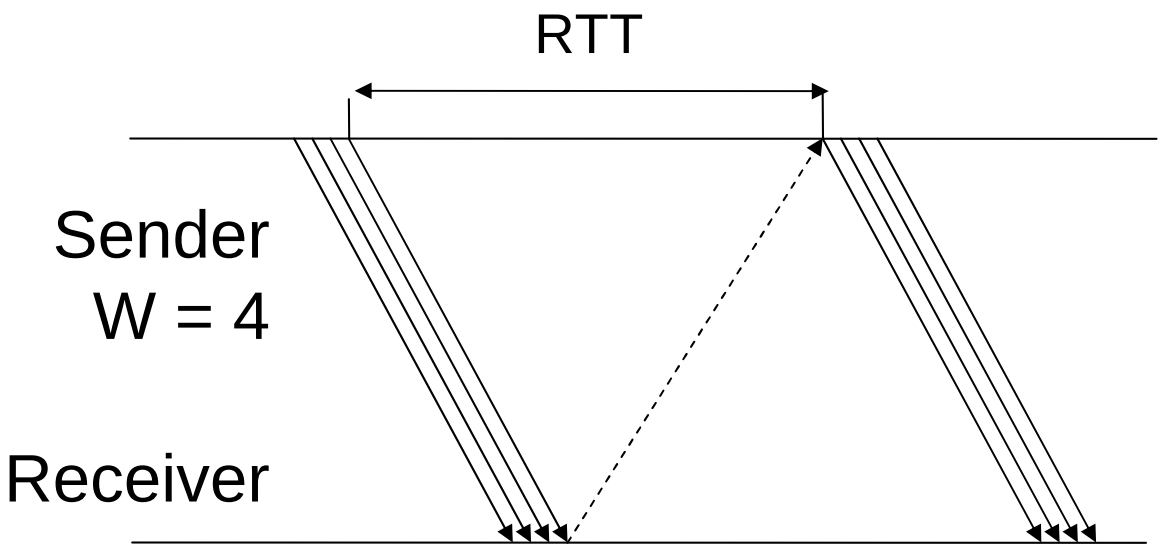
- Approaches for TCP:
  - Implicit mechanisms:
    - Increase of sending credit after receipt of an ACK
    - Decrease of sending credit upon lack of an ACK
  - Explicit mechanisms:
    - Explicit Congestion Notification (RFC 2481)

AQM: Alternate Queue Management

© 2017 Burkhard Stiller

Window-based Congestion Control (1)

- Window W defines the number of packets allowed to be sent upon receipt of an ACK
- Bursty Window Control:
  - ACK step = W

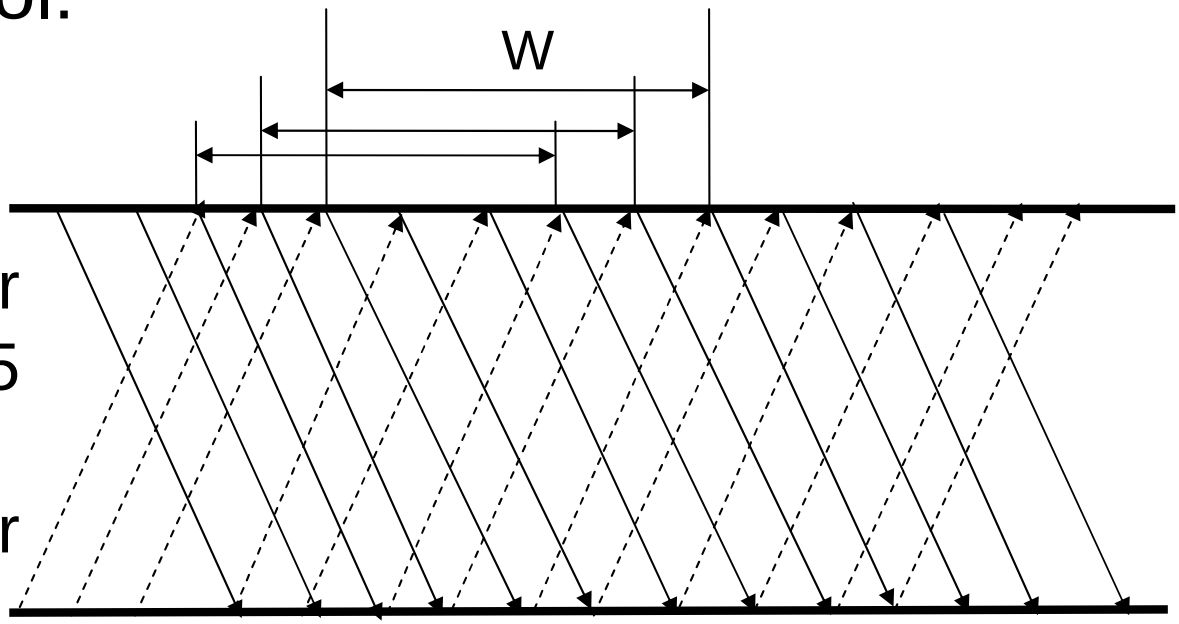


- Advantage:
  - Small overhead due to ACKs
- Drawbacks:
  - Bursty nature
  - Loss of ACKs

© 2017 Burkhard Stiller

Window-based Congestion Control (2)

- Sliding Window Control:
  - ACK step = 1



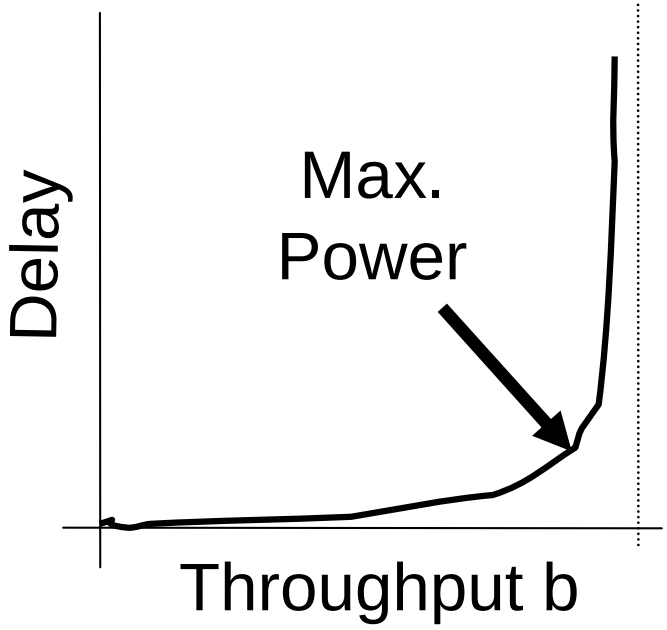
- Advantages:
  - Not bursty
  - Loss of ACKs bearable
- Drawbacks:
  - Large overhead due to many ACKs
  - Interdependency between error control and window size

© 2017 Burkhard Stiller

Window-based Congestion Control (3)

- Control of transmission volume, bytes or packets
- Effective rate = W(t) / RTT(t);
  - W(t) = f(p(t))
  - RTT(t) = q(t) + 2*d
- Optimal Window: W = b * 2d
  - Optimal link utilization:
    - Effective rate = b*(2d/2d) = b
  - No queue at bottleneck:
    - Optimal network performance (power)

b: Bottleneck Bandwidth
q: Queue Size
p: Loss Probability
d: Propagation Delay

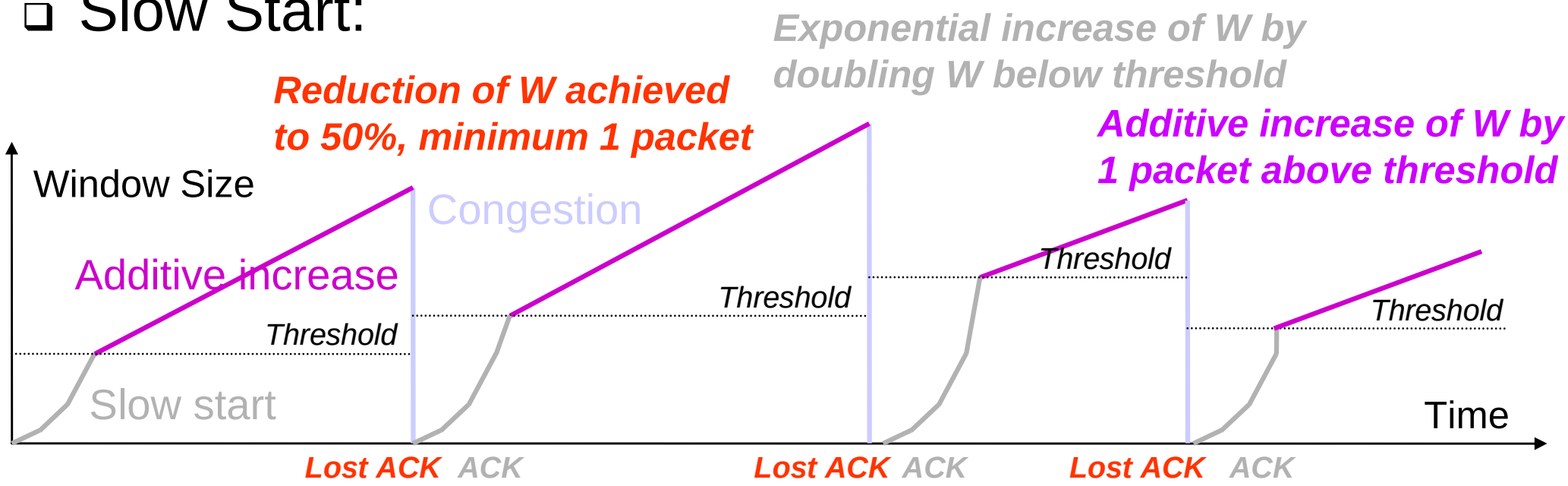


Power = Throughput / Delay:

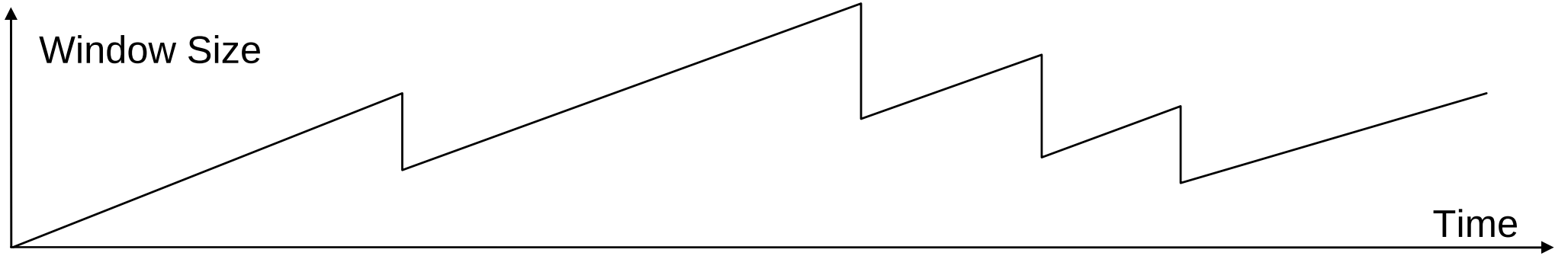
© 2017 Burkhard Stiller

Adaptation of TCP Window Size

- Slow Start:



- Additive Increase Multiplicative Decrease (AIMD):



Practical TCP Considerations

- Wrap around of TCP sequence numbers:
  - High-speed networks may show an overflow within MSL
- Bandwidth delay product:
  - Advertised Window (16 Bit) defines max window of 64 kByte

Bandwidth	Time until Wrap Around	Bandwidth Delay Product
T1 (1.5 Mbit/s)	6.4 h	18 kByte
Ethernet (10 Mbit/s)	57 min	122 kByte
FDDI (100 Mbit/s)	6 min	549 kByte
STM-1 (155 Mbit/s)	4 min	1200 kByte
STM-3 (622 Mbit/s)	55 s	1800 kByte
STM-16 (2.49 Gbit/s)	14 s	29800 kByte

RTT = 100 ms

← MSL = 2 min

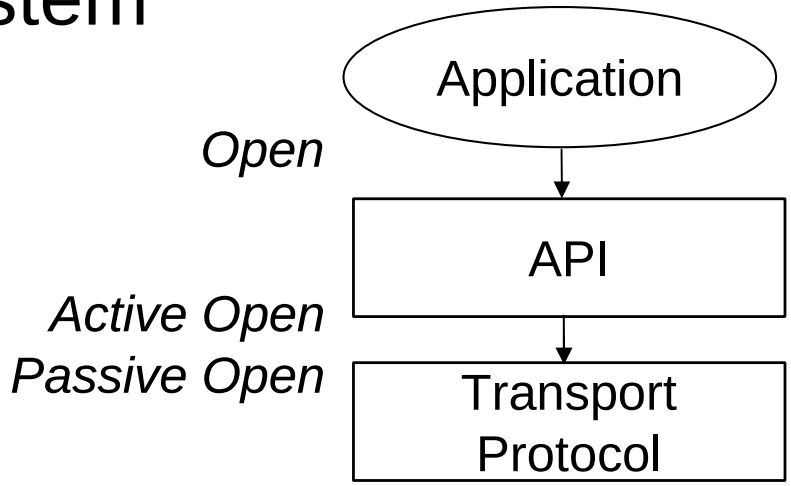
MSL: Maximum Segment Lifetime

© 2017 Burkhard Stiller

Protocol Implementations and Interfaces

Implementation ...

- To separate protocol implementations from protocol interfaces being exported:
  - Importance at transport layer due to application access to the network
  - Interface called Application Programming Interface (API)
- API defined by the operating system
- Specific API: sockets
  - Defined by BSD Unix
  - Ported to many other systems



© 2017 Burkhard Stiller

Socket Calls

socket	Create a new communications end point
bind	Bind server address to socket
listen	Listen to incoming connections
accept	Blocking at passive open, creates a new socket for a server
connect	Active connection setup
send	Send
receive	Receive
close	Connection teardown

```
socket (domain, type, protocol)
bind   (socket, address, addr_len)
listen (socket, backlog)
accept (socket, address, addr_len)
connect(socket, address, addr_len)
send   (socket, message, msg_len, flags)
recv   (socket, buffer, buf_len, flags)
close  (socket)
```

Code examples in C may be found at http://cs.baylor.edu/~donahoo/practical/CSockets/textcode.html

© 2017 Burkhard Stiller

Sample Socket Calls

- Creating a socket:

```
int socket(int domain, int type, int protocol)
domain=PF_INET, PF_UNIX
type=SOCK_STREAM, SOCK_DGRAM
```

- Passive open on server:

```
int bind(int socket, struct sockaddr *address,
         int addr_len)
int listen(int socket, int backlog)
int accept(int socket, struct sockaddr *address,
           int *addr_len)
```

- Active open on client:

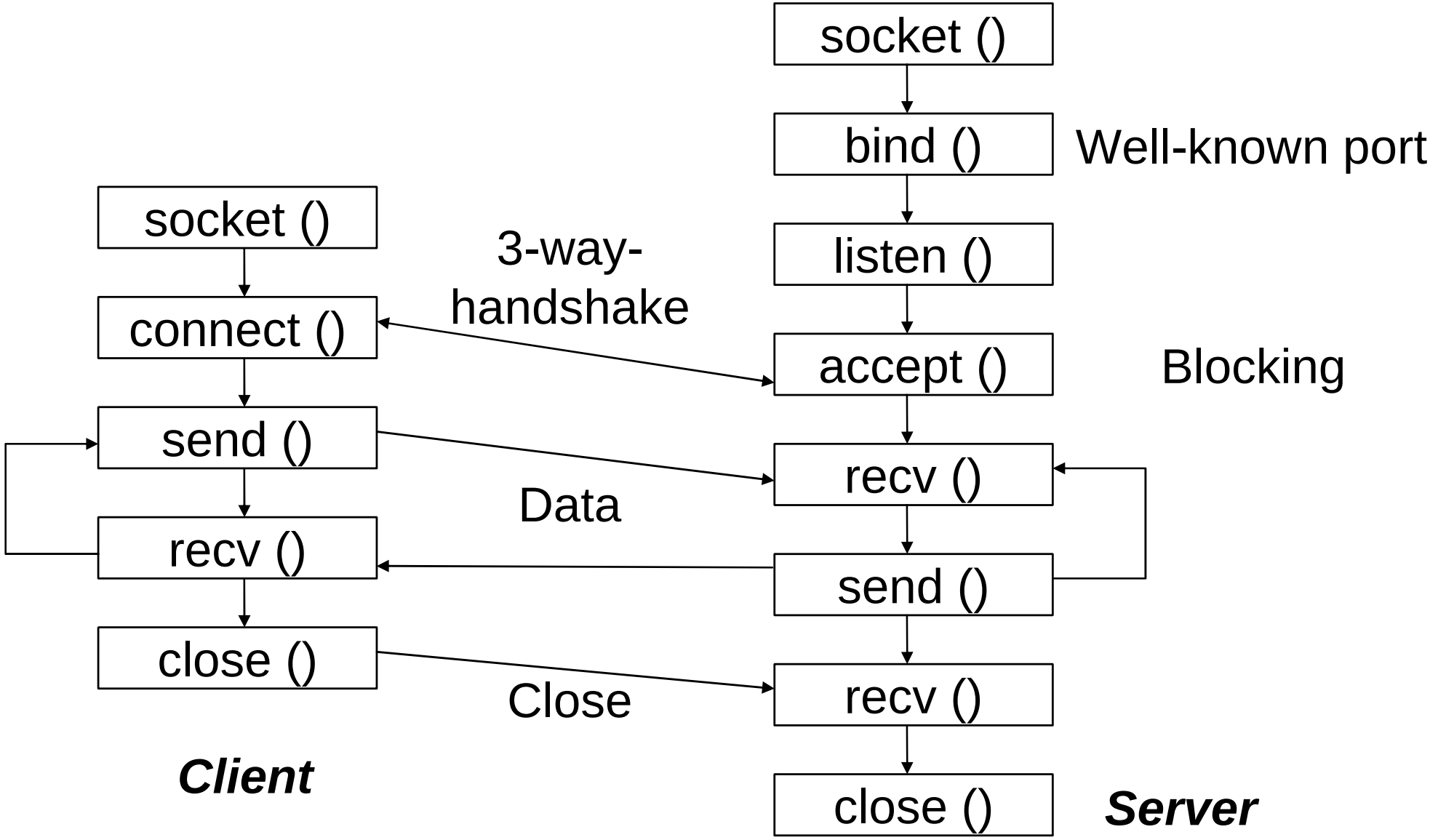
```
int connect(int socket, struct sockaddr *address,
            int addr_len)
```

- Sending and receiving messages:

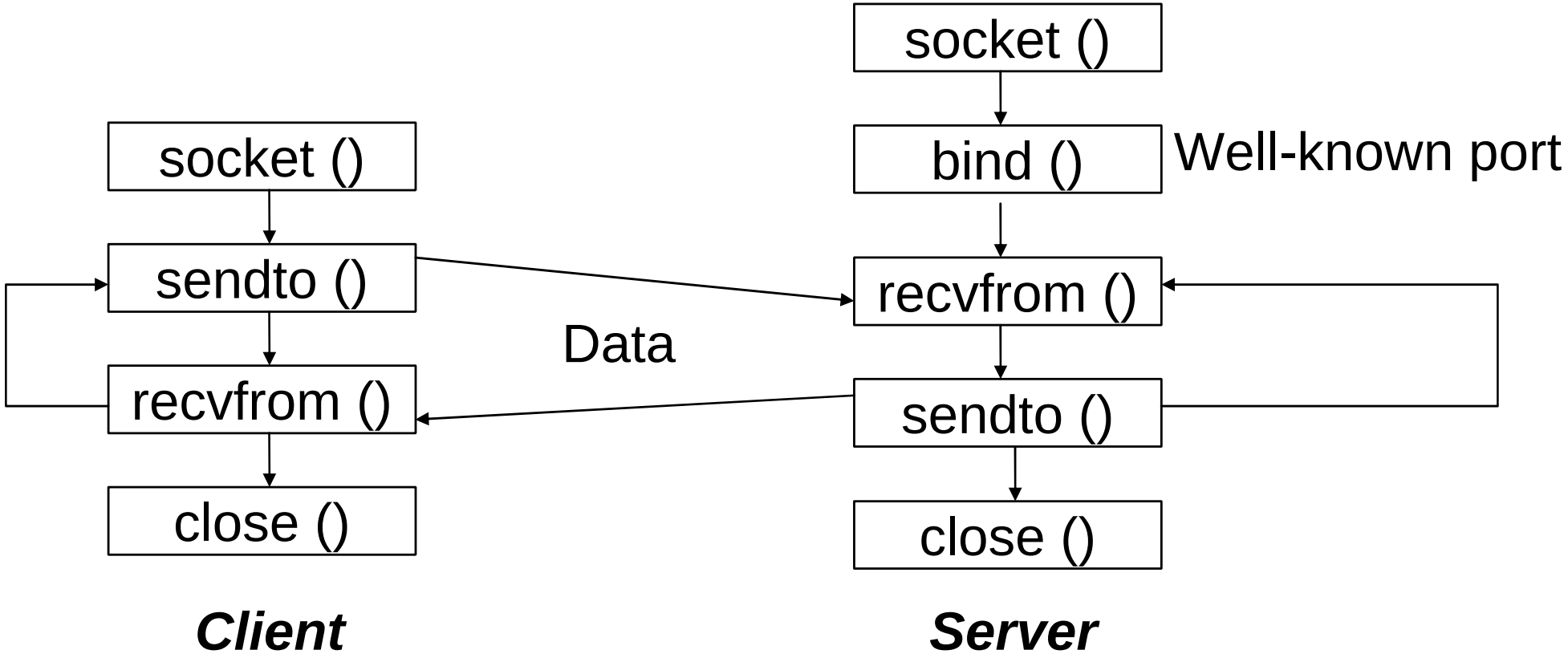
```
int write(int socket, char *message, int msg_len, int flags)
int read(int socket, char *buffer, int buf_len, int flags)
```

© 2017 Burkhard Stiller

TCP Client and Server



UDP Client and Server




```
recvfrom (socket, buffer, buf_len, flags, from, from_len)
sendto (socket, message, msg_len, flags, to, to_len)
```

© 2017 Burkhard Stiller

12. Security Mechanisms and Protocols

Security Motivation, Threats, Attacks

Authentication, Encryption, Signatures

Access Control, SSH, SSL, IPSec

© 2017 Burkhard Stiller

Security for Networks and Distributed Systems

- Traditional view:
  - Public networks: Closed networks maintained centrally
  - Internet: Research network, no target, no attacks ”useful“, a certain degree of user trust
- Current view:
  - Decentralization of public networks driven by deregulation of telecommunication markets
  - Strong commercial use of open, decentralized, and “anarchic“ Internet
  - Distributed applications vulnerable due to their distribution!
- Security mechanisms are inevitable for modern communication networks and distributed systems.

© 2017 Burkhard Stiller

Vulnerability, Threat, and Risk

- Vulnerability:
  - A quality or characteristic of a system that provides an opportunity for misuse.



- Threat:
  - Any potentially malicious or otherwise occurrence that can have an undesirable effect on the assets and resources of an IT system.
- Risk:
  - = Threat X Vulnerabilities ***OR*** Likelihood X Impact



IT Security defines a process of risk management, supported by a set of suitable technical measures!

© 2017 Burkhard Stiller

OWASP Risk Rating Methodology

OWASP: Open Web Application Security Project

- Software life cycle
  - Early stages: Identification of security concerns in the architecture or by design by using threat modeling.
  - Later stages: Finding security issues using code review or penetration testing
  - Operations: Problems may not be discovered until the software is in production and being compromised
- 6 Steps
  - Step 1: Identifying a risk, Step 2: Factors for estimating like-lihood of that risk, Step 3: Determining factors for estimating technical and business impact, Step 4: Deter-mining severity of risk at scale 0 … 9, Step 5: Deciding on prioritized list what to fix, Step 6: Customize risk rating model

Attacks and Damage

- Statistics of attacks (Angriff):

CERT: Computer Emergency Response Team
http://www.cert.org/stats/

  - 22.000 attacks onto Pentagon in 2000
  - 1999: 10.000 incidents 417 Cataloged Vulnerabilities

2000:	20.000 incidents	1.090
2001:	53.000 incidents	2.437
2002:	82.000 incidents	4.129
2003:	138.000 incidents	3.784
2007		7.263

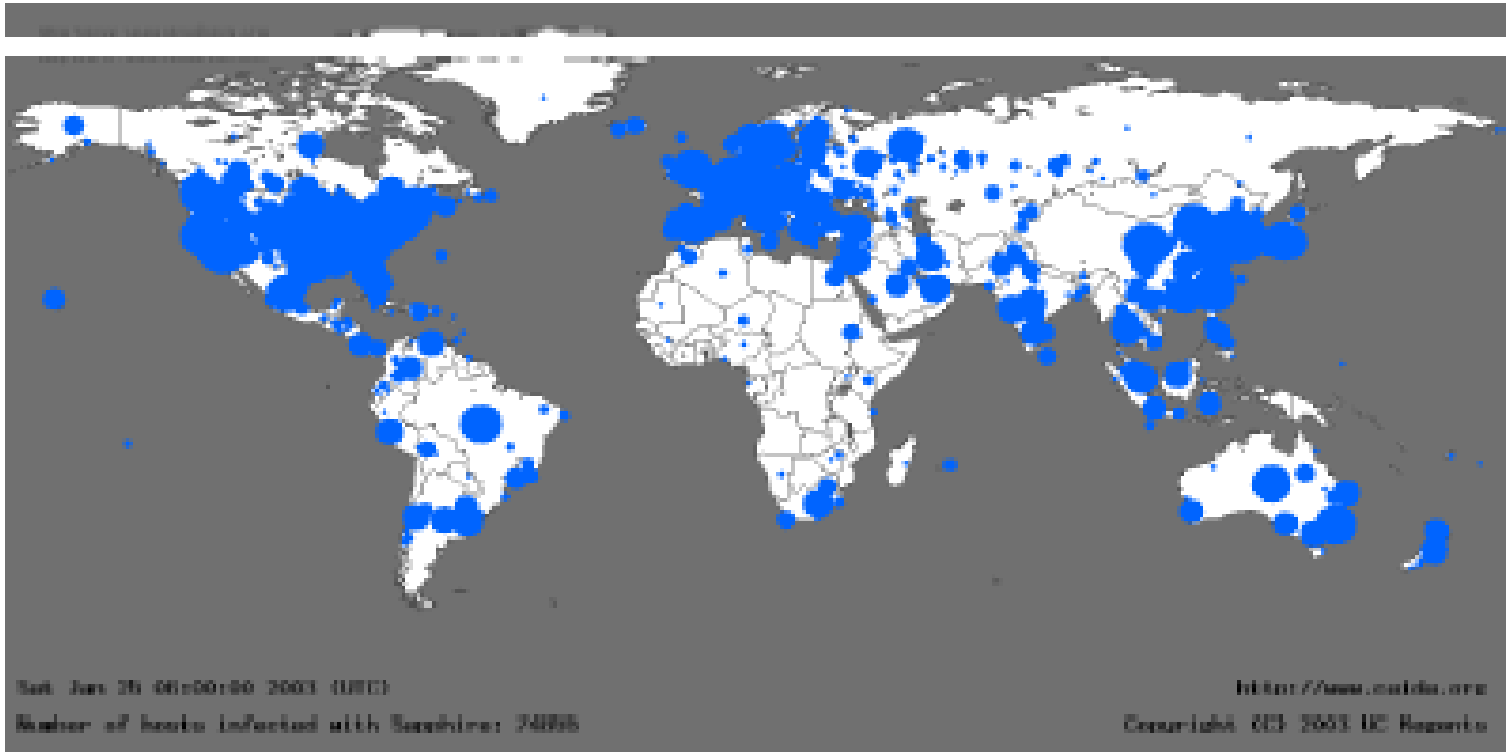
- Estimation of damage (Schaden):
  - Melissa (1999): Word 97/2000
    - 300.000.000 US$ with 150.000 systems infected for about 4 days
  - ILOVEYOU (2000): Outlook
    - 10.000.000.000 US$ with 500.000 systems infected for 24 hours
  - Targeted attacks (2013) at 1 company can cost up to US $2.4 million in damages per attack/incident

http://www.kaspersky.com/about/news/virus/2013/

© 2017 Burkhard Stiller

Motivating Example – SQL Slammer (1)

- SQL Slammer, 2003:
  - Saturday, January 25, 2003: just before 5:30am (UTC), say 6:30am (MEZ) first instance of worm discovered
  - Large-scale distribution from 6:00am (UTC) onwards

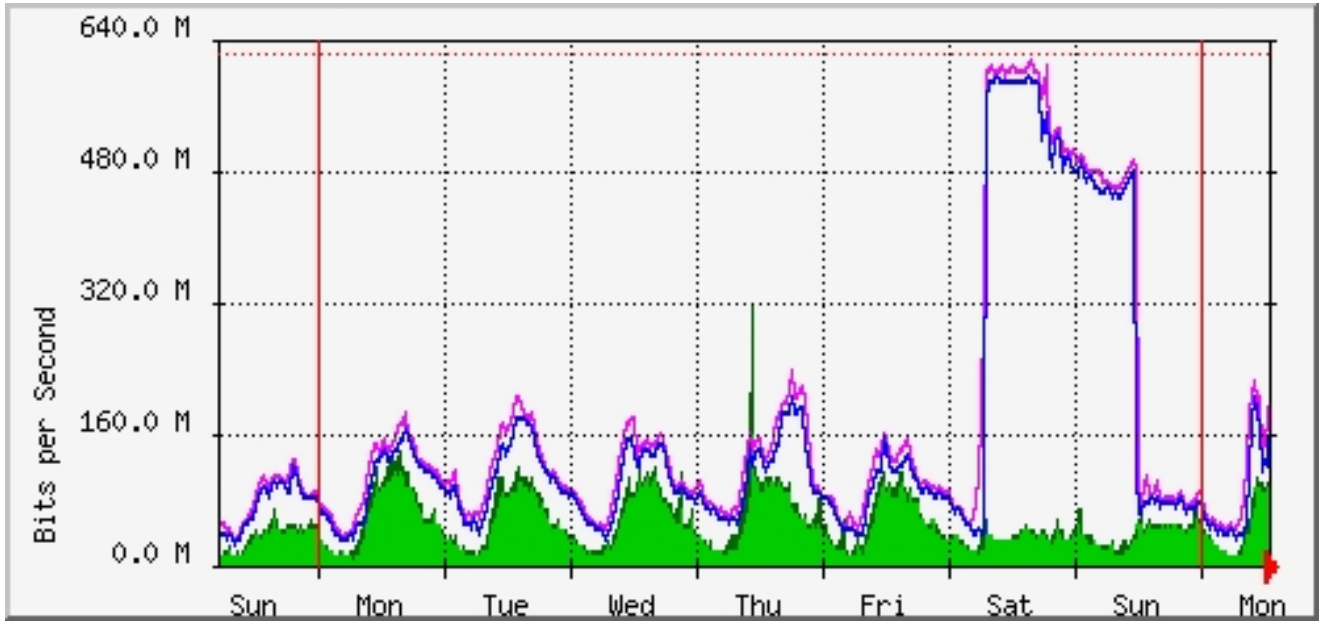


Quelle:

MPSS 03

Motivating Example – SQL Slammer (2)

- Munich Research Network (MWN)
  - For local Munich educational units
  - Massive problems from Saturday January 25, 2003 at 6.30am until Sunday January 26, 2003 at 11.30am
- Traffic statistics at central backbone router (1 week):



  - **Output traffic**
  - **Input traffic (maximum peak for every 5 min)**
  - **Output traffic (maximum peak every 5 min)**

© 2017 Burkhard Stiller

Motivating Example – SQL Slammer (3)

- Fastest worm ever so far:
  - 1st min: duplication of population all 8.5 s
  - From 3rd min: slower duplication due to network capacity
  - All 10 min: about 90% of all susceptible hosts infected
- Consequences:
  - Large parts of the Internet can not be reached any more
  - Energy supply disrupted, ATMs interrupted, …
- Characteristics:
  - Slammer (SQL Server) suits exactly a single UDP packet
    - 376 Byte, Assembler code
  - Slammer ”exploits“ buffer overflow of UDP Port 1434
  - After infection:
    - ”Guessing“ of arbitrary IP addresses and attack via UDP

© 2017 Burkhard Stiller

Motivating Example – Stuxnet

- Stuxnet (June 2010)
  - Worm attacking a Supervisory Control and Data Acquisition (SCADA) system (Leitsystem)
    - Target: industrial control systems, infected over 200,000 Windows machine causing app. 1,000 machines to physically degrade
  - Explicitly programmed for a Siemens control technology (Simatic-S7), addressing a particular industry in Iran
    - Extreme efforts for developments
    - Very detailed knowledge of Simantic-S7 necessary
- Unproven status of investigations:
  - An intention to sabotage Iranian nuclear power plants
  - Dedicated damage seems to happen in *some* systems only!
  - Other systems remain infected, but were not damaged!

© 2017 Burkhard Stiller

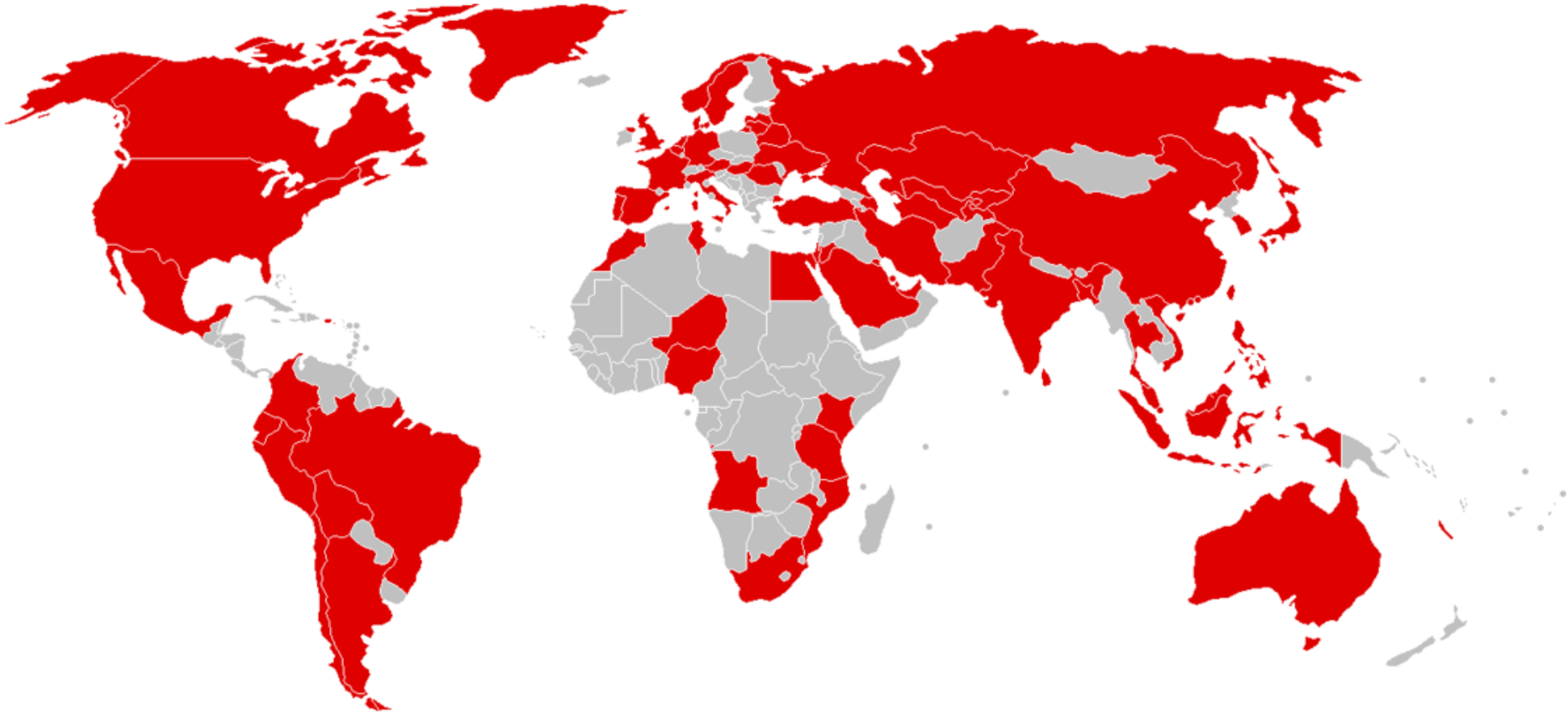
Motivating Example – Wanna Cry (1)

- WannaCry ransomware attack (May 2017)
  - Ransomware crypto-worm, targeting Windows machines
  - Encrypting data and demanding ransom payments
    - Infected more than 230,000 computers in over 150 countries
    - "kill switch" (mechanism incorporated in software to be able to disable Software) was used to slow down the migration (domain name reg.)
  - Propagation by EternalBlue, an exploit of Windows‘ Server Message Block (SMB) protocol, which enables the execution of arbitrary code on the target computer
  - Approach: Worm‘s payload finds and encrypts a range of data files, displays a “ransom note”, informing the user and demanding a payment in bitcoin with different payment schemes (300 US$ within 3 days, 600 US$ within 7 days).

© 2017 Burkhard Stiller

Motivating Example – Wanna Cry (2)

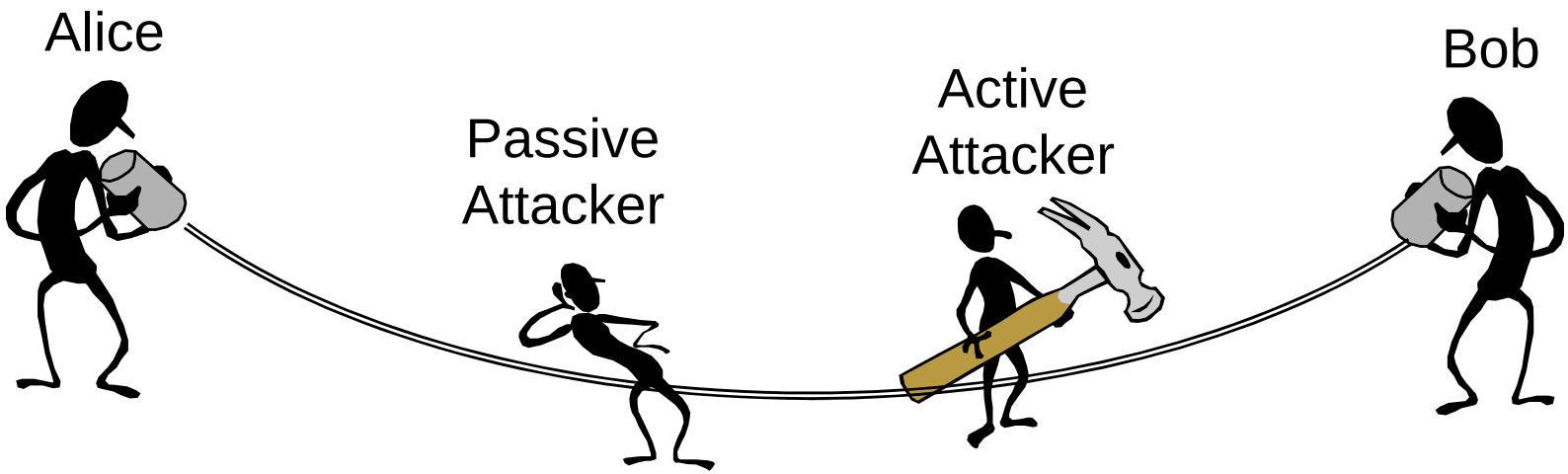
- Countries initially impacted



https://en.wikipedia.org/wiki/WannaCry_ransomware_attack#/media/File:Countries_initially_affected_in_WannaCry_ransomware_attack.png

© 2017 Burkhard Stiller

Example: Attacking the Data Transfer



- One possible passive attack:
  - Eavesdropping (Abhören) only, no change of data
    - ➢ Threat for confidentiality
- Multiple possible active attacks:
  - Changing, deletion, insertion
    - ➢ Threat for confidentiality, integrity, authenticity

© 2017 Burkhard Stiller

Example: Attacking Service Provisioning

- Denial-of-Service (DoS) attacks, *e.g.*, of Web Server:
  - Principle:
    - Generate as much as possible work for any infrastructure to prevent the processing of ”normal“ tasks.
  - SYN-Flooding (Synchronization):
    - Send TCP-SYN messages to TCP (Transmission Control Prot.) Server
      - Spoofed IP addresses
      - Server allocates resources for partially opened connections
  - “smurf“ DoS attack:
    - End systems reply to ICMP (Internet Control Message Protocol) Echo-Requests messages
    - Spoofed address:
      - End system received large amount of ICMP Echo-Reply messages
  - Distributed Denial-of-Service attacks (DDoS)

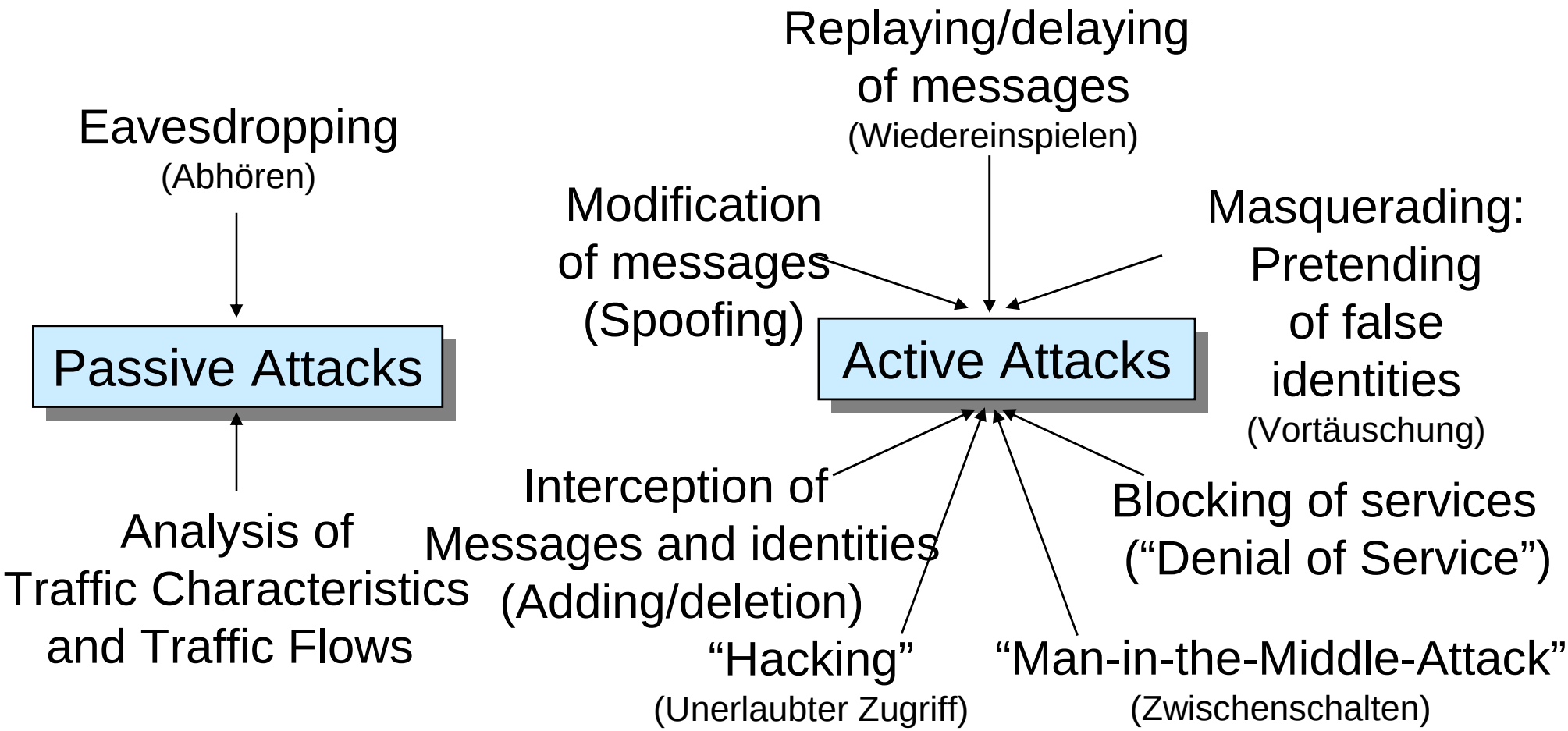
© 2017 Burkhard Stiller

Threats — Examples/Countermeasures

- Packet Snooper:
  - Reading of packet content (data) → Encryption
- Packet Sniffer:
  - Reading of source and destination addresses (protocol header) → Encapsulation of packet and encryption
- Session Hijacking:
  - Session with multiple messages between two parties, a third party may gain control of this session → Authentication
- Data Tempering:
  - Similar to session hijacking, however, only a part of the data transfer will be intercepted → Authentication and encryption

© 2017 Burkhard Stiller

Threats and Attacks



⇨ High risk!

© 2017 Burkhard Stiller

Intrusion and Data Loss



- Intruder Classification (Eindringling):
  - Temporary sniffing of non-technical personnel
  - Sniffing of insiders
  - Dedicated attempts to achieve an economic success
  - Military, industrial, and commercial espionage

- Data loss due to operation failures:
  - Act of God
  - Hard- or software errors
  - Human errors

© 2017 Burkhard Stiller

Major 7 Security Pillars (1)



- Authentication (Authentifizierung/Authentifikation):
  - Authentication ensures that partners involved in communications can prove that the peer is that it claims to be.
- Authorization (Autorisierung):
  - Authorization ensures that a partner with a known ID is enabled to utilize a service.
- Integrity (Unversehrtheit, Fälschungssicherheit):
  - Integrity provides protection against the modification of a message along a transmission path.
- Privacy (Privatheit):
  - Privacy defines the degree of publication of personal information and data.

© 2017 Burkhard Stiller

Major 7 Security Pillars (2)



- Confidentiality (Vertraulichkeit):
  - Confidentiality protects transmitted data against eavesdroppers in a communication channel ensuring that only an authorized receiver can interpret the message received.
- Non-repudiation (Nicht-Zurückweisbarkeit/Nicht-Abstreitbarkeit):
  - Non-repudiation provides that neither the sender nor the receiver can deny that a communication has taken place.
- Anti-replay protection (Schutz gegen Wiedereinspielung):
  - Anti-replay protection protects a receiver from the duplicated reception of a previously obtained and already authenticated message.

© 2017 Burkhard Stiller

Additional Security Aspects (1)

- System security (Systemsicherheit):
  - To obtain a full-fledged security degree, the entire system, going beyond the communication security, has to be protected by means of security mechanisms and protocols.
- Anonymity (Anonymität):
  - Anonymity defines a condition in which a person's true identity is not known.
- Pseudonymity (Pseudonymität):
  - Pseudonymity defines a condition in which a person has taken on an assumed identity.
- Auditing (Diensteprüfung):
  - A. defines the process to collect unforged events and facts.

© 2017 Burkhard Stiller

Additional Security Aspects (2)

- Identity (Identität):
  - The collective aspect of the set of characteristics by which an object/human is definitively recognizable or known.
- Identity Management (Verwaltung der elektronischen Identität):
  - Encompasses different mechanisms, *a.o.,* password management, user provisioning, and access management. Enables/maintains user access to resources, which includes creation of the user entity, authorization, and permissions, and a single point of administration for de-/provisioning accounts.
- Trust and Trusted Third Party (TTP) (Vertrauenswürdiger Dritter):
  - Besides two arbitrary parties a third one (entity, instance, administration), whom those two parties fully trust.

© 2017 Burkhard Stiller

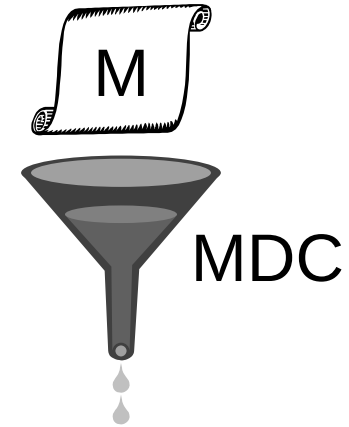
Authentication (1)

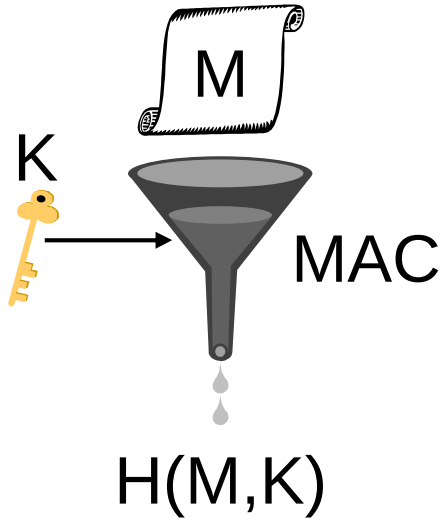
- Mechanisms to prove that the peer that it claims to be is the peer:
  - Ownership (Besitz):
    - *E.g.*, smart card, physical device
  - Knowledge (Wissen):
    - *E.g.*, password, account
  - Biometrics (Körperliche Merkmale):
    - *E.g.*, finger print, iris scan
  - Location or context:
    - *E.g.*, being a well-known person at a certain place for a certain reason
  - Proficiency (Können):
    - *E.g.*, signing

© 2017 Burkhard Stiller

Authentication (2)

- Cryptographic hash function (Message Digest Code, MDC):
  - Message M (arbitrarily long) → Hash H(M) (minimum of 128 bit length)
  - Note: "One-way" feature of function:
    - Efficient generation
    - Very low collision possibility: M, M' with H(M)=H(M')
    - Examples: MD5, SHA-1, SHA-2, RIPEMD-160
- Key-dependent hash function (Message Authentication Code, MAC):
  - Message M, key K → Hash H(M, K)
  - May be constructed out of MDC
  - HMAC (RFC 2104), *e.g.*, HMAC-MD5





© 2017 Burkhard Stiller

Authentication (3)

- Authentication and integrity of packets:
  - Adding of a Sequence Number (SN) to ensure order (re-use)
    - Securing against replay attacks by time stamps (synchronized clocks) or challenge-response mechanisms utilizing random numbers
  - Adding of MAC (Message Authentication Code) or signature, calculated from data, SN, key

- Authentication of systems or users:
  - Application of non-cryptographic mechanisms:
    - Username and password, biometric approaches (finger print, iris)
  - Application of cryptographic mechanisms:
    - Login messages with MAC and signature or PKI, use only once passwords

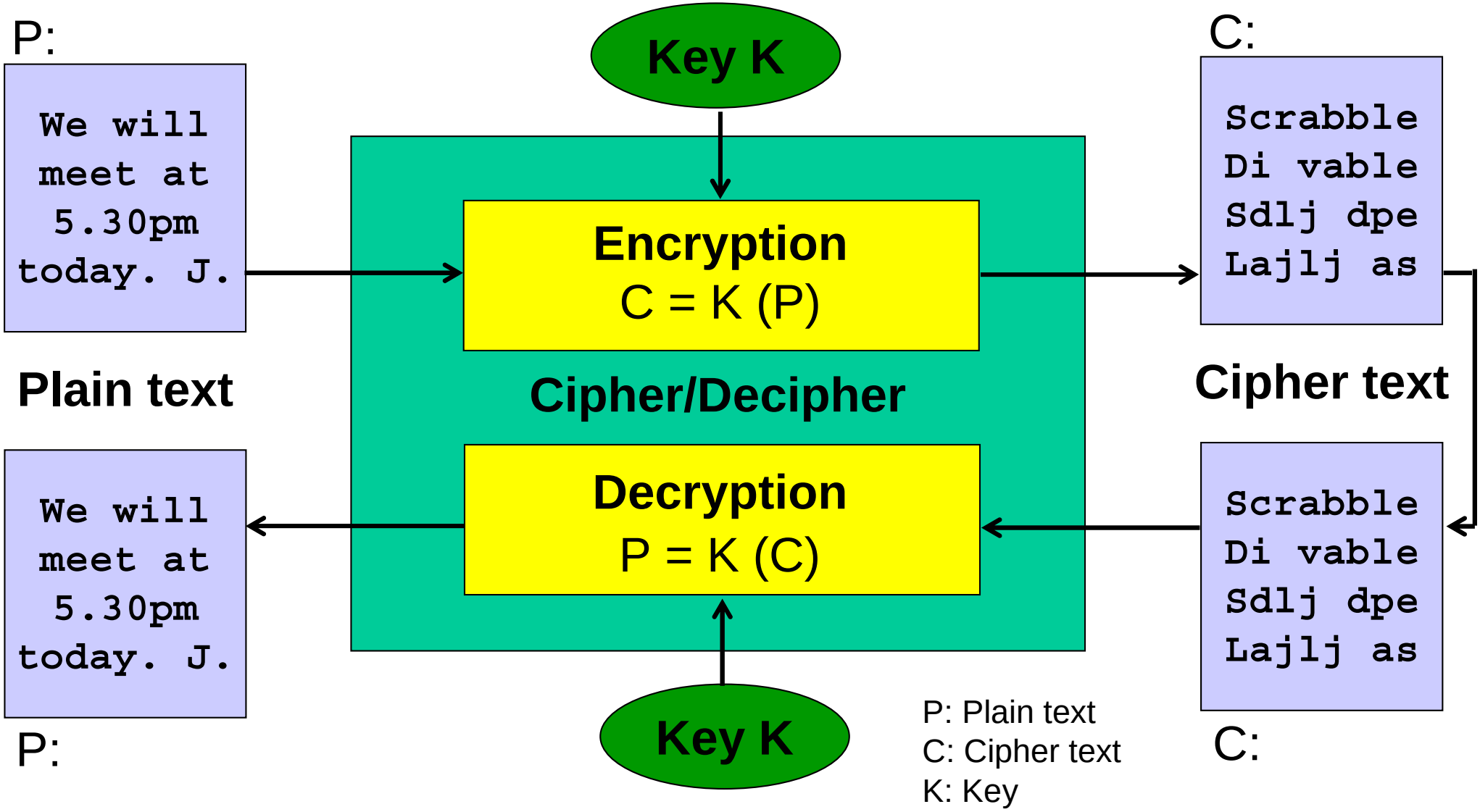
PKI: Public Key Infrastructure

Encryption and Decryption

- En-/Decoding (En-/Decryption) (Verschlüsselung/Entschlüsselung) of data to ensure confidentiality and privacy:
  - Encoding of plain text:
    - Only possible with the knowledge of a key (the “secret”)
    - Easy to do and fast to process
  - Decoding of cipher text (encrypted data):
    - Only successful with the right key
    - Extremely large, dedicated, and specific calculation effort, iff the key is not known (attack situation only), otherwise easy and fast to process
  - Respective algorithms
    - In the past, based on alphabet shifts (Cesar’s Shiffre)
    - Today, typically based on cryptography

© 2017 Burkhard Stiller

Cryptography

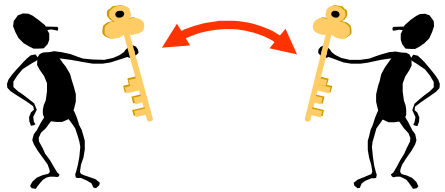


© 2017 Burkhard Stiller

ifi

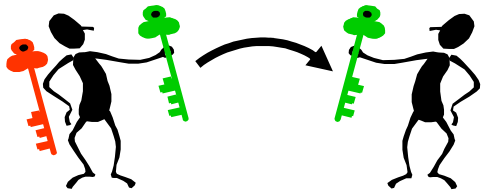
Cryptographic Variants

- Symmetric cryptography:
  - Entities own a shared, secret key



- Advantages:
  - Small overhead/calculation
  - Short keys
- Drawbacks:
  - Key exchange complicated
  - No commitment

- Asymmetric cryptography (public key cryptography):
  - Key pair of private/public parts

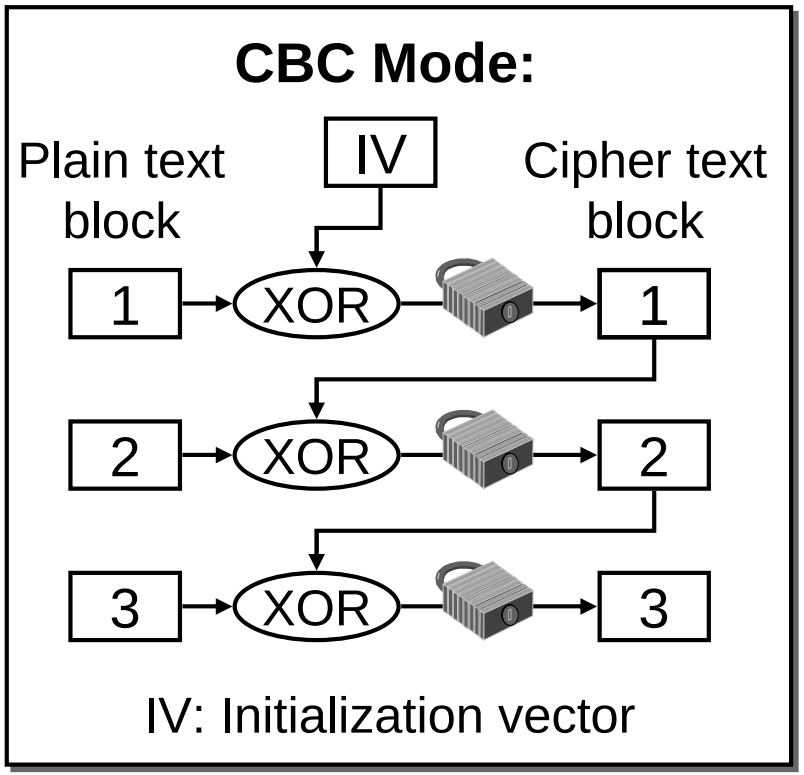


- Advantages:
  - Public keys easy to publish
  - Commitment possible (CA)
- Drawbacks:
  - Longer keys
  - Larger overhead/calculation

© 2017 Burkhard Stiller

Symmetric Encryption

- Symmetric encryption:
  - Current minimum key length 80 or better 128 bit
  - Secure algorithms: 3DES (Digital Encryption Standard), IDEA
- Operation:
  - Block cipher with 64 bit blocks
  - Electronic Code-book (ECB):
    - Block-wise encryption
    - Attacker may interchange blocks
  - Cipher Block Chaining (CBC):
    - "More" secure: every block is dependent on preceding block
  - Byte-wise encryption



© 2017 Burkhard Stiller

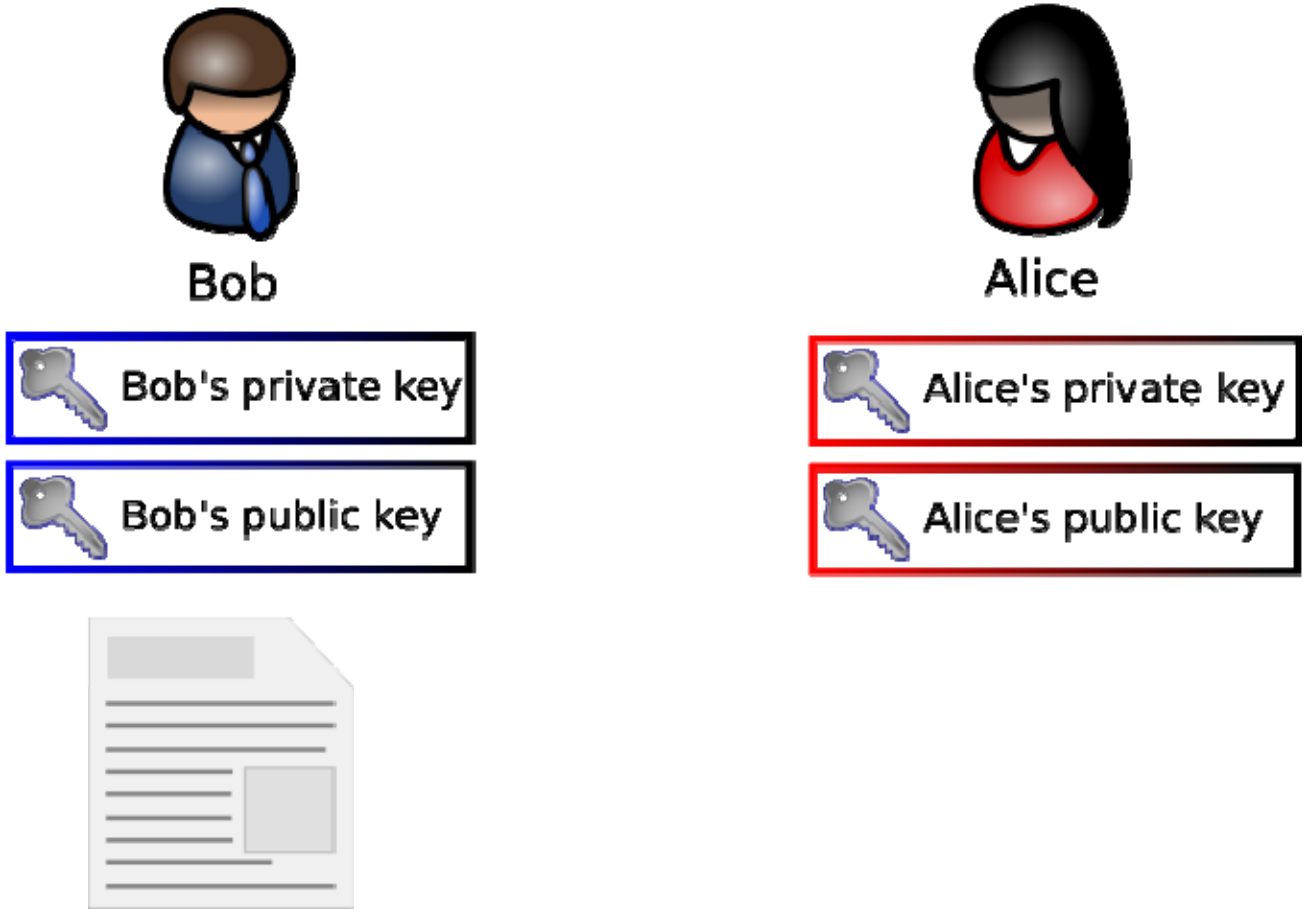
Asymmetric Encryption

- Asymmetric encryption (Public Key Encryption):
  - Encryption easy and publicly accessible to everyone
  - Decryption difficult for everyone except the intended recipient
  - Current minimum key length 1024 bit (309 decimal digits)
  - Secure algorithms: RSA (Rivest-Shamir-Adelman), ElGamal
- Practical encryption: Hybrid approaches
  - First: User authentication and exchange of a session key, public-key-based (in a non hybrid version: symmetric)
  - Second: Symmetric encryption of user data by session key and further authentication required with session key
  - Note: Longer sessions should change session key on a periodical basis, *e.g.*, once per 30 min or 1 hour

© 2017 Burkhard Stiller

Example – Asymmetric Encryption (1)

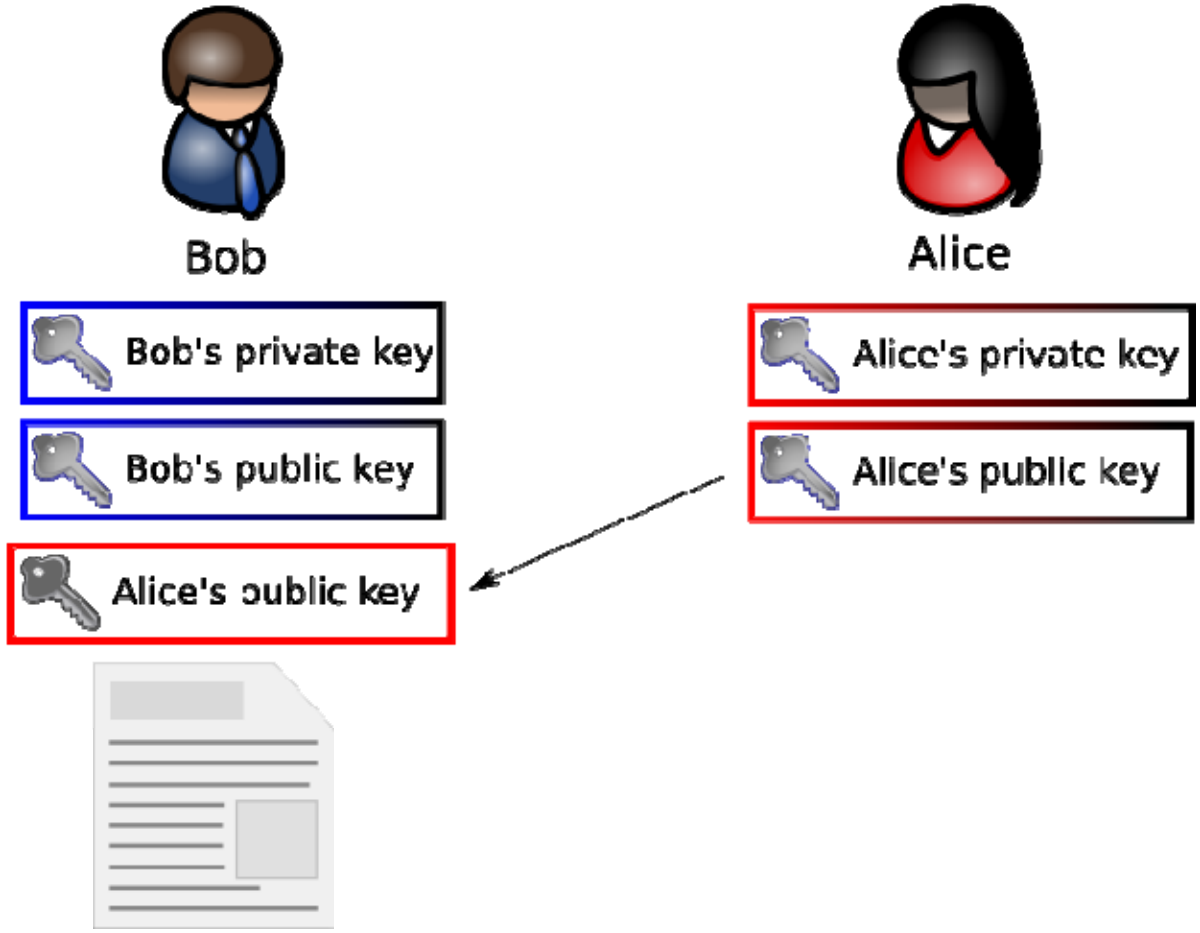
- Encryption demand:
  - Bob plans to sent Alice an encrypted document
- Initial set-up:



© 2017 Burkhard Stiller

Example – Asymmetric Encryption (2)

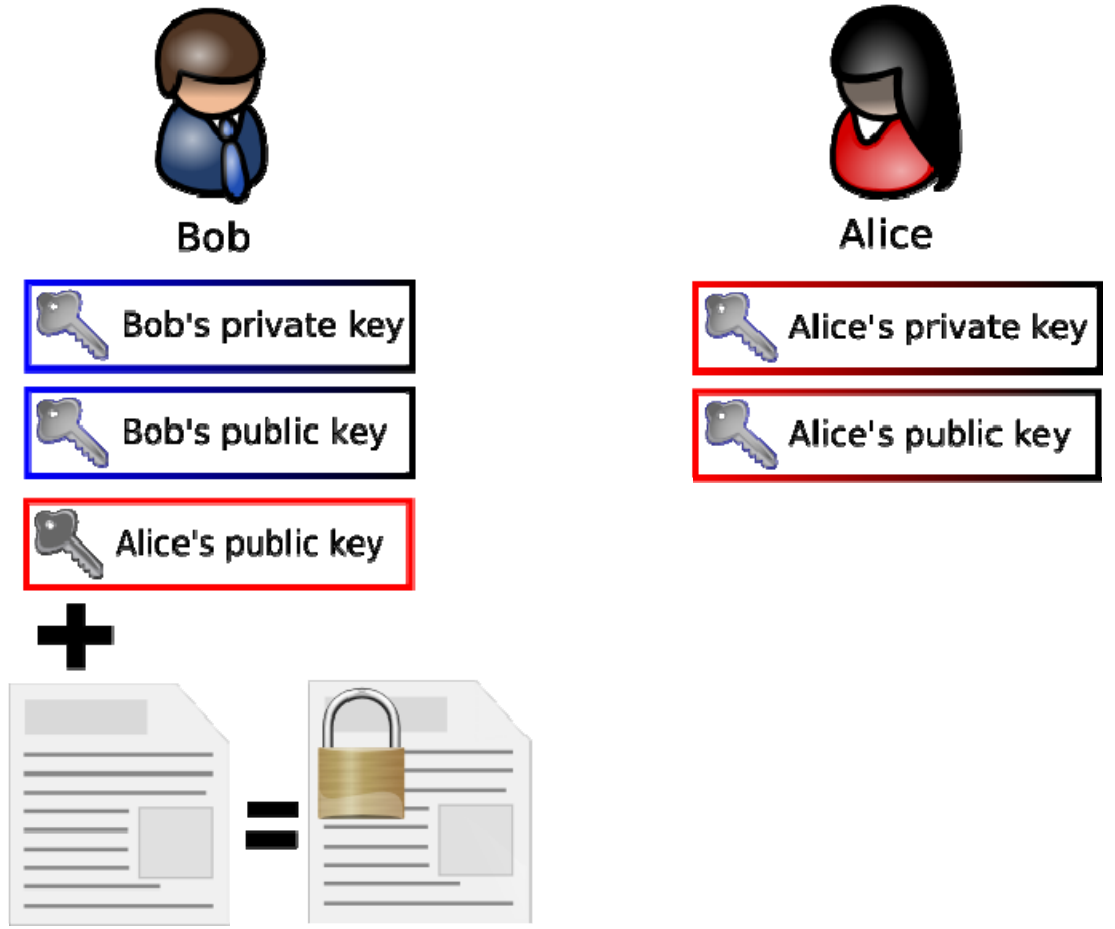
- Key exchange:
  - Alice hands over to Bob her public key (K-Alice$_{pub}$)
  - Alternative: Bob may access Alice‘s public key in a key data base, too



© 2017 Burkhard Stiller

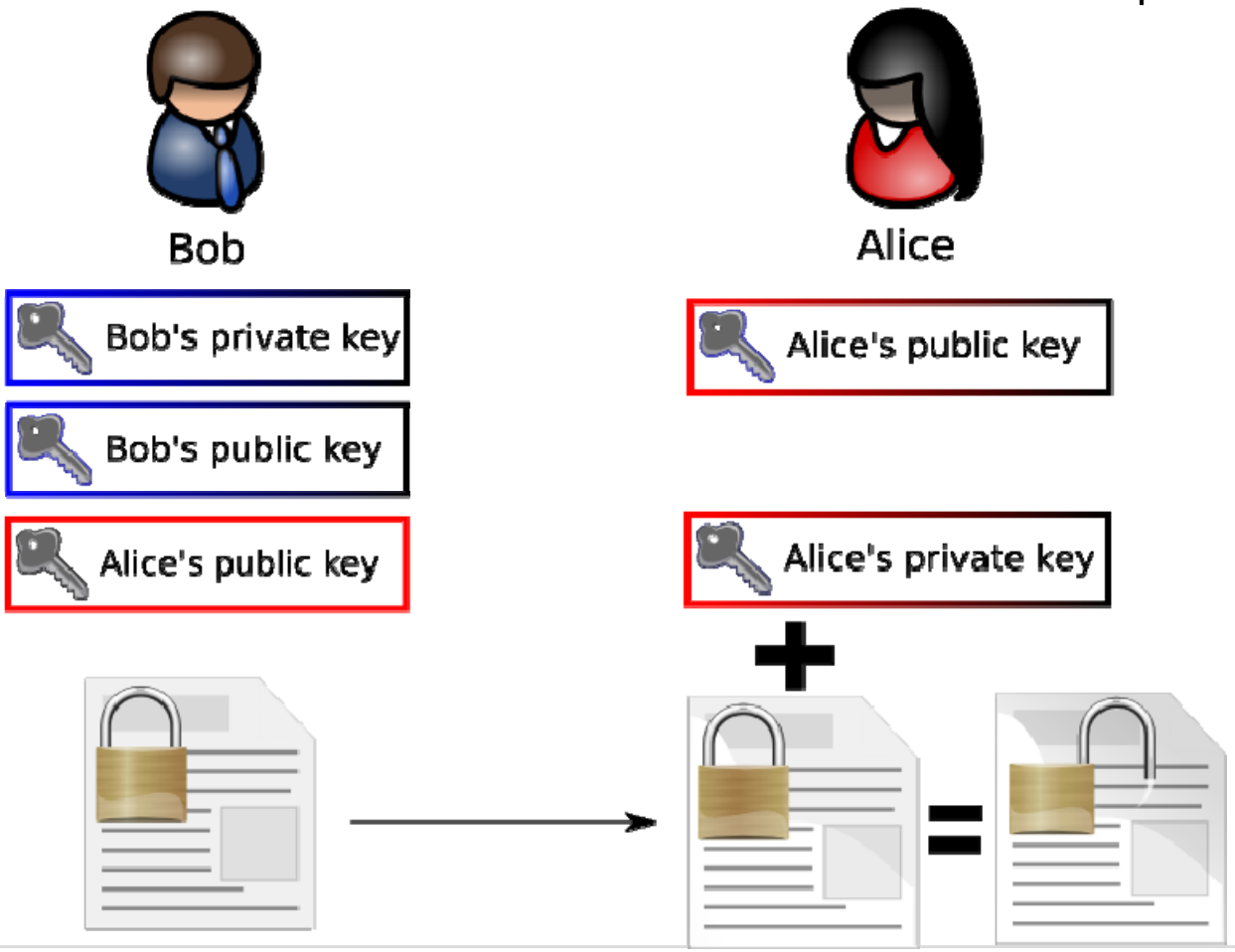
Example – Asymmetric Encryption (3)

- Encryption:
  - Bob encrypts the document with Alice‘s public key (K-Alice$_{pub}$)



Example – Asymmetric Encryption (3)

- Decryption:
  - Alice decrypts the document with her private key (K-Alice$_{priv}$)



RSA (1)

- Encryption defines a function f, mapping plain text to cipher text; decryption defines the inverse of f.

RSA: Rivest-Shamir-Adelman

- f requires 5 properties:
  1. f is one-to-one (uniquely invertible)
  2. f is easy to compute (encryption easy)
  3. f^{-1} is difficult to compute (decryption difficult for senders)
  4. f has a domain that is easy to sample from (Bob easily generates a key)
  5. Existence of an easy-to-compute function d of the input of f making computing f^{-1} easy (Bob decrypts easily)

f is one-way

f is a trapdoor function

RSA (2)

- Example trapdoor function: a mod b: remainder after dividing a by b
  - $\boldsymbol{f}(x,e,p,q) = (x^e \bmod pq, pq, e)$,
  - p and q are prime numbers, x (encoded) plaintext, $y=x^e \bmod pq$ is the cipher text
  - $\boldsymbol{d}(x,e,p,q)=e^{-1} \bmod (p-1)(q-1)$ (property 5.) that is $e\boldsymbol{d} \bmod (p-1)(q-1) = 1$
- f is the basis for the RSA Cryptosystem:
  - (e, pq) public encryption key (public key)
  - (p, q, d) secret decryption key (secret key)
  - Alice generates a private/public key pair once (re-use later)
  - Alice encrypts/signs message with her private key
  - Bob decrypts this message with Alice's public key

© 2017 Burkhard Stiller

Example (Simplified) – RSA (1)

- 2 (large, very large) prime numbers p,q are selected
  - p=11, q=13
- The product N is calculated as N = p * q = 143
- Value e ist calculated to: 1 < e < (p-1) (q-1)
  - e = 23
- Value d ist calculated to: d*e mod (p-1)(q-1) = 1
  - d = 47
- Thus, the number pair (e, N) defines the public key:
  - K_{pub} = (23, 143)
- Thus, the number triple d defines the private key:
  - K_{priv} = (11, 13, 47)

© 2017 Burkhard Stiller

Example (Simplified) – RSA (2)

- Text (plain text) to be decrpyted: P = 75
- Calculation of the cipher text y as: $y = P^e \bmod N$
  - $y = 75^{23} \bmod 143$
  - $y = 69$
- Cipher text C = 69 will be transmitted as the message
- Calculation of the plain text x as: $x = C^d \bmod N$
  - $x = 69^{47} \bmod 143$
  - $x = 75 \rightarrow P$

© 2017 Burkhard Stiller

Theoretical Security

- Basis: Prime Number Factoring
  - N = p * q, with p and q prime numbers
  - Existing algorithms cannot do this in “finite” time
    - Exponential algorithms in place, but no polynomial ones? No proof!
    - But: Shor-Algorithm for quantum computers factorizes polynomial time
    - 200 digit binary number factorization took from December 2003 until May 2005 at University of Bonn with 80 2.2 GHz machines.
    - Typical key length: 300 binary digits
    - RSA Laboratories: Contest RSA 576 bit digits and RSA-2048 (617 decimal digits), RSA-768, 3 years, 2000 2.2GHz (Dec 19, 2009)
  - Moore‘s Law (duplication of processing power in 2 years) does not determine a problem for RSA, however, quantum computers may simply break the system at any time.

© 2017 Burkhard Stiller

Signatures and Certificates (1)

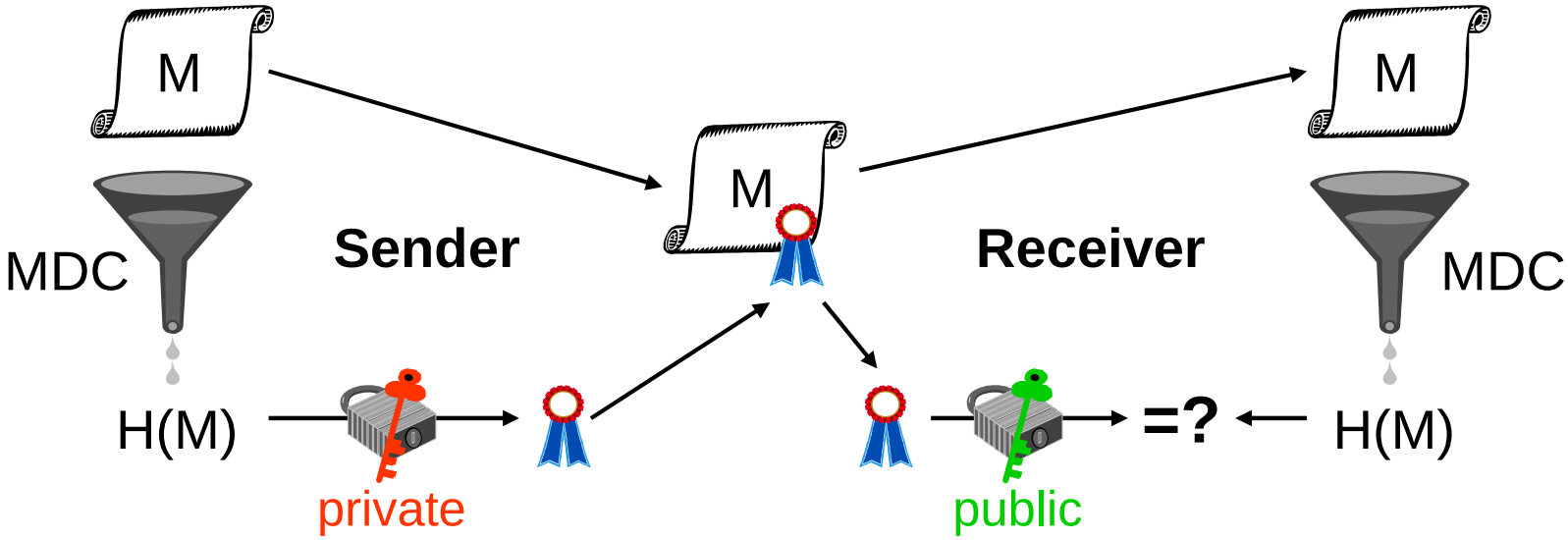
- Signatures:
  - Digital signatures
  - Achieved by asymmetric encryption
    - RSA
    - DAS (Digital Signature Algorithm)
- Certificates:
  - To certify the association between public key and person
    - Certification Authorities (CA) release certificates
    - CAs typically organized hierarchically, they form a Trusted Third Party
  - Authentication of a sender for the signing process
    - Trusted mechanism to relate signature and signee
    - Applied to secure the signature, not the confidentiality

© 2017 Burkhard Stiller

Signatures and Certificates (2)

- Digital signature:
  - Hash value H(M) signed with private key of sender
  - Receiver checks signature with public key of sender
  - Guarantees the commitment
  - Major algorithms: RSA, DSA
  - Minimum key length: 1024 bit

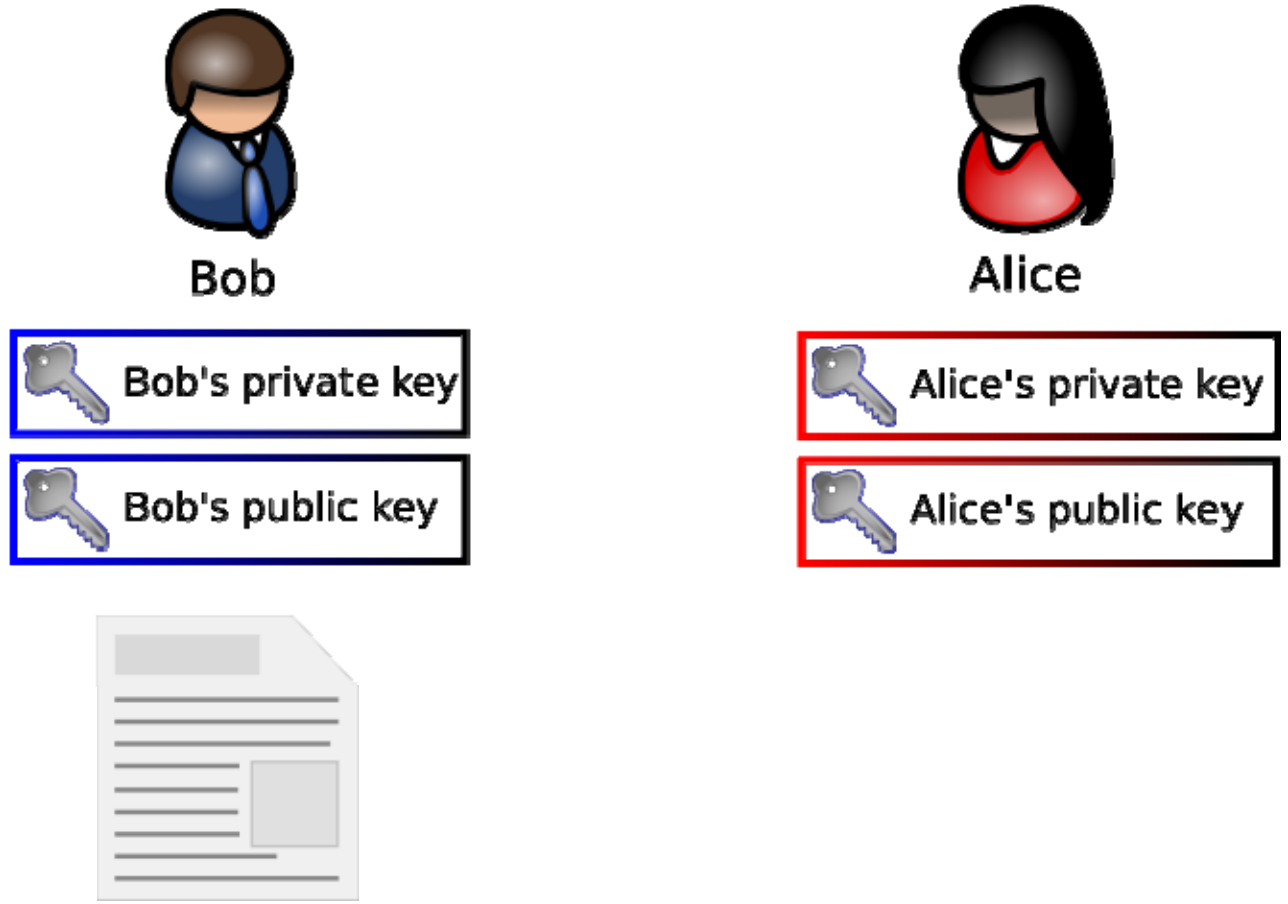
M: Message
MDC: Message Digest Code
(Cryptographic Hash Function)



© 2017 Burkhard Stiller

Example – Signature (1)

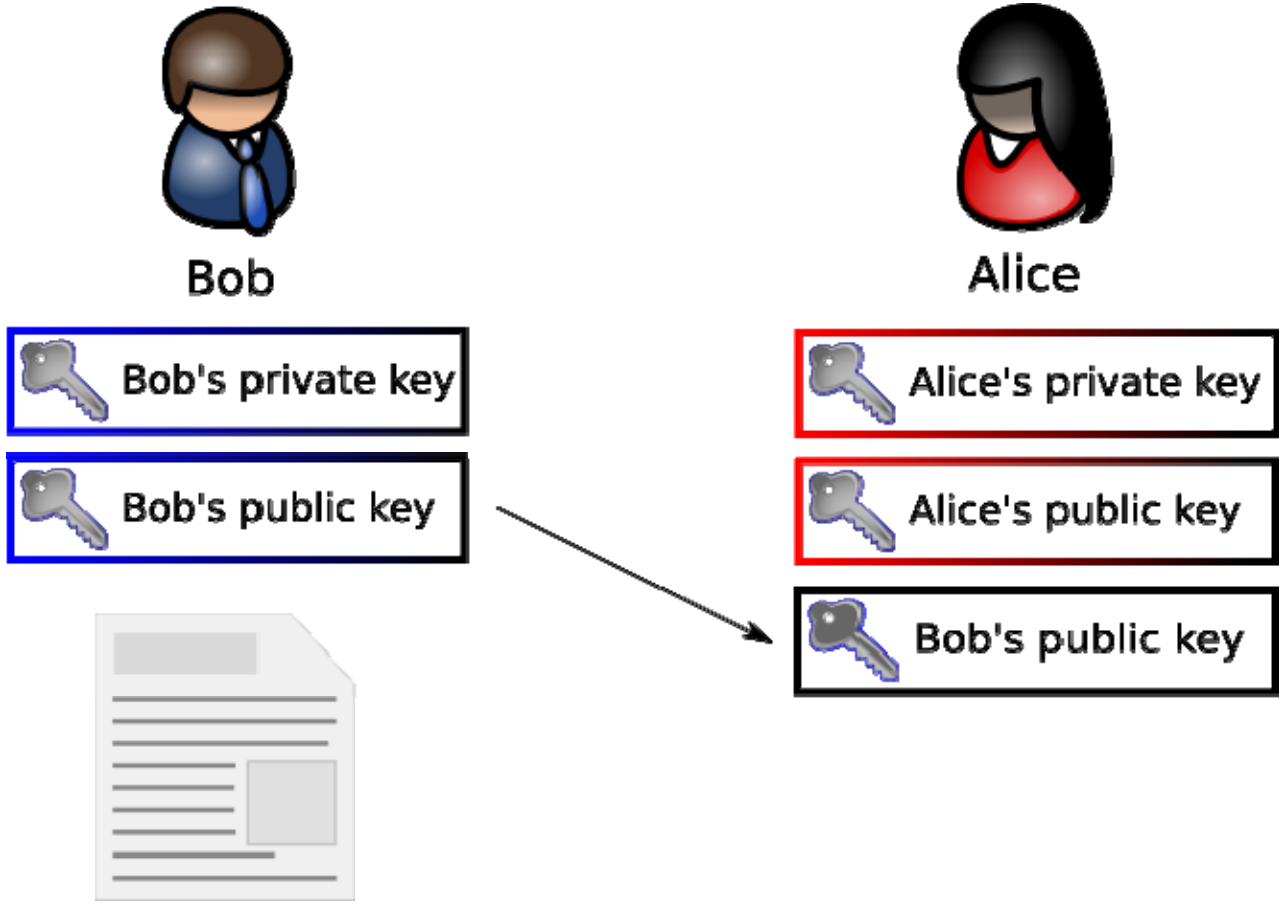
- Digital signature
  - Bob plans to sent a digitally signed document M to Alice
- Initial set-up:



© 2017 Burkhard Stiller

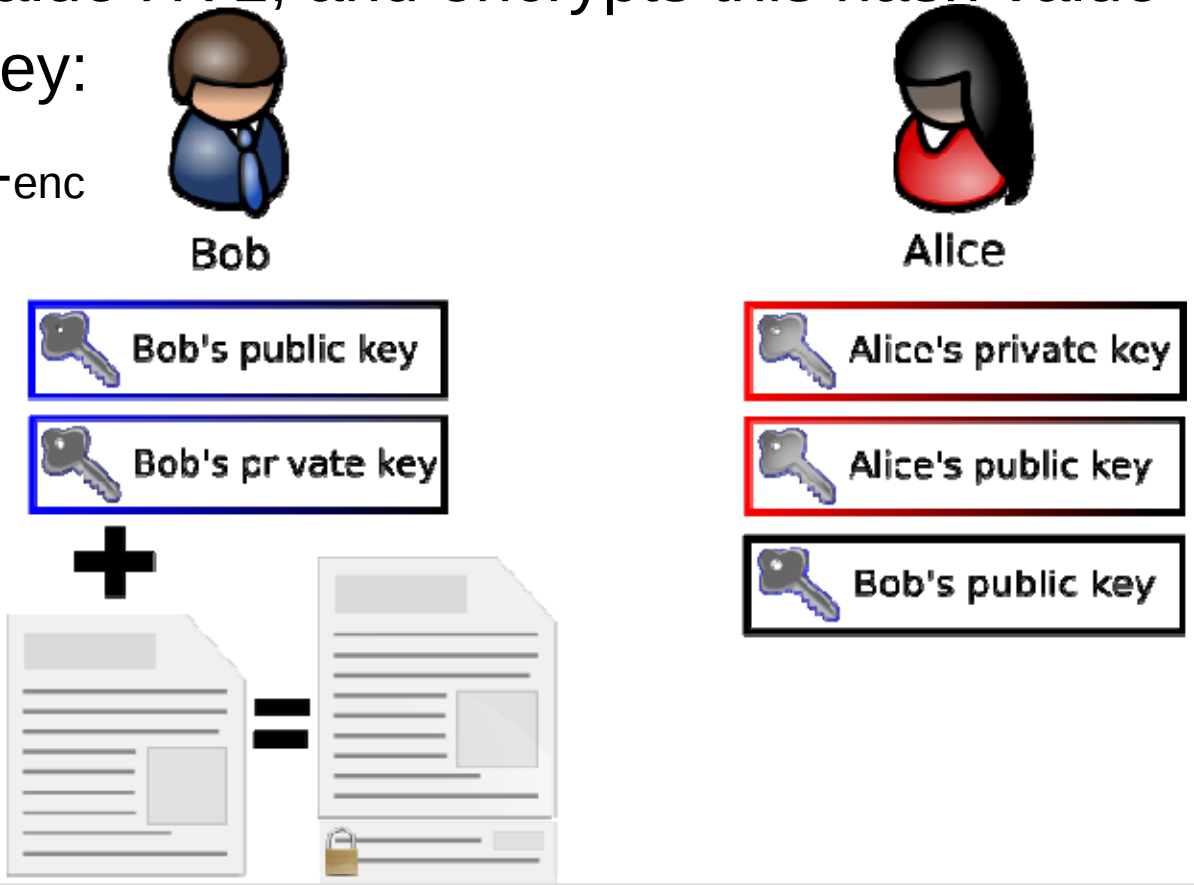
Example – Signature (2)

- Key exchange:
  - Bob hands over to Alice his public key (K-Bob$_{pub}$)



Example – Signature (3)

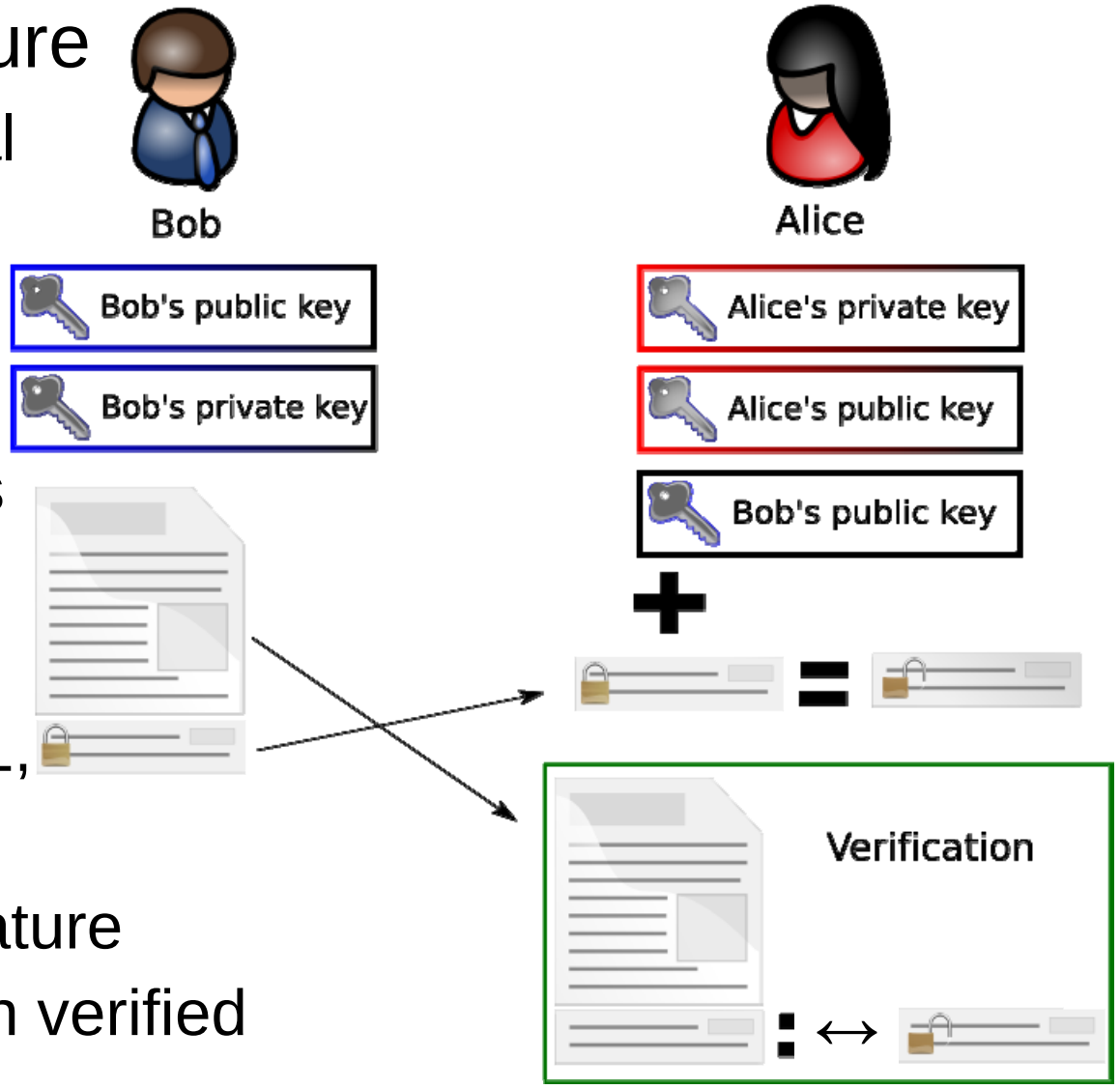
- Bob signs the document
  - Bob applies a cryptographic hash function H to the document M, obtaining a hash value HV1, and encrypts this hash value HV1 with his private key:
    $K\text{-}Bob_{priv}\ (HV1) = HV1_{enc}$



© 2017 Burkhard Stiller

Example – Signature (4)

- Alice verifies the signature
  1. Alice applies the identical hash function H onto the document M and obtains the hash value HV2
  2. Alice decrypts with Bob‘s public key the encrypted hash value $HV1_{enc}$: $K\text{-}Bob_{pub}$ ($HV1_{enc}$) = HV1, and obtains HV1
  3. Iff HV1 ↔ HV2, the signature on document M has been verified



Example – Signature (5)

- Signed e-mail via the public domain tool PGP (Pretty Good Privacy):

```
-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Dieses Mail ist signiert.
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (GNU/Linux)

iD8DBQFHYACBQvL3WYjGY/sRAuyBAJ0UsM/rdXyrqTez79gybb
RStb0SJgCg0xb+dqOTAsRvM8O6w1d7V+FPTrM==oxnW
-----END PGP SIGNATURE-----
```

© 2017 Burkhard Stiller

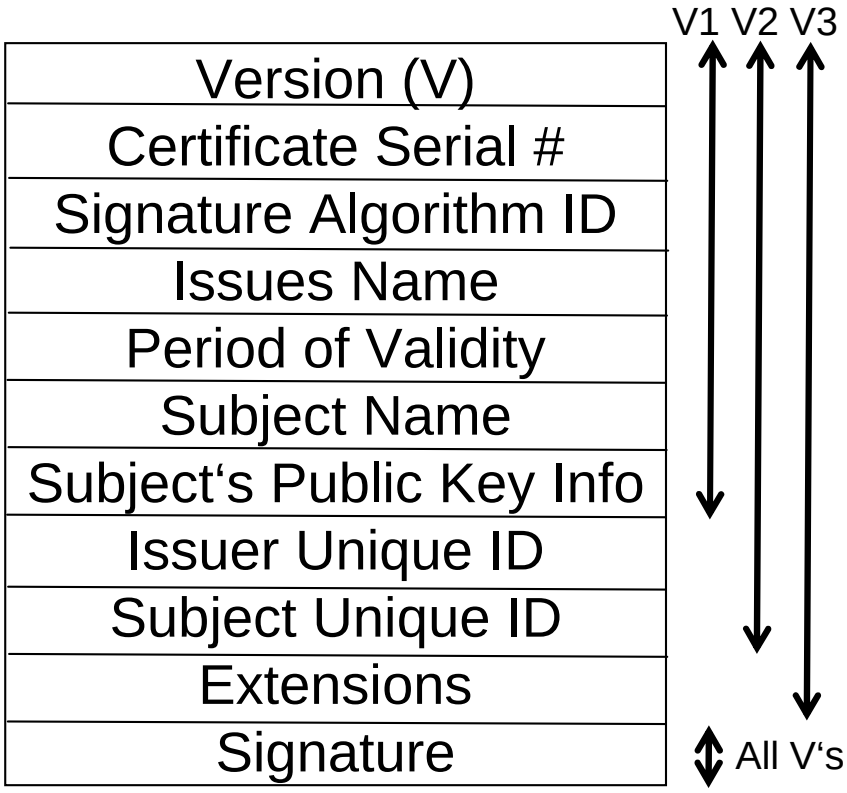
Signatures and Certificates (1)

- Signatures:
  - Digital signatures
  - Achieved by asymmetric encryption
    - RSA
    - DAS (Digital Signature Algorithm)
- Certificates:
  - To certify the association between public key and person
    - Certification Authorities (CA) release certificates
    - CAs typically organized hierarchically, they form a Trusted Third Party
  - Authentication of a sender for the signing process
    - Trusted mechanism to relate signature and signee
    - Applied to secure the signature, not the confidentiality

© 2017 Burkhard Stiller

Standard for Certificates

- ITU-T X.509 Certificates
  - International Telecommunications Union – Telecom Sector



X.509 Certificate

Signature Algorithm ID
Issues Name
This Update Date
Next Update Date
Revoked Certificate #1
…
Revoked Certificate #n
Signature

X.509 Certificate Revocation List

Subject: Distinguished Name, Public Key
Issuer: Distinguished Name, Signature
Period: Not before/not after date
Signature: Algorithm, parameters, encrypted

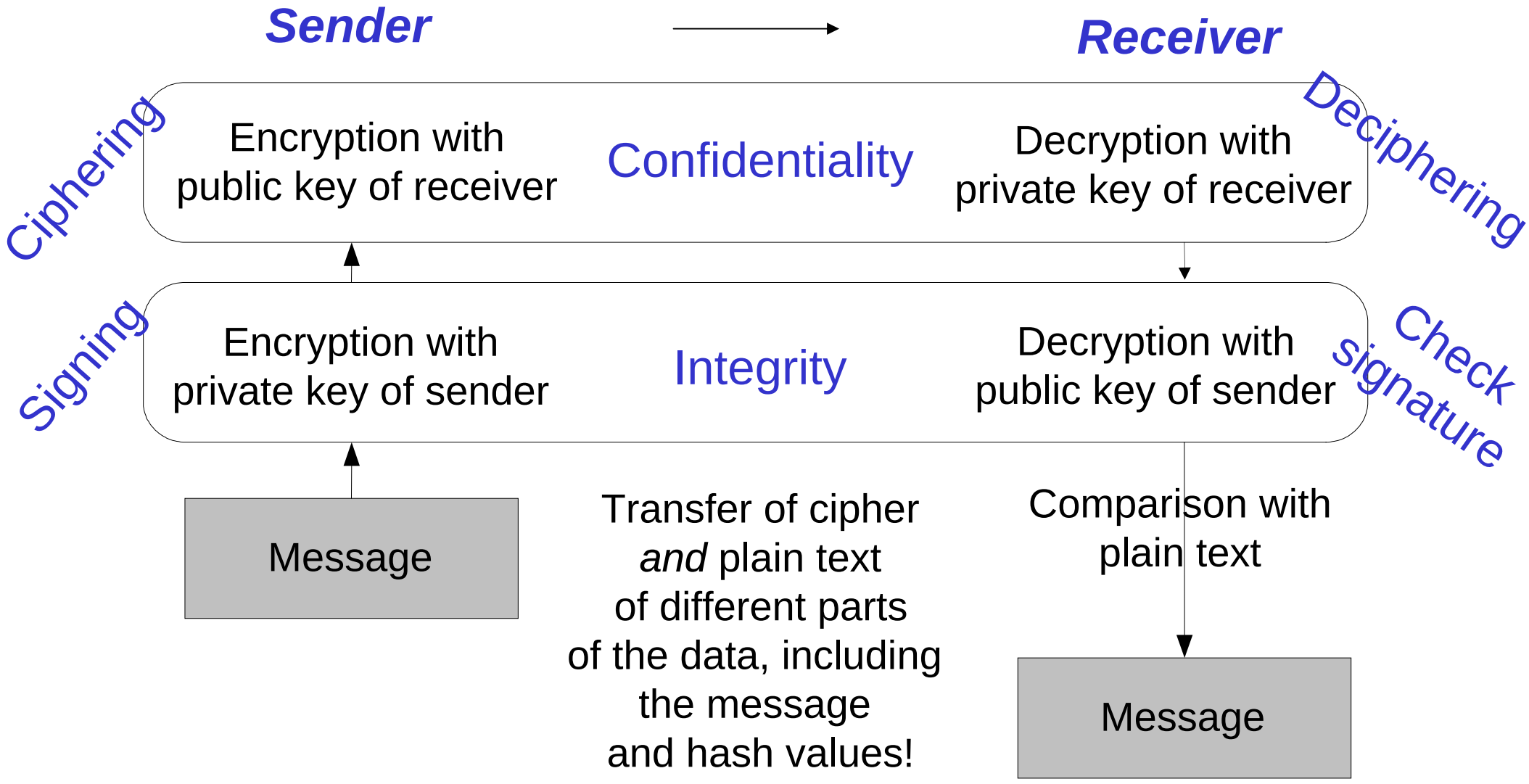
© 2017 Burkhard Stiller

Revocation Procedure for Certificates

- A Certificate Authority (CA) not only generates certificates it needs to revoke them as well.
  - Private key or CA was compromised
  - Service the certificate grants access to may not exist any more
  - CA may have erroneous certificates
- An expiration time does not solve these problems
- Solution: Certificate Revocation Lists (CRL)
  - Lists all certificates being revoked, not expired!
- Problems (examples):
  - CRLs grow large quickly!
  - Periodical CRL distribution, which periodicity?

© 2017 Burkhard Stiller

Relation between Ciphering and Signing

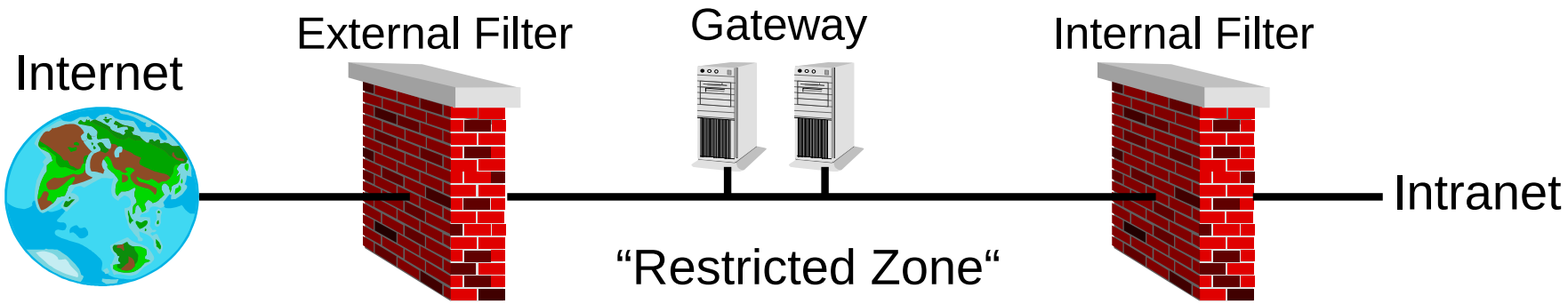


© 2017 Burkhard Stiller

ifi

Access Control

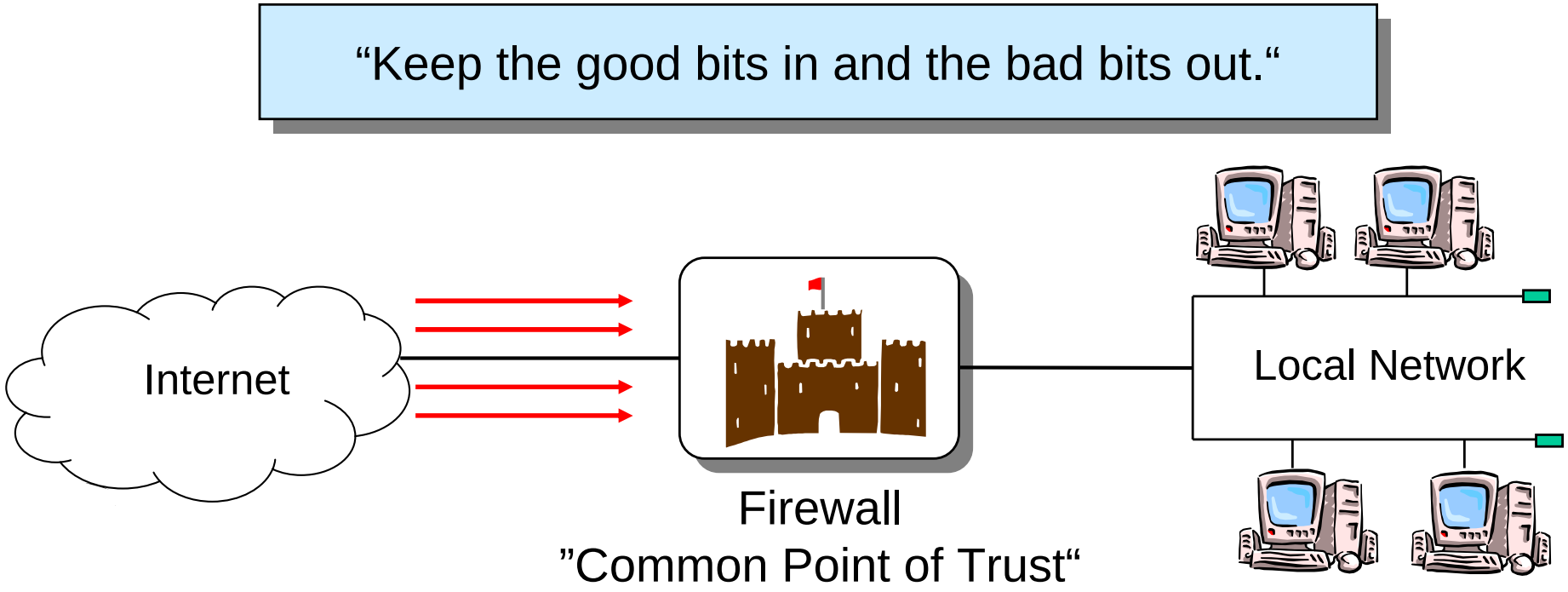
- Based on applications: Model of access rights (AR)
  - Examples: Unix/NT file system AR, SNMP objects AR
- Based on network/transport layer: Firewalls
  - Packet filter based on source/destination address and ports (TCP/UDP)
  - Topology-driven ingress/egress filtering
  - Gateways with access control and logging
  - Use of private IP addresses and address translations (NAT)



© 2017 Burkhard Stiller

Firewalls (1)

- Affiliations, government agencies, universities, private persons are connected via TCP/IP similarly
- Goal: Securing the local network
  - Mainly addressing external active and passive attacks



Firewalls (2)

- Most secure solution:
  - Physical separation of internal and external hosts

- Practical networking solution:
  - Firewalls located in routers:
    - Filtering based on IP address and port number possible
    - Additional logging of information useful
    - Example: Filtering of port 80 packets will stop the access to a WWW server hidden behind the firewall.
  - Application-level gateways:
    - Translation between two systems, *e.g.*, e-mail systems
    - Dynamic mapping of IP address onto multiple hosts

© 2017 Burkhard Stiller

Firewalls (3)

- Example:

Source Address	Destination Address	Service	Action	Protocol
all	Web server	http	accept	short
Intranet	Intranet	smtp	encrypt	normal
Intranet	all	http	accept	short
Extranet	Intranet	smtp, http	accept, Virus scan	normal
all	all	all	delete	Alarm, long

- Notes:
  - Actions only in accordance with addresses and applications
  - Dedicated firewall solutions, *e.g.*, hardware boxes or routers, available with good performance

© 2017 Burkhard Stiller

Secure Shell (SSH)

- Task: Secure and remote host usage (remote login)
  - rsh/rlogin do not offer an authentication
  - telnet transfers passwords insecure
- Operation:
  - Exchange of session key based on Diffie-Hellmann key exchange, server authentication (digital signature)
  - Symmetric encryption plus MAC for all packets
  - User authentication based on digital signature or password
  - Options:
    - Encrypted data transfer by scp
    - Encrypted tunnel for distinct TCP ports

MAC: Message Authentication Code

- Versions 1.0/2.0 not compatible (http://www.ssh.fi)

© 2017 Burkhard Stiller

Secure Socket Layer (SSL) (1)

- Task: Encryption/integrity for transfers over dedicated sockets
  - Mainly applied to encrypted HTTP connections
- Operation:
  - Exchange of session key based on Diffie-Hellmann (DH)
  - Optional server/user authentication (digital signature)
  - Symmetric encryption plus MAC for all packets

MAC: Message Authentication Code

- Versions:
  - Netscape: SSL V1.0 up to SSL V3.0
  - Transport Layer Security (RFC 2246) based on SSL V3.0

© 2017 Burkhard Stiller

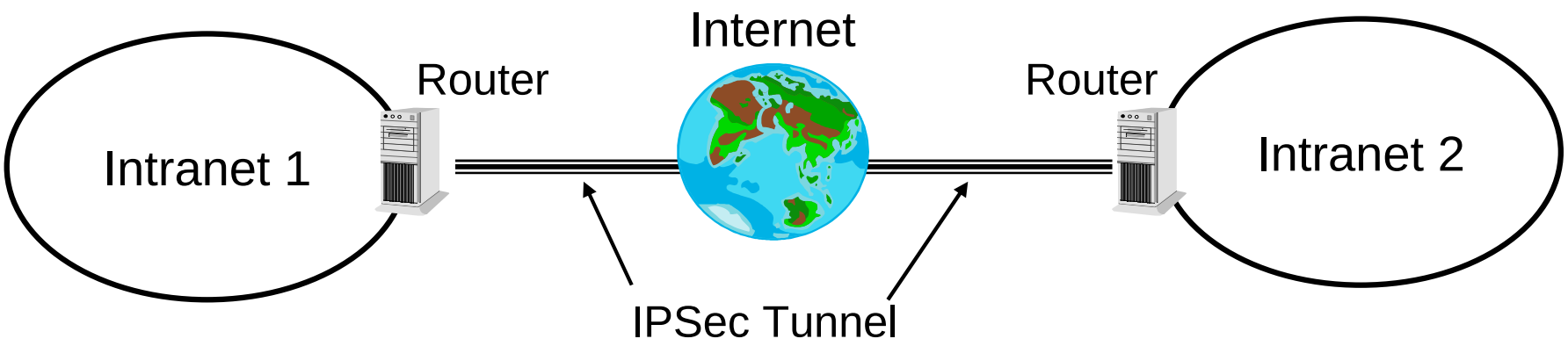
Secure Socket Layer (SSL) (2)

- Remark: Security products have undergone a stringent export and selling policy.
- History:
  - Fortify for Netscape (http://www.fortify.net):
    - Patch for international (non-US) Netscape versions 3 and 4
    - Symmetrical encryption with 128 (instead of 40) bit key length
    - Since 02/2000 not required any more
  - IE and Netscape support:
    - SSL 1.0-3.0
    - TLS 1.0
    - 128 bit key length
    - US export regulations relaxed

© 2017 Burkhard Stiller

Tunneling

- General:
  - Embedding an IP packet into a second IP packet
- Specific:
  - Encryption of IP packets at sending tunnel end-point
  - Decryption of IP packets at receiving tunnel end-point
  - Automated processing, valid for dedicated applications



© 2017 Burkhard Stiller

IP Security (IPSec)

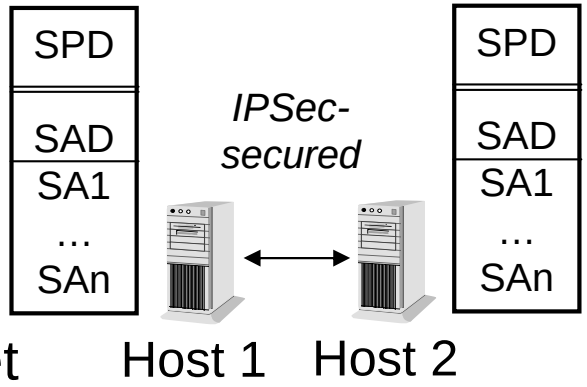
- Goal: Secure transmission of data transferred between any two locations
- Operation:
  - Tunneling of IP packets
  - Packets may not be readable within the Internet by decryption/encryption
- Example IP Security:
  - MAC and/or symmetrical encryption
  - Two security services and protocols
  - Software packages:
    - FreeS/WAN (http://www.freeswan.org)

MAC: Message Authentication Code

© 2017 Burkhard Stiller

IPSec — Communication

- IPSec Architecture includes:
  - Security Policy Database (SPD):
    - Defines, which IP packets shall use IPSec
    - Determines details of the Security Association
  - Security Association (SA):
    - Defines security services to be applied to packet
    - Identified by Security Parameter Index (SPI)
  - Security Association Database (SAD):
    - Stores active SAs, which was generated on-demand
    - Stores security protocol-relevant data, such as key and mechanism
  - Security Protocols: AH and ESP
    - Provide the security service selected for an SA
    - Utilize parameter information stored in SAD



© 2017 Burkhard Stiller

IPSec — AH and ESP (1)

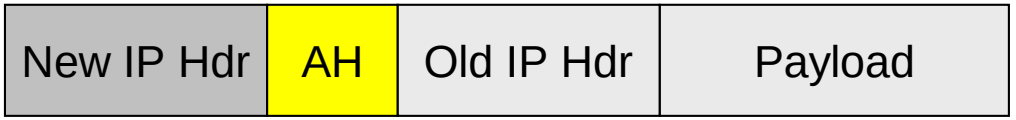
- Authentication Header (AH):
  - Authentication and data integrity

RFC 2402

Transport mode

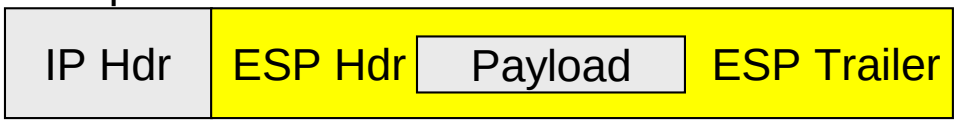


Tunnel mode



- Transport mode:
  - No change of addresses in case of direct communication

Transport mode



- Tunnel mode:
  - New IP addresses between partner

Tunnel mode

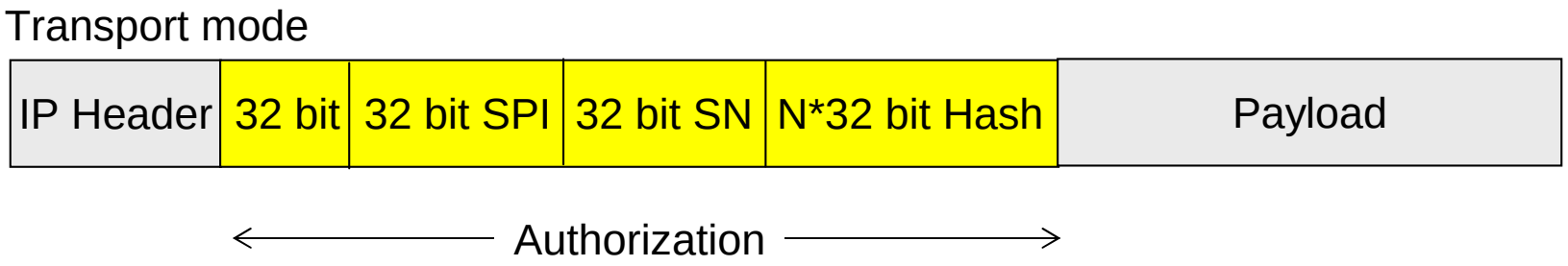


- Encapsulating Security Payload (ESP):
  - Authentication, data integrity, and privacy

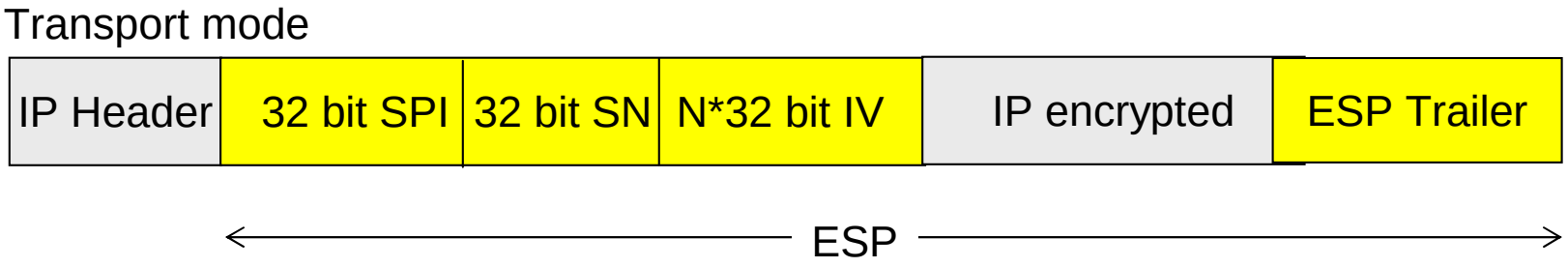
RFC 2406

IPSec — AH and ESP (2)

- IPSec IP packet: AH in Transport Mode:



- IPSec IP packet: ESP in Transport Mode:



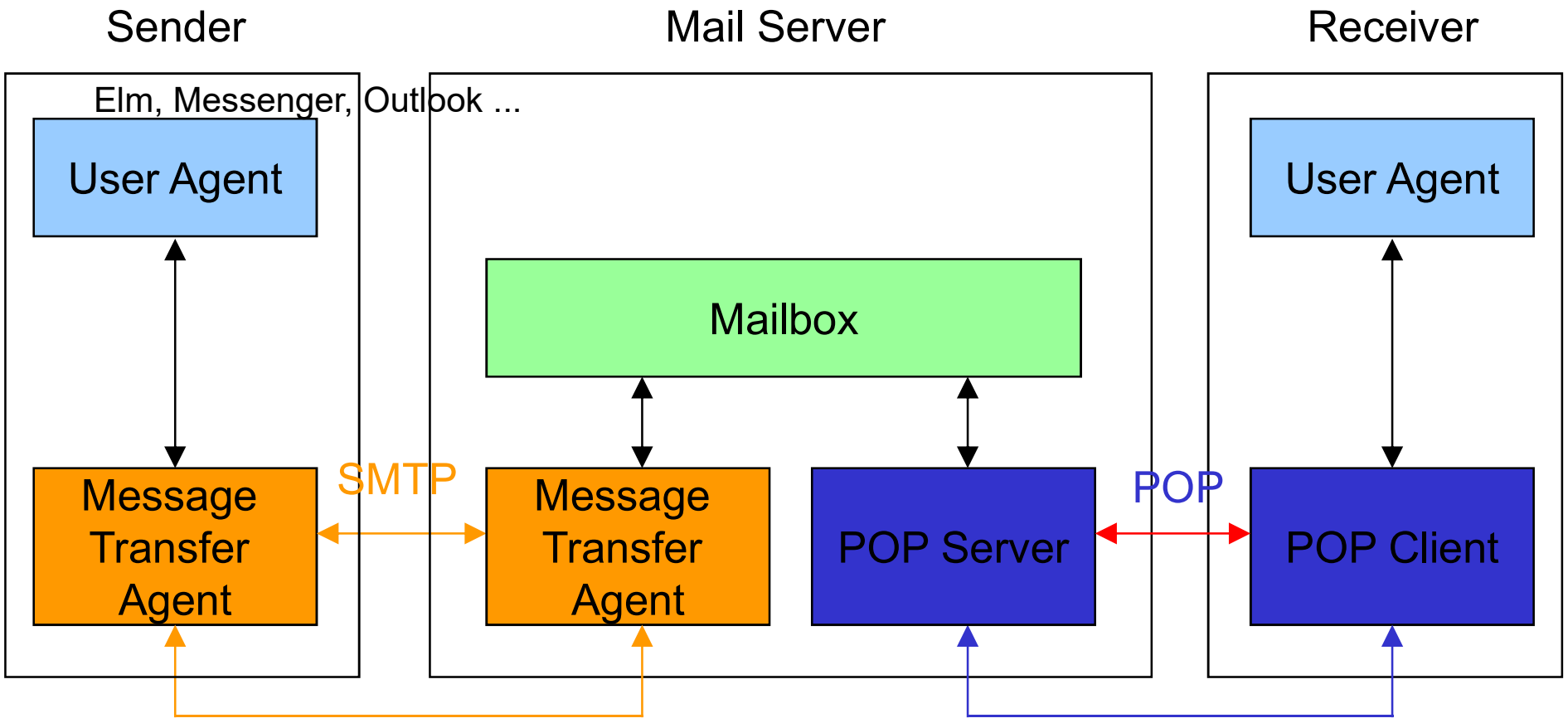
13. Applications and Mechanisms

E-Mail
Domain Name System (DNS)
WWW (World Wide Web)

© 2017 Burkhard Stiller

E-mail ...

Electronic Mail (E-mail) System (1)



Multiple mail gateways may exist between sender and mail server

POP: Post Office Protocol
SMTP: Simple Mail Transfer Protocol

© 2017 Burkhard Stiller

ifi

Electronic Mail (E-mail) System (2)

- Goal:
  - International exchange of electronic messages between humans
- Major characteristic:
  - Support of asynchronous sender and receiver behavior
  - Store-and-forward switching
- Basic functions:
  - Writing of e-mails
  - Transfer to destination
  - Indication of success/erroneous cases/received messages
  - Storage of messages
- Protocols: SMTP (Internet) and X.400 (OSI)

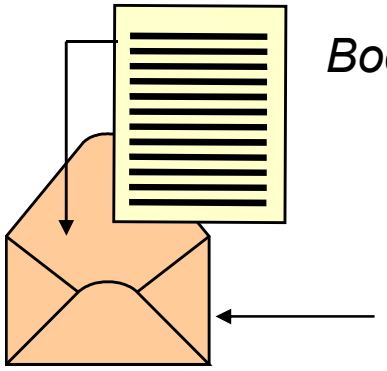
© 2017 Burkhard Stiller

Simple Mail Transfer Protocol (SMTP)

- Direct TCP connections between mail server:
  - Sending multiple messages within TCP connection
  - TCP connection used for backward direction
- Messages:
  - Contain ASCII text only
  - Maximum message size may be limited by 64 kByte
  - Message formats:
    - Text (RFC 822)
    - Multi-purpose Internet Mail Extensions (MIME)
- Exchange of messages between client and server:
  - HELO: Introduction
  - MAIL FROM: Information about sender
  - RCPT TO: Information about receiver
  - DATA: Sending messages
  - QUIT: Finalization
  - VRFY: Verification of user name
  - EXPN: Expand, information about lists
- Receiver acknowledges every message

SMTP-based E-mails

- Envelope
  - Contains information to transport a message to the receiver
  - Addressing based on DNS, *e.g.*, stiller@acm.org
  - Interpretation by Message Transfer Agent (MTA)
- Header
  - Contains additional field, *e.g.*, subject, CC
  - Interpretation of User Agent (UA)
  - SMTP integrates envelope fields in header
- Body
  - Contains message



Client Operation

- Problem
  - E-mail server always have to be able to receive messages
  - PCs, laptops, though, are connected temporarily
- Solution
  - E-mails are transferred from a server to the client based on a client protocol and stored over there
- Client protocols
  - Post Office Protocol Version 3 (POP3)
  - Interactive Mail Access Protocol (IMAP)
  - Hyper-text Transfer Protocol (HTTP)-based:
    - No dedicated software, a simple WWW browser does the job
    - Slow processing of scripts

© 2017 Burkhard Stiller

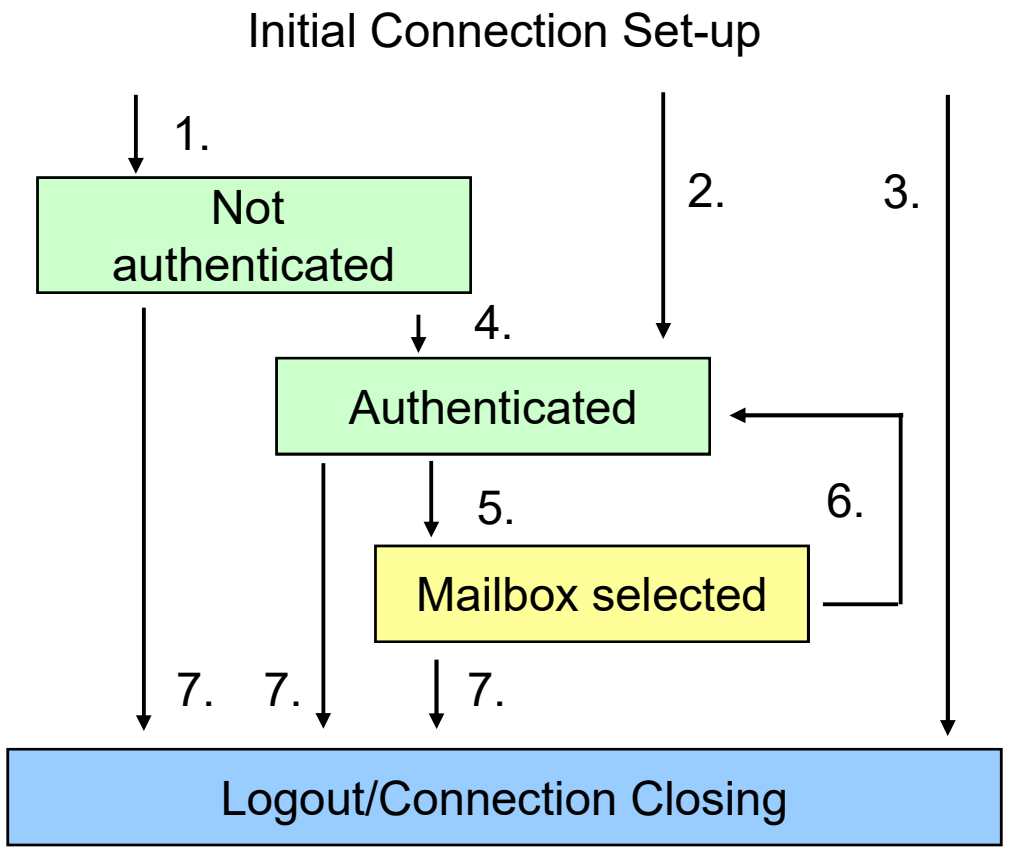
Internet Mail Access Protocol (IMAP) (1)

- Support of nomadic users
- Definition of mail folders on server
- Commands for remote folder manipulation available
- E-mails may be stored on server side
- Partly down-load of e-mails, *e.g.*, header only
- IMAP server states:
  - Non-authenticated → Authentication required
  - Authenticated → Selection of a directory/folder
  - Selected → Manipulate and down-load of e-mails
  - Logout: Closing-down of session

© 2017 Burkhard Stiller

Internet Mail Access Protocol (IMAP) (2)

- Connection set-up
  - Regular “OK Greeting”
  - No authentication “PREAUTH Greeting”
    - Authentication performed external
  - Set-up rejection
  - Successful authentication
  - Successful selection of mailbox
  - Closing of mailbox
  - Connection closing/logout



© 2017 Burkhard Stiller

Multipurpose Internet Mail Extensions (MIME)

- Problems of RFC 822-based standards
  - No transmission of binary data or executables
  - ASCII only
  - No country-specific characters, *e.g.*, ä, ö, ü, Â, È, Ü, ø, ...
  - Mail servers with limited message size processing capabilities

- Multipurpose Internet Mail Extensions (MIME)
  - Additional header information:
    - Content type: Text, multi-part, message, application, image, audio, ...
    - MIME version
  - Definition of content transfer encoding:
    - Mechanisms to code binary data by ASCII characters
    - Transfer syntax: Base64, Quoted Printable, 7 bit ASCII, 8 bit ASCII

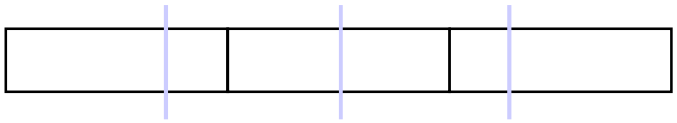
© 2017 Burkhard Stiller

MIME Transfer Encoding

- 7 bit: ASCII characters, short rows
- 8 bit: ASCII and non-ASCII characters, short rows

No coding

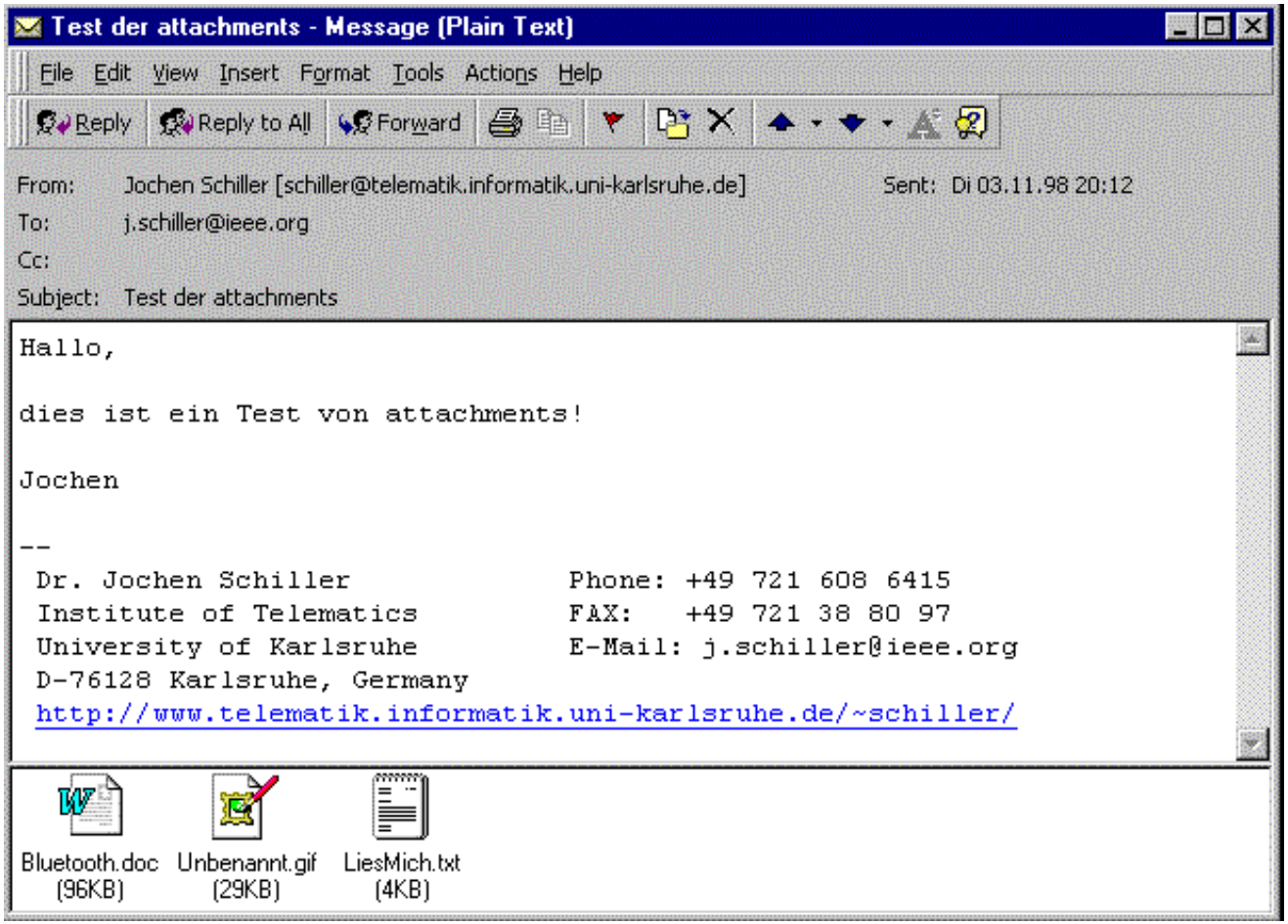
- Binary:
  - ASCII and non-ASCII characters, long rows
- Quoted printable:
  - ASCII characters plus hex codes
  - *E.g.*, representation of form feed (decimal 12): “=0C“
  - Still readable
- Base 64:
  - Mapping of 8 bit binary values onto ASCII characters:
    - 24 Bits (3 Byte) → 4*6 Bits → 4 ASCII Char (restricted to values of 0 … 63)



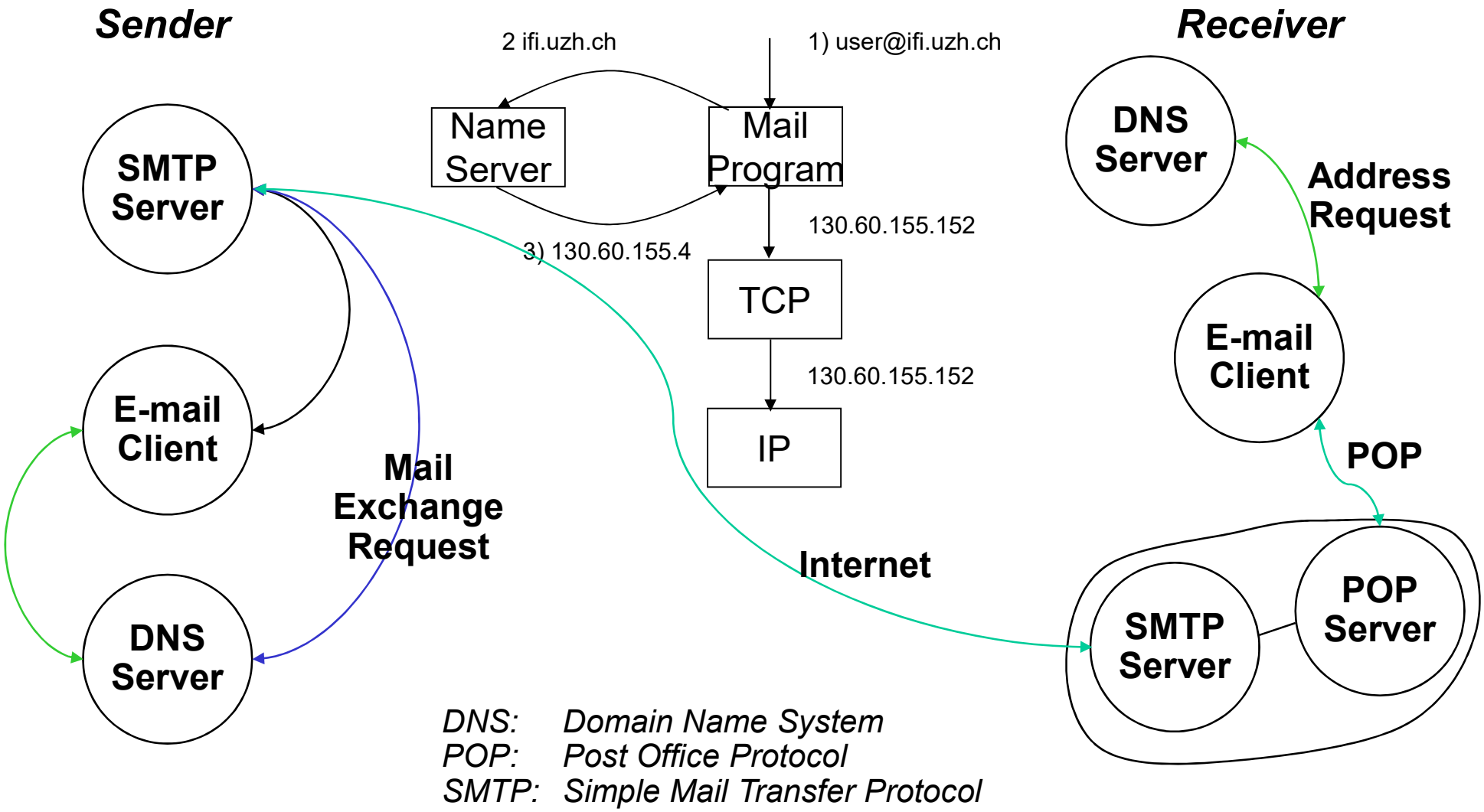
© 2017 Burkhard Stiller

MIME-based E-mail Example

- MIME E-mail:
  - Text
  - Word attachment
  - Image
  - Text attachment
- Graphical User Interface: MS-Outlook



DNS Motivation — E-mail System



© 2017 Burkhard Stiller

Domain Name System (DNS) — Overview

- "Addresses" versus "names":
  - Addresses are fixed length
    - Tied to routing
    - Easy for computers to process
    - Often changed (*e.g.*, change of server machine, re-use as client)
    - Transport addresses 2-tupel (port, IP address)
  - Names are variable length
    - Mnemonic
    - Easy for humans to remember (*e.g.*, hosts, mail boxes, servers)
- Name Space:
  - Defines set of possible names
- DNS hides addresses from Internet applications

RFC 1034
RFC 1035

Name Space

- Consists of a set of names to value bindings
- Flat name space:
  - Centralized co-ordination to avoid overlaps
  - Not scalable for many hosts, max alphabet, max length
  - Requests addressed to a centralized directory not practicable
  - World-wide consistency of copies difficult to achieve
- Hierarchical name space:
  - Hierarchical names enable a structured name space
  - Localization of objects made simpler
  - Distribution of control (name assignments) made easier
  - Maintenance of names stored distributed

Domain Names in the Internet

- DNS defines syntax and rules for the delegation of a name authority and the implementation of a distributed system to map names onto addresses
- DNS-conforming names:
  - "Sub-domain. … .Sub-domain.Domain": ifi.uzh.ch

Top-level domains (TLD):

uTLD, sTLD:

Various special ones

cf. next two slides

ccTLD:

cc 2 letter country code ISO 3166 (cc – country code) (*e.g.*, "CH" Switzerland, "DE" Germany, "UK" Great Britain)

Top-level Domains (Unsponsored: uTLD)

- **.arpa** arpanet Address and Routing Parameter Area
- **.biz** business for companies, open to all
- **.com** commercial for US companies, open to all
- **.info** information for information providers, open to all
- **.int** international for international organizations
- **.name** names for private persons
- **.net** network for network organization, open to all
- **.org** organization for non-commercial organizations, open to all
- **.pro** professionals for lawyer, tax accountants, doctors, engineers (only for U.S.A., Canada, Germany, U.K.)

© 2017 Burkhard Stiller

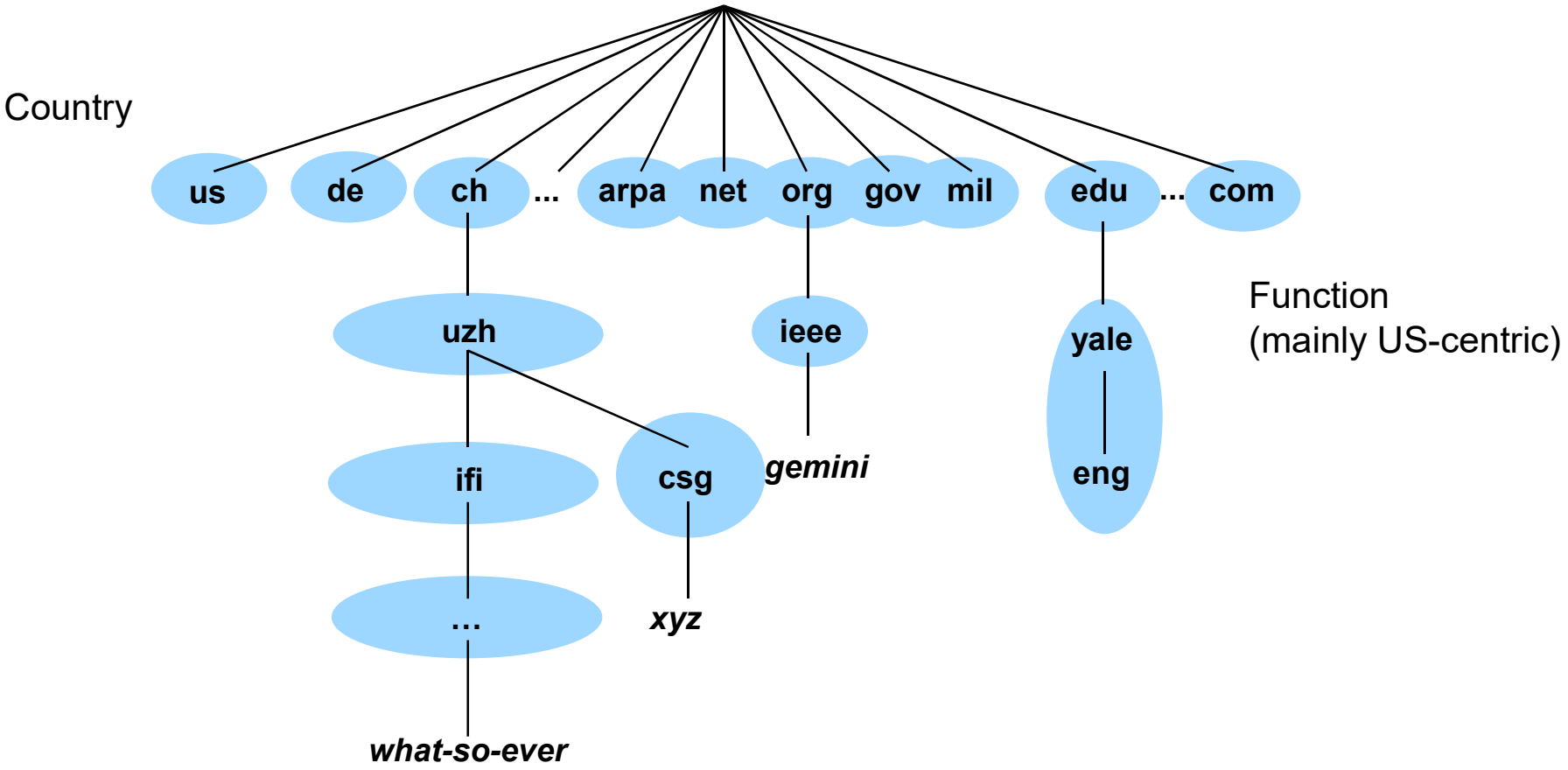
Top-level Domains (Sponsored: sTLD)

- **.aero** aeronautics for SITA only
- **.asia** asia ICANN region Asia/Australia/Pacific
- **.cat** catalan for katalan language and culture
- **.coop** cooperatives, for .com LLC only
- **.edu** educational for "*post-secondary*" accredited US
- **.gov** government only for Government of the USA
- **.jobs** jobs only for companies with job offers
- **.mil** military for US military only
- **.mobi** mobile for services within mobile devices
- **.museum** museums, for association members only
- **.tel** telephone simplified calling of persons/companies
- **.travel** travel for travel organizations/companies
- **.post** postal for all post and logistic companies (plan)

© 2017 Burkhard Stiller

DNS Name Space

- Hierarchical name space separates domains
- Zone represents a sub-tree of name tree



DNS Name Server (NS)

- Every zone maintains
  - One primary NS
  - Minimum two secondary NSs

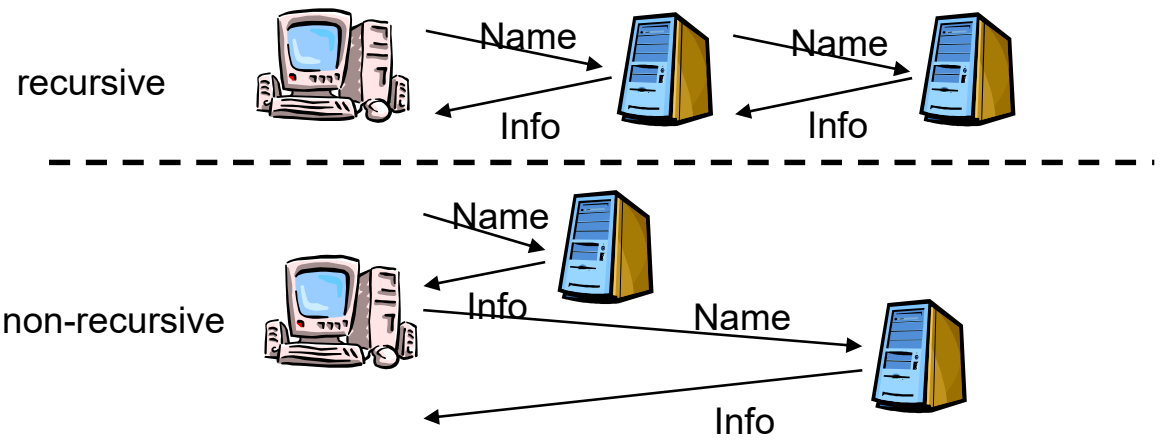
- Operation:
  - Every NS knows a sub-part of the name space
  - Every NS knows those IP addresses of all NS in the directly attached sub-domains
  - Every NS caches already known addresses
  - Secondary NS maintain periodic updates (zone transfer) of local data bases (driven by the primary NS)

© 2017 Burkhard Stiller

Name Resolution

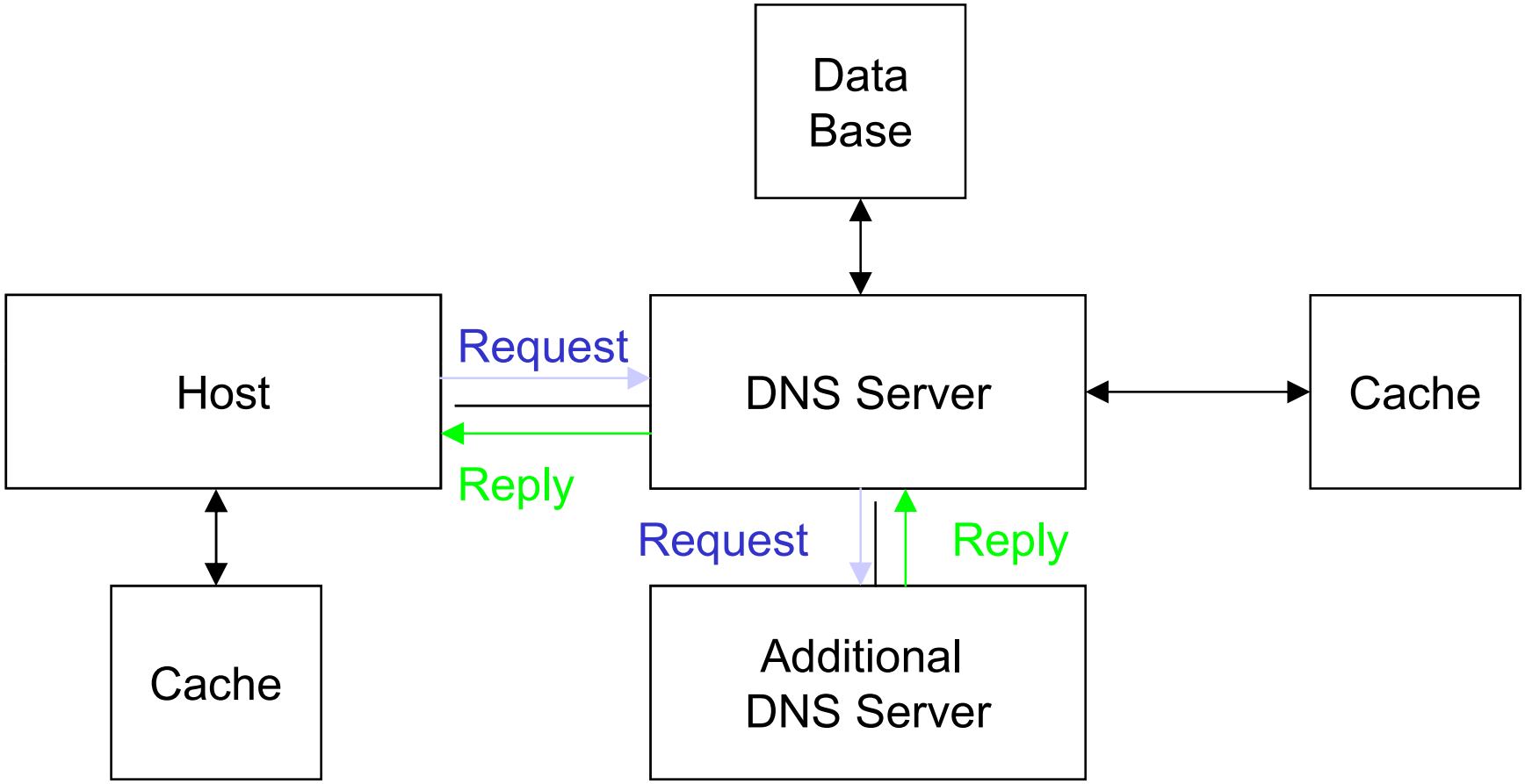
- Name resolution starts at tree root
- Name resolution initiated by DNS resolver within application

- Requests:
  - Addressed to dedicated NS
  - Recursively addressing the full DNS



- For incoming request NS checks local sub-tree, if not able to resolve, request addressed to higher level NS
  - Resolver needs to know only “his” NS address (/etc/resolv.conf).

DNS Request


Data Base
Host
Request
DNS Server
Cache
Reply
Request
Reply
Cache
Additional DNS Server

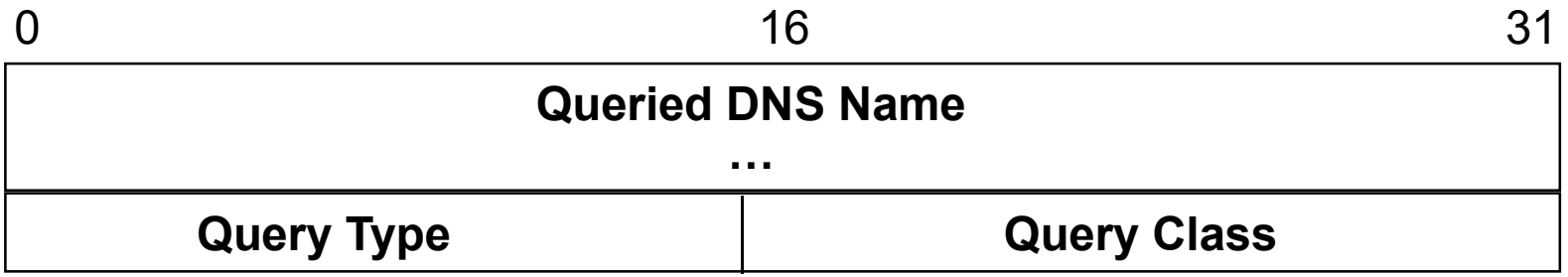
Format of DNS Messages (1)

Query and responses apply similar format:

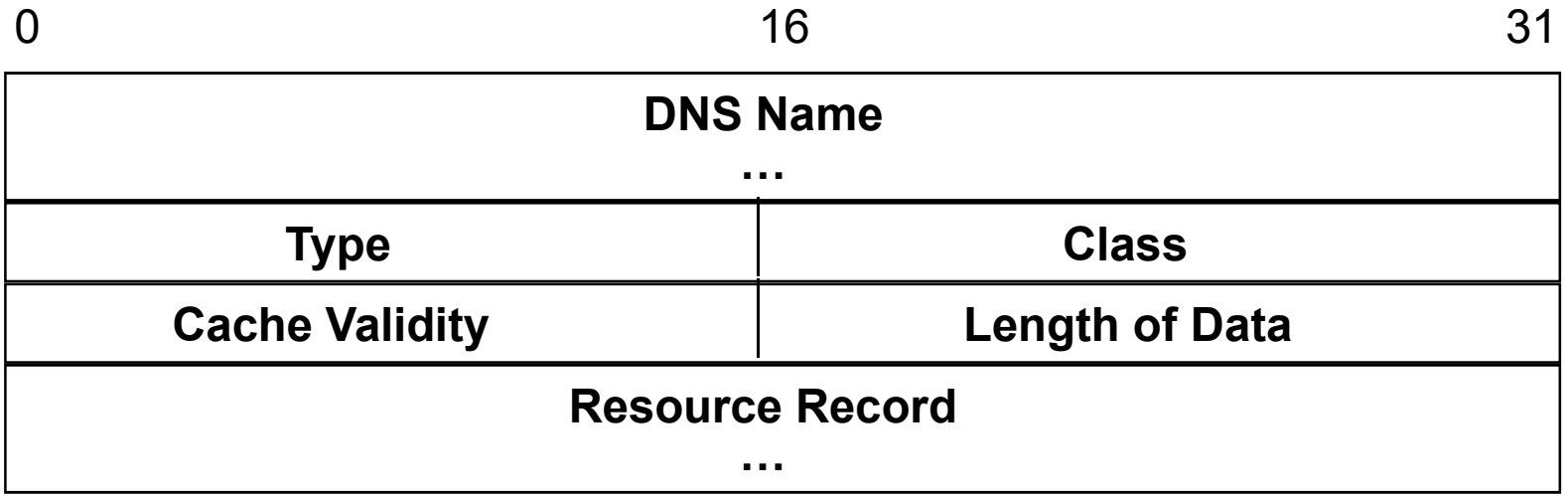
0	16	31
Identification	**Parameter**	
Number of Queries	**Number of Responses**	
Number of Name Authorities	**Number of Additional Data**	
Query Block …		
Reply Block …		
Authority Block …		
Additional Data Block …		

Format of DNS Messages (2)

DNS Query

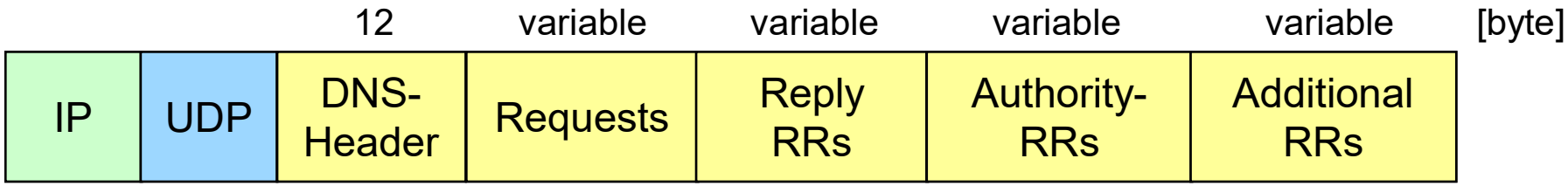


DNS Response



© 2017 Burkhard Stiller

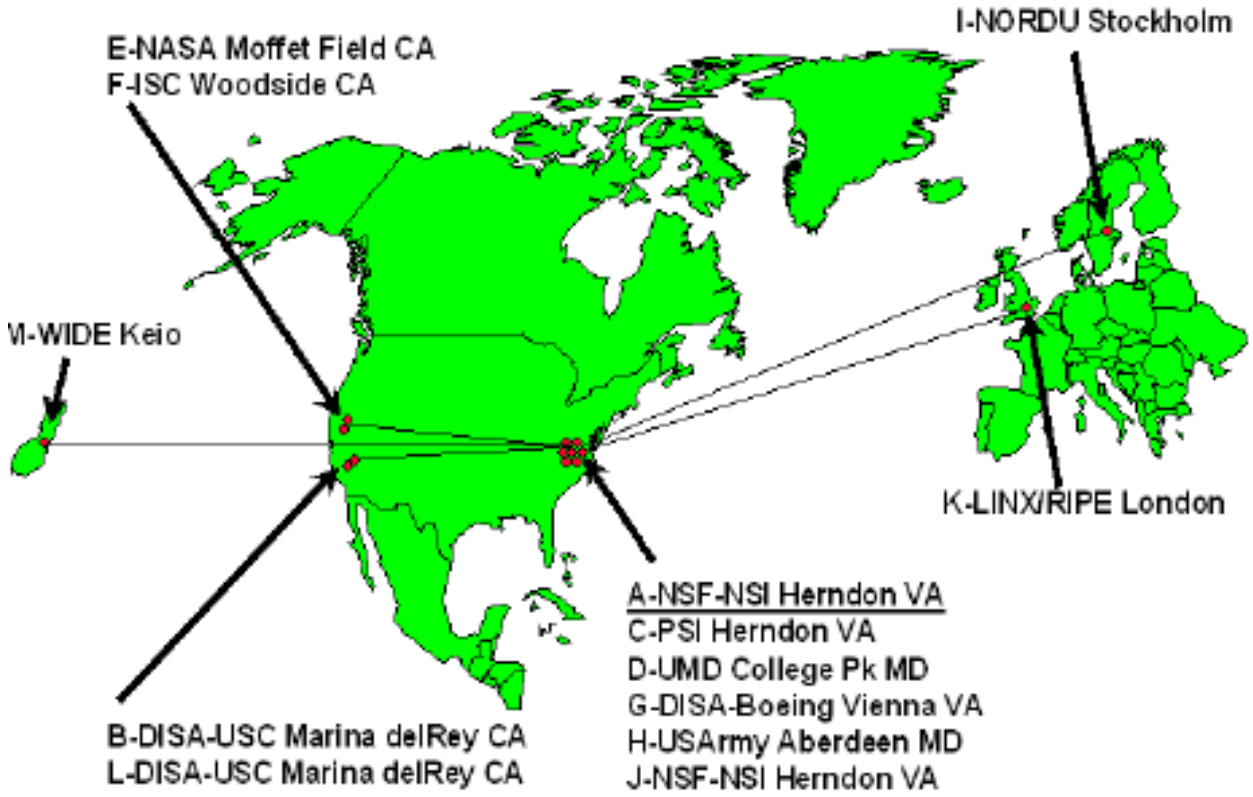
DNS Packet Format



- DNS utilizes UDP:
  - Efficient, no connection set-up or reliability required
- DNS header:
  - Contains number of request, number of fields, and control information
- Requests:
  - Contain DNS names (*e.g.*, www.amazon.de) and type of request (*e.g.*, A, MX, PTR)
- Reply/Authority/Additional RRs (Resource Records):
  - One/multiple resource records with required DNS information
  - Authority RRs: Name(s) of responsible name server(s) for this request

Root Server

- Location of DNS Root Servers
  - http://www.root-servers.org
  - World-wide 13 root servers
  - Root server maintain TLD (Top Level Domain) only

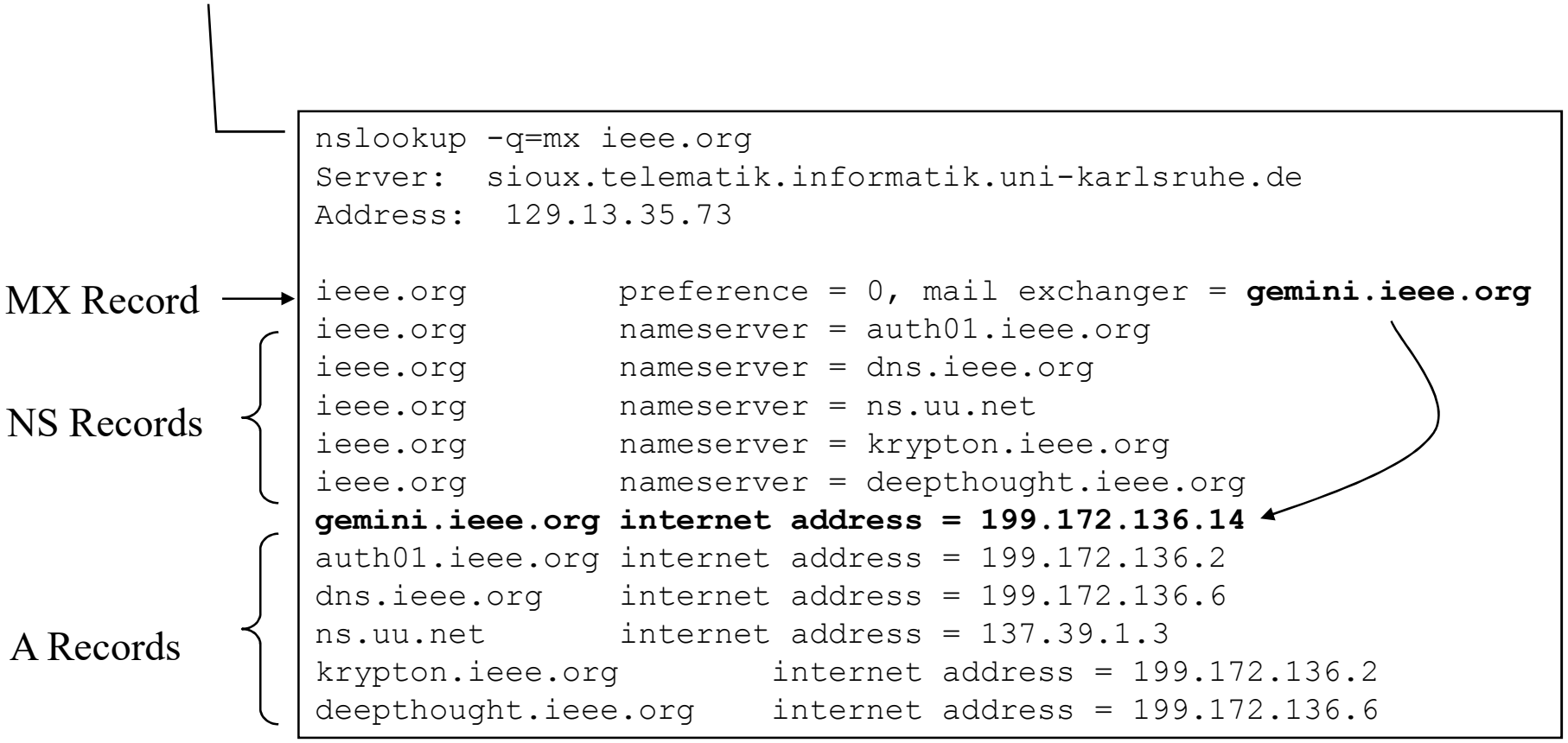


© 2017 Burkhard Stiller

ifi

Name Resolution — Example

– DNS maps names onto information (*e.g.*, IP addresses)
  - **`www.ieee.org` ⇨ `199.172.136.14`**



© 2017 Burkhard Stiller

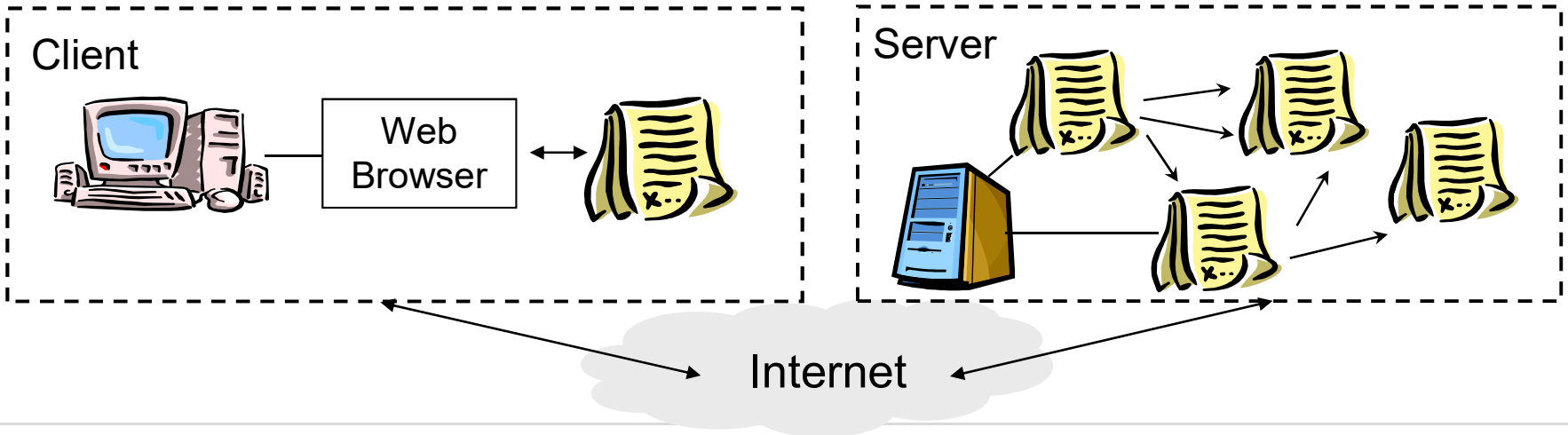
WWW …

World Wide Web (WWW)

- Goal:
  - Simple and world-wide exchange of documents
  - Proposal by Tim Berners-Lee (UK) at CERN, CH
- Implementation:
  - First prototype 1990
  - Graphical version (NEXTStep) and command-line based
- User interface:
  - WWW-client Mosaic by Marc Andreesen und Eric Bina (University of Illinois)
  - X-Windows systems
  - Source code publicly available ⇨ fast distribution
  - Marc Andreesen founded Netscape in 1995

Client/Server-based WWW Architecture

- Client/Server-based architecture:
  - Web browser displaying hypertext documents/hypermedia objects
  - Hyperlinks enable navigation
- Solutions required for:
  - Addressing a web page uniquely
  - Transfer of a web page
  - Description of web page content, especially of hyperlinks

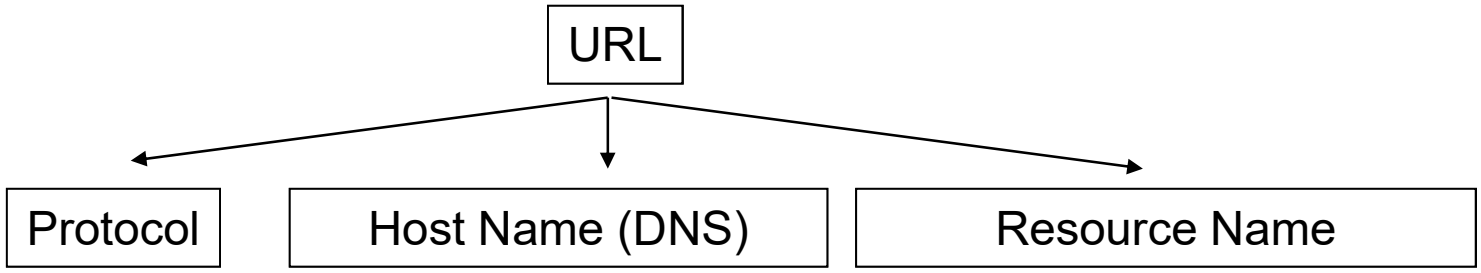


© 2017 Burkhard Stiller

Addressing a Web Document

- A resource description identifies an object to be accessed within a server:
  - WWW: selected web page
  - FTP: to be transmitted file
  - Mail: Receiver of this message

- Uniform Resource Locator (URL):
  - Determines the resource‘s location as a name, the protocol to access this resource, and a name of the resource
    - E.g., http://www.ifi.uzh.ch/



Uniform Resource Locator (URL)

- Compact representation of location and access to Internet resources based on URLs:

<scheme>:<scheme-specific-part>

```
ftp://<user>:<password>@<host>:<cwd1>..
      <cwdN>/<name>;type=<typecode>
http://<host>:<port>/<path>?<searchpart>
mailto:<rfc822-addr-spec>
nntp://<host>:<port>/
      <newsgroup-name>/<article-number>
telnet://<user>:<password>@<host>:<port>
file://<host>/path
```

  - Web browser support a large number of protocols

WWW Application Protocol HTTP

- HTTP (Hypertext Transport Protocol):
  - Version 0.9/1.0 RFC1945, since January 1997 version 1.1 RFC2068
  - Mainly transfer of Web pages
- Characteristics:
  - ASCII-based application protocol (layer 7)
  - Utilized a reliable TCP connection at default port 80
  - Short-lived connection: HTTP server closes connection after a reply on a request of a client has been issued
- Sample commands:
  - **`GET`**: Request of a document
  - **`HEAD`**: Request of header information of this document
  - **`POST`**: Attachment of data to an existing document
  - **`PUT`**: Generation of a document

© 2017 Burkhard Stiller

HTTP Messages

Request (Status) Line	General Header	Request (Response) Header	Entity Header	Entity Body

- Request Line:
  - Message type, requested resource, ...
- Status Line:
  - State information (error messages, HTTP version, ...)
- General Header:
  - Cache control, proxy information, ...
- Request Header:
  - Request- and client information, *e.g.*, acceptable media/coding, ...
- Response Header:
  - Response information (location, server software, retry information, ...)
- Entity Header:
  - Information on resources and body
    (Compression, date of modification, language, length)
- Entity Body:
  - Message body

© 2017 Burkhard Stiller

Example of a HTTP Request/HTTP Reply

- Request:
  - <Command><URL> <Version>
  - Client requests non-cached, current version
- TCP connection already established:
  - Reply line
  - Reply, date, server
  - Encoding information
  - Content type

HTTP Client

```
GET /index.html HTTP/1.1
Host: www.ifi.uzh.ch
Pragma: no-cache
....
```

HTTP Server

```
HTTP/1.1 200 OK
Date: Fri, 24 Sep 1999 09:45:51 GMT
Server: Apache/1.3.6 (Unix)
Transfer-Encoding: chunked
Content-Type: text/html

<HTML>
Structured document (HTML-based)
</HTML>
```

© 2017 Burkhard Stiller

Document Description Language HTML

- HTML (Hypertext Markup Language):
  - HTML documents describe structured text documents
  - HTML tags describe the representation of document parts
    - HTML tags contained as clear text in document
    - *E.g.*, Bold Font
  - Documents are separated in header and body part:
    - Header contains general characteristics of document
    - Body contains “real” content:
      - *E.g.*, Title, subtitle, paragraphs, tables, items, lists
  - Text elements and further documents are concatenated by hyperlinks
  - Non-text parts, *e.g.*, graphs, figures, or video, are inter-linked
  - Advantage: Representation of client (browser) locally decided
- Standardization of version 4.0:
  - Integration of scripting languages and Cascading Style Sheets (CSS)

© 2017 Burkhard Stiller

Example of a HTML Document

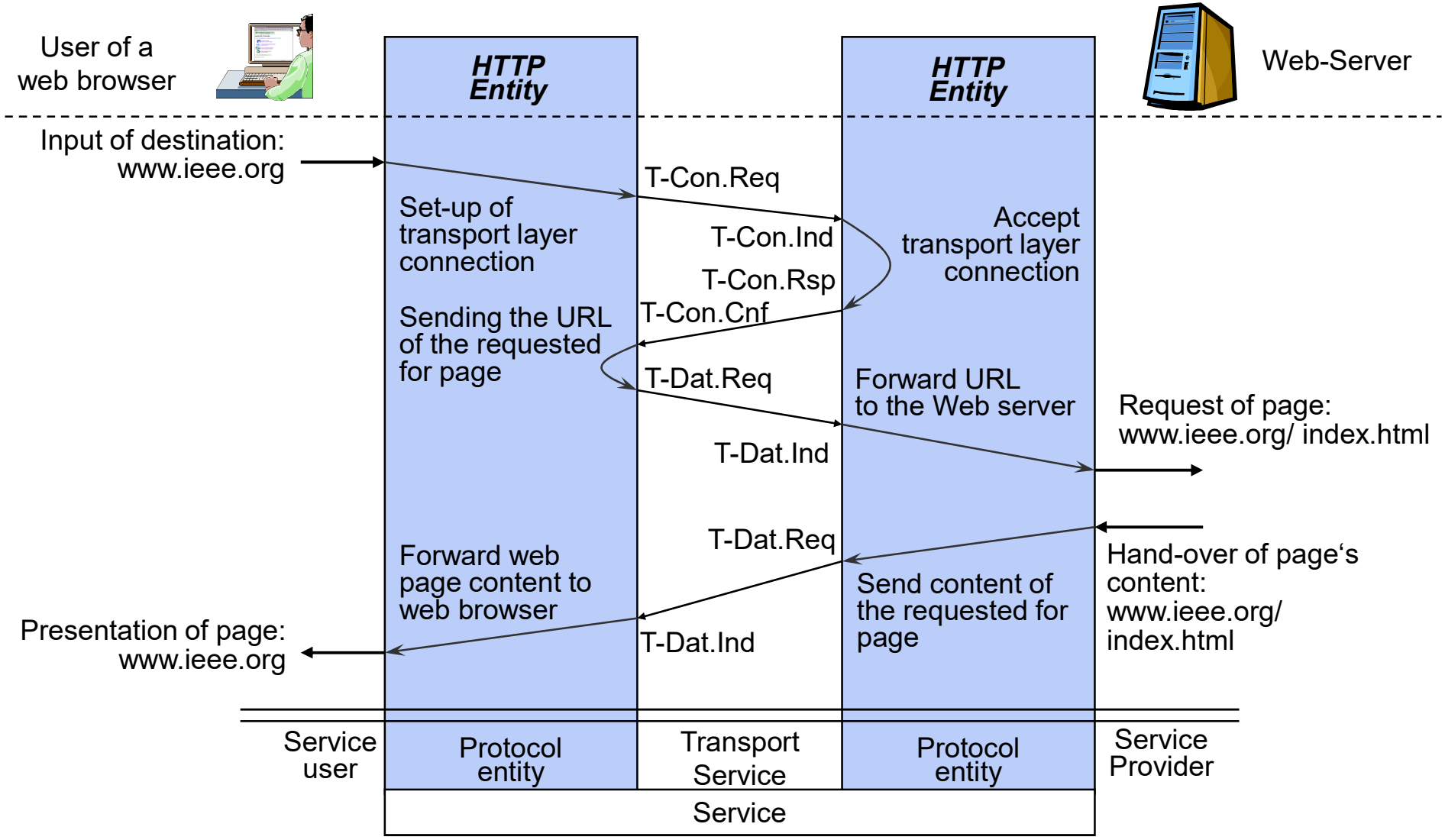
- Tags:
  - Start: <...>
  - End: </...>
- Structural information:
  - Paragraph: <p>
  - New line:
  - Header level *n*: <h*n*>

```
<html>
 <head>
 <title> Document Title </title>
 </head>
 <body>
 <p>This is a HTML document.</p>
 </body>
</html>
```

- Text options:
  - Italic <i>, bold <b>, emphasized , strong
  - Character set ISO 8859-2 8 bit, ASCII as a subset
- References to other documents:
  - <A HREF="iis.unibw-muenchen.de/index.html">Index</A>

© 2017 Burkhard Stiller

Surfin' … Surfin' … Surfin' the Net :-)



© 2017 Burkhard Stiller