Rechnernetze I

SoSe 2025

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 30. April 2025

Rechnernetze I

SoSe 2025

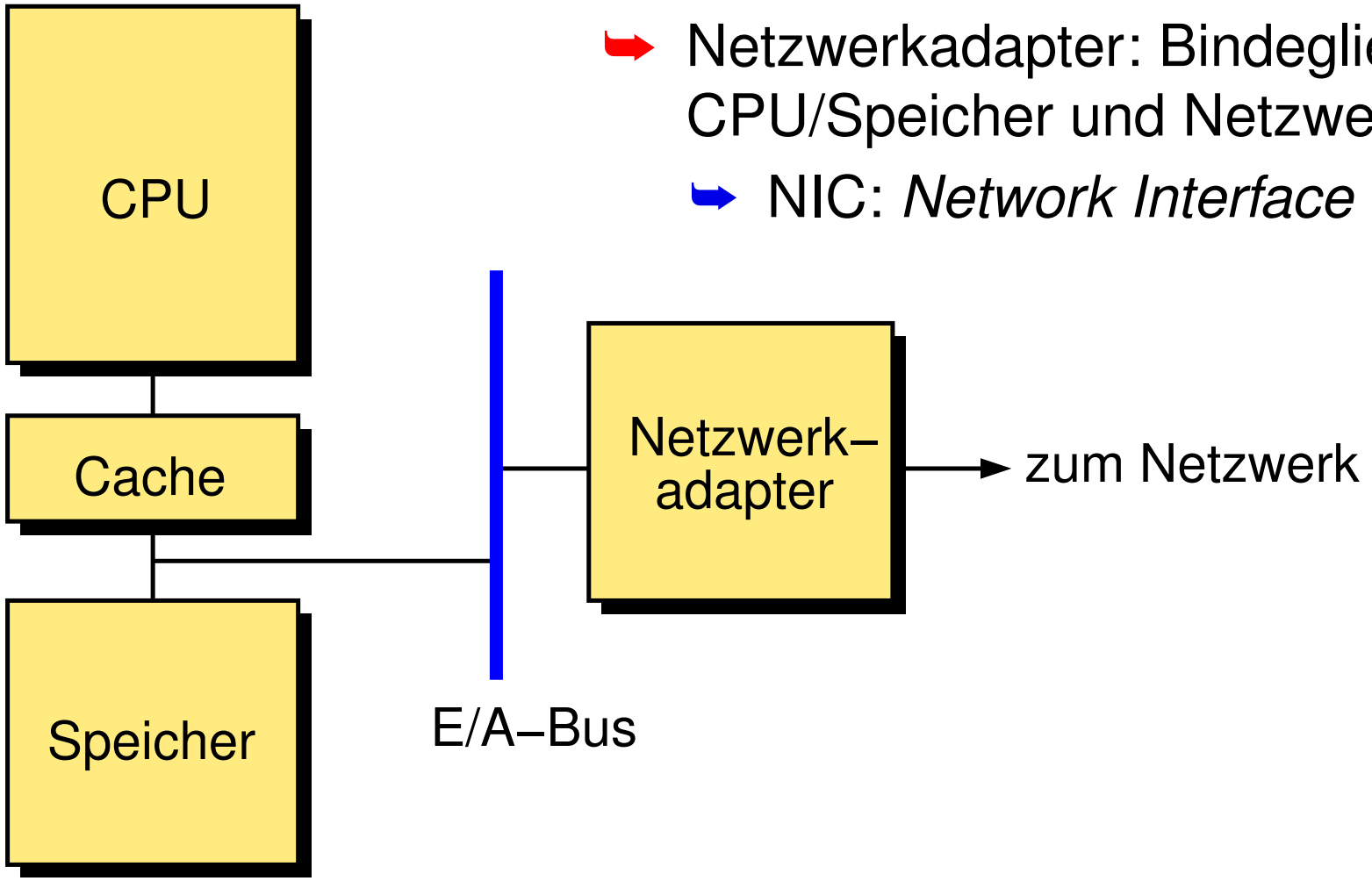
3 Direktverbindungsnetze

Inhalt

- Hardware-Bausteine: Knoten und Verbindungsleitungen
- Grundlagen zur Datenübertragung
- Modulation
- Codierung
- Framing
- Fehlererkennung und Fehlerkorrektur
- Medienzugriffssteuerung (MAC)
  - Allgemeines
  - Ethernet (CSMA-CD)

- Peterson, Kap. 2.1 – 2.6, 2.7.2
- CCNA, Kap. 4, 5.1

Aufbau eines Knotens



- Netzwerkadapter: Bindeglied zwichen CPU/Speicher und Netzwerk-Schnittstelle
  - NIC: *Network Interface Controller*

Rechnernetze I

SoSe 2025

24.04.2025

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

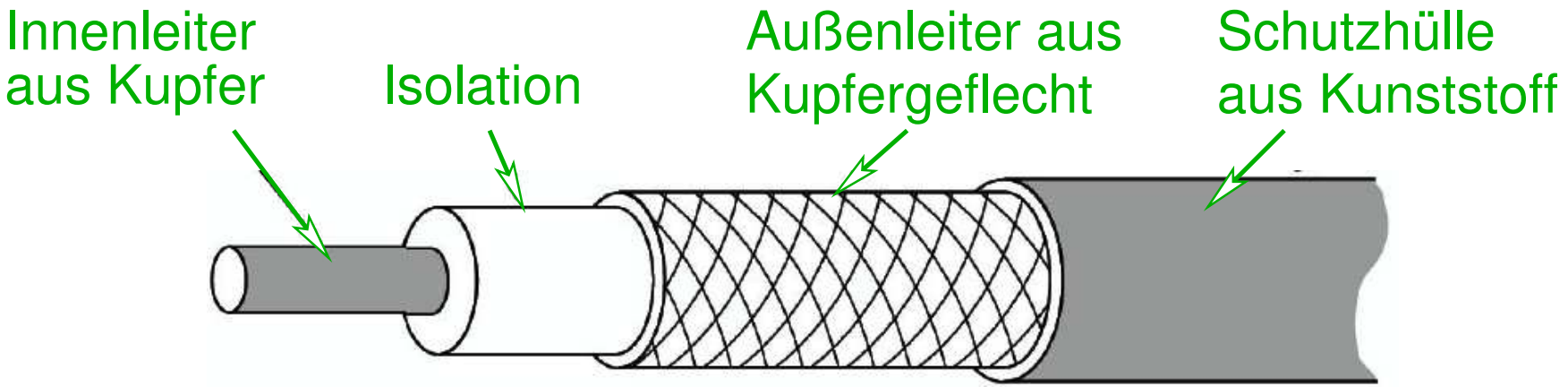
Stand: 30. April 2025

Verbindungs„leitungen“

- Übertragen Signale als elektromagnetische Wellen
- Typische Attribute:
  - Frequenz- bzw. Wellenlängenbereich (Bandbreite)
  - Dämpfung (max. Kabellänge)
  - Richtung des Datenflusses
    - **Simplex**: nur in eine Richtung
    - **Vollduplex**: in beide Richtungen, gleichzeitig
    - **Halbduplex**: in beide Richtungen, abwechselnd
- Grundlegende Arten:
  - Kupferkabel
  - Glasfaserkabel (Lichtwellenleiter)
  - Drahtlose Verbindung (Funk, IR) (☞ **RN_II**)

Kupferkabel: Koaxialkabel

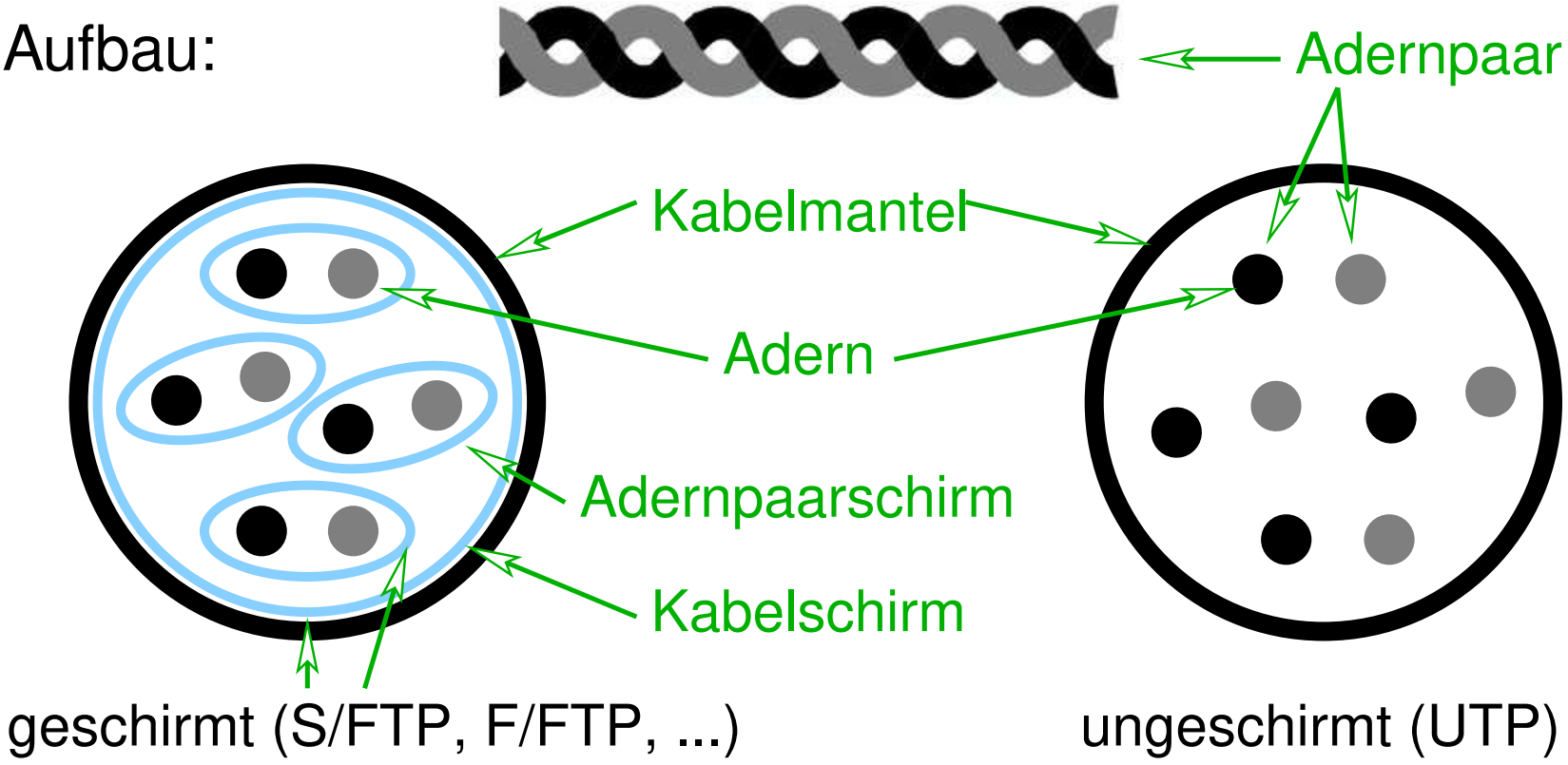
- Aufbau:



- Hohe Bandbreite, geringe Dämpfung, teuer
- Basisband-Kabel (direkte Übertragung, 1 Kanal, <500m)
  - Beispiele: Ethernet (10BASE-5, 10BASE-2)
- Breitband-Kabel (Modulation auf Träger, mehrere Kanäle, mehrere km)
  - Beispiel: Fernsehkabel

Kupferkabel: Twisted-Pair (verdrilltes) Kabel

- Aufbau:

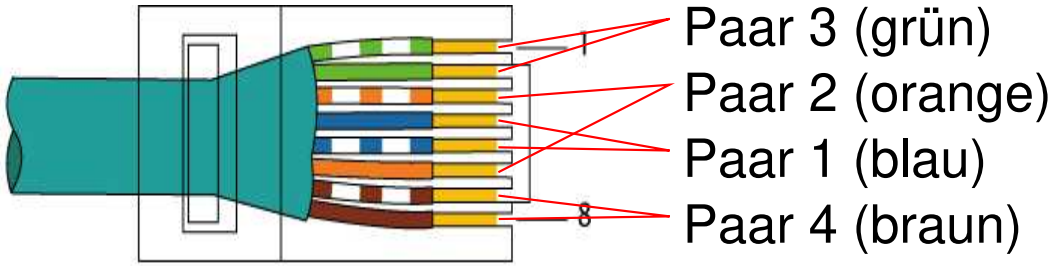


- Geringe Kosten, relativ gute Bandbreite
- Beispiel: Fast Ethernet (100BASE-TX)

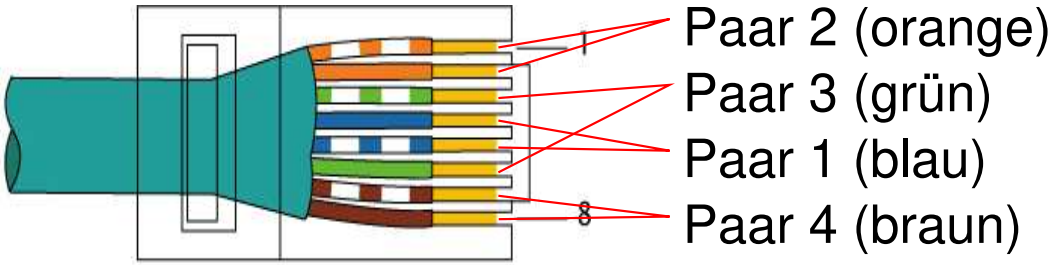
Beispiel: Fast Ethernet (100 Mb/s)

- Twisted-Pair Kabel (mind. Cat 5, UTP) mit 4 Adernpaaren
  - Paar 2: Signal vom Switch zum NIC
  - Paar 3: Signal vom NIC zum Switch

  } d.h. 2 Simplex-Leitungen
  - Paar 1 früher für analoges Telefon genutzt
- Stecker: RJ-45, zwei verschiedene Belegungen:
  - TIA-568A:

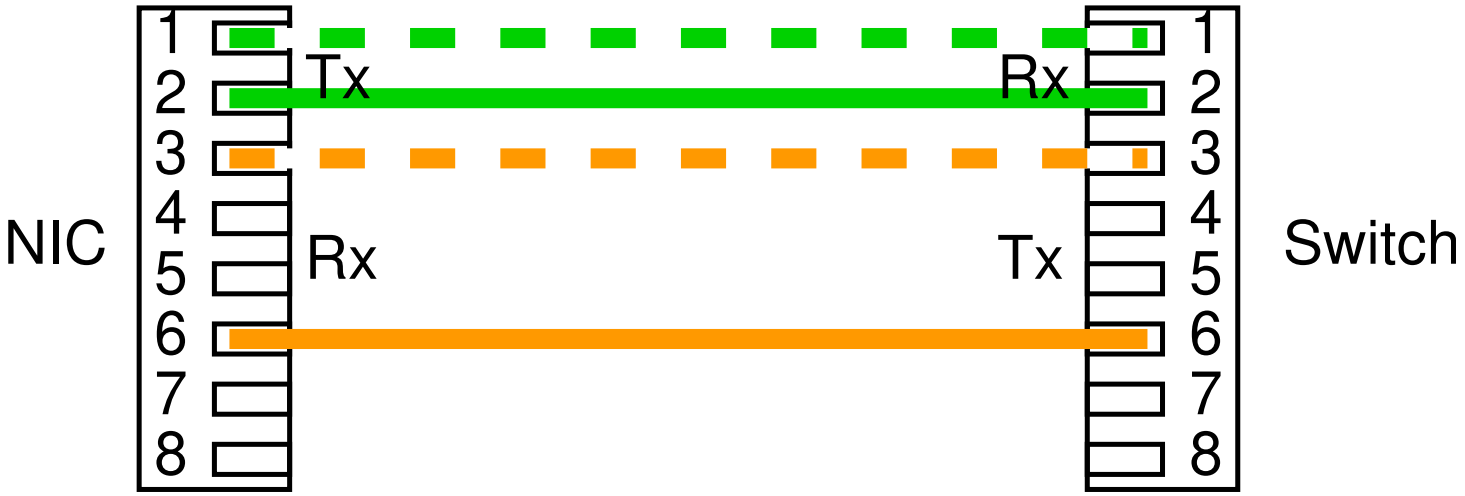


  - TIA-568B:



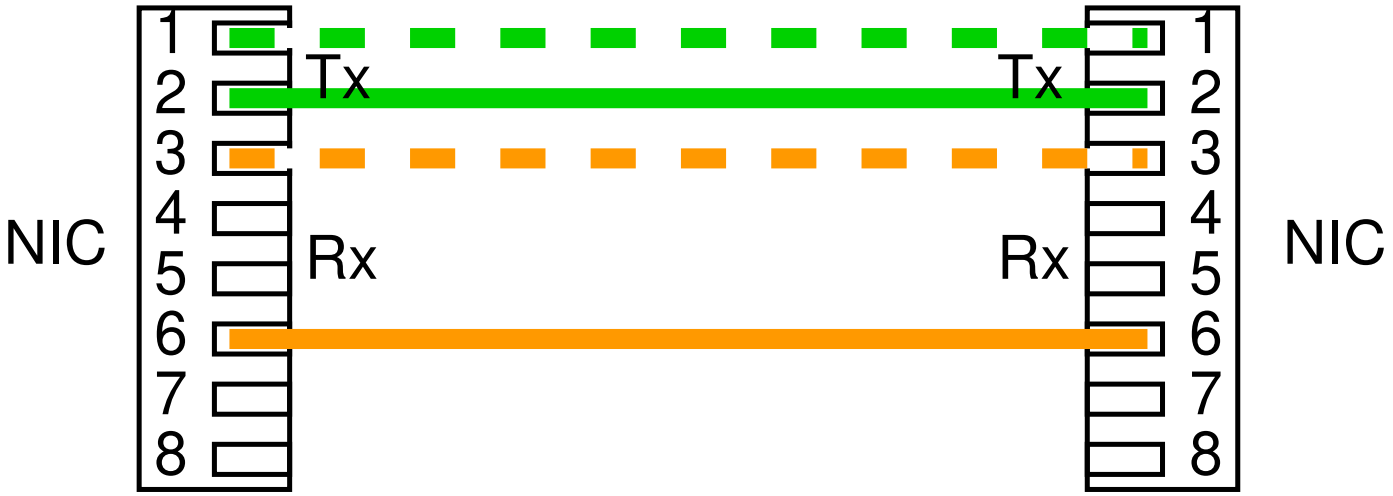
Straight-Through und *Cross-Over* Kabel

- Belegung der RJ-45 Buchsen:
  - NIC (PC, Router): Pin 1,2 = Transmit, Pin 3,6 = Receive
  - Switch/Hub: Pin 1,2 = Receive, Pin 3,6 = Transmit
- Straight-Through Kabel (beide Stecker nach 568A bzw. 568B)



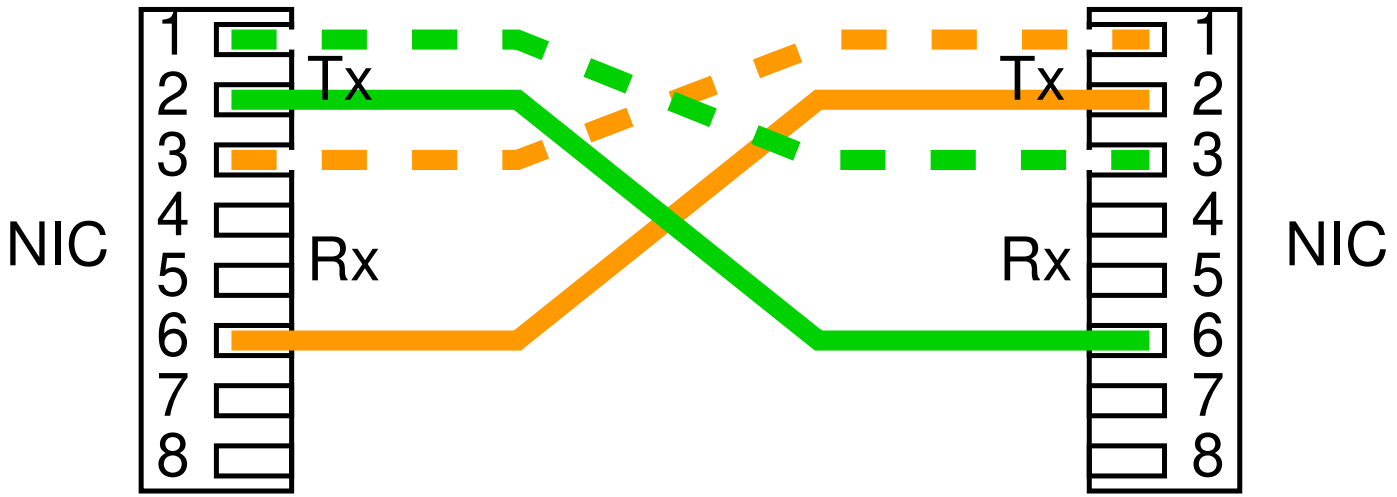
Straight-Through und *Cross-Over* Kabel

- Belegung der RJ-45 Buchsen:
  - NIC (PC, Router): Pin 1,2 = Transmit, Pin 3,6 = Receive
  - Switch/Hub: Pin 1,2 = Receive, Pin 3,6 = Transmit
- Straight-Through Kabel (beide Stecker nach 568A bzw. 568B)



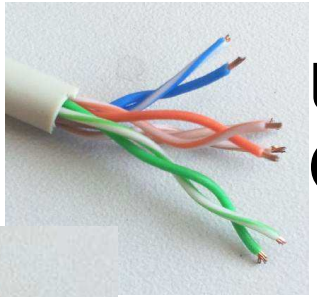
Straight-Through und *Cross-Over* Kabel

- Belegung der RJ-45 Buchsen:
  - NIC (PC, Router): Pin 1,2 = Transmit, Pin 3,6 = Receive
  - Switch/Hub: Pin 1,2 = Receive, Pin 3,6 = Transmit
- Straight-Through Kabel (beide Stecker nach 568A bzw. 568B)
- Cross-Over Kabel (ein Stecker 568A, ein Stecker 568B)

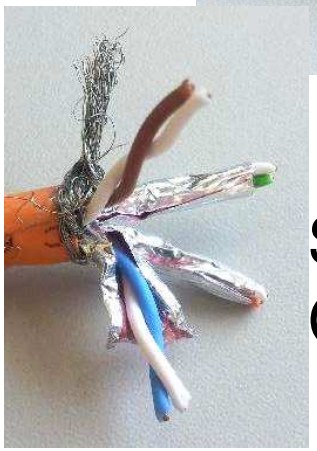


Twisted-Pair Kabelkategorien

- Cat. 3: bis 16 MHz, Telefon + 10 Mb/s Ethernet
- Cat. 5: 100 MHz, 100 Mb/s Ethernet
- Cat. 6: 250 MHz, 1 Gb/s Ethernet
- Cat. 6a: 500 MHz, 1 Gb/s Ethernet
- Cat. 7: 600 MHz, 10 Gb/s Ethernet
- Cat. 7a: 1 GHz, 10 Gb/s Ethernet
- Cat. 8: 2 GHz, 40 Gb/s Ethernet



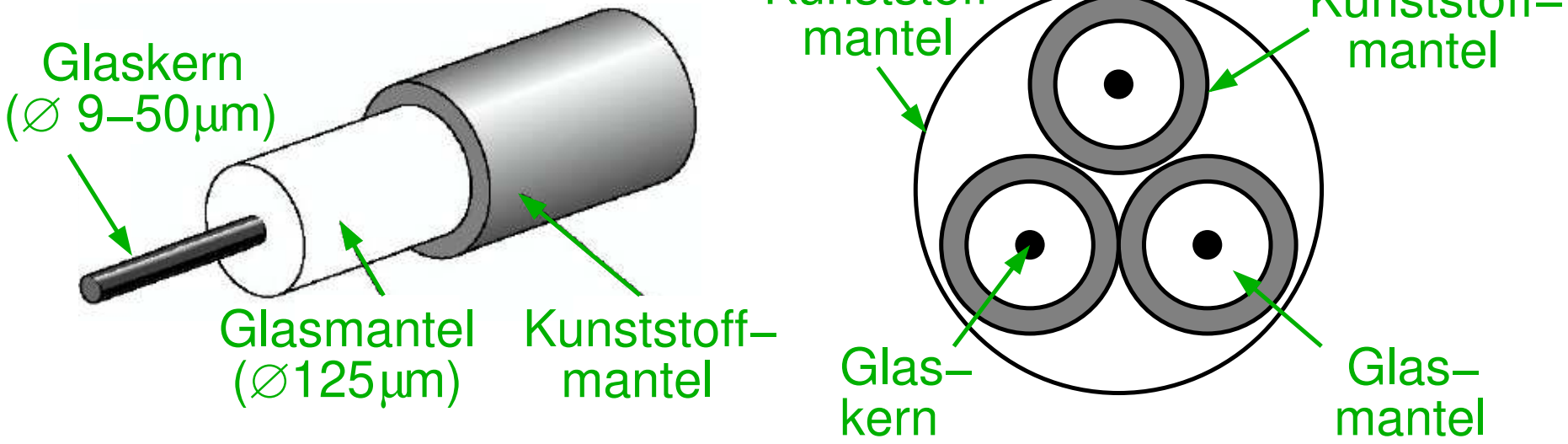
UTP
Cat. 5



S/FTP
Cat. 7

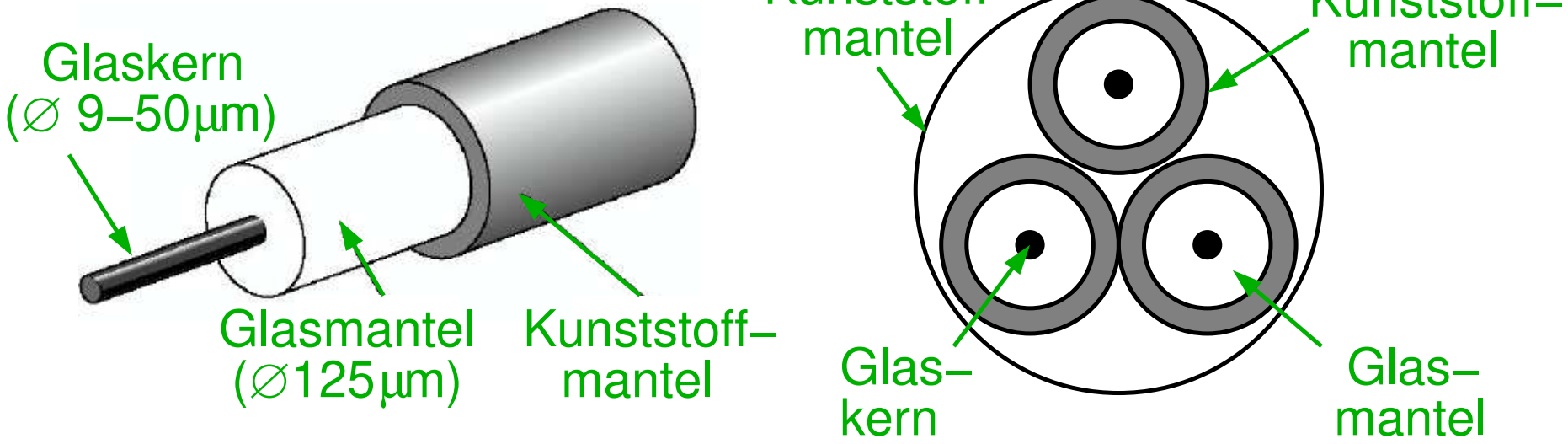
- Unterschiede: Material, Verdrillung, Adernschirmung
- Maximale Länge bei Ethernet: 100 m

Glasfaserkabel

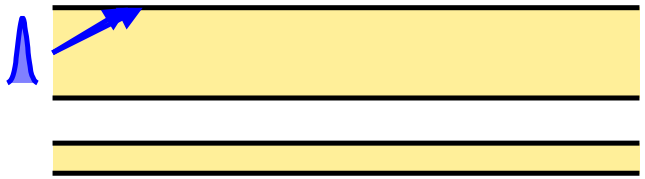


- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)

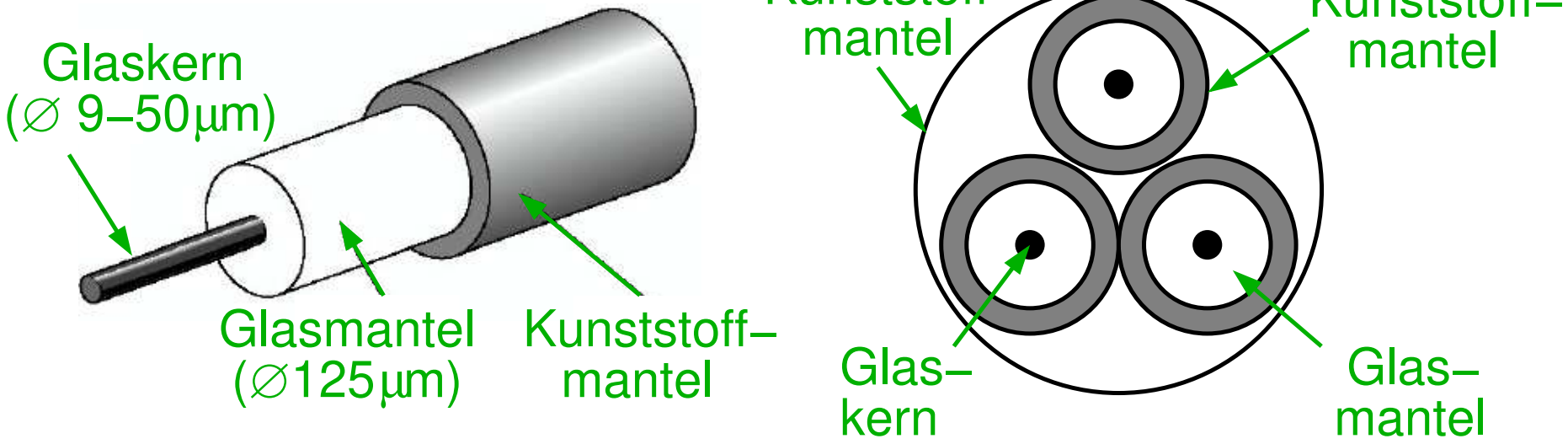
Glasfaserkabel



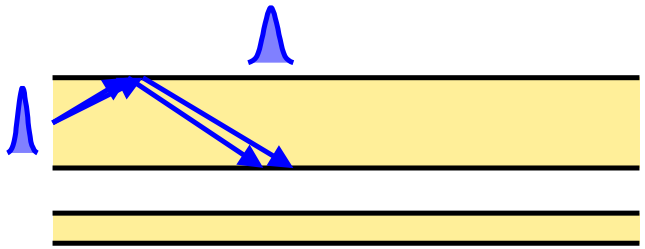
- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



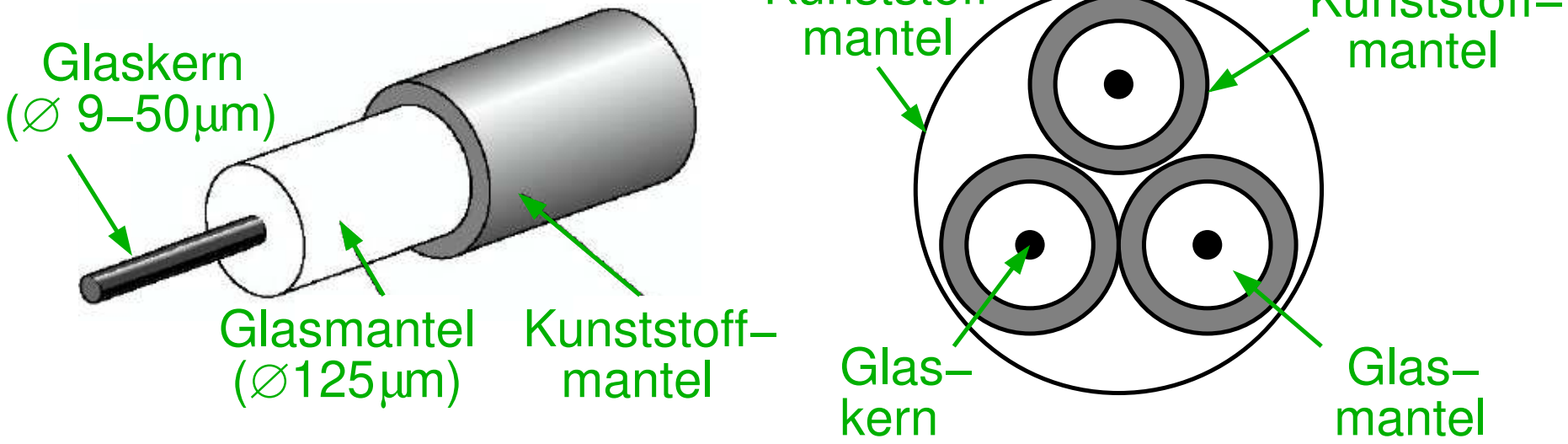
Glasfaserkabel



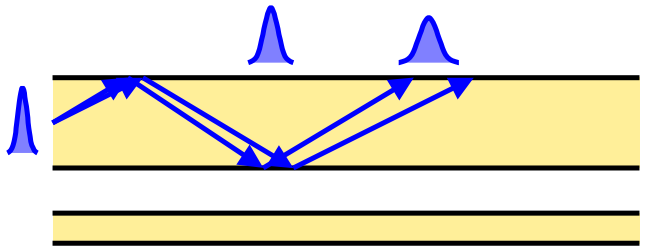
- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



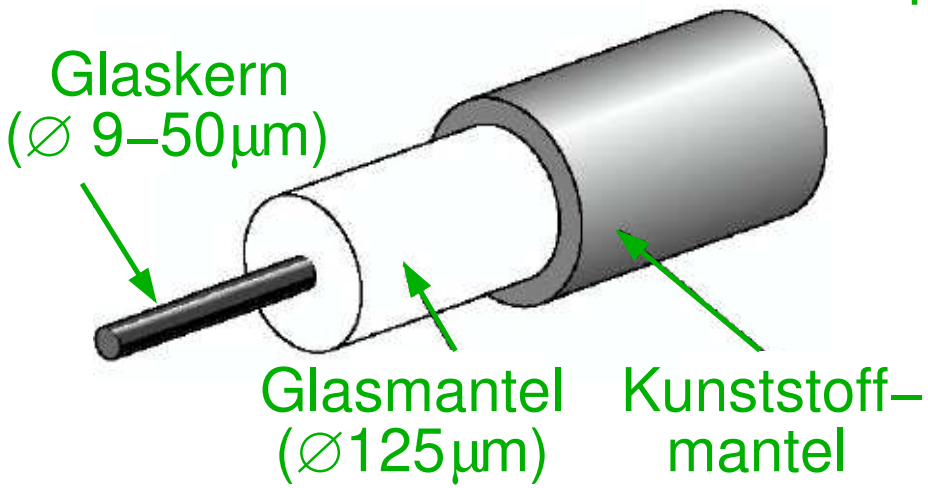
Glasfaserkabel

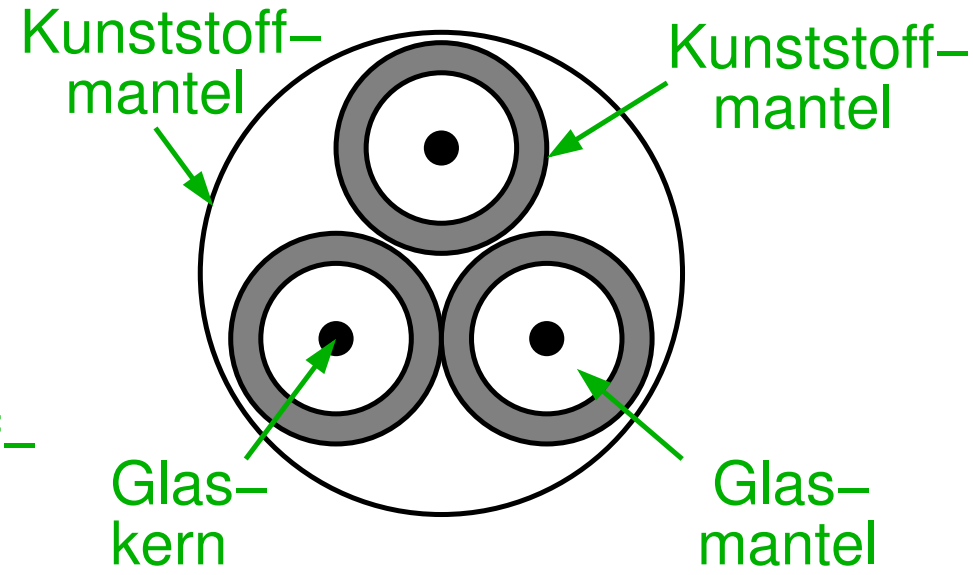


- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)

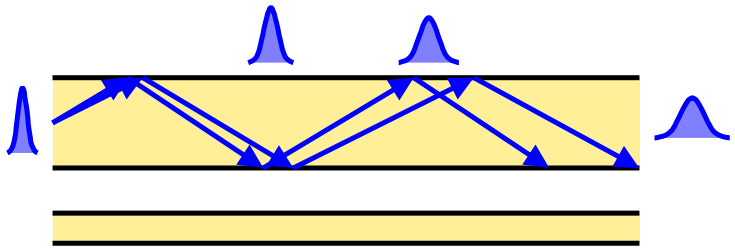


Glasfaserkabel

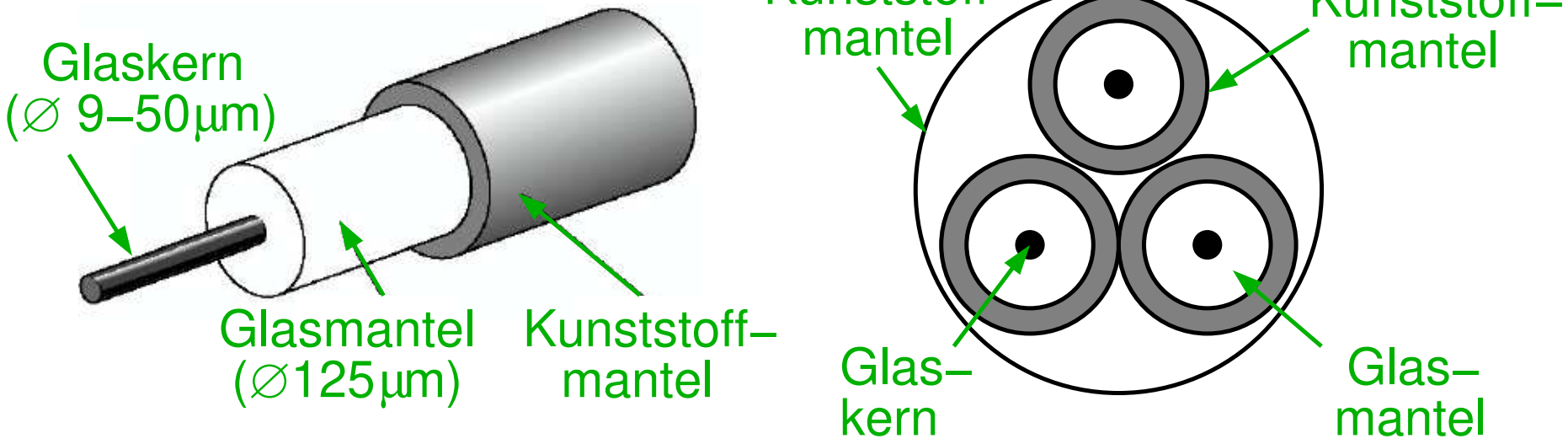




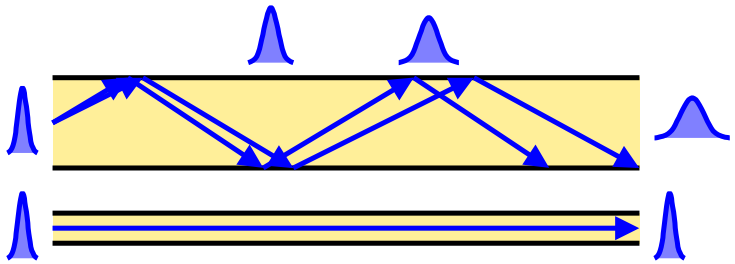
- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



Glasfaserkabel

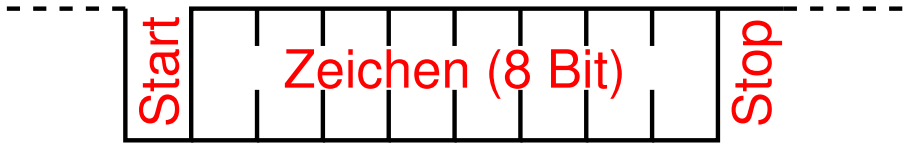


- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



Synchrone und asynchrone Übertragung

- Synchrone Übertragung:
  - serielle Übertragung eines **Bit**stroms
  - Bits müssen taktsynchron gesendet werden
  - Empfänger muss Sendetakt rekonstruieren können
  - z.B. Ethernet
- Asynchrone Übertragung:



  - serielle Übertragung eines Stroms von **Zeichen**
  - Zeichen kann zu beliebiger Zeit gesendet werden
  - Anfangsmarkierung durch Startbit
  - Empfänger muss Takt nur für ein einzelnes Zeichen halten
  - z.B. serielle Schnittstelle (V.24 / RS 232)

Qualitätsmerkmale von Leitungen

- Grenzfrequenz / Bandbreite (in Hz)
  - bei Basisbandübertragung (ohne Modulation): höchste Frequenz, die die Leitung noch übertragen kann
  - bei Modulation auf ein Trägersignal (Breitbandkabel, Funk): Breite des verfügbaren Frequenzkanals
- Dämpfung (in dB)
  - welcher Teil der eingespeisten Leistung kommt am anderen Ende an?
  - logarithmisches Maß: 10 dB = Faktor 10, 20 dB = Faktor 100 ...
- Übersprechen (in dB)
  - bei Kabeln mit mehreren Adernpaaren: Einstrahlung von Leistung aus anderen Adernpaaren

Maximal erreichbare Bitrate einer Leitung

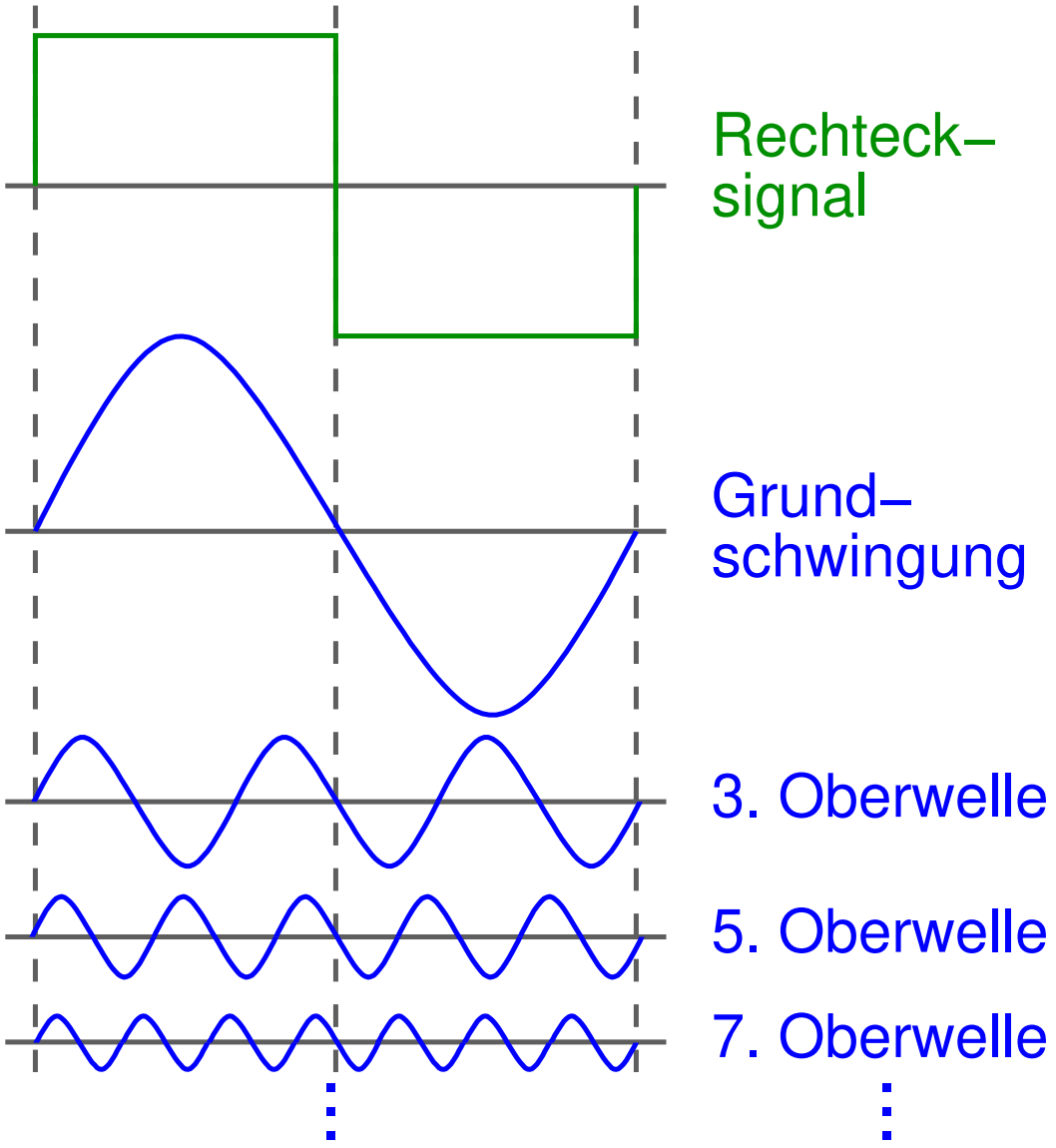
- Frage: Welche Übertragungsrate (bit/s) ist auf einer Leitung mit gegebener Grenzfrequenz (Bandbreite) möglich?
- Antworten liefern:
  - **Nyquist-Theorem**
    - maximal sinnvolle Abtastrate (bei exakten Abtastungen)
  - **Shannon'sches Theorem**
    - maximal erreichbare Bitrate

Hilfsmittel: Fourier-Analyse

- jedes (periodische) Signal ist als Summe von Sinus-schwingungen darstellbar
- Z.B. Rechtecksignal:

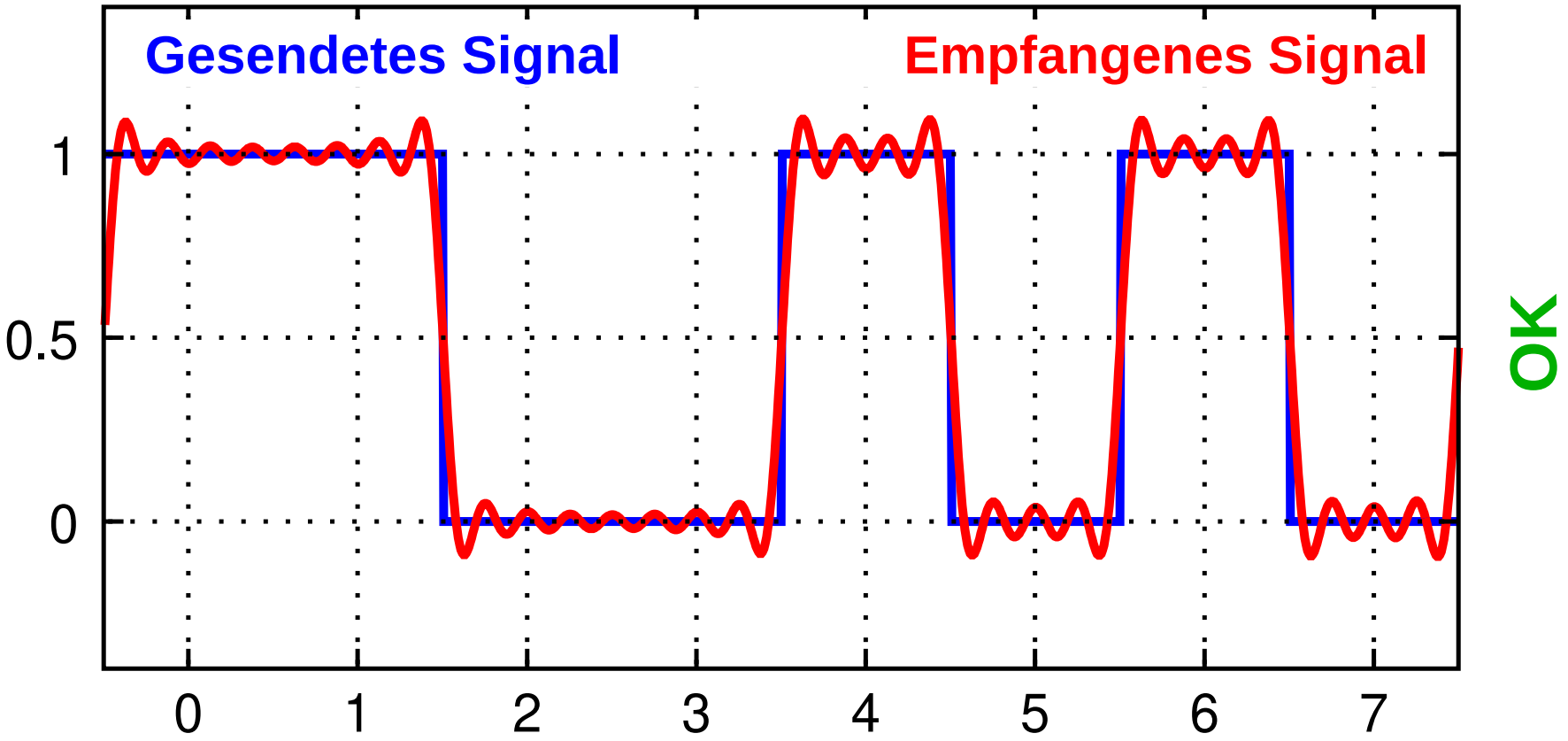
$$\sum_{k=1}^{\infty} \frac{4 \cdot \sin((2k-1)\omega t)}{(2k-1)\pi}$$

- Damit u.a. Auswirkungen der Grenzfrequenz einfach zu ermitteln
  - summiere nur die Ober-wellen mit kleinerer Frequenz



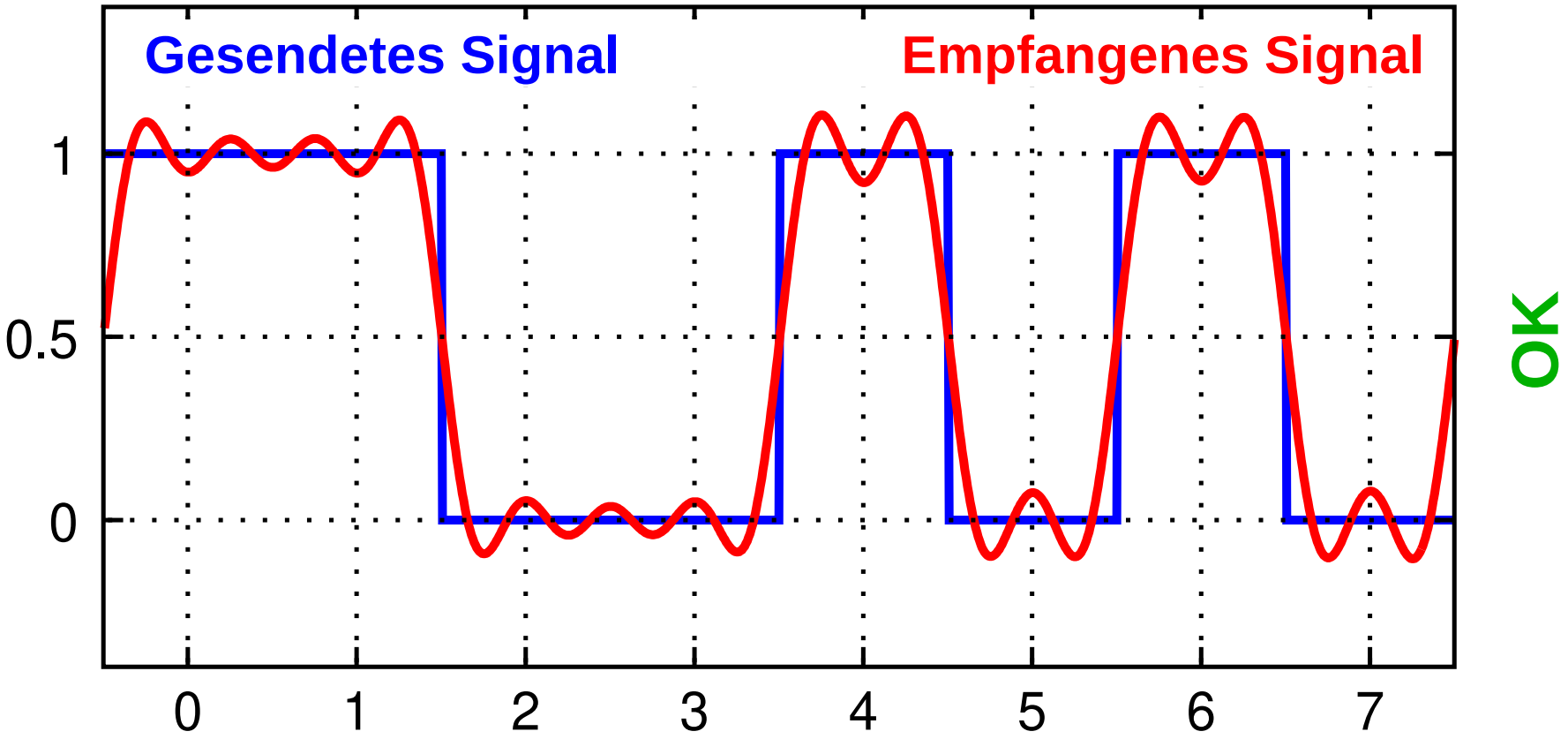
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes mit 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 4 MHz



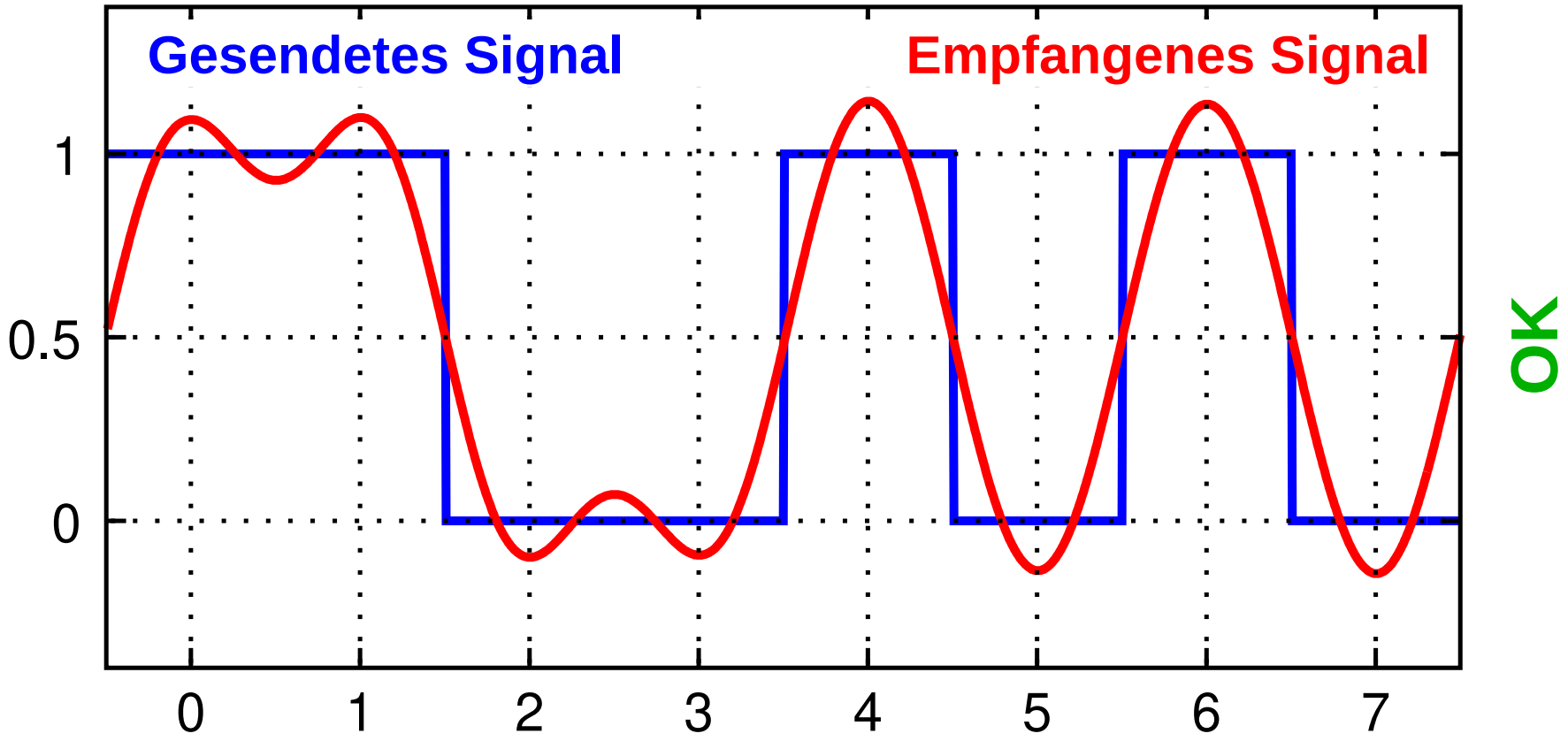
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes mit 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 2 MHz



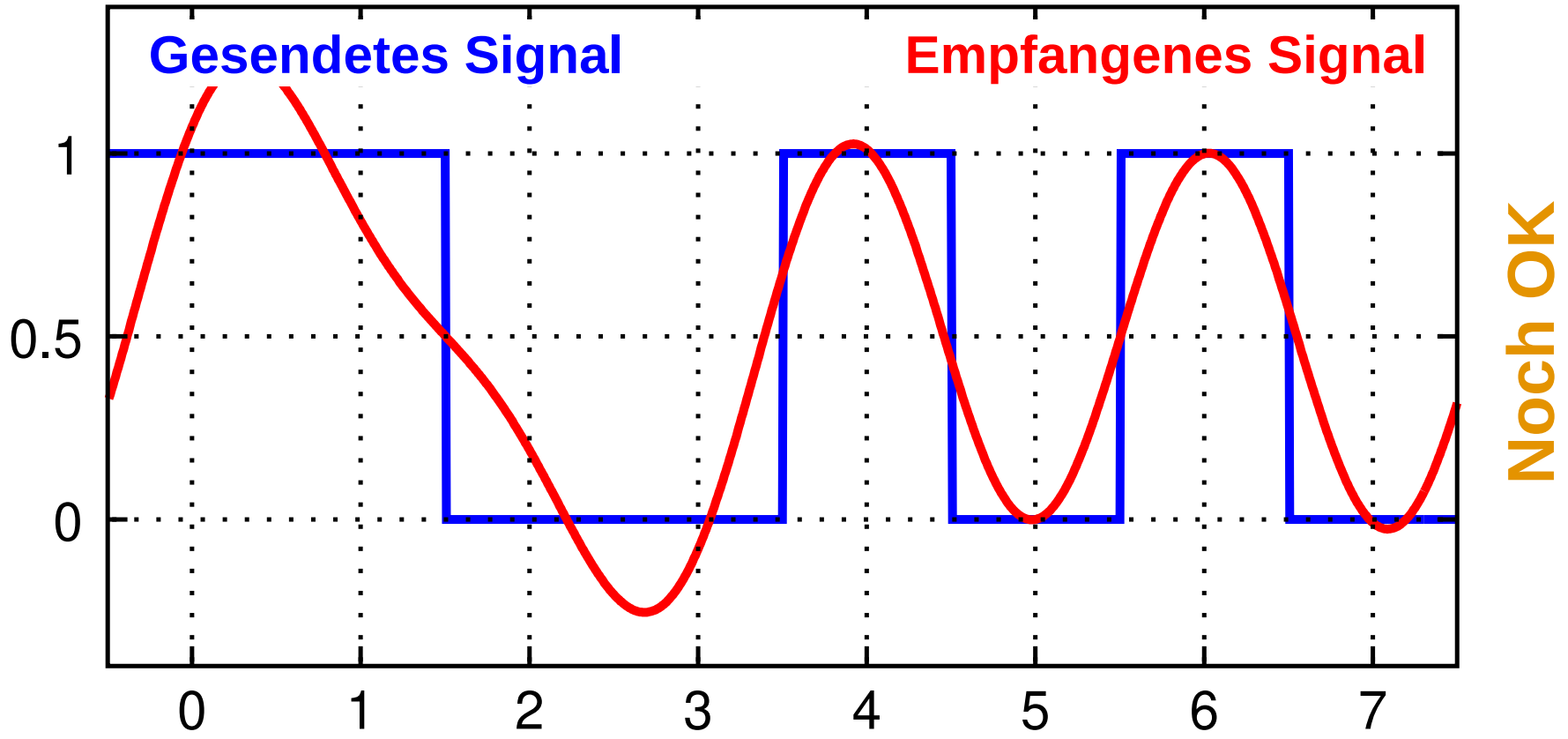
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes mit 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 1 MHz



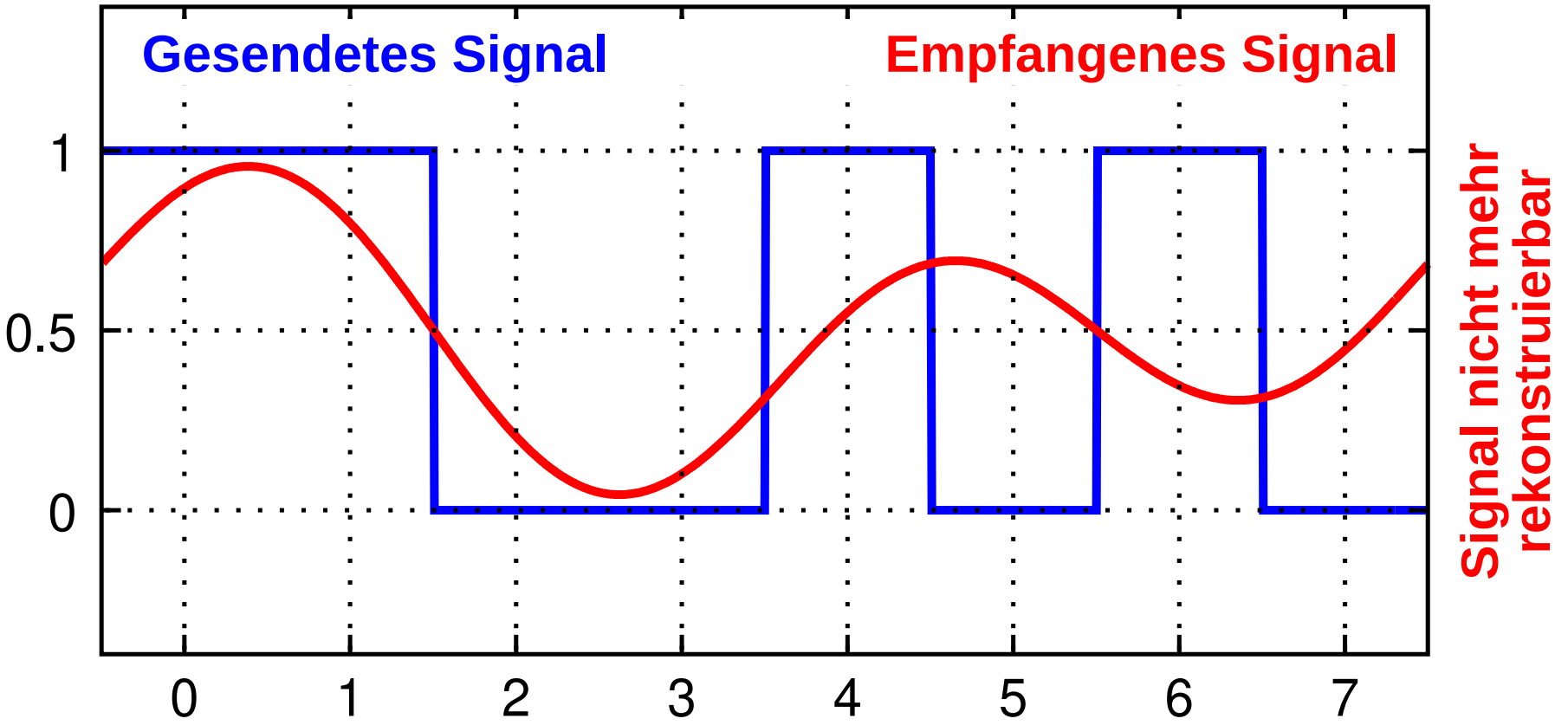
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes mit 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 500 kHz



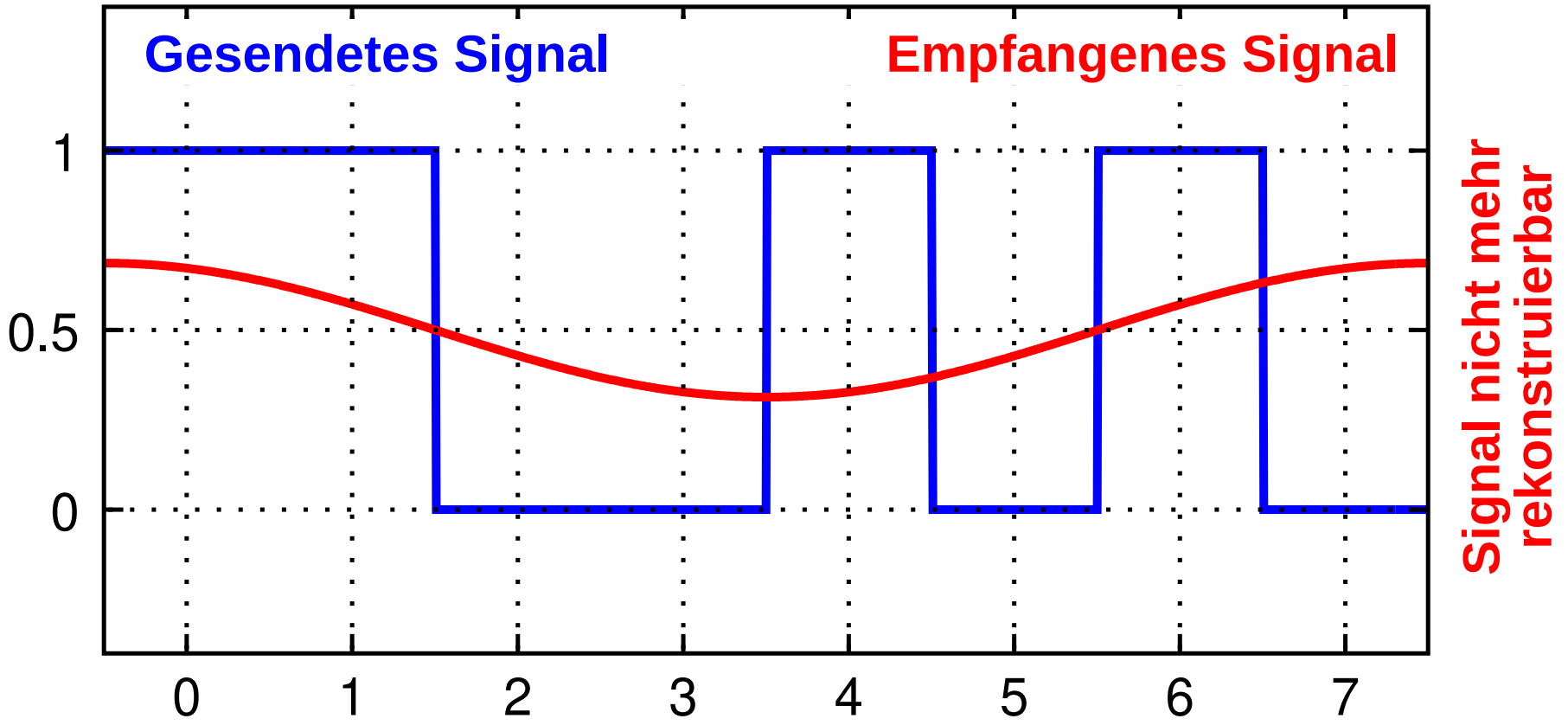
Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes mit 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 250 kHz



Zur Auswirkung der Grenzfrequenz

- Übertragung eines 8-Bit Wortes mit 1 Mb/s
- Bandbreite der Leitung (Grenzfrequenz): 125 kHz



Nyquist-Theorem (Abtasttheorem)

- Ein Signal mit Grenzfrequenz H [Hz] kann mit $2 \cdot H$ (exakten) Abtastwerten pro Sekunde vollständig rekonstruiert werden
- Die maximal sinnvolle Abtastrate ist daher $2 \cdot H$ [1/s]
- Folgerung für Übertragung mit 1 Bit pro Abtastung:
  - maximale Datenübertragungsrate = $2 \cdot H$ [b/s]
  - siehe Beispiel: 1 Mb/s erfordert 500 kHz Grenzfrequenz
- Höhere Übertragungsraten sind möglich, wenn pro Abtastung mehr als 1 Bit gewonnen wird (genauere Abtastung)
  - für n Bit pro Abtastung: 2^n Zustände (z.B. Spannungspegel)
  - Übertragungsrate ist dann begrenzt durch das **Rauschen**
    - z.B. Störeinstrahlung, thermisches Rauschen

Shannon'sches Theorem

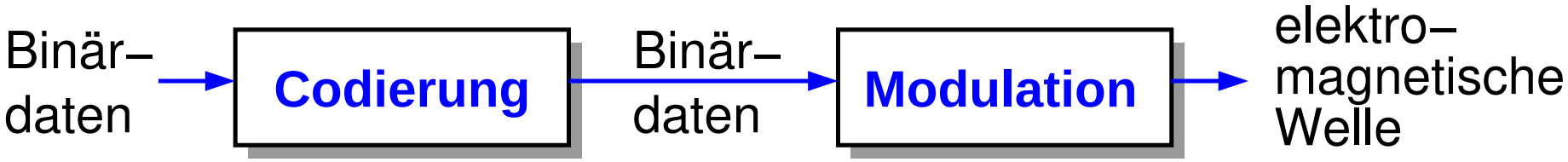
- Max. Datenübertragungsrate = $H \cdot \log_2(1 + S/N)$
- S/N = **Rauschabstand** (**Signal/Rauschverhältnis**)
  - (Leistungs-)Verhältnis von Signalstärke zu Rauschen
    - beeinflusst u.a. durch Dämpfung und Übersprechen der Leitung
  - definiert maximale Genauigkeit der Abtastung

Zur Unterscheidung von Übertragungs- und Abtastrate

- Einheit **b/s** für Übertragungsrate
- Einheit **Baud** (Zeichen/s) für Abtastrate

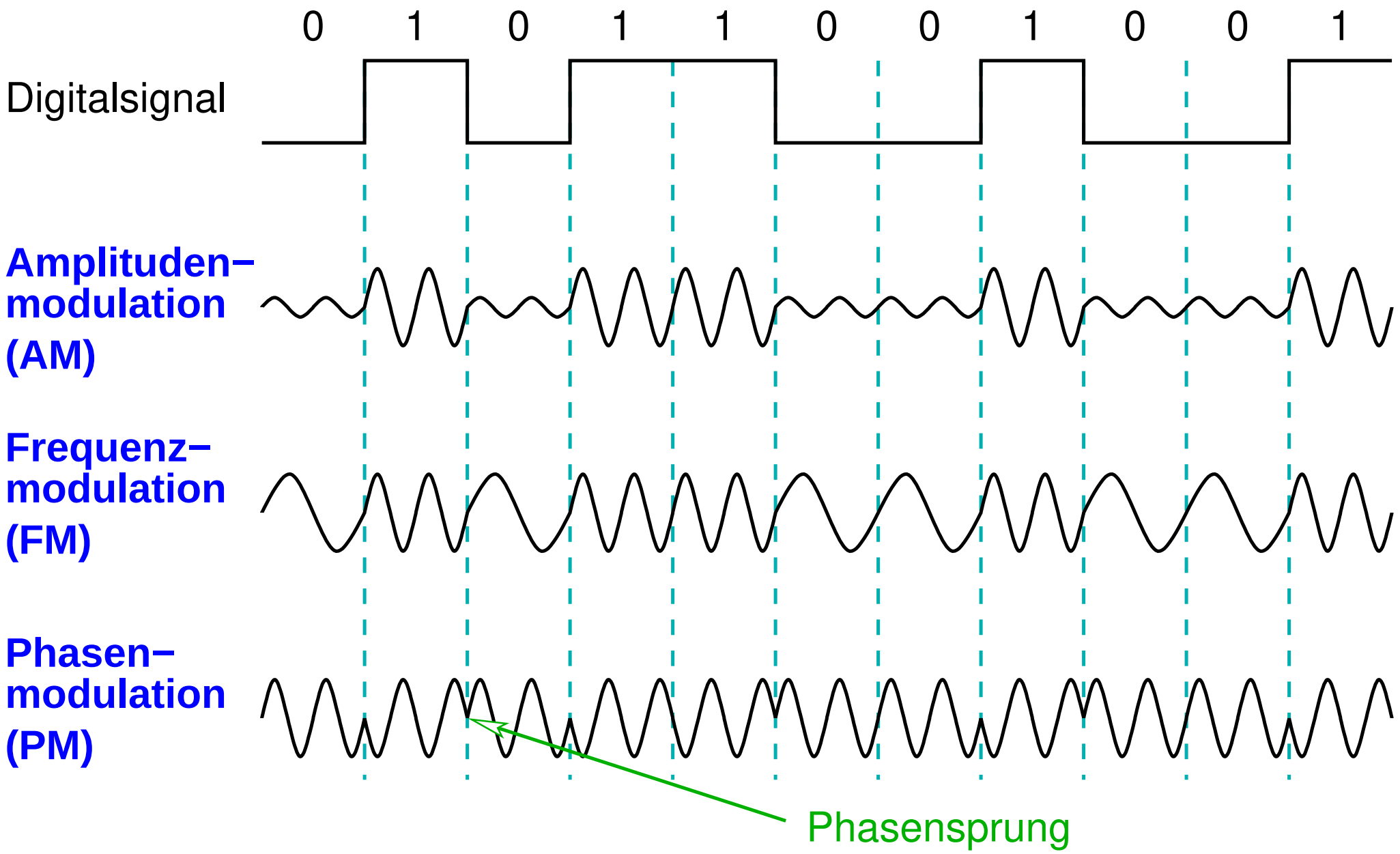
★★

- Zur Übertragung müssen Binärdaten (digitale Signale) in analoge elektrische Signale (elektromagnetische Wellen) umgesetzt werden
- Umsetzung in zwei Schritten:



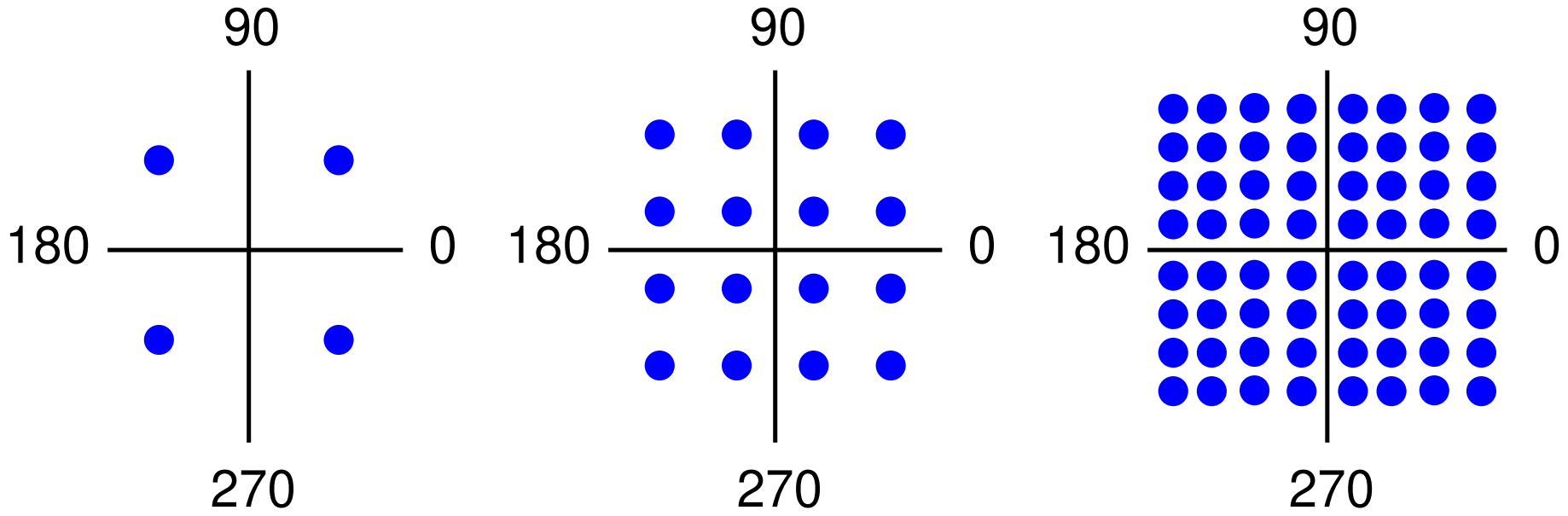
- **Modulation:**
  - Variation von Frequenz, Amplitude und/oder Phase einer Welle
  - zur Überlagerung der (Träger-)Welle mit dem Nutzsignal
    - z.B. bei Funk, Modem, Breitbandkabel, ...
  - (entfällt bei Basisband-Übertragung)

3.3 Modulation ...



Modulationsverfahren QPSK und QAM

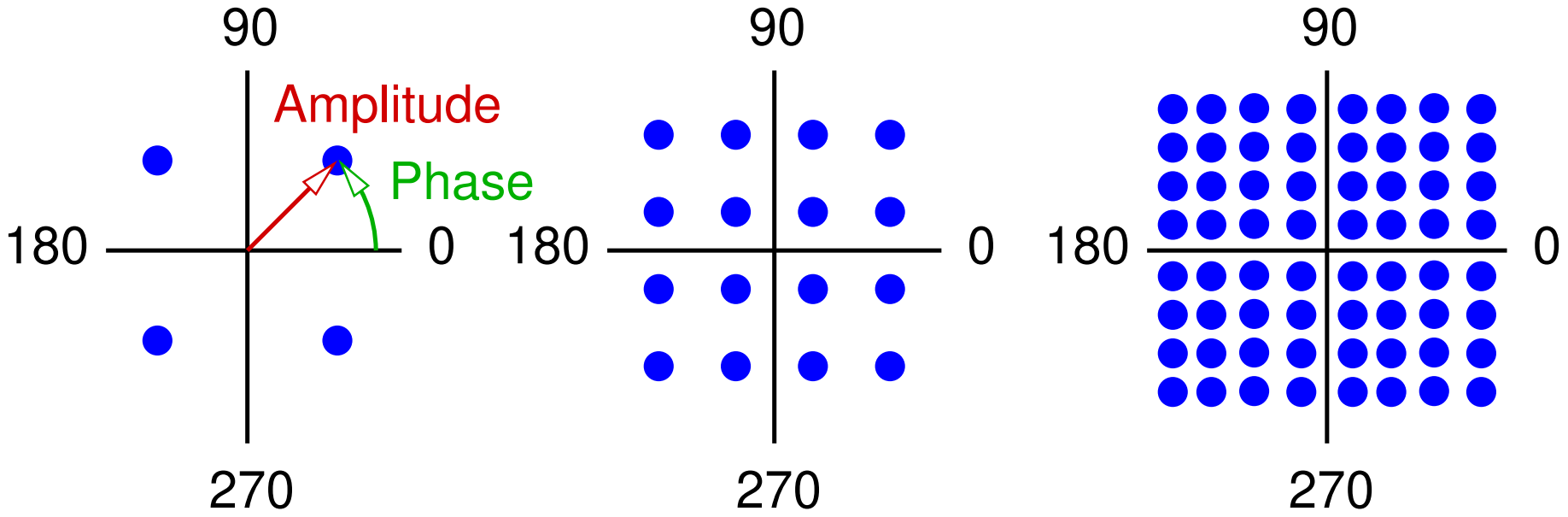
- Erleichtern die Gewinnung von mehreren Bits pro Abtastung
- Funktionsprinzip:
  - jeweils n Bits bestimmen **Amplitude** und **Phase** des Signals
- Beispiele:



QPSK (2 Bit) **QAM−16** (4 Bit) **QAM−64** (6 Bit)

Modulationsverfahren QPSK und QAM

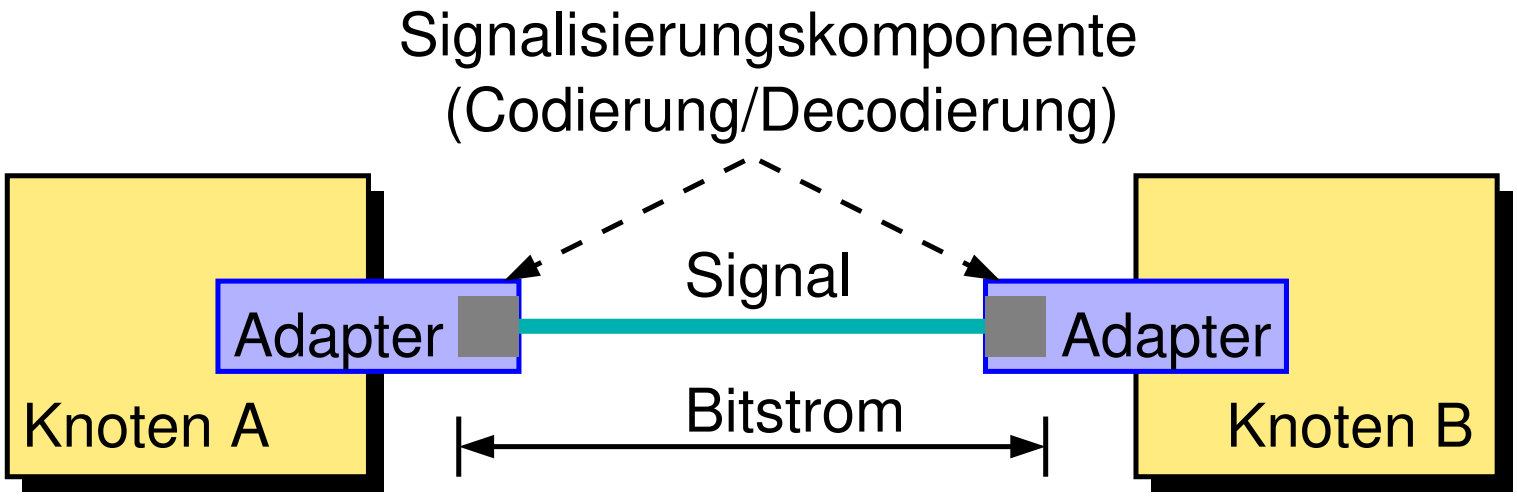
- Erleichtern die Gewinnung von mehreren Bits pro Abtastung
- Funktionsprinzip:
  - jeweils n Bits bestimmen **Amplitude** und **Phase** des Signals
- Beispiele:



QPSK (2 Bit) **QAM−16** (4 Bit) **QAM−64** (6 Bit)

★★★

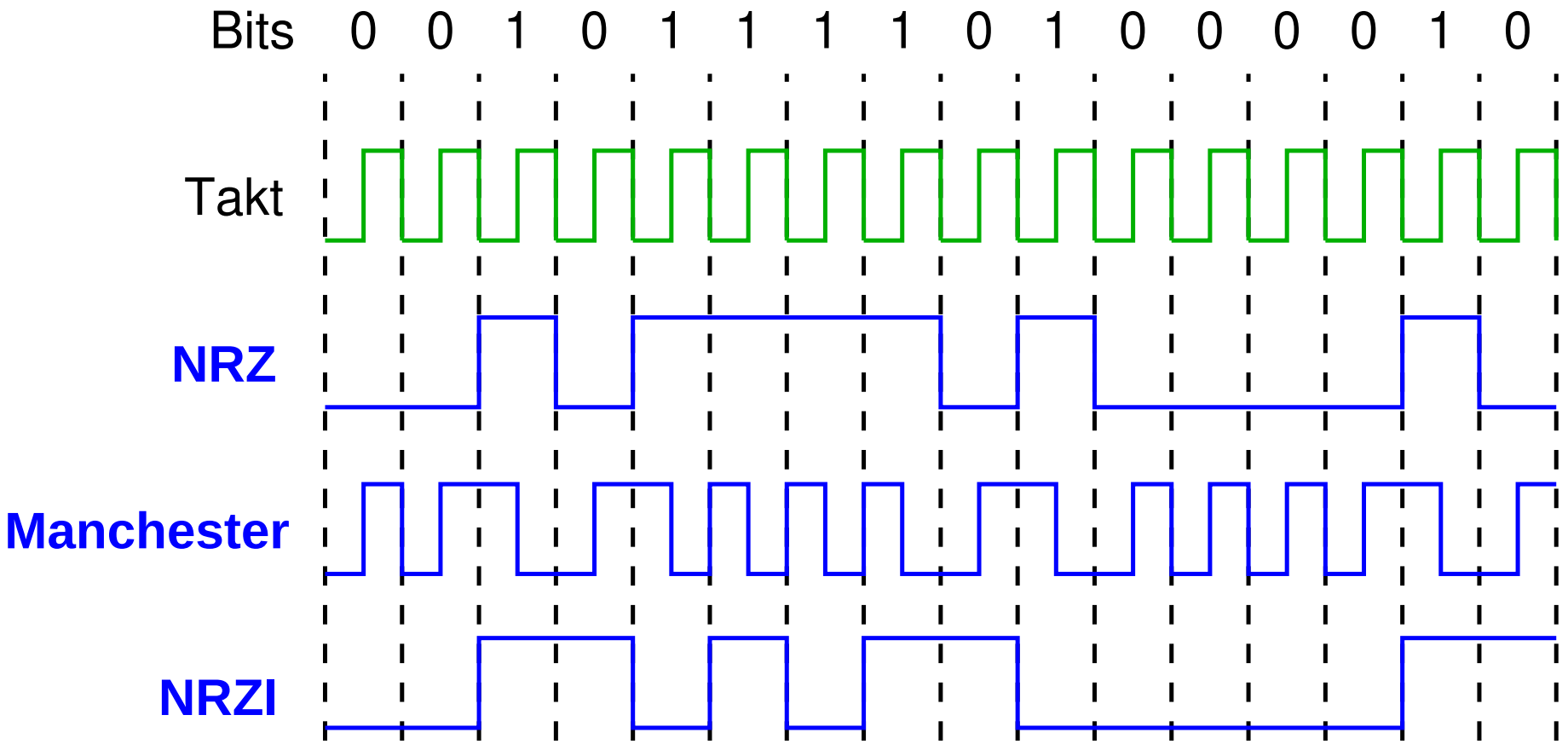
- Übertragung eines Bitstroms zwischen zwei Knoten:



- Einfachste Codierung:
  - ***Non-Return to Zero* (NRZ)**: **1** ≙ *high*, **0** ≙ *low*
- Probleme:
  - Festlegung der Spannungspegel für *high* und *low*
  - **Taktwiederherstellung** (**Synchronisation**)
    - wo ist die „Grenze“ zwischen zwei Bits?

3.4 Codierung ...

- Abhilfe: Codierungen mit Taktwiederherstellung



NRZI: *Non–Return to Zero Inverted*

★★★

Manchester-Codierung

- Bitstrom wird mit Taktsignal EXOR-verknüpft
- Anwendung z.B. bei 10 Mb/s Ethernet
- Problem:
  - **Baudrate** (Rate, mit der das Signal abgetastet werden muß) ist doppelt so hoch wie die Bitrate
  - verschwendet Bandbreite

NRZI

- Signal wird bei jedem **1**-Bit invertiert
- Problem: keine Taktwiederherstellung bei aufeinanderfolgenden Nullen möglich

★★★

4B/5B-Codierung

- 4 Datenbits werden auf 5-Bit Codeworte so abgebildet, daß nie mehr als 3 aufeinanderfolgende Nullen übertragen werden müssen
  - jedes der 5-Bit Codeworte hat
    - höchstens eine Null am Anfang
    - höchstens zwei Nullen am Ende
  - Übertragung der Codeworte z.B. mit NRZI
  - Overhead nur noch 25%

- Bei schnellen Netzen (z.B. Fast Ethernet, GBit-Ethernet) oder auch schnellen Modems werden noch effizientere Verfahren zur Taktrückgewinnung eingesetzt

Ziele der Codierung

- Taktrückgewinnung beim Empfänger
- Reduktion der benötigten elektrischen Bandbreite
  - Erhöhung der Bitrate (mehrere Bits pro Abtastung, z.B. 8B6T)
  - Reduktion der Stör-Ausstrahlung (z.B. MLT-3)
- Gleichspannungsfreiheit
  - Ziel: konstanter Mittelwert des Signals
  - erleichtert Übertragung und Erkennung von Low- und High-Pegel
  - bei 4B5B-Codierung nicht gegeben, daher z.B. 8B10B
- Fehlerkorrektur

Leitungs-codierung

Kanalcodierung

Rechnernetze I

SoSe 2025

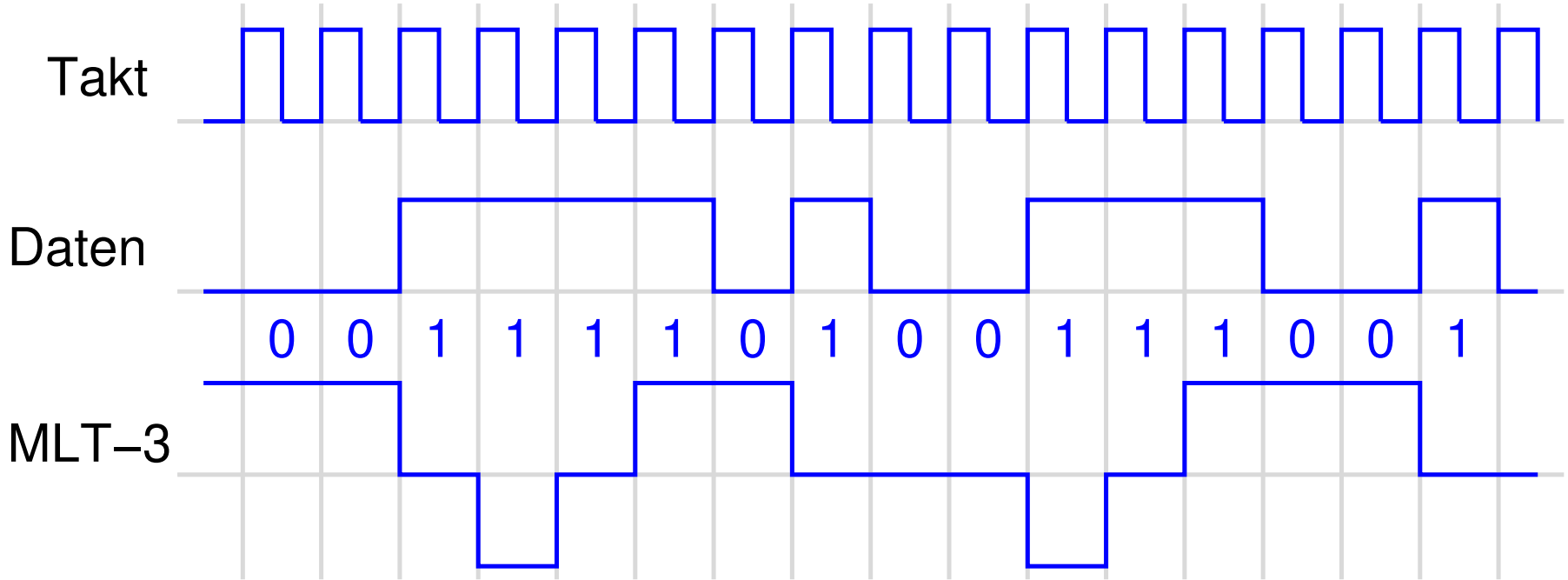
28.04.2025

Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 30. April 2025

Leitungscodierung beim Ethernet

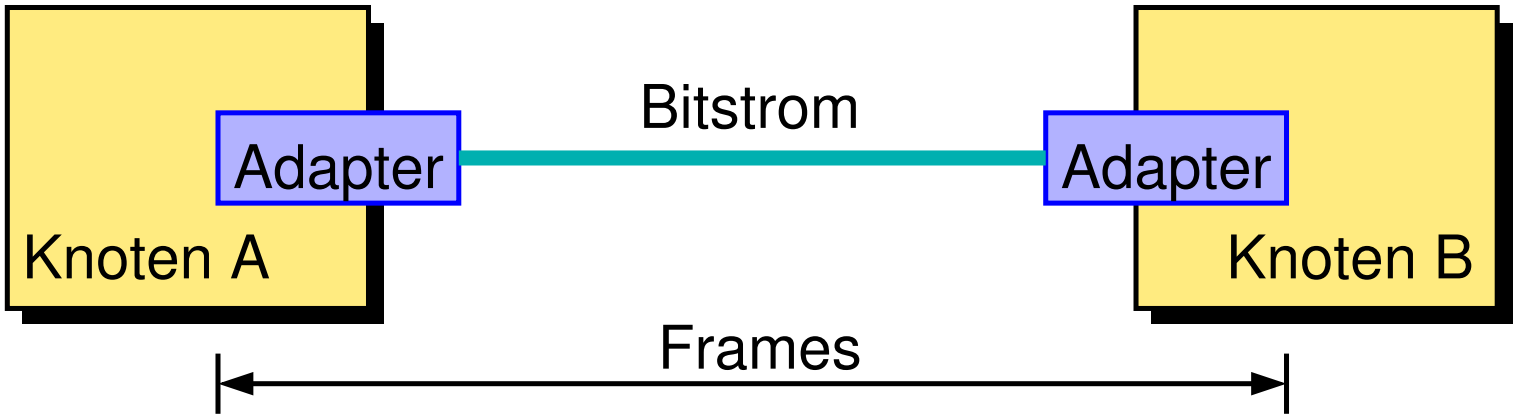
- Ursprünglich: Manchester-Codierung, 20 MBaud bei 10 Mb/s
- Fast Ethernet: 4B5B-Codierung, 125 MBaud bei 100 Mb/s
  - anschließend MLT-3 Codierung (3 Spannungspegel)



  - damit Grundfrequenz nur noch maximal 31,25 MHz (statt 62,5 MHz mit NRZI)

★★★

- Wir betrachten nun die Übertragung von Datenblöcken (**Frames**) zwischen Rechnern:

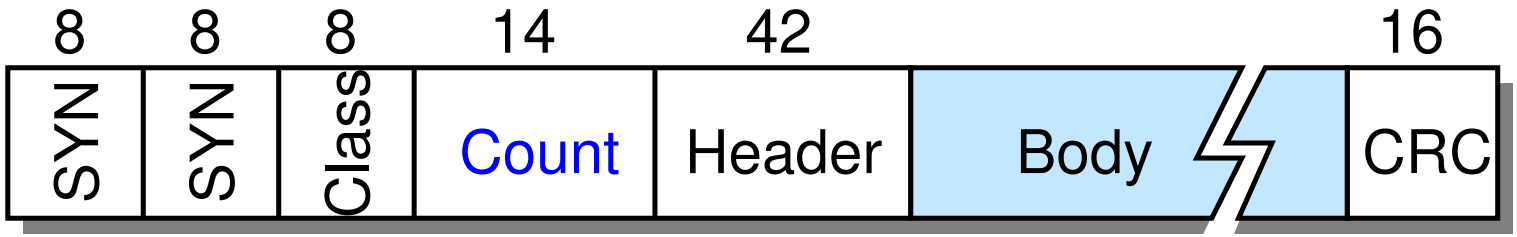


- Gründe für die Aufteilung von Daten in Frames:
  - einfaches Multiplexing verschiedener Kommunikationen
  - bei Fehler muss nur betroffener Frame neu übertragen werden
- Zentrale Aufgabe des Framings:
  - Erkennung, wo Frame im Bitstrom anfängt und wo er aufhört
  - dazu: Framegrenzen müssen im Bitstrom erkennbar sein

★★

Byte-Count Methode

- Frame-Header enthält Länge des Datenteils
- Beispiel: (Frame im DDCMP-Protokoll, DECNET)

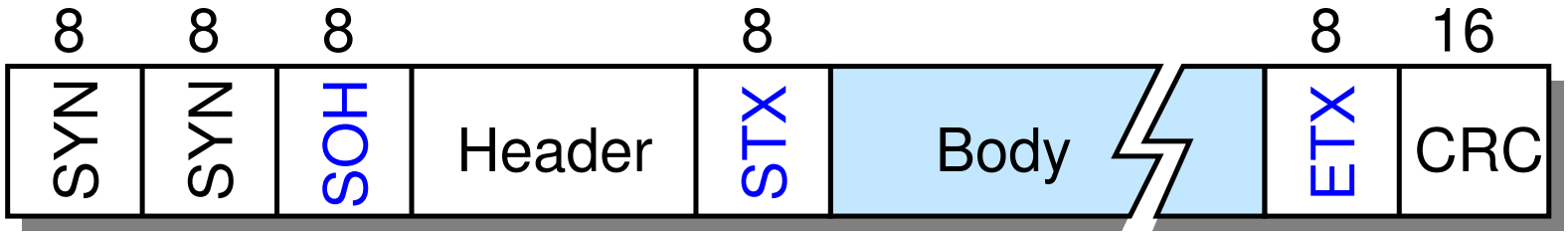


- Problem: was passiert, wenn die Länge fehlerhaft übertragen wird?
  - Frame-Ende wird nicht korrekt erkannt
  - SYN-Zeichen am Beginn jedes Frames, um (wahrscheinlichen!) Anfang des Folgeframes zu finden
- Verwendet u.a. beim ursprünglichen Ethernet

★★

Sentinel-Methode

- Frame-Ende wird durch spezielles Zeichen markiert
- Beispiel: (Frame im BISYNC-Protokoll, IBM)



- Problem: Das Endezeichen kann auch im Datenteil (Body) vorkommen
- Lösung: ***Byte-Stuffing***
  - ersetze ETX im Datenteil durch DLE ETX
  - ersetze DLE im Datenteil durch DLE DLE
  - verwendet u.a. bei PPP

★★

Sentinel-Methode ...

- Lösung: ***Bit-Stuffing***
  - Eindeutigkeit durch Einfügen von Bits in den Bitstrom erreicht
  - Beispiel: (HDLC-Protokoll)
    - Anfangs- und Endemarkierung ist 01111110_2
    - nach 5 aufeinanderfolgenden 1-Bits wird vom Sender ein 0-Bit in den Bitstrom eingeschoben
    - wenn Empfänger 5 aufeinanderfolgende 1-Bits gelesen hat:
      - nächstes Bit = 0: ignorieren, da eingeschoben
      - nächstes Bit = 1: sollte Endemarkierung sein (prüfe, ob die 0 folgt; falls nicht: Fehler)
- Lösung: Nutzung „ungültiger“ Codeworte
  - Anfang und Ende durch Codeworte markiert, die sonst nicht vorkommen (z.B. bei 4B/5B-Codierung)

Framing beim Ethernet

- Ein Ethernet-Frame enthält i.d.R. keine Längenangabe (☞ **3.1.6**)
- Ebenso wird kein Bit-/Bytestuffing verwendet
- Erkennung des Frame-Endes erfolgt auf OSI-Schicht 1!
  - 10 Mb/s Ethernet: Ausbleiben des Signalwechsels (Manchester-Codierung!)
  - Fast Ethernet: Ende durch Bitfolge 01101 00111 gekennzeichnet (ungültige 4B/5B Codeworte)

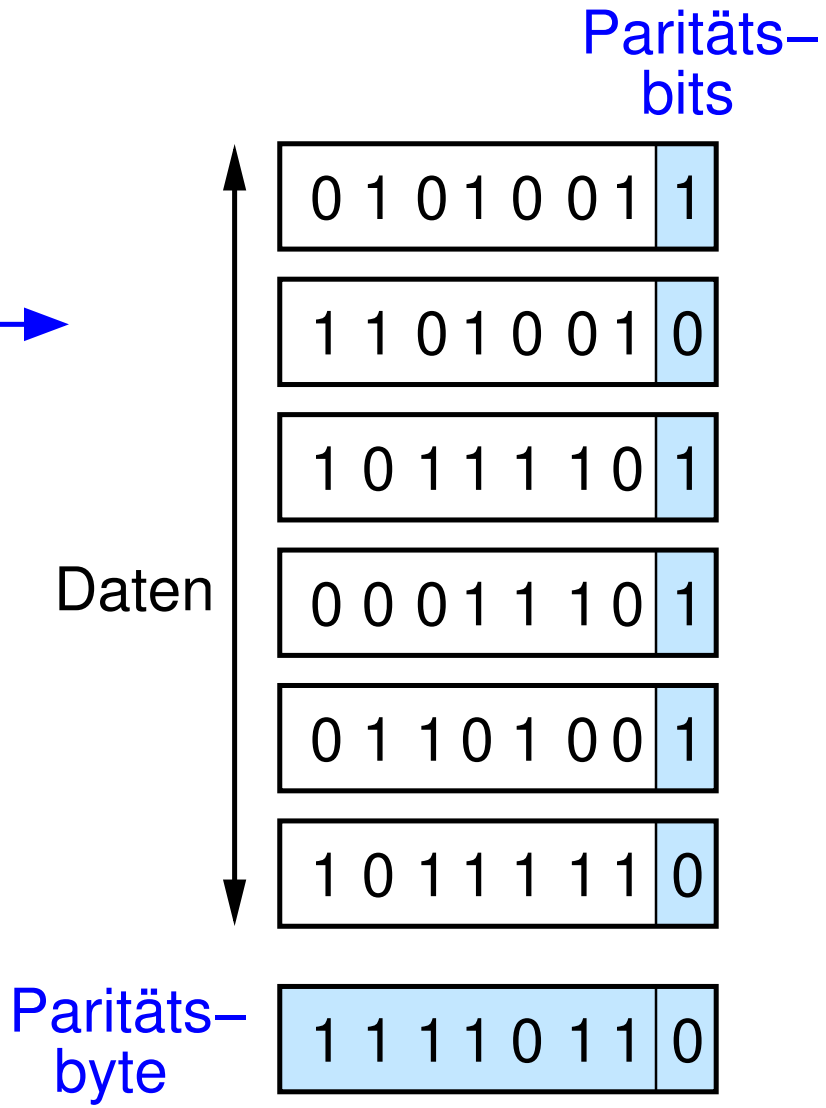
3.6 Fehlererkennung

★★

- Ziel: Übertragungsfehler in Frames erkennen (und behandeln)
- Möglichkeiten zur Fehlerbehandlung:
  - Korrektur des Fehlers beim Empfänger
  - Verwerfen der fehlerhaften Frames, Neuübertragung durch das Sicherungsprotokoll (☞ **7.4**)
- Vorgehensweise: Hinzufügen von **Redundanzbits** (Prüfbits) zu jedem Frame
- Theoretischer Hintergrund: **Hamming-Distanz**
  - Hamming-Distanz d = Minimale Anzahl von Bits, in denen sich zwei Worte eines Codes unterscheiden
  - $d \geq f + 1 \quad \Rightarrow$ f Einzelbitfehler erkennbar
  - $d \geq 2 \cdot f + 1 \Rightarrow$ f Einzelbitfehler korrigierbar
- Beispiel: Paritätsbit führt zu $d = 2$

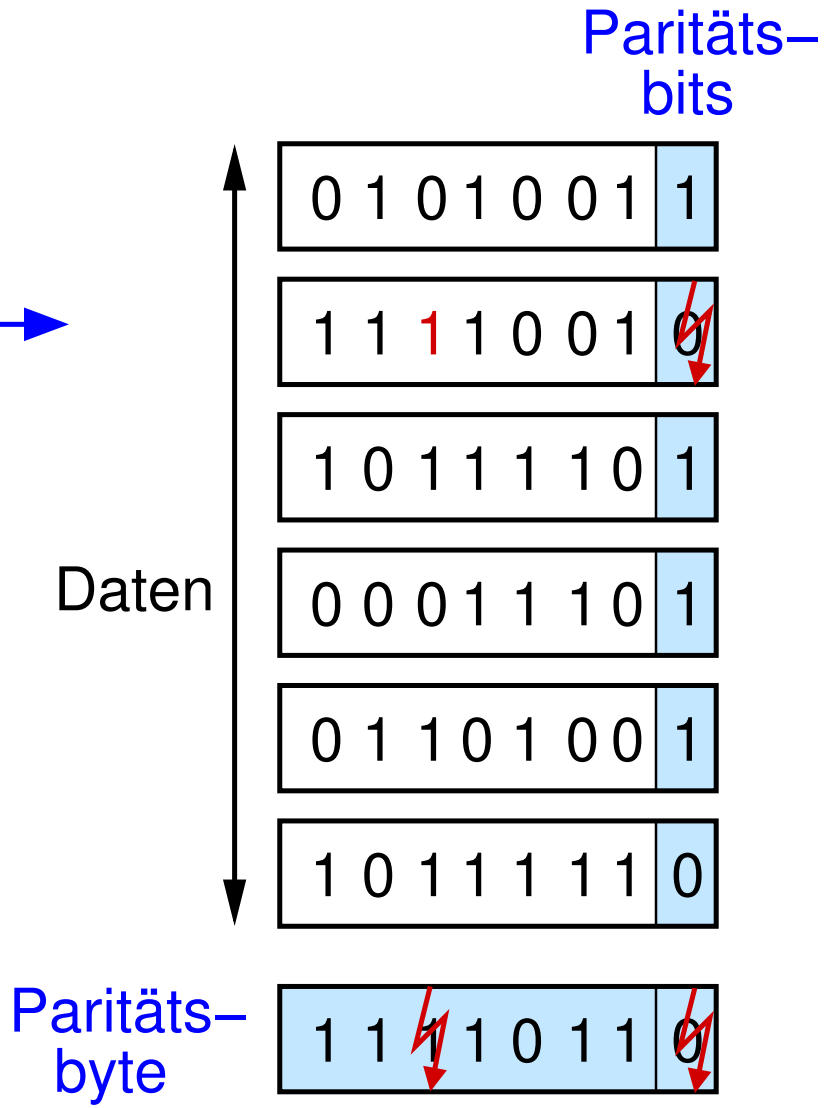
Zweidimensionale Parität

- Erweiterung der einfachen Parität
- Beispiel: 6 Worte á 7 Bit
- Erkennt alle 1, 2, 3 sowie die meisten 4-Bit-Fehler
- Erlaubt auch die Korrektur von 1-Bit-Fehlern



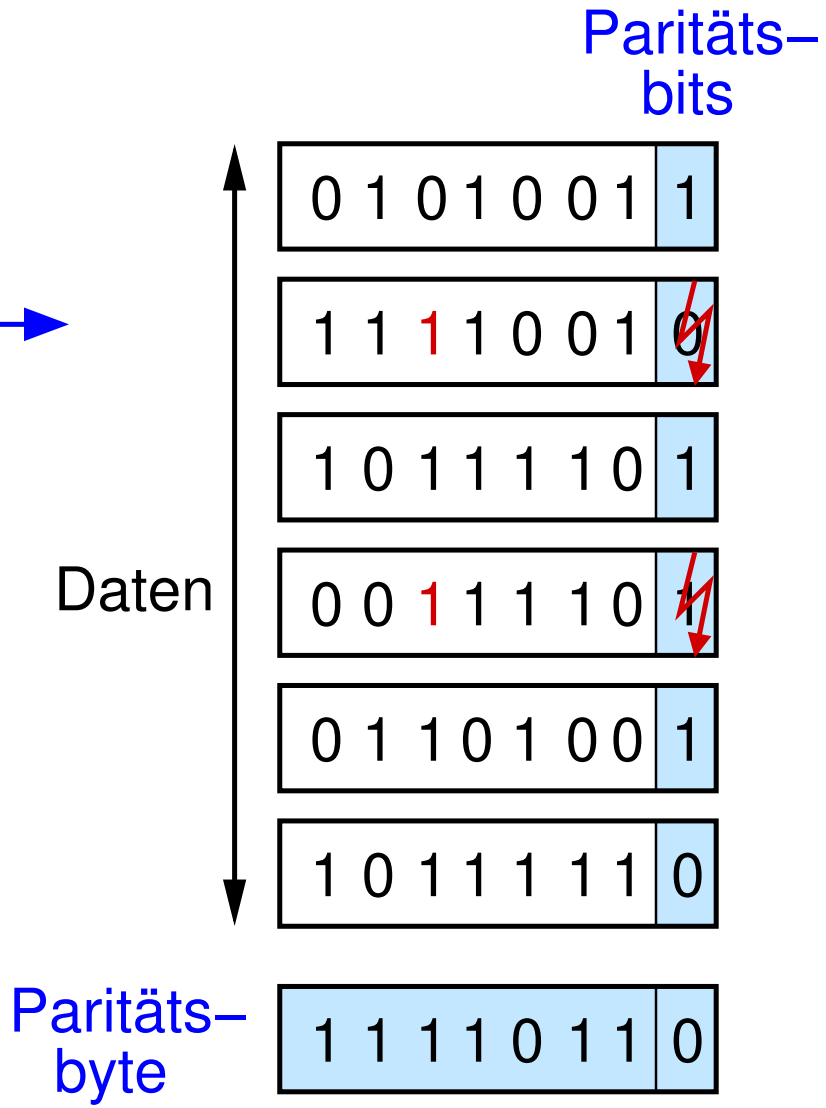
Zweidimensionale Parität

- Erweiterung der einfachen Parität
- Beispiel: 6 Worte á 7 Bit
- Erkennt alle 1, 2, 3 sowie die meisten 4-Bit-Fehler
- Erlaubt auch die Korrektur von 1-Bit-Fehlern



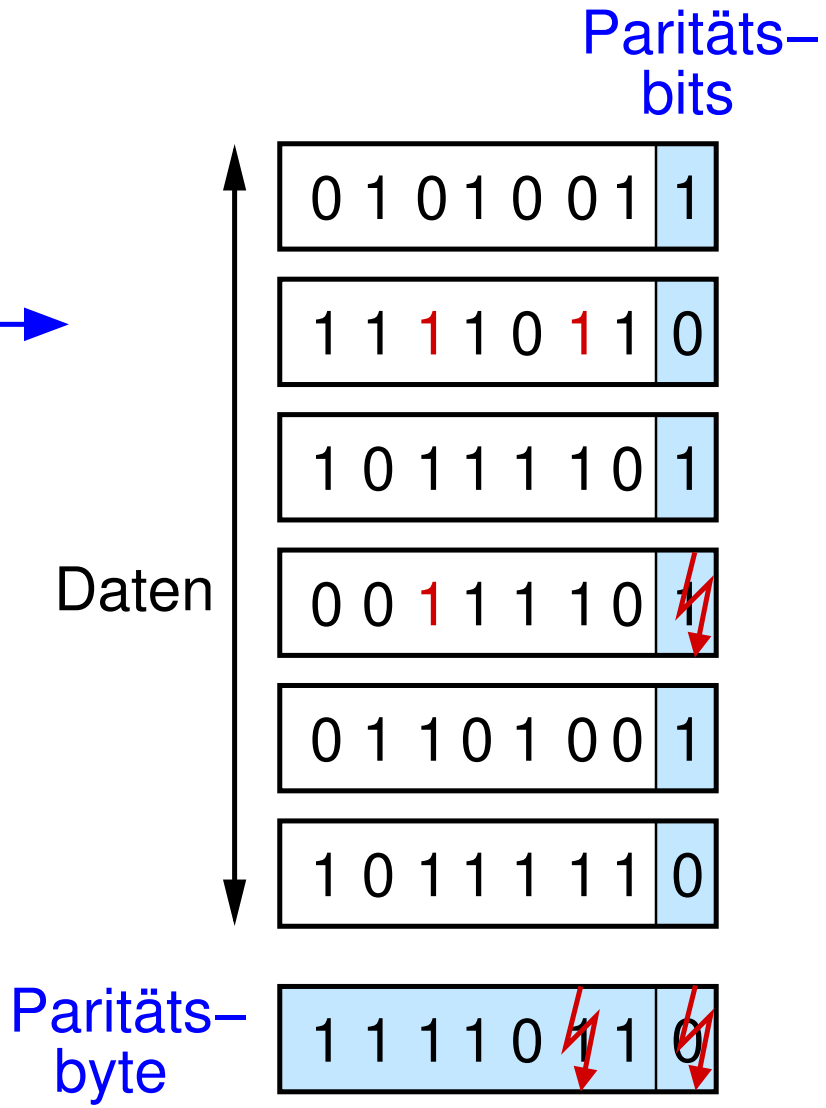
Zweidimensionale Parität

- Erweiterung der einfachen Parität
- Beispiel: 6 Worte á 7 Bit
- Erkennt alle 1, 2, 3 sowie die meisten 4-Bit-Fehler
- Erlaubt auch die Korrektur von 1-Bit-Fehlern



Zweidimensionale Parität

- Erweiterung der einfachen Parität
- Beispiel: 6 Worte á 7 Bit
- Erkennt alle 1, 2, 3 sowie die meisten 4-Bit-Fehler
- Erlaubt auch die Korrektur von 1-Bit-Fehlern



★★

CRC (*Cyclic Redundancy Check*)

- Ziel: hohe Wahrscheinlichkeit der Fehlererkennung mit möglichst wenig Prüfbits
- Basis des CRC-Verfahrens: Polynomdivision mit Modulo-2-Arithmetik (d.h. Add./Subtr. entspricht EXOR)
- Idee:
  - jede Nachricht *M* kann als Polynom $M(x)$ aufgefaßt werden, z.B.
    - *M* = 1 0 0 1 1 0 1 0 (Bits 7, 4, 3, 1 sind 1)
    - $M(x) = x^7 + x^4 + x^3 + x^1$
  - wähle Generatorpolynom $C(x)$ vom Grad k
  - erweitere *M* um k Prüfbits zu Nachricht *P*, so daß $P(x)$ ohne Rest durch $C(x)$ teilbar ist

★★

CRC (*Cyclic Redundancy Check*)

- Ziel: hohe Wahrscheinlichkeit der Fehlererkennung mit möglichst wenig Prüfbits
- Basis des CRC-Verfahrens: Polynomdivision mit Modulo-2-Arithmetik (d.h. Add./Subtr. entspricht EXOR)
- Idee:
  - jede Nachricht *M* kann als Polynom $M(x)$ aufgefaßt werden, z.B.
    7 6 5 4 3 2 1 0
    - *M* = 1 0 0 1 1 0 1 0 (Bits 7, 4, 3, 1 sind 1)
    - $M(x) = x^7 + x^4 + x^3 + x^1$
  - wähle Generatorpolynom $C(x)$ vom Grad k
  - erweitere *M* um k Prüfbits zu Nachricht *P*, so daß $P(x)$ ohne Rest durch $C(x)$ teilbar ist

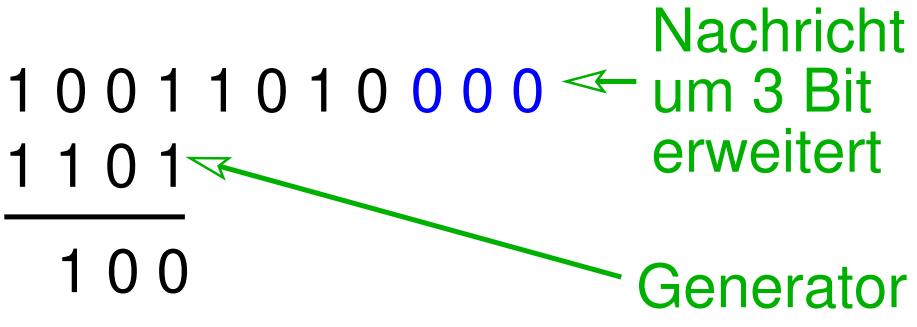
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101

1 0 0 1 1 0 1 0 0 0 0 ← Nachricht um 3 Bit erweitert

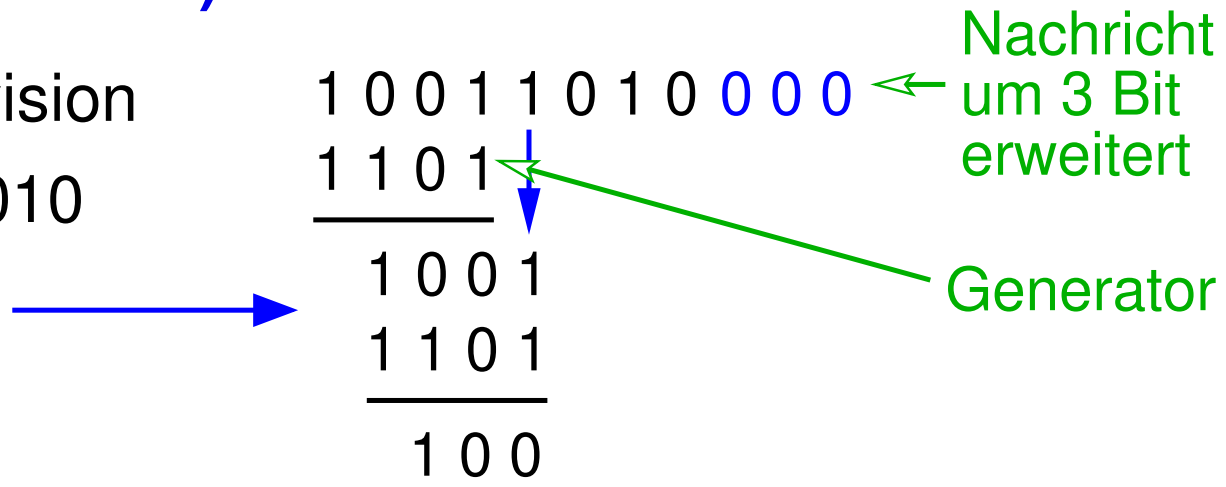
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101



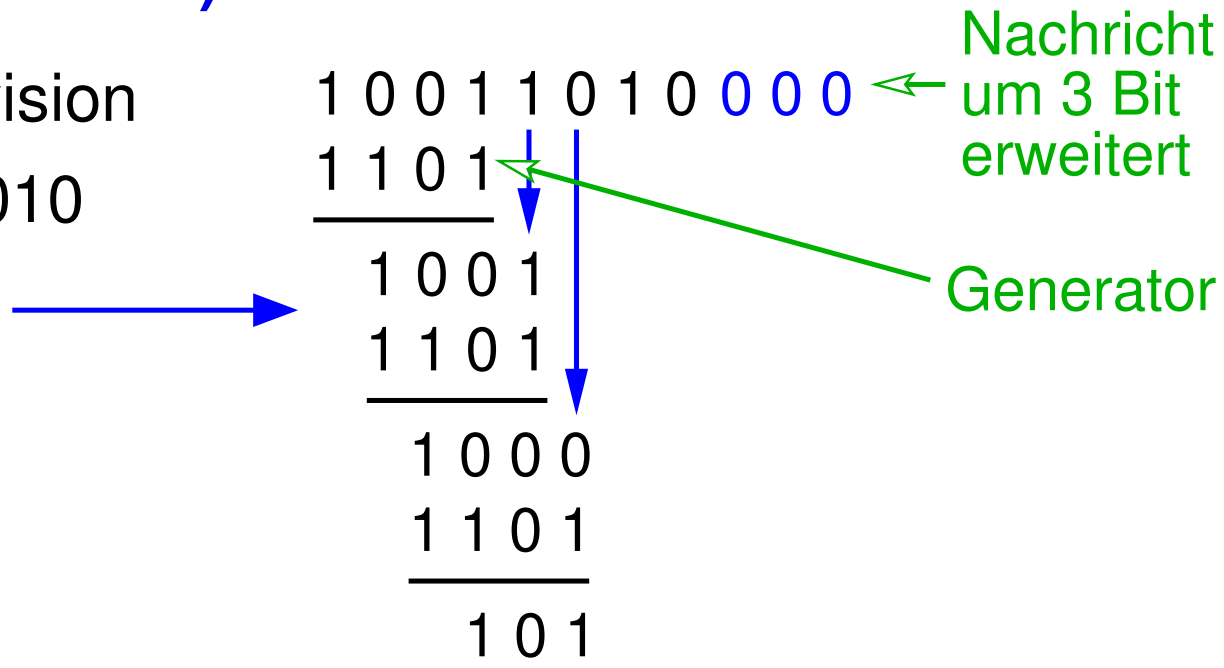
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101



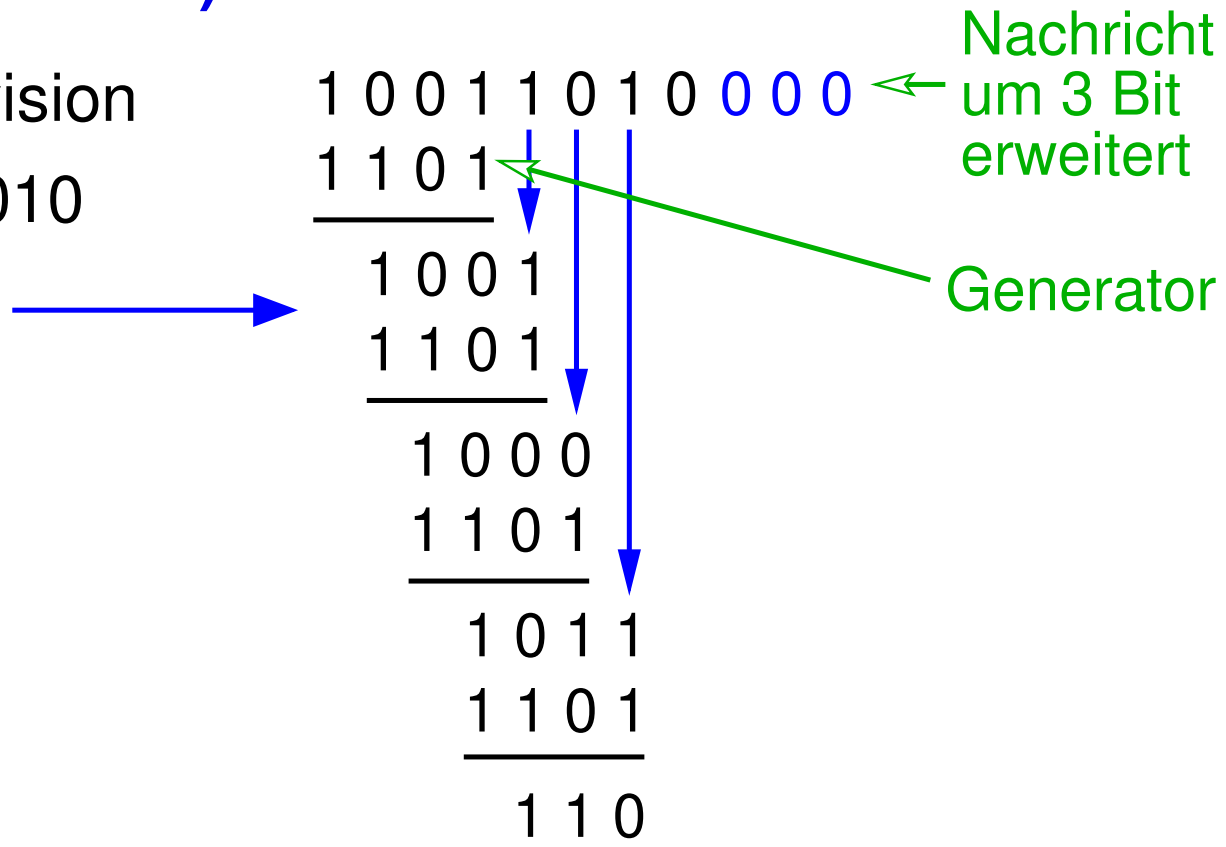
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101



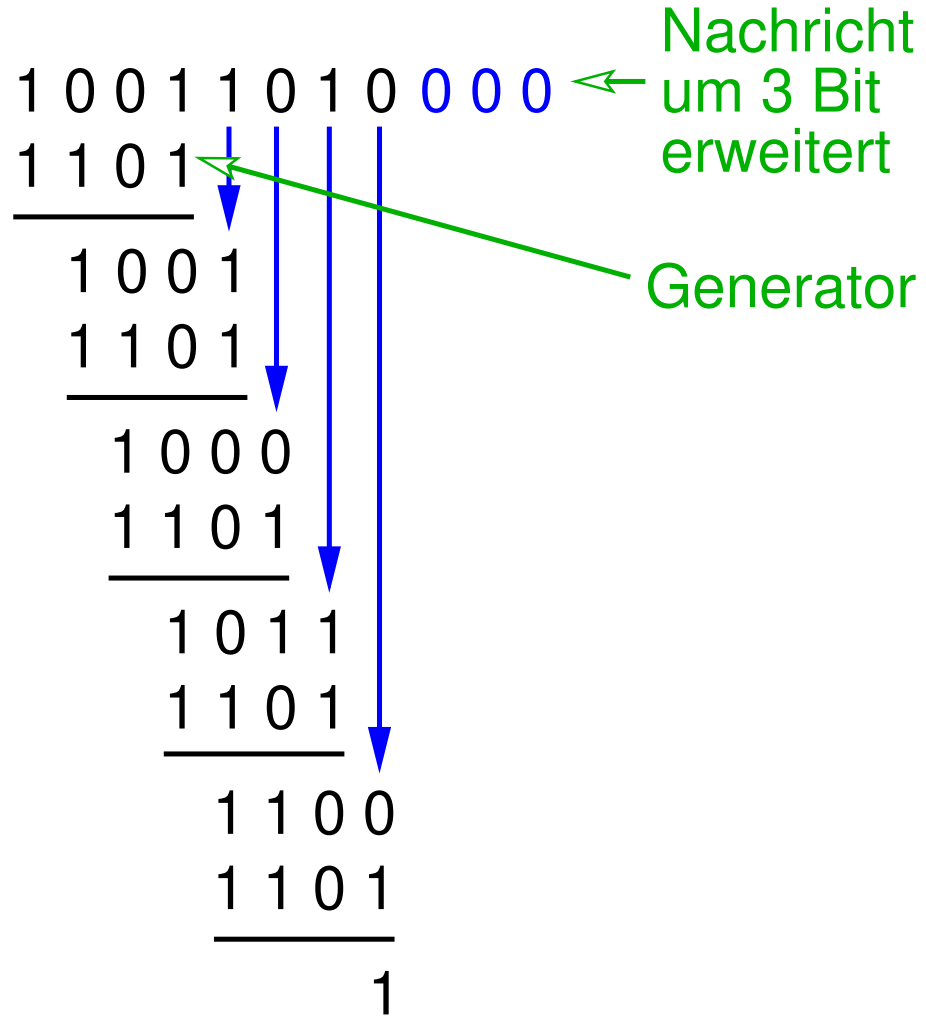
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101



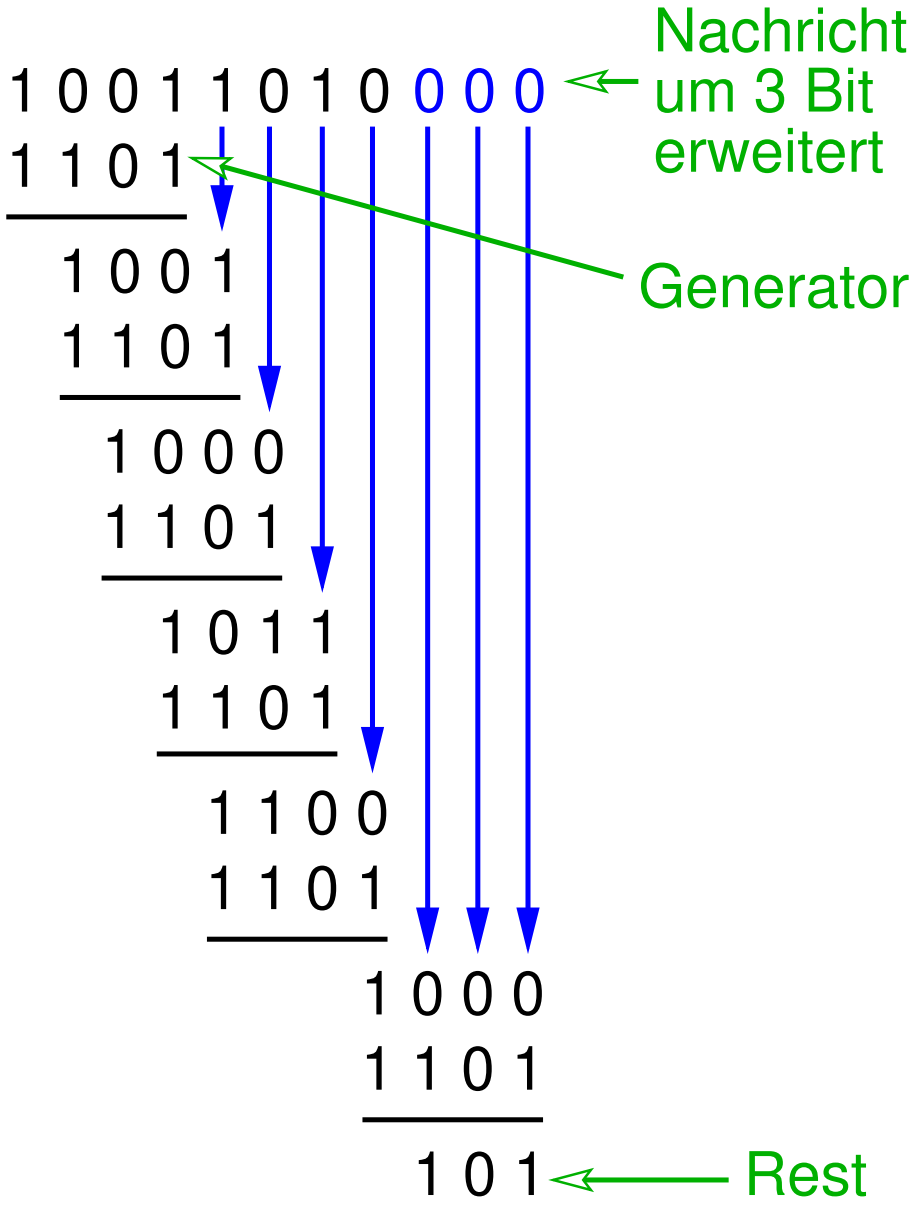
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101



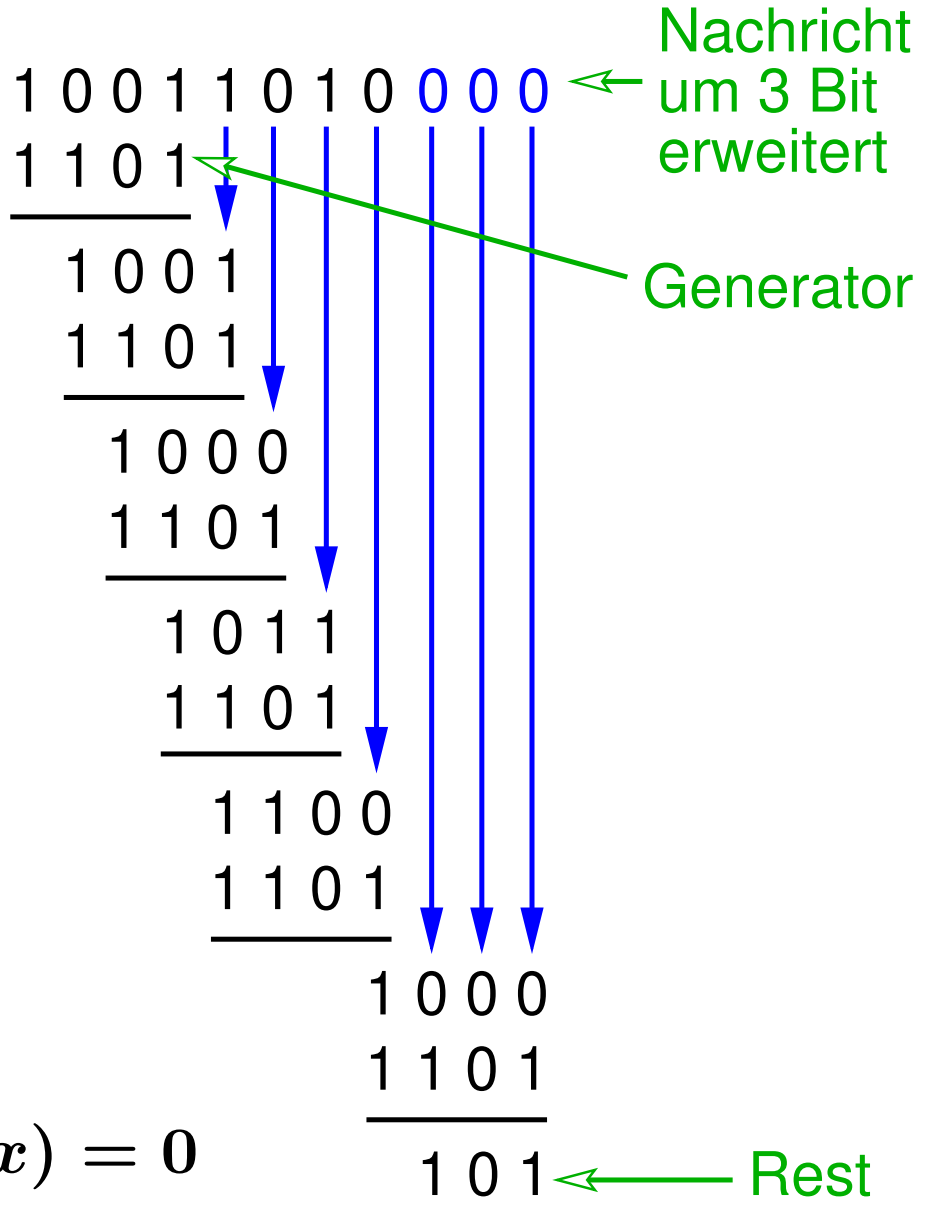
CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht M: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator C: 1101



CRC (*Cyclic Redundancy Check*) ...

- Beispiel zur Polynomdivision
  - Nachricht *M*: 10011010
  - 3 Prüfbits ($k = 3$)
  - Generator *C*: 1101
- Divisionsrest *R* wird an die Nachricht *M* angefügt
- Versendete Nachricht *P*: 10011010101
- Diese Nachricht ist durch den Generator ohne Rest teilbar:
  - $R(x) = M(x) \mod C(x)$
    $\Rightarrow (M(x) - R(x)) \mod C(x) = 0$



CRC (*Cyclic Redundancy Check*) ...

- Wahl des Generatorpolynoms?
  - so, daß möglichst viele Fehler erkannt werden!
  - Beispiel für ein übliches CRC-Polynom:
    - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-16 erkennt:
    - alle Ein-und Zweibitfehler
    - alle Fehler mit ungerader Bitanzahl
    - alle Fehlerbündel mit Länge $\leq$ 16 Bit

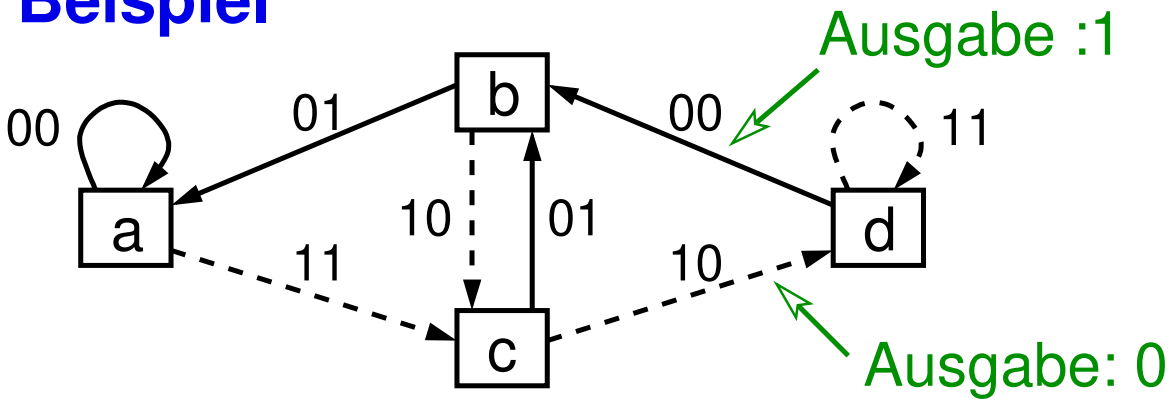
- Gründe für den Einsatz von CRC:
  - gute Fehlererkennung
  - sehr effizient in Hardware realisierbar

3.7 Nachtrag: Kanalcodierung

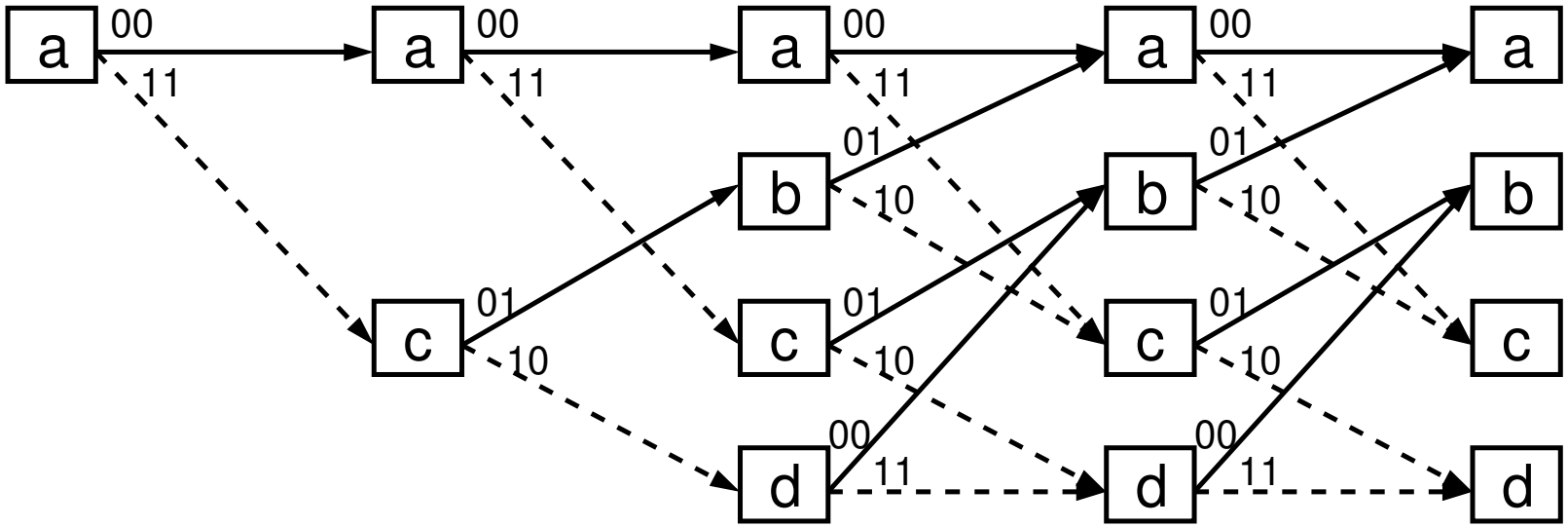
- Ziel: Fehlerkorrektur bei der Bitübertragung (OSI-Schicht 1)
- *Linear Block Codes*
  - Codeworte fester Länge werden durch längere Codeworte mit Redundanz ersetzt
    - z.B. Anfügen eines Paritätsbits an jedes übertragene Byte
- *Convolutional Codes*
  - **Strom** von Eingabezeichen (bzw. -bits) wird durch Zustands-automat um Redundanzbits erweitert
    - Redundanzbit kann von allen bisherigen Eingabezeichen abhängen
  - beim Dekodieren wird der wahrscheinlichste **Pfad** durch die Zustände gesucht (Viterbi Decoder)

Convolutional Codes: Beispiel

- Automat zur Decodierung:

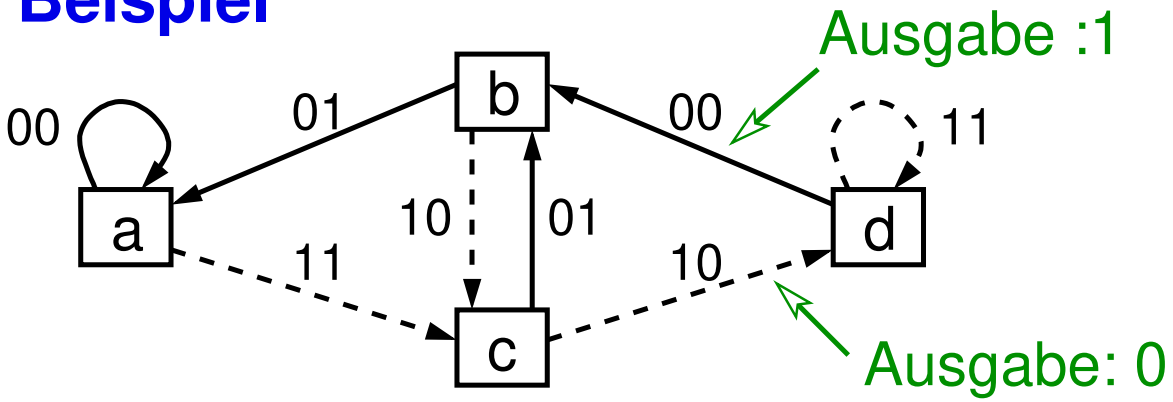


- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):

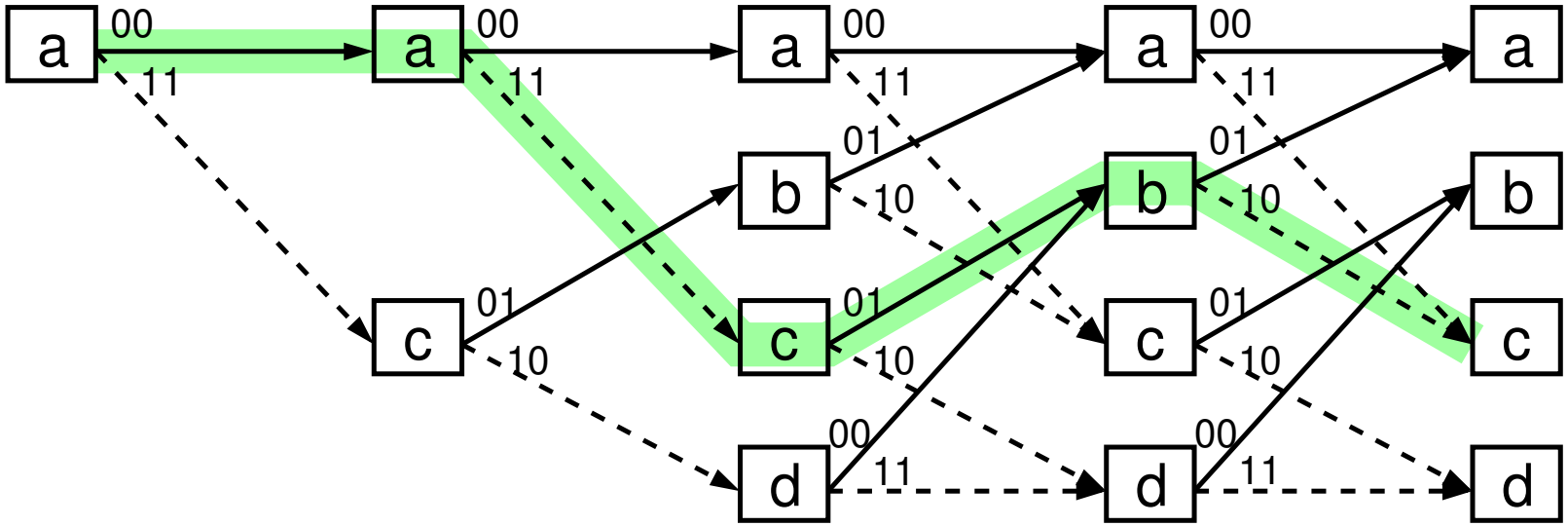


Convolutional Codes: Beispiel

- Automat zur Decodierung:



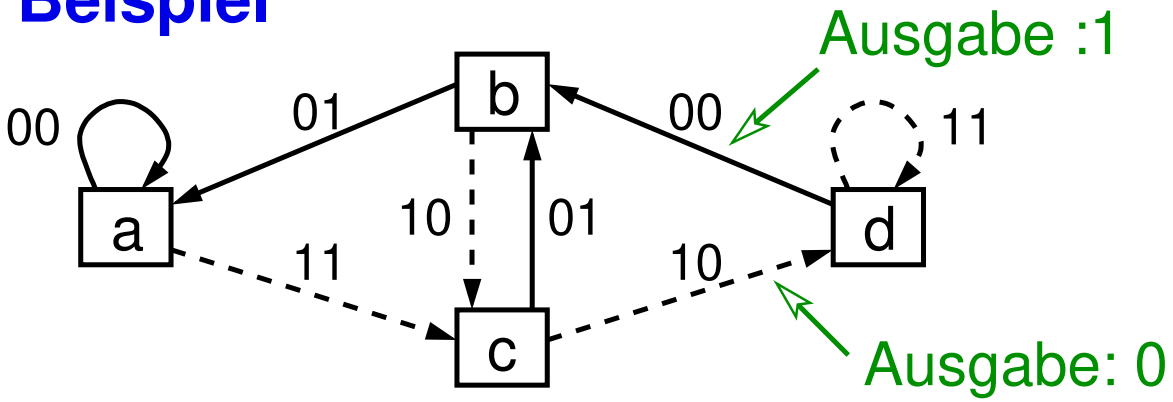
- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):



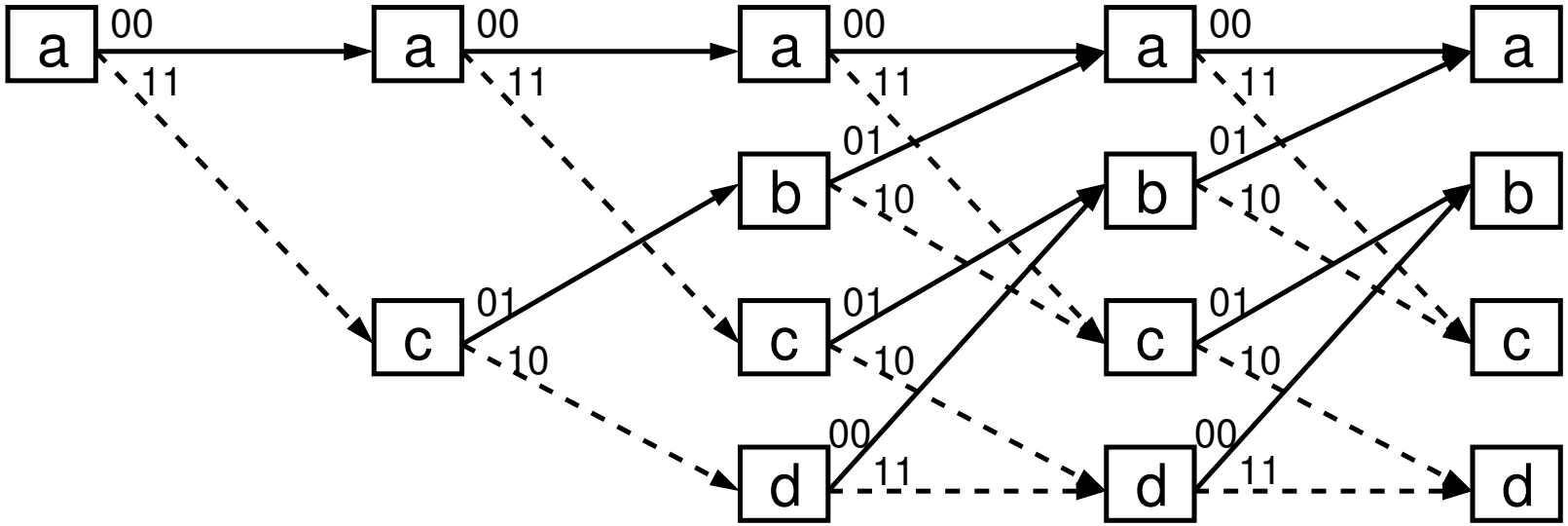
Empfangener Bitstrom: 00 11 01 10 –> 1 0 1 0

Convolutional Codes: Beispiel

- Automat zur Decodierung:



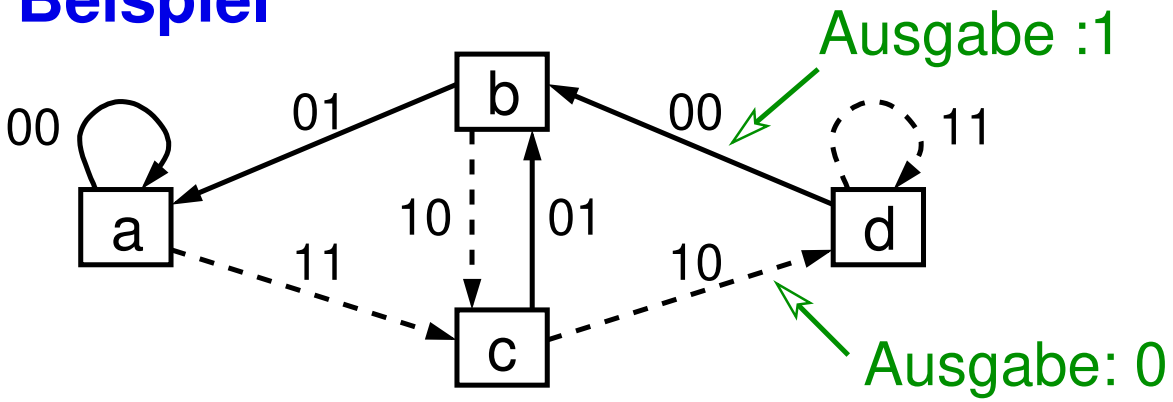
- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):



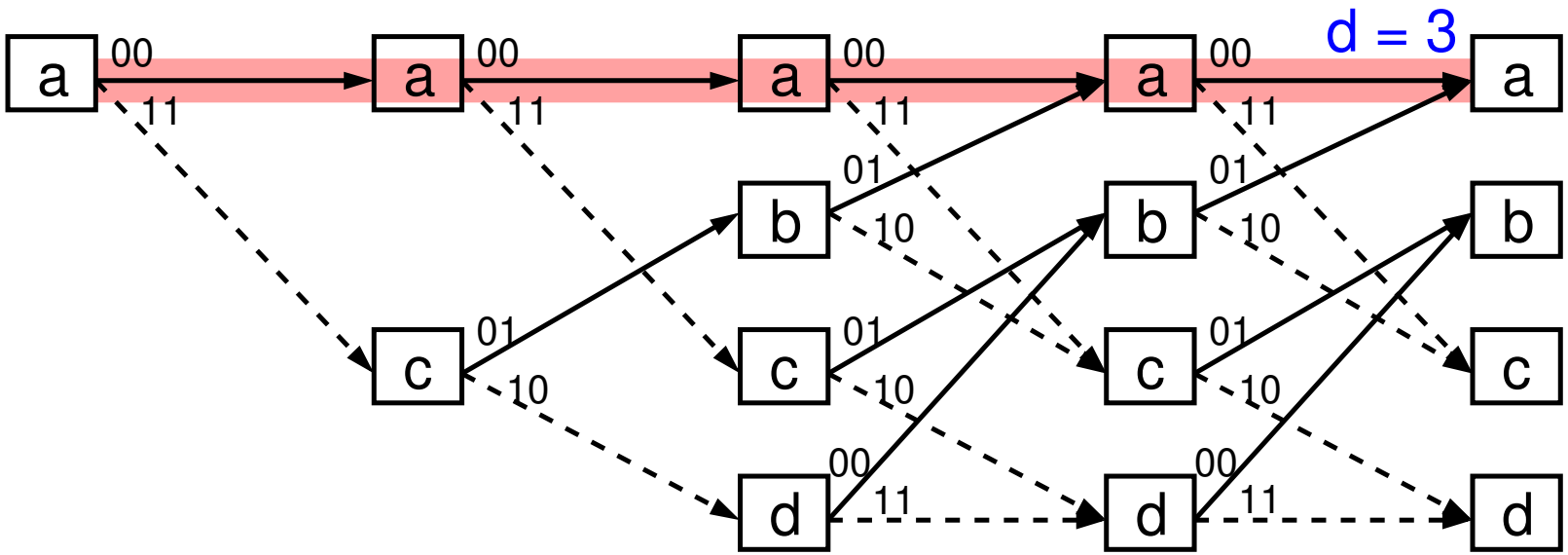
Empfangener Bitstrom: 00 01 01 10

Convolutional Codes: Beispiel

- Automat zur Decodierung:



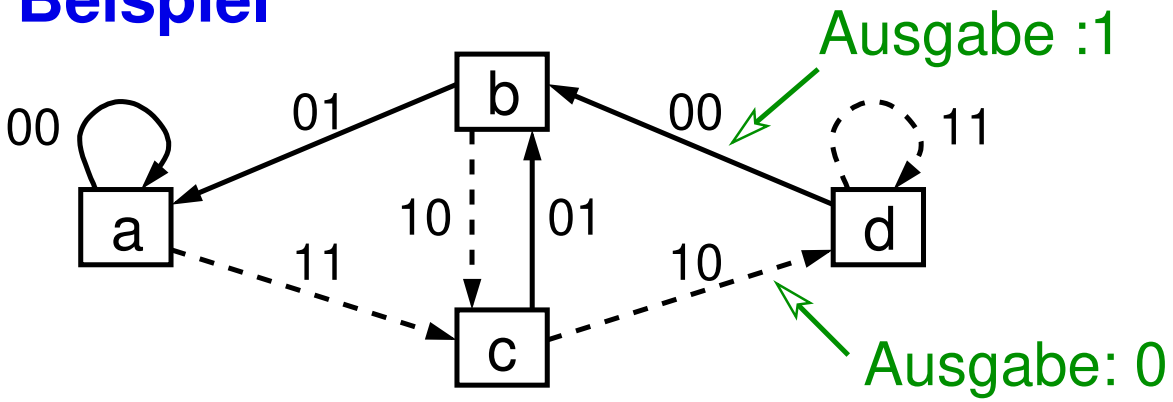
- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):



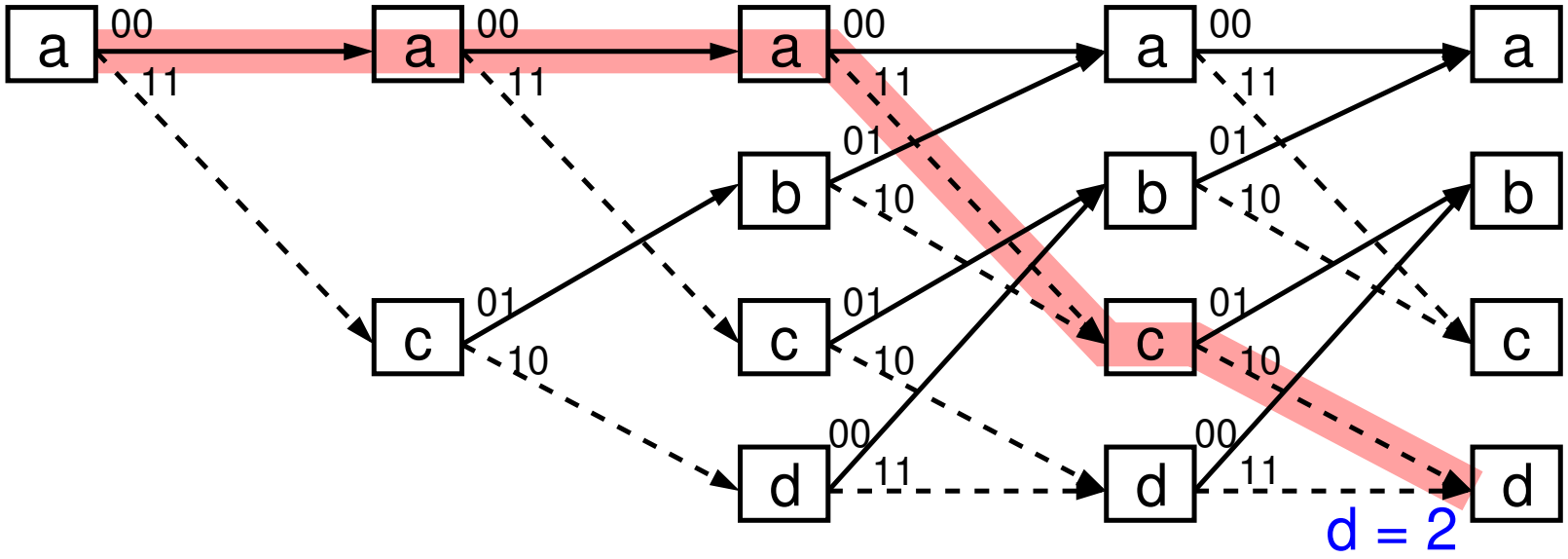
Empfangener Bitstrom: 00 01 01 10

Convolutional Codes: Beispiel

- Automat zur Decodierung:



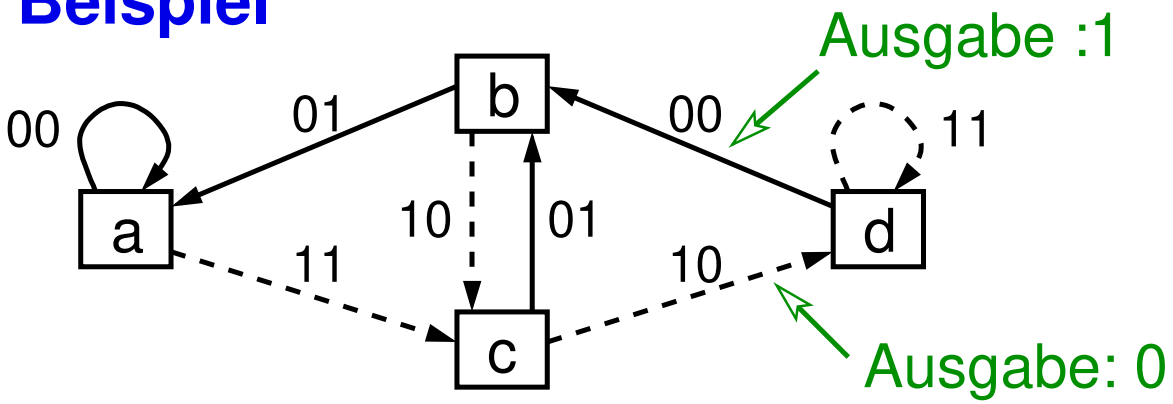
- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):



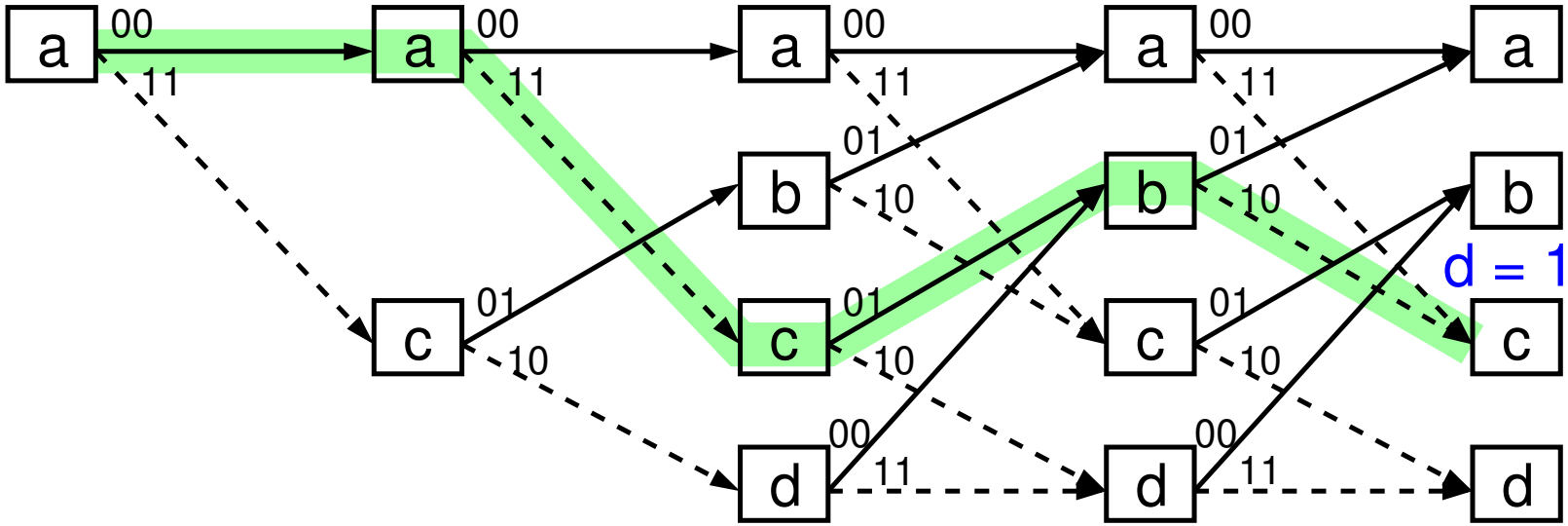
Empfangener Bitstrom: 00 01 01 10

Convolutional Codes: Beispiel

- Automat zur Decodierung:



- „Abgewickelter“ Zustandsgraph (Trellis-Diagramm):



Empfangener Bitstrom: 00 01 01 10

★★

- In vielen LANs:
  - Knoten greifen auf ein gemeinsames Medium zu
  - Zugriff muß geregelt werden, um **Kollisionen** zu vermeiden:
    - zu jeder Zeit darf nur jeweils ein Knoten senden

- Typische Vorgehensweisen:
  - statische Aufteilung
  - *Random Access* Verfahren
    - z.B. CSMA/CD (Ethernet)
  - kollisionsfreie Verfahren (für Echtzeit-Anwendungen)
    - z.B. Token-Ring

3.8.1 Statische Verfahren

- Feste Aufteilung des Mediums auf die Stationen ($\sim$ Multiplexing)
  - häufig aufgrund vorheriger Reservierung ($\rightarrow$ Dynamik)
- FDMA (*Frequency Division Multiple Access*): Zuteilung eines (unterschiedlichen) Frequenzbands
  - z.B. Kabelfernsehen, Mobilfunk
- TDMA (*Time* $\sim$): Zuteilung fester Zeitschlitze
  - häufig in Netzen für Automatisierungssysteme
- CDMA (*Code* $\sim$): gleichzeitiges Senden auf gespreiztem Frequenzband mit verschiedenen Codierungen
- SDMA (*Space* $\sim$): Fokussierung des Funkstrahls auf das Gebiet des jeweiligen Station
  - z.B. Mobilfunk-Netze

3.8.2 Dynamische Verfahren mit *Random Access*

- Unabhängige Stationen versuchen zu zufälligen Zeiten auf dasselbe Medium zu senden
  - gleichzeitiges Senden führt zu Kollision
- Unterschiedliche Voraussetzungen:
  - Trägerprüfung (sendet schon jemand?) möglich / nicht möglich
  - Sendezeitpunkt beliebig / nur zu Beginn fester Zeitscheiben
  - Kollision beobachtbar / nicht beobachtbar
- Führt zu vielen verschiedenen Verfahren
  - ohne Kollisioserkennung: z.B. Aloha, slotted Aloha, CSMA
  - mit Kollisioserkennung: CSMA/CD (☞ **3.8.5**)

Aloha

- Entwickelt in Hawaii in den 1970'er Jahren
  - Verbindung zum Hauptrechner über Inseln hinweg
- Station sendet zu beliebigem Zeitpunkt
  - keine Trägerprüfung, keine Zeitscheiben
- Empfänger quittiert den Empfang
  - keine Kollisionserkennung
- Ggf. Neuübertragung nach zufälliger Wartezeit
  - verhindert sofortige Neu-Kollision
- Bei fester Framegröße und konstanter mittlerer Framerate: Auslastung max. 18%
  - (diese Annahmen sind in der Praxis unrealistisch)

Rechnernetze I

SoSe 2025

08.05.2025

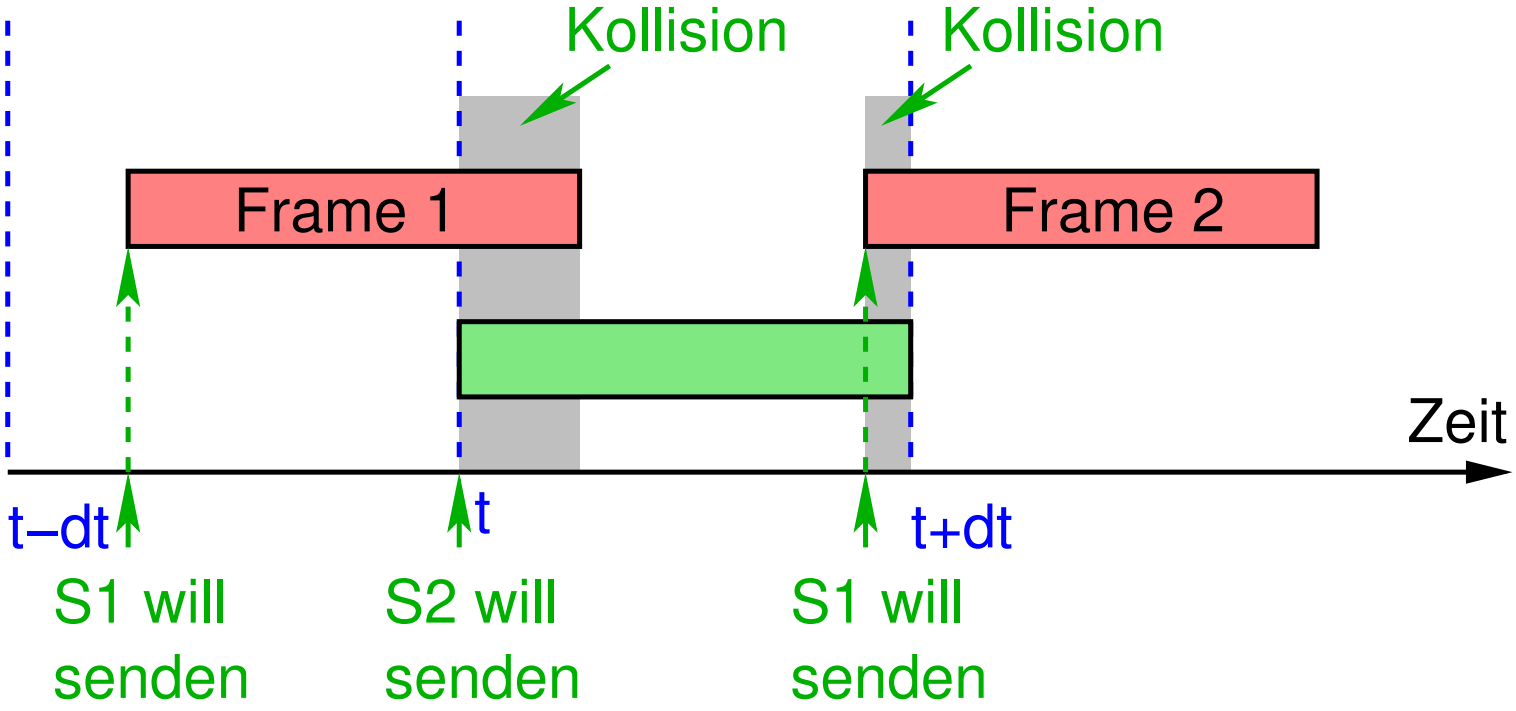
Roland Wismüller
Universität Siegen
roland.wismueller@uni-siegen.de
Tel.: 0271/740-4050, Büro: H-B 8404

Stand: 30. April 2025

Slotted Aloha

- Wie Aloha, aber mit Zeitscheiben
  - benötigt Synchronisation der Stationen
- Verdopplung der max. Auslastung gegenüber Aloha

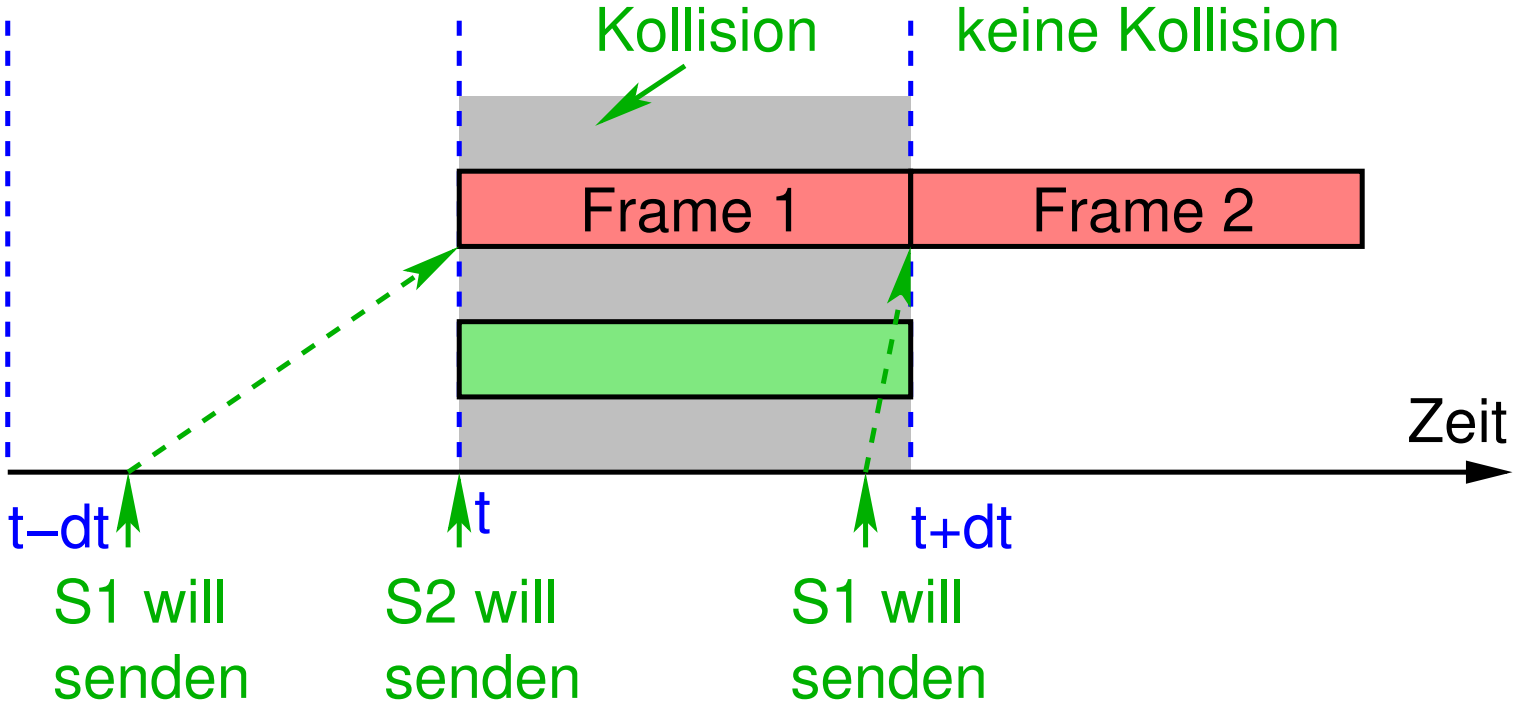
Ohne Zeitscheiben:



Slotted Aloha

- Wie Aloha, aber mit Zeitscheiben
  - benötigt Synchronisation der Stationen
- Verdopplung der max. Auslastung gegenüber Aloha

Mit Zeitscheiben:



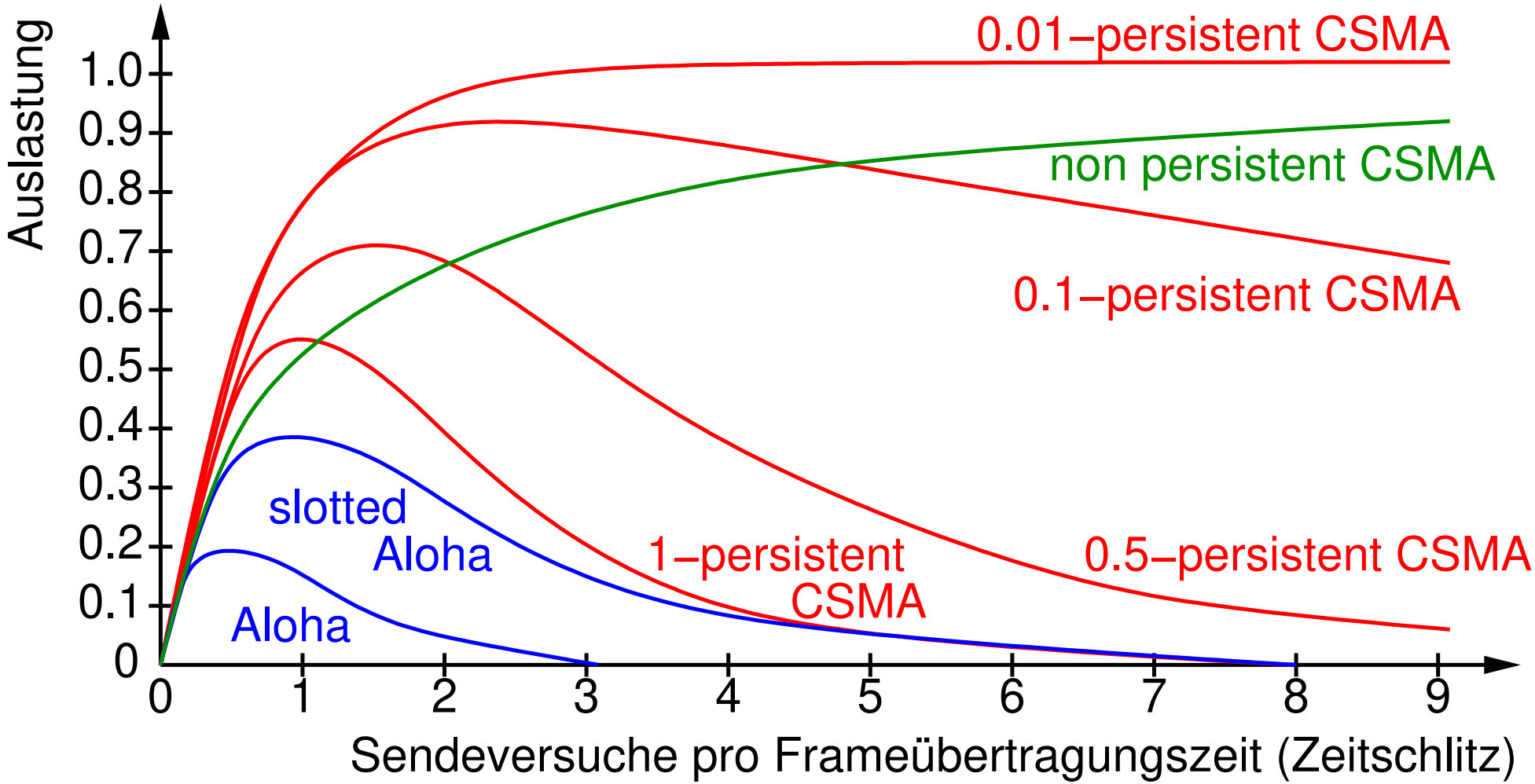
Beispiel: RFID-Tags

- Lesegerät sendet durchgehendes Signal aus
  - zur Energieversorgung der Tags
- Tags können Signal reflektieren oder absorbieren
  - Informationsübertragung an das Lesegerät
- Tags können nicht feststellen, ob ein anderes Tag „sendet“
- Vorgehensweise:
  - Lesegerät sendet Anfrage mit Zahl der Zeitscheiben; wiederholt Anfrage periodisch (für jede Zeitscheibe)
  - Tag wählt zufällige Zeitscheibe und sendet kurze Antwort
  - Lesegerät bestätigt, falls keine Kollision
  - nach Bestätigung sendet Tag seine ID

Carrier Sense Multiple Access (CSMA)

- Grundidee: höre das Medium vor dem Senden ab
  - falls frei: sende; falls belegt: sende, wenn wieder frei
- Verschiedene Designentscheidungen möglich:
  - sende immer, oder sende nur mit Wahrscheinlichkeit p (setzt Zeitscheiben voraus)
  - sende sofort, wenn Medium frei wird, oder warte zufällige Zeit
- Varianten
  - **1-persistent CSMA**: sende immer sofort mit $p = 1$
  - **nonpersistent CSMA**: falls frei, sende mit $p = 1$, falls nicht, warte zufällige Zeit und wiederhole
  - **p-persistent CSMA**: falls frei, sende mit Wahrscheinlichkeit p in diesem Zeitschlitz; wiederhole im nächsten Zeitschlitz

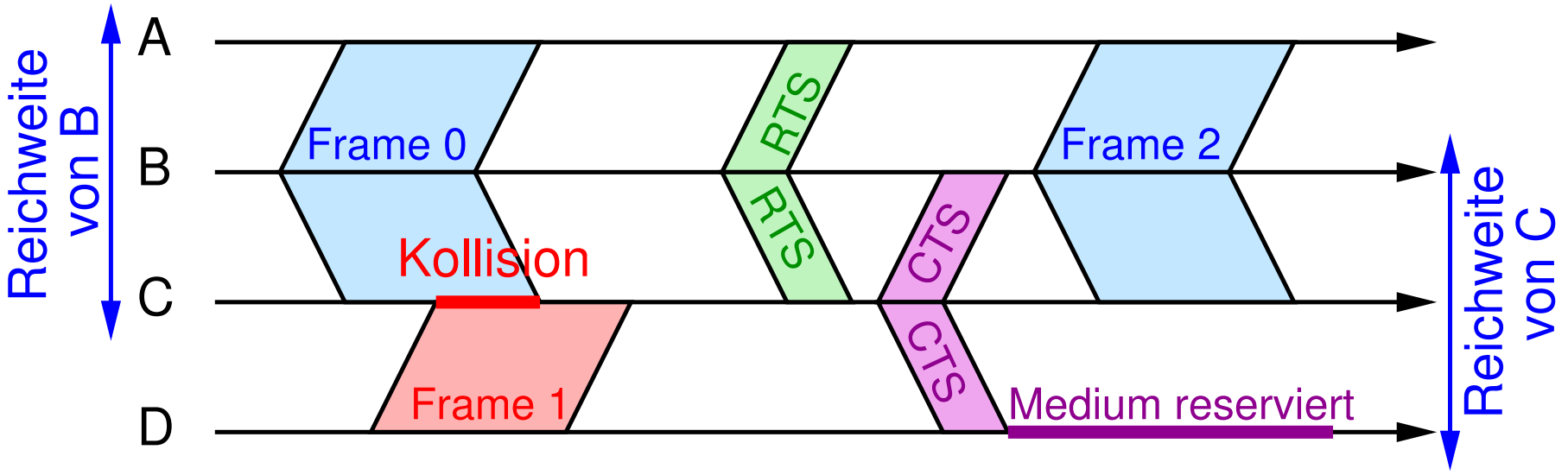
Vergleich



➥ Hier vorausgesetzt: keine Bursts, einheitliche Framelänge

CSMA bei drahtloser Übertragung

- Problem: Reichweite des Signals ist begrenzt
  - Station kann daher evtl. andere sendende Station nicht „hören“
  - Folge: erhöhte Zahl von Kollisionen (CSMA → Aloha)
- Lösung: Reservierung des Mediums durch den Empfänger
  - MACA-Protokoll mit RTS/CTS (*Request / Clear To Send*)



3.8.3 Kollisionsfreie dynamische Verfahren

Master/Slave

- eine Station im Netz ist Master
  - fordert andere Stationen (Slaves) zum Senden auf
- Slave darf nur nach Anforderung senden
- Verwendung z.B. bei Bluetooth

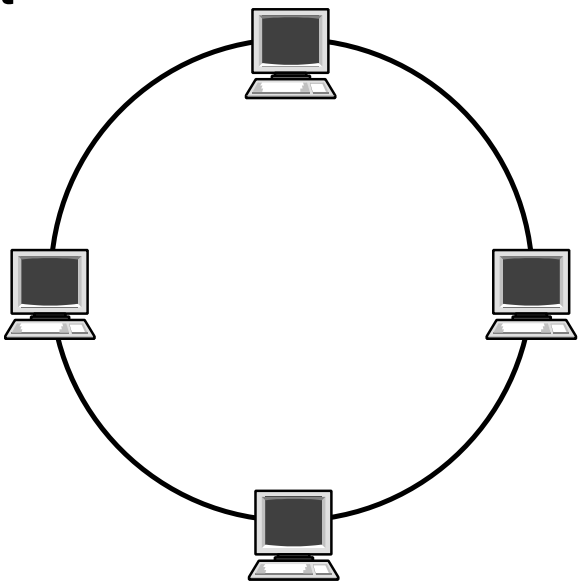
Tokenweitergabe

- Token = Erlaubnis zum Senden
- Token wird im Netz zyklisch weitergegeben
- Beispiel: Token-Ring

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

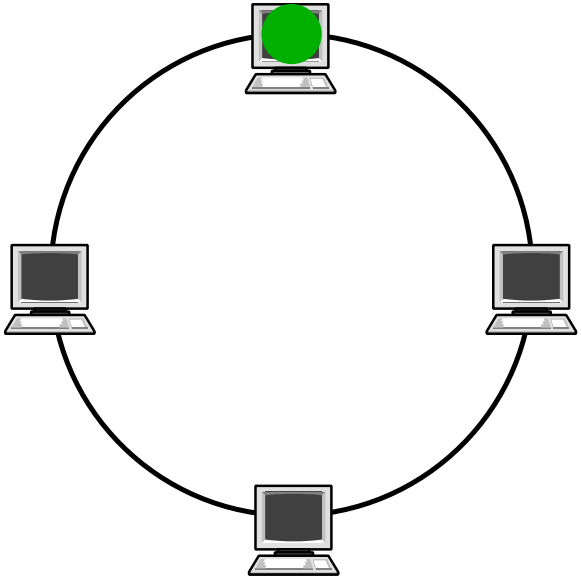


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

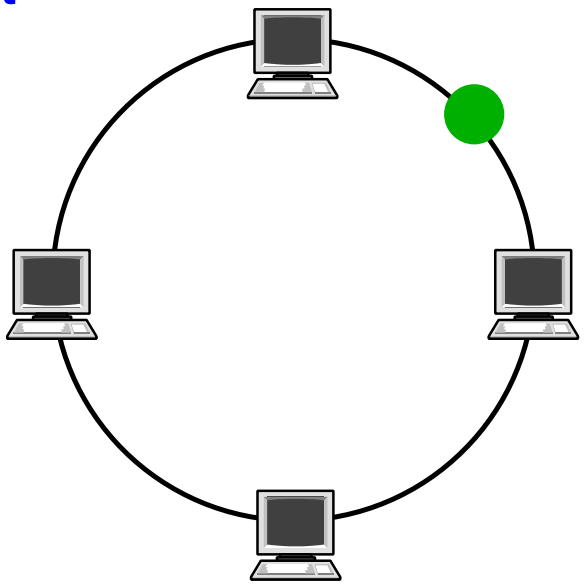


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

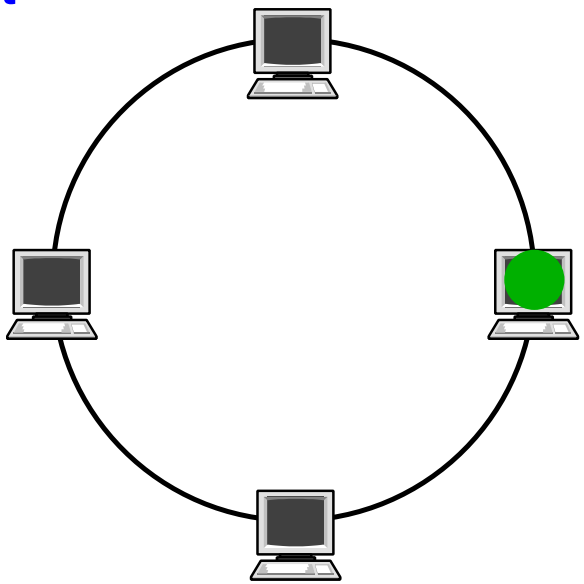


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

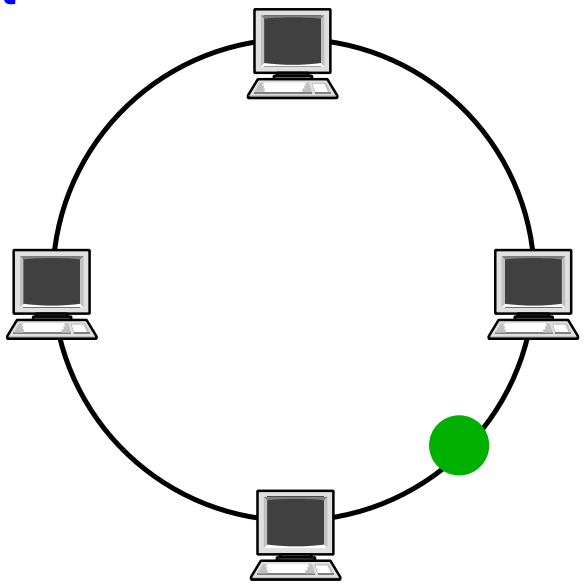


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

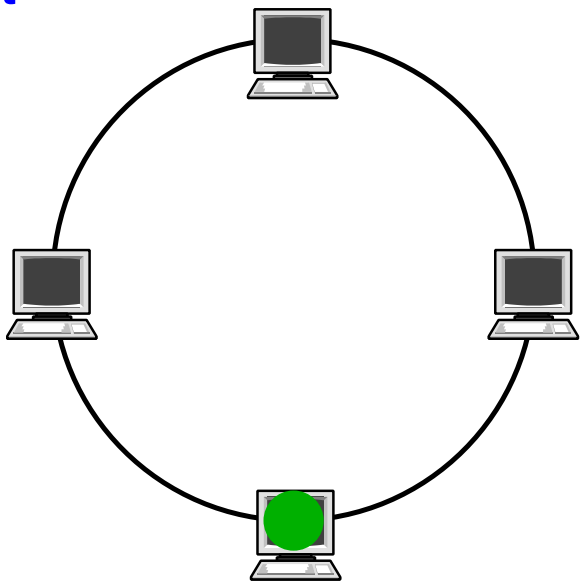


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

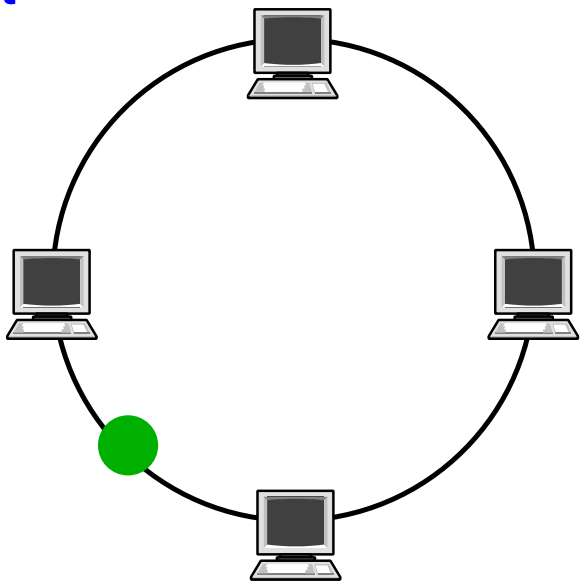


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

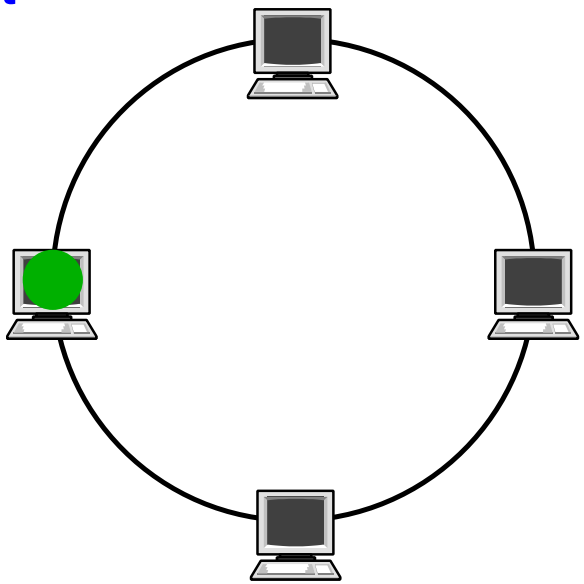


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

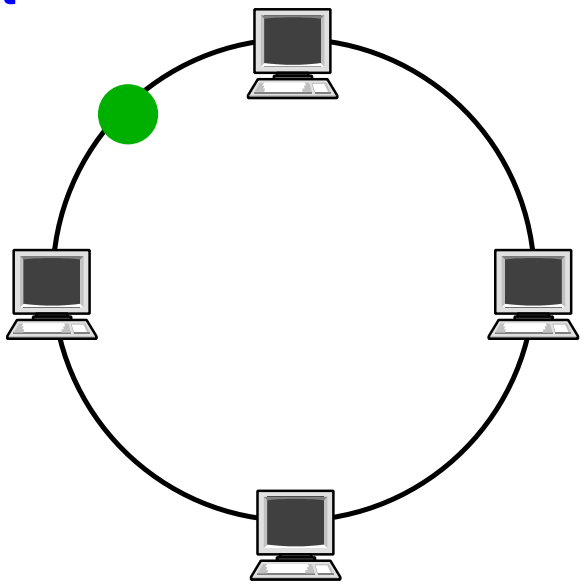


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

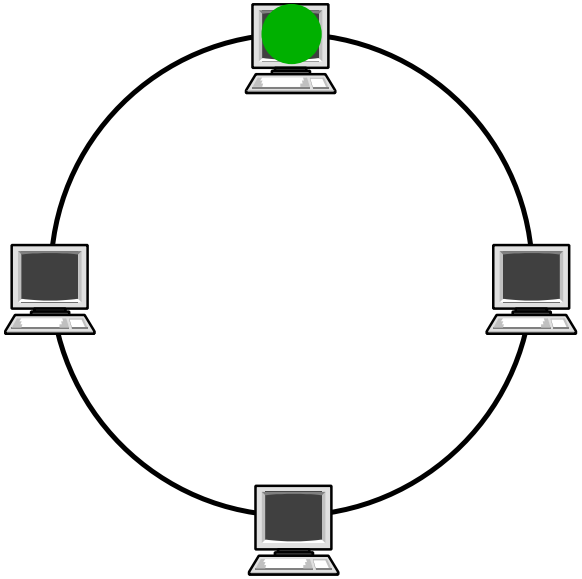


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

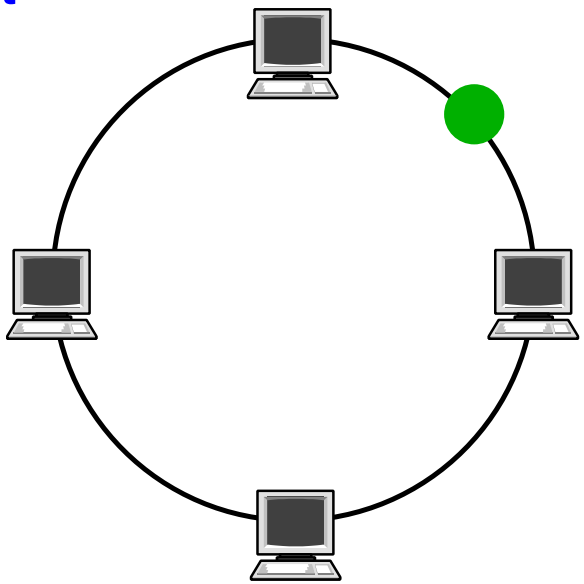


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

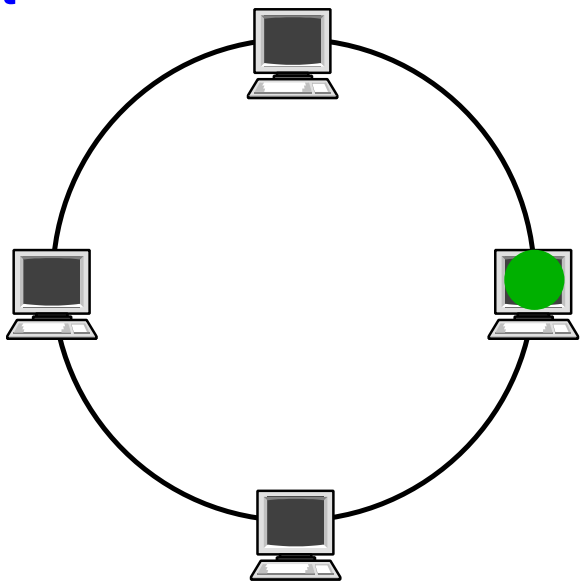


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

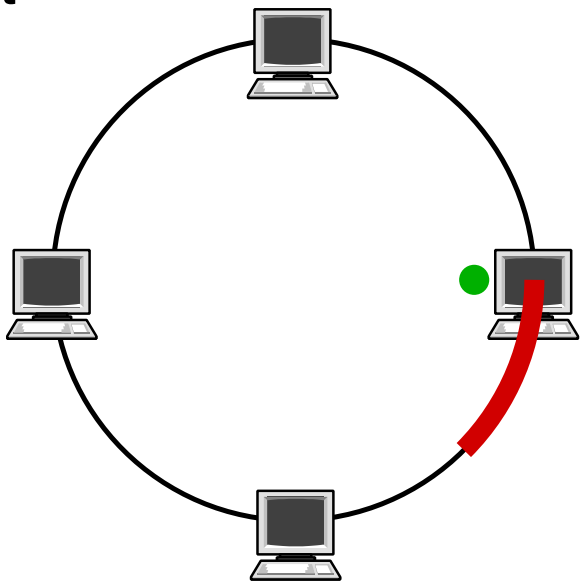


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

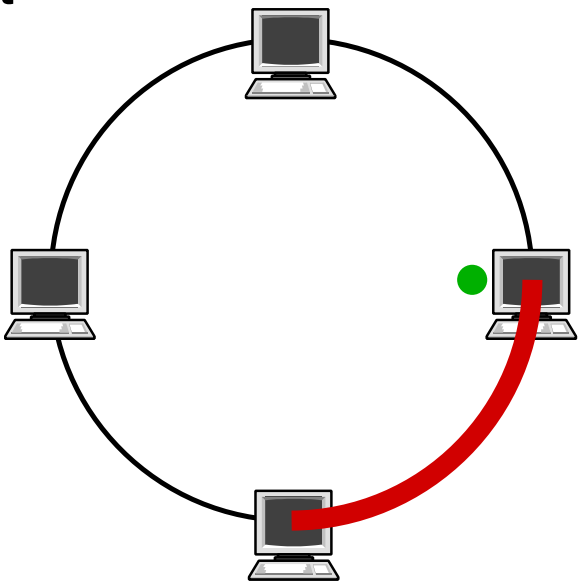


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

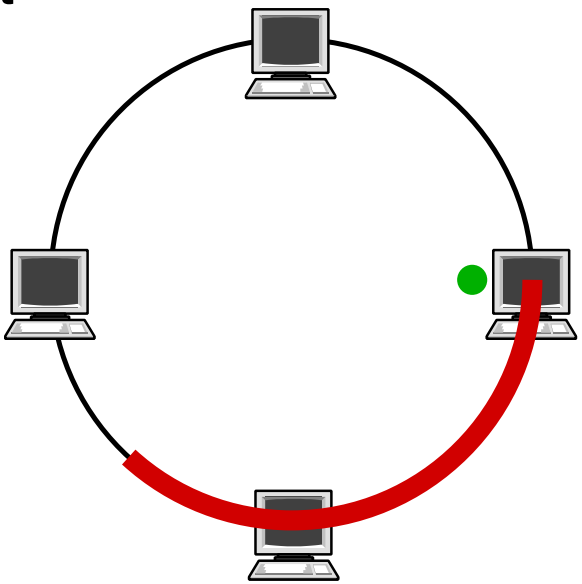


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

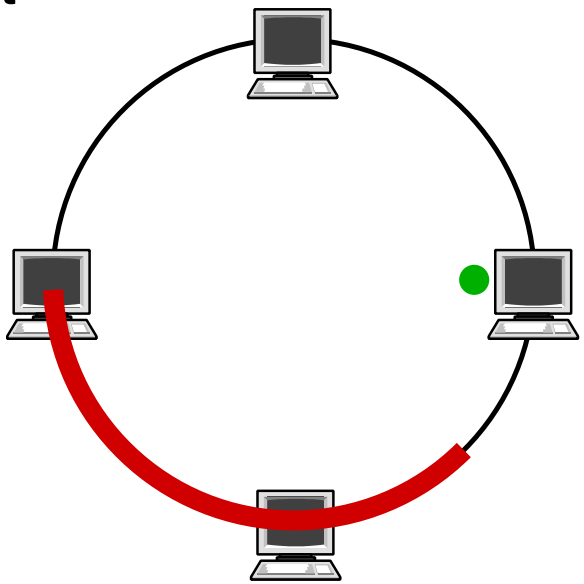


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

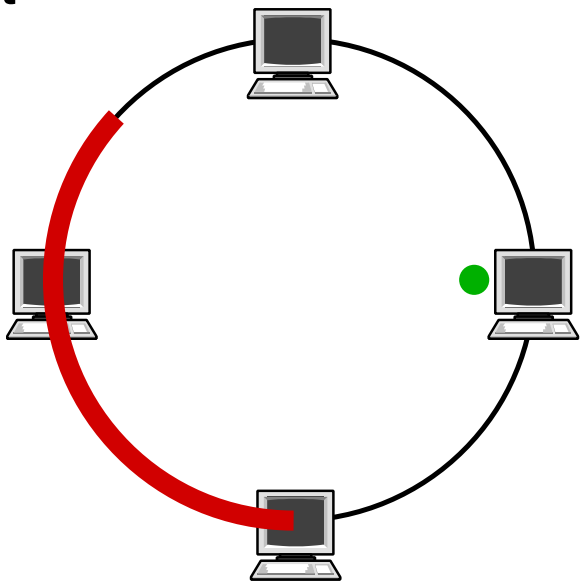


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

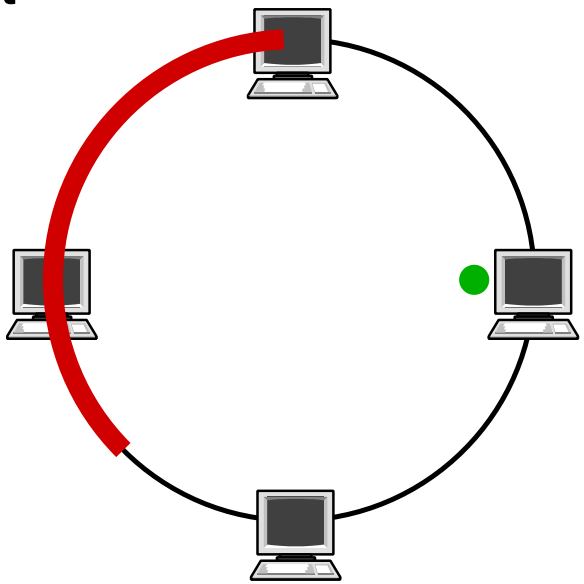


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

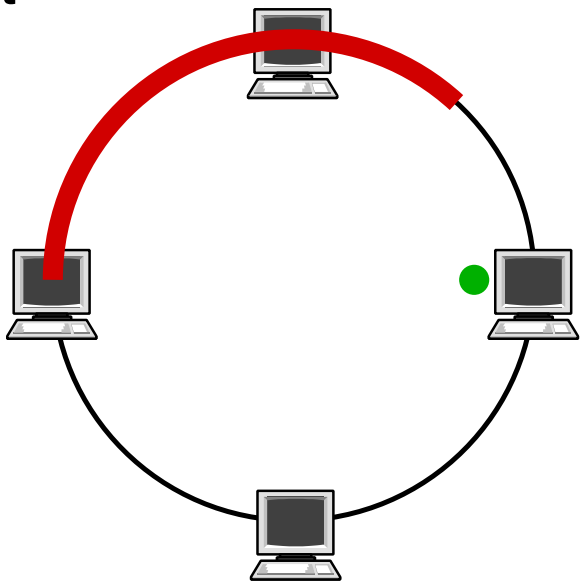


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

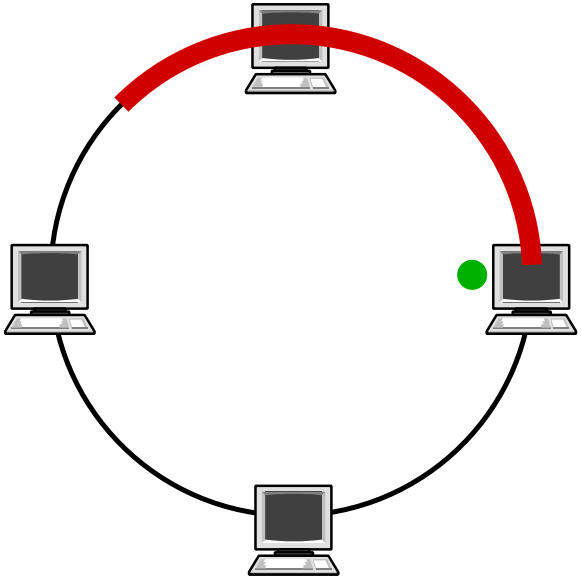


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

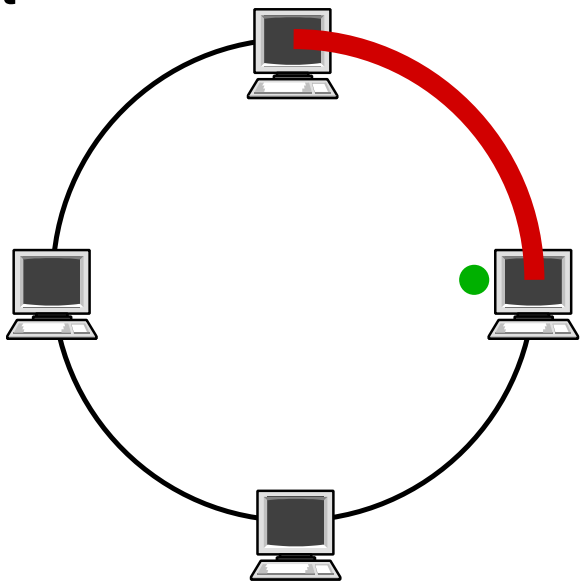


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

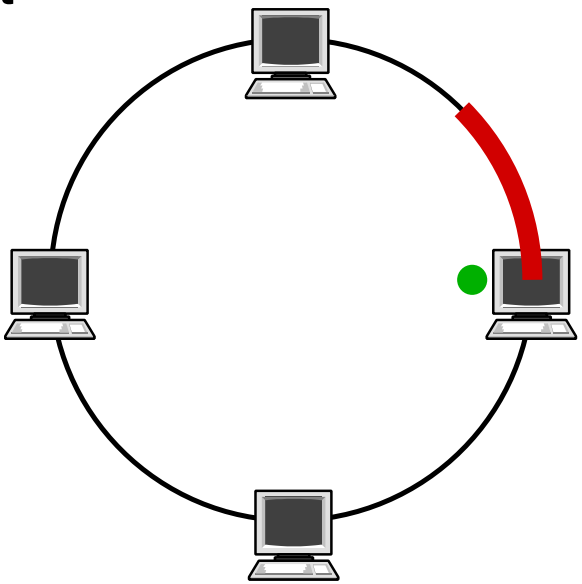


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

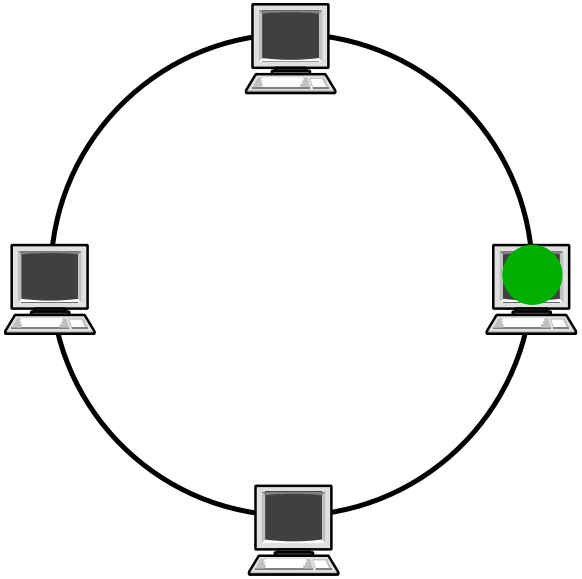


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

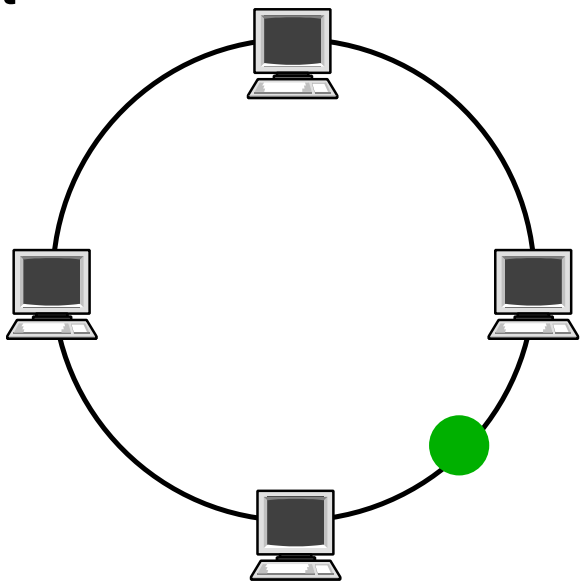


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie

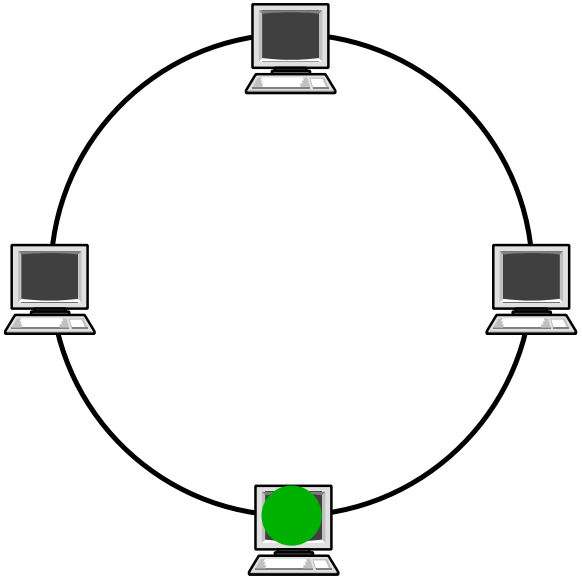


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

★★

Beispiel: Token-Ring

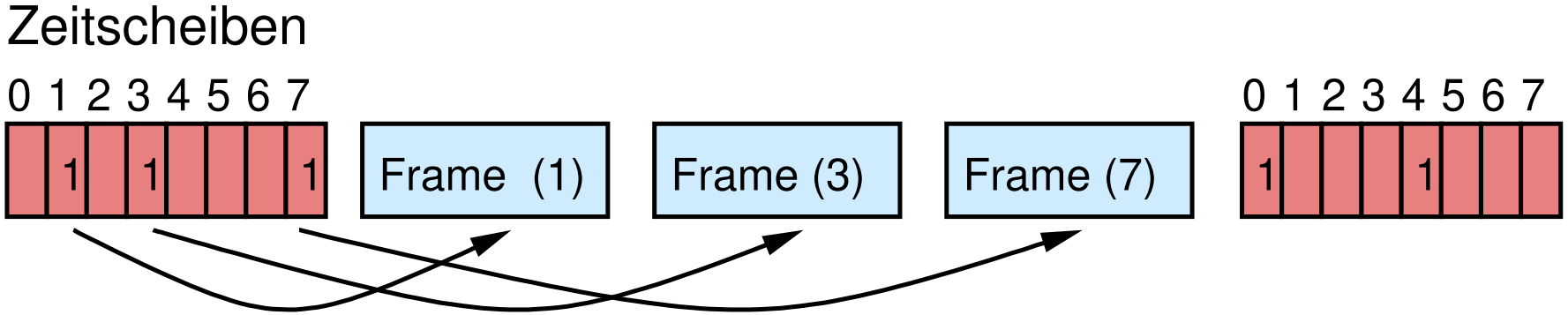
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezieller Steuerframe) umkreist ständig den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie



- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

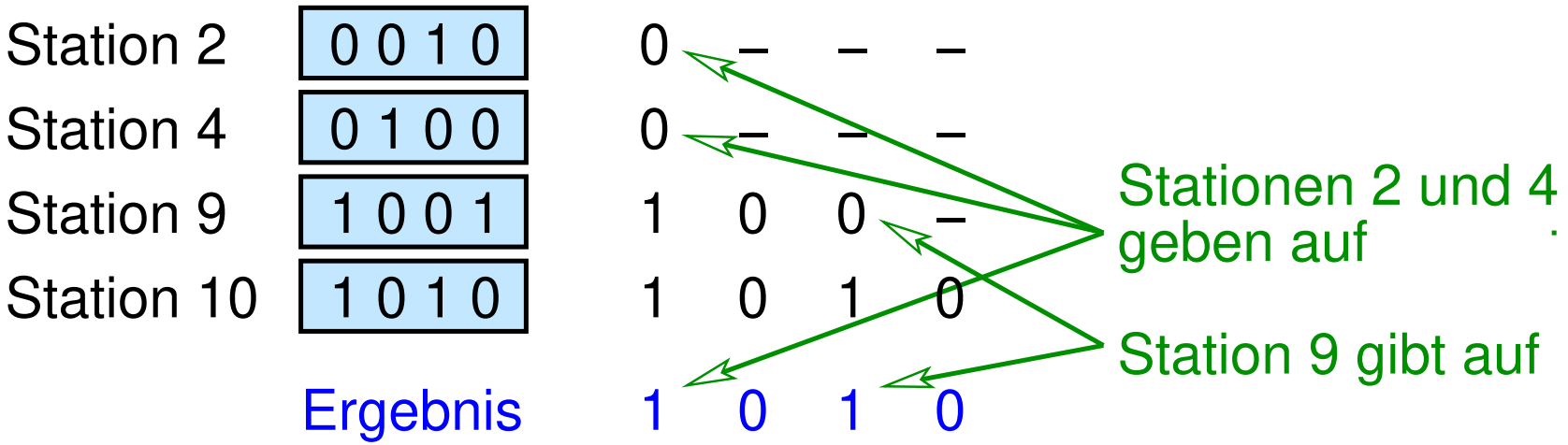
Bitmusterprotokoll

- Während der Konkurrenzphase werden alle Sendewünsche kollisionsfrei bekanntgegeben
- Jede Station erhält dazu eine eigene Zeitscheibe
  - Länge muß $\geq$ RTT/2 sein
  - Anzahl der Teilnehmer im Netz limitiert
- Im Anschluss kennen alle Stationen alle Sendewünsche
  - Abarbeitung in Reihenfolge der IDs



Binärer Countdown

- In Konkurrenzphase: Arbitrierung anhand der Sender-ID
- Gewinner kann anschließend kollisionsfrei senden
- Voraussetzungen:
  - Kanal bildet logisches ODER aller Signale
  - Bitzeit muss $\geq$ RTT/2 sein ($\Rightarrow$ geringe Übertragungsrate)
- Beispiel: CAN-Bus



★★

Diskussion

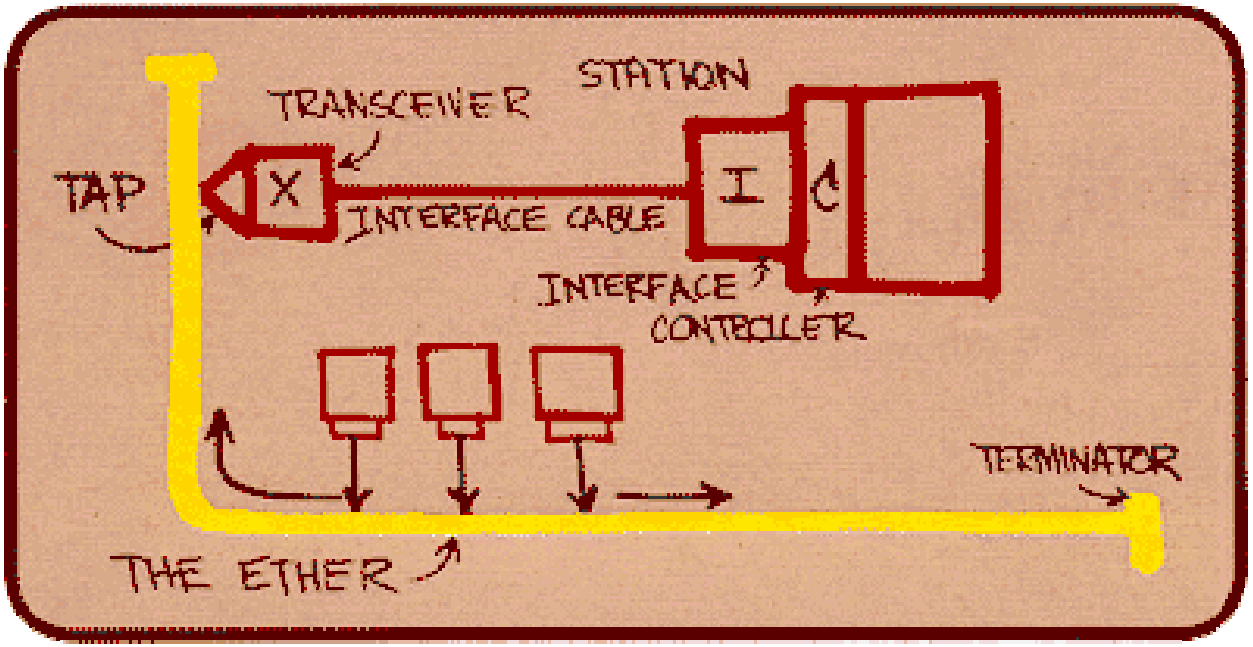
- Kollisionsfreie Verfahren gut geeignet für Echtzeit-Anwendungen
  - maximale Sendeverzögerung kann garantiert werden
- Beispiel Token-Ring: mit gegebener Token-Haltezeit THT kann jeder Knoten garantiert nach Ablauf der Zeit

  $$TRT \leq \text{Ringlatenz} + (\text{AnzahlKnoten} - 1) \cdot THT$$

  seinen Frame senden
- Mit CSMA-Verfahren sind keine Echtzeit-Garantien möglich
  - dafür kann ein Knoten bei unbelastetem Netz immer sofort senden

3.8.4 Ethernet

- Erfolgreichste LAN-Technologie der letzten Jahre
- Im Folgenden: Grundlagen, 10 Mb/s und 100 Mb/s Ethernet
- Ursprünglich Mehrfachzugriffsnetz: alle Rechner nutzen eine gemeinsame Verbindungsleitung

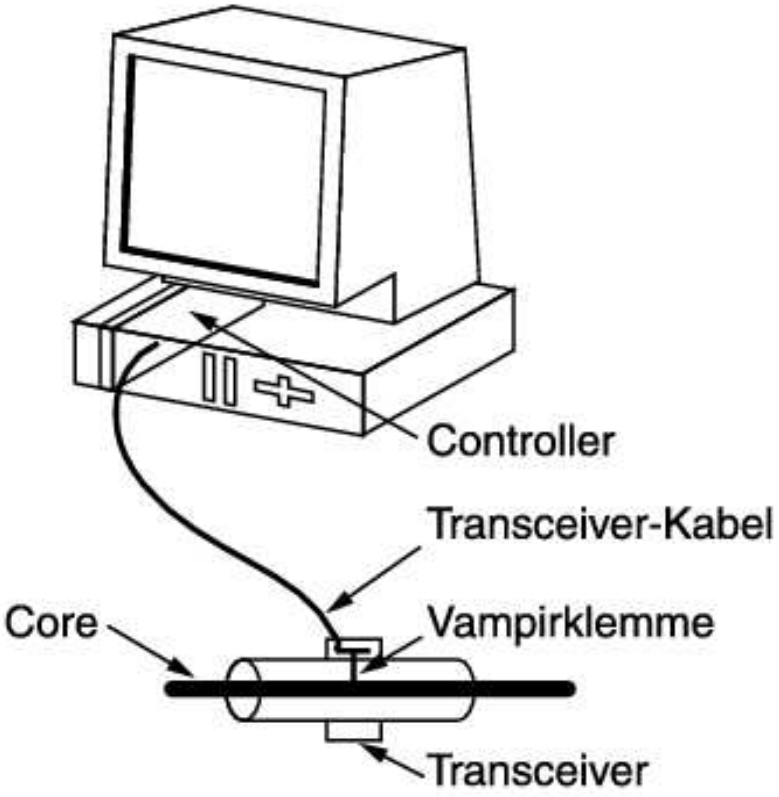


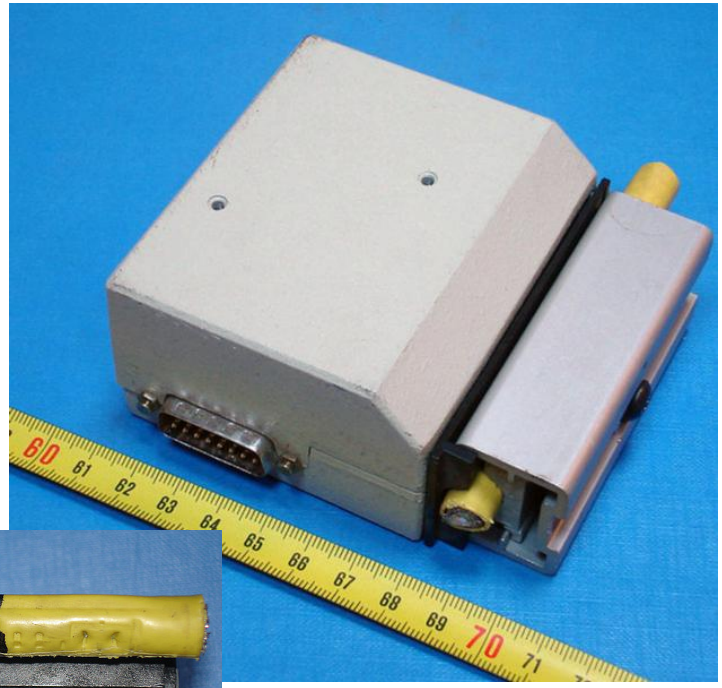
Bob Metcalfe
Xerox, 1976

- Heute: Punkt-zu-Punkt Verbindungen

Verkabelung

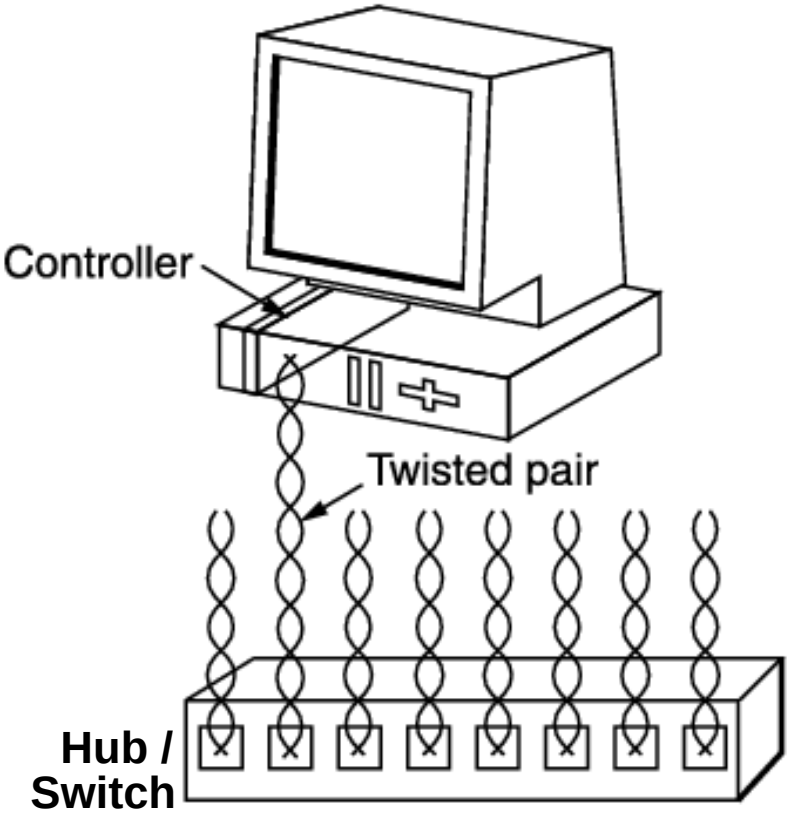
10Base5 (max. 500m)

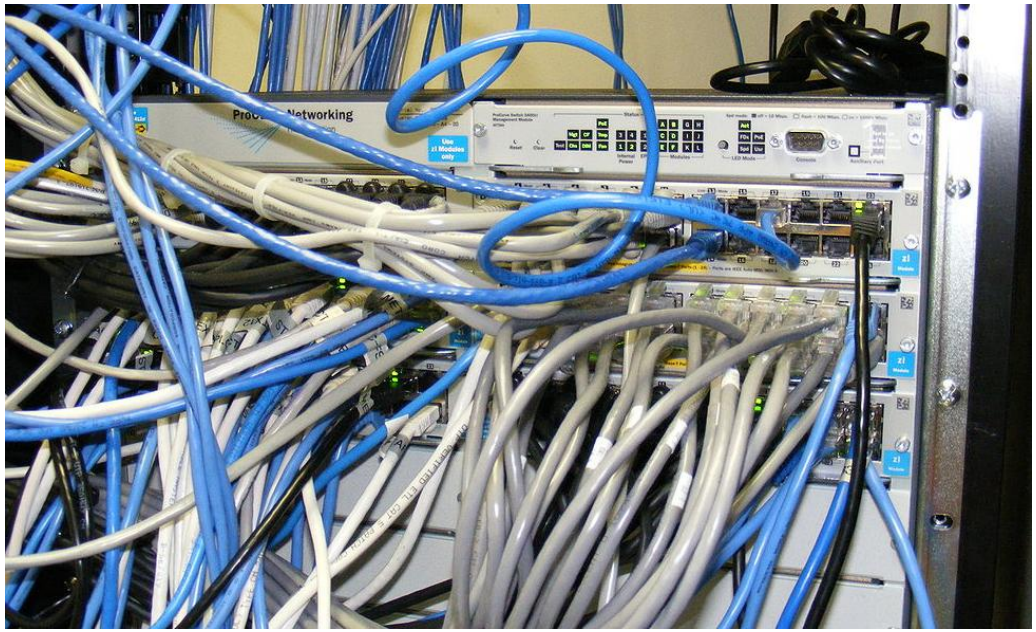




Verkabelung

10BaseT / 100BaseTx (100m)



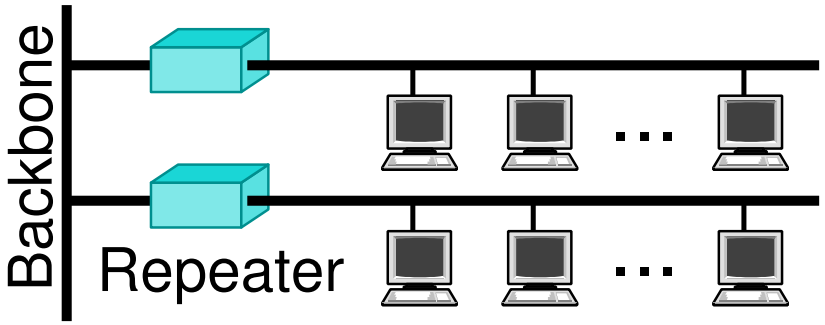


- **Hub**: „Verstärker“ und Verteiler (OSI-Schicht 1)
- **Switch**: Vermittlungsknoten (OSI-Schicht 2)

Physische Eigenschaften

10BASE-5 (10 Mb/s)

- Segmente aus Koaxial- kabel, je max. 500m
- Segmente über **Repeater** (Hub mit 2 Ports) verbindbar

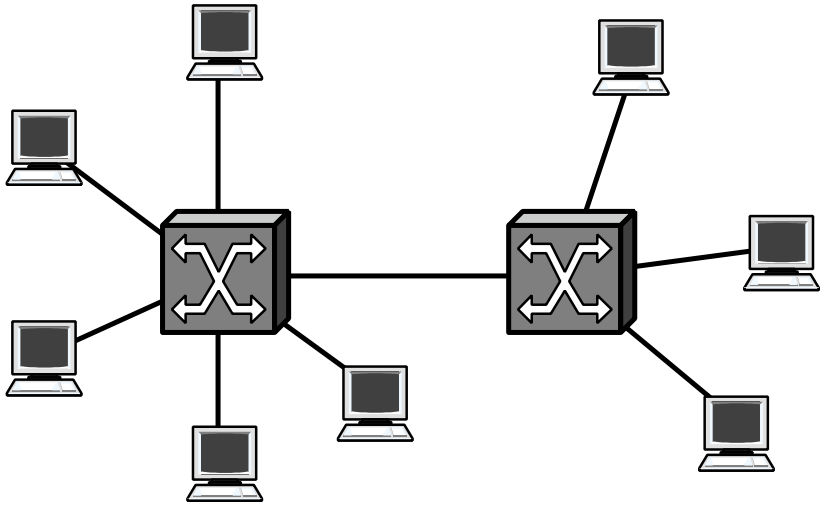


  - max. 4 Repeater zwischen zwei Knoten erlaubt

- Manchester-Codierung

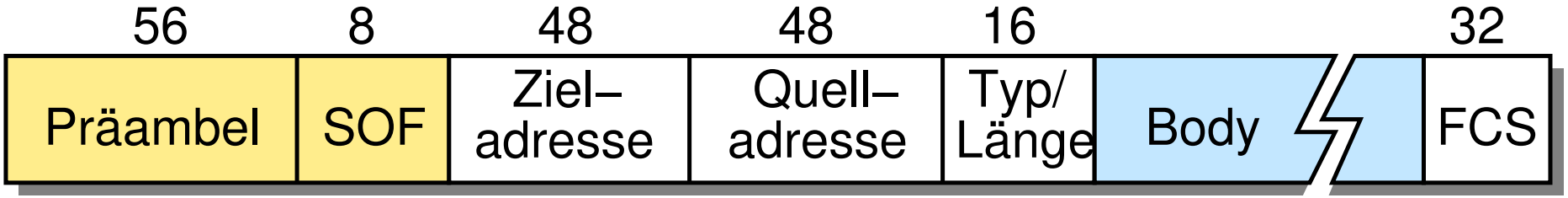
100BASE-TX (100 Mb/s)

- Twisted-Pair Kabel, je max. 100m
- Sternförmige Verkabelung mit Hubs / Switches



- 4B5B-Codierung

Frame-Format



- **Präambel/SOF**: Bitfolge 10101010 ... 10101011
  - zur Takt- und Frame-Synchronisation des Empfängers
  - letztes Byte (SOF, *Start of Frame*) markiert Frame-Anfang
- **Typ/Länge**:
  - Wert $<$ 1536 (0600_{16}): Framelänge
  - Wert $\geq$ 1536: *EtherType*, spezifiziert Protokoll im *Body*
- **FCS** (*Frame Check Sequence*): 32-Bit CRC Wert

Ethernet-Adressen (MAC-Adressen)

- Identifizieren die Netzwerkkarte
- 6 Byte (48 Bit) lang, weltweit eindeutig
- Schreibweise: byteweise hexadezimal, mit ‘:’ oder ‘-’ als Trennzeichen, z.B. 01:4A:3E:02:4C:FE
- jeder Hersteller erhält ein eindeutiges 24 Bit Präfix und vergibt eindeutige Suffixe
- Niedrigstwertiges Bit = 1: Multicast-Adresse
- Adresse ff:ff:ff:ff:ff:ff als Broadcast-Adresse
- Die Netzwerkkarte bestimmt, welche Frames sie empfängt

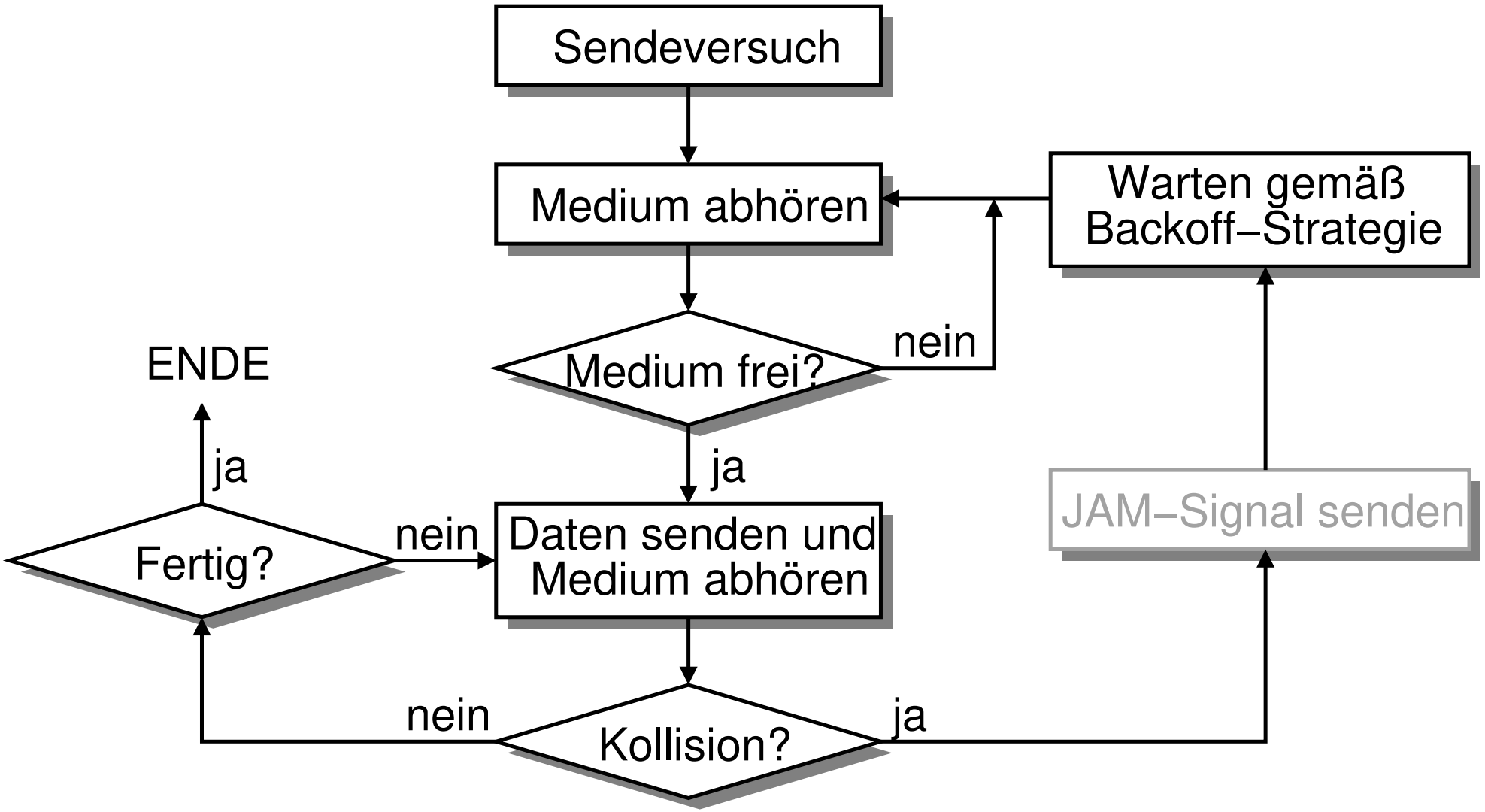
★★★

Begriffsdefinition

- **Zugangsprotokoll** zum gemeinsamen Übertragungsmedium beim Ethernet
  - *Carrier Sense Multiple Access*
    - jede Netzwerkkarte prüft zunächst, ob die Leitung frei ist, bevor sie einen Frame sendet
    - wenn die Leitung frei ist, sendet die Netzwerkkarte ihren Frame
  - *Collision Detection*
    - beim Senden erkennt der Sender Kollisionen mit Frames, die andere Netzwerkkarten evtl. gleichzeitig senden
    - bei Kollision: Abbruch des Sendens, nach einiger Zeit neuer Sendeversuch

★★★

CSMA/CD – Algorithmus



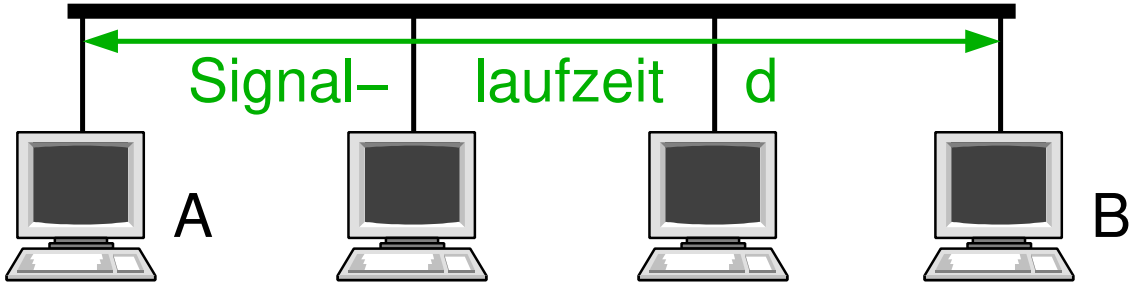
★★

Exponential Backoff-Strategie

- Aufgabe: Bestimmung der Wartezeit zwischen Kollision und erneutem Sendeversuch
- Ziel: erneute Kollision möglichst vermeiden
- Vorgehensweise:
  - c = Anzahl der bisherigen Kollisionen für aktuellen Frame
  - warte $s \cdot 51{,}2\ \mu$s (bei 10 Mb/s), wobei s wie folgt bestimmt wird:
    - falls $c \leq 10$: wähle $s \in \{ 0, 1, \ldots, 2^c\text{-}1 \}$ zufällig
    - falls $c \in [11, 16]$: wähle $s \in \{ 0, 1, \ldots, 1023 \}$ zufällig
    - falls $c = 17$: Abbruch mit Fehler
  - damit: bei geringer Netzlast nur kurze Wartezeit, geringe Wahrscheinlichkeit einer erneuten Kollision

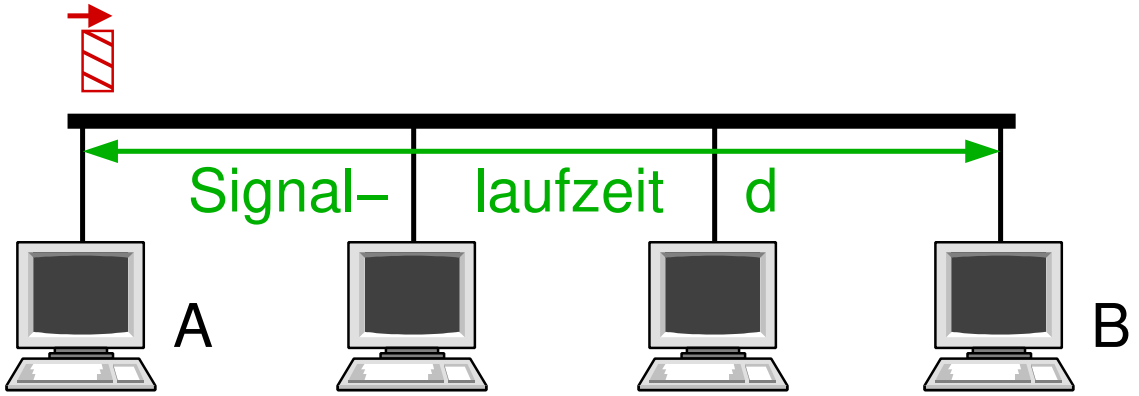
★★

Kollisionserkennung: Worst-Case Szenario



★★

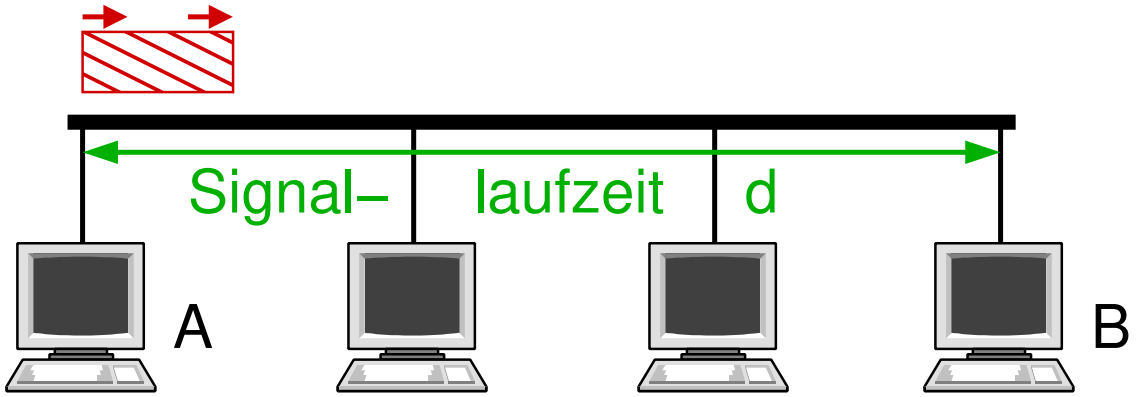
Kollisionserkennung: Worst-Case Szenario



➥ $t = 0$: A beginnt, einen Frame zu senden

★★

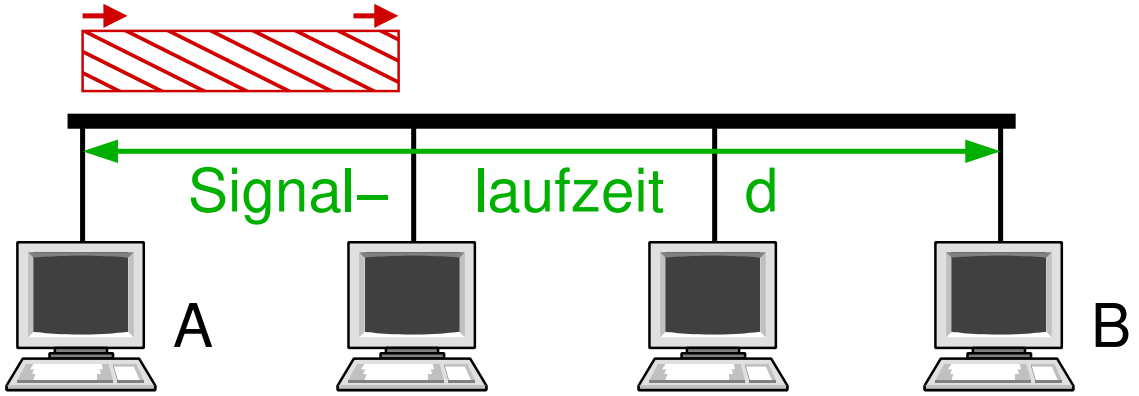
Kollisionserkennung: Worst-Case Szenario



➥ $t = 0$: A beginnt, einen Frame zu senden

★★

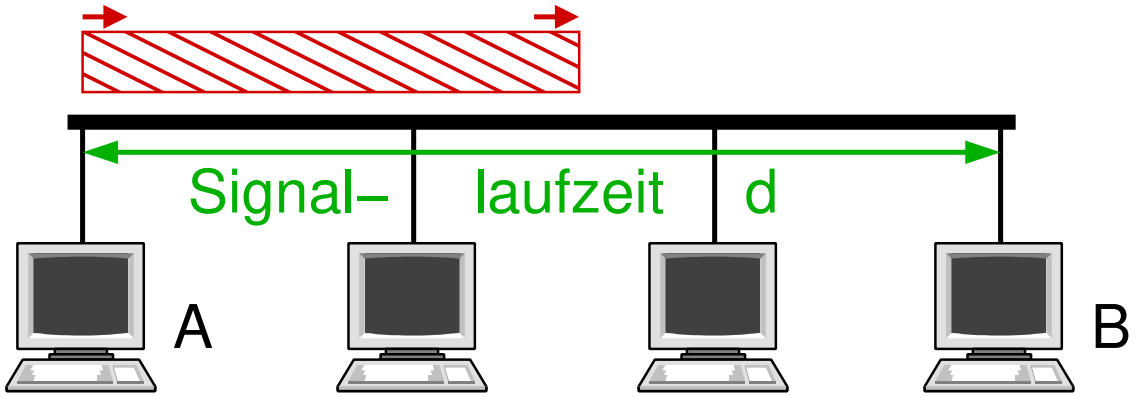
Kollisionserkennung: Worst-Case Szenario



➥ $t = 0$: A beginnt, einen Frame zu senden

★★

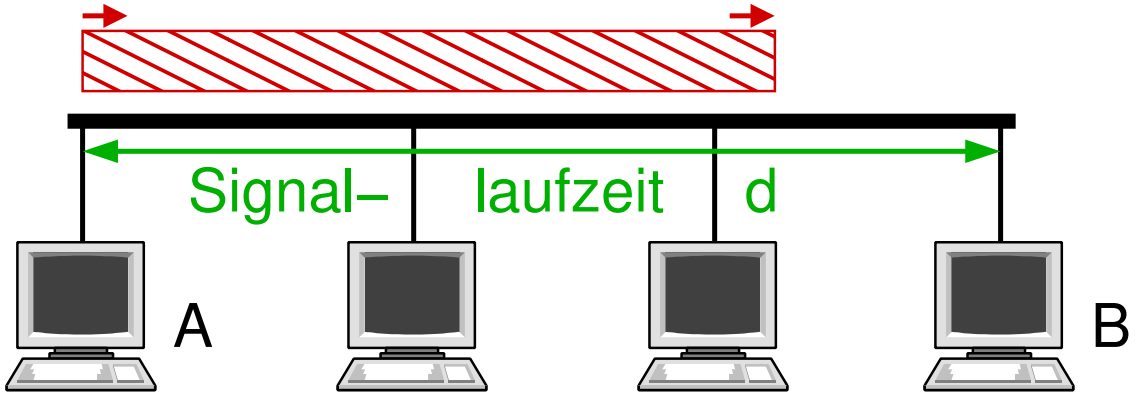
Kollisionserkennung: Worst-Case Szenario



➥ $t = 0$: A beginnt, einen Frame zu senden

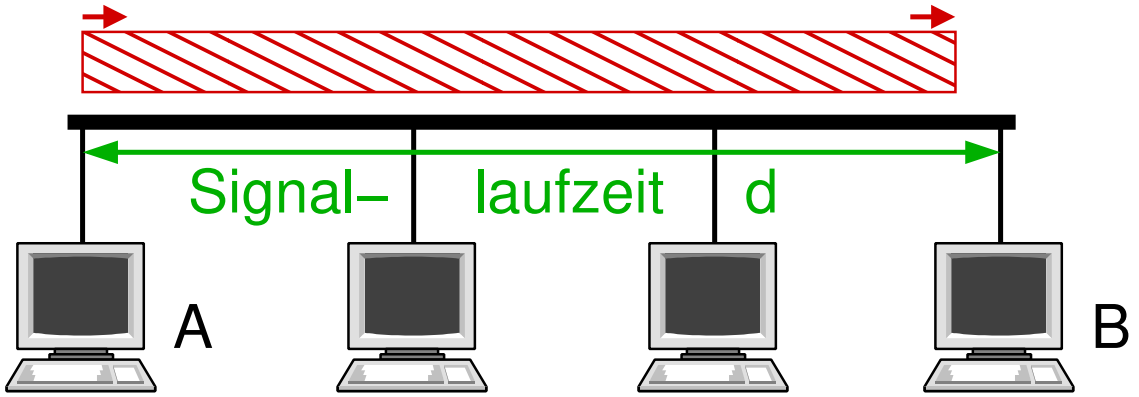
★★

Kollisionserkennung: Worst-Case Szenario



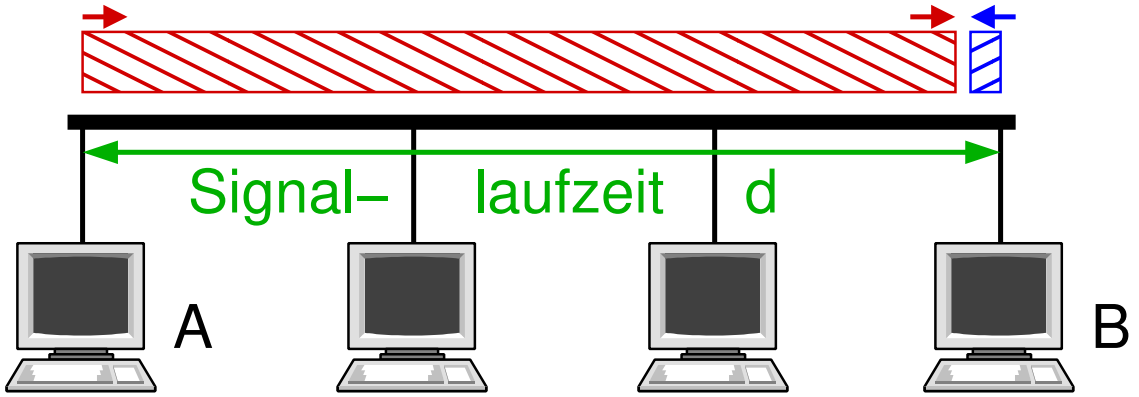
➥ $t = 0$: A beginnt, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: Das Medium ist bei B noch frei

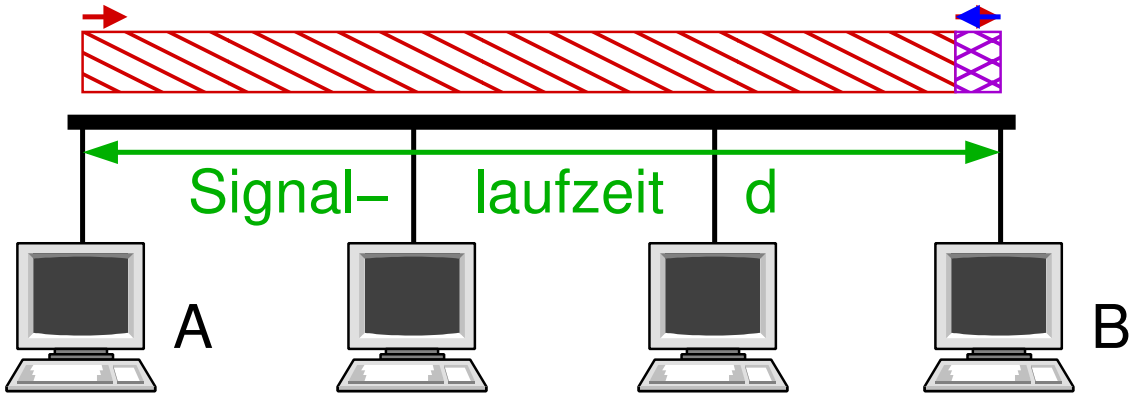
Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden

★★

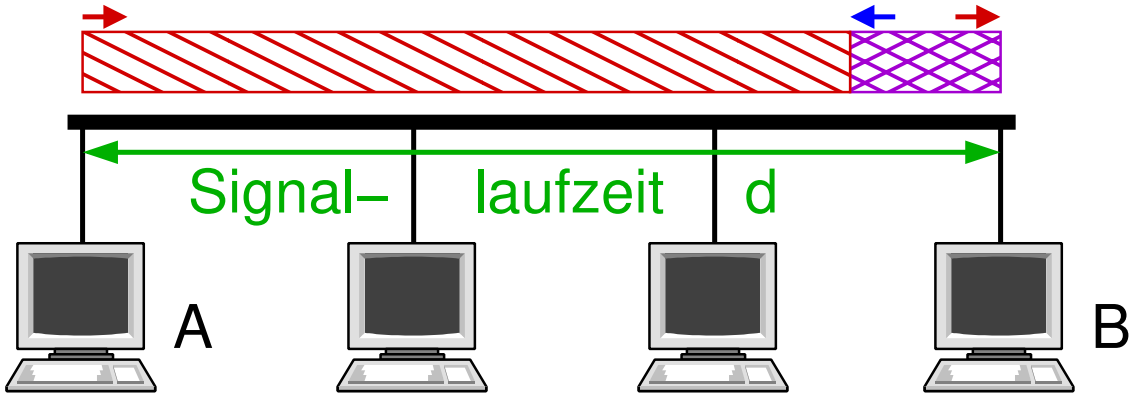
Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

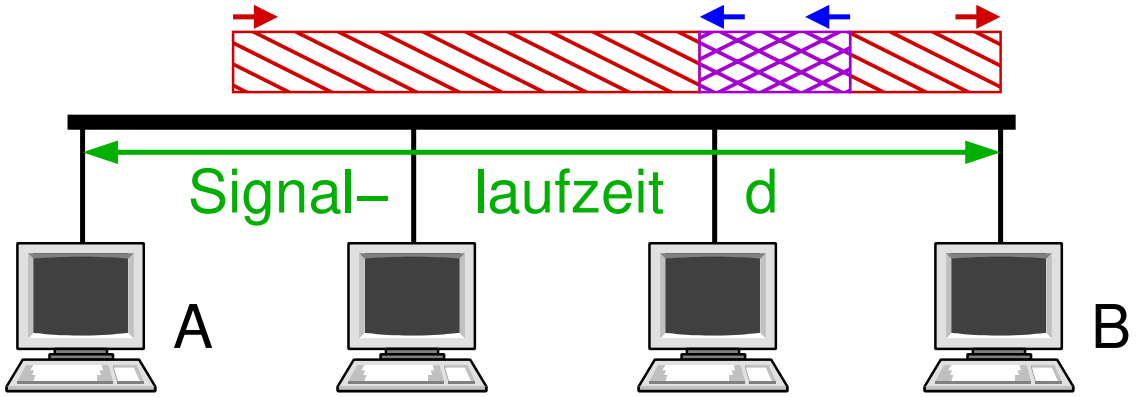
★★

Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

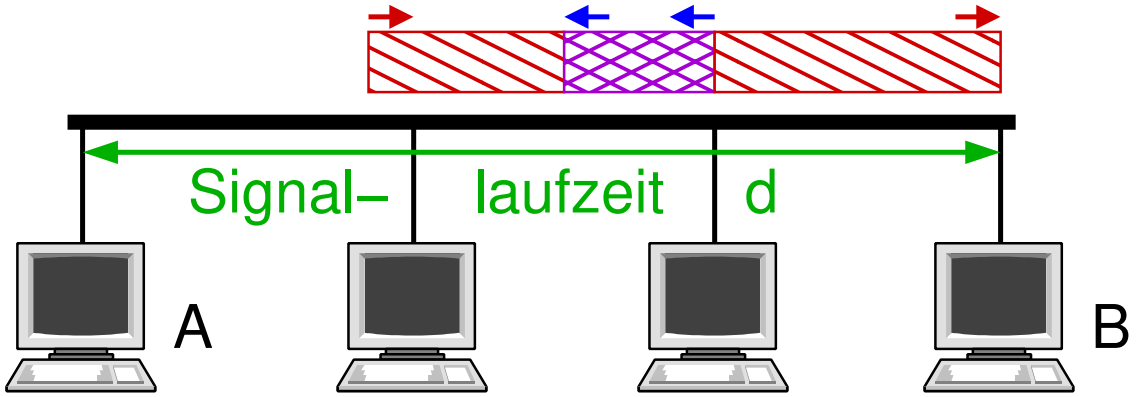
Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

★★

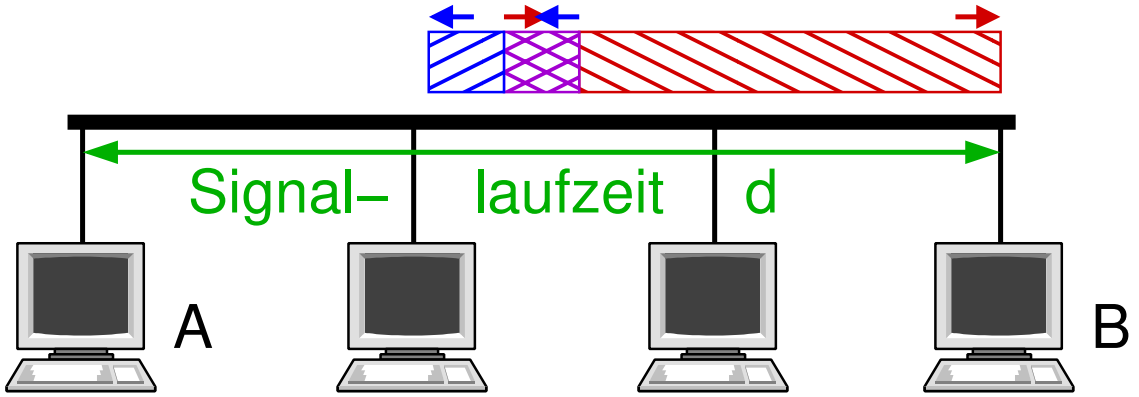
Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

★★

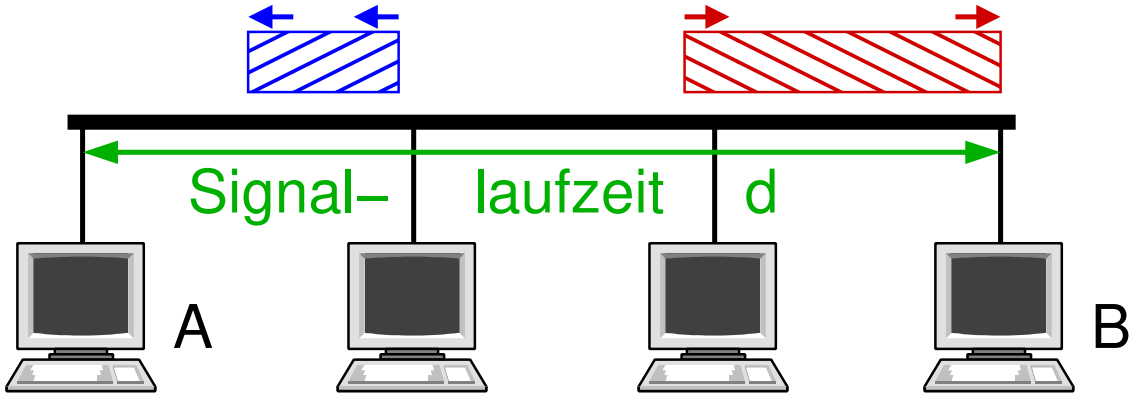
Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

★★

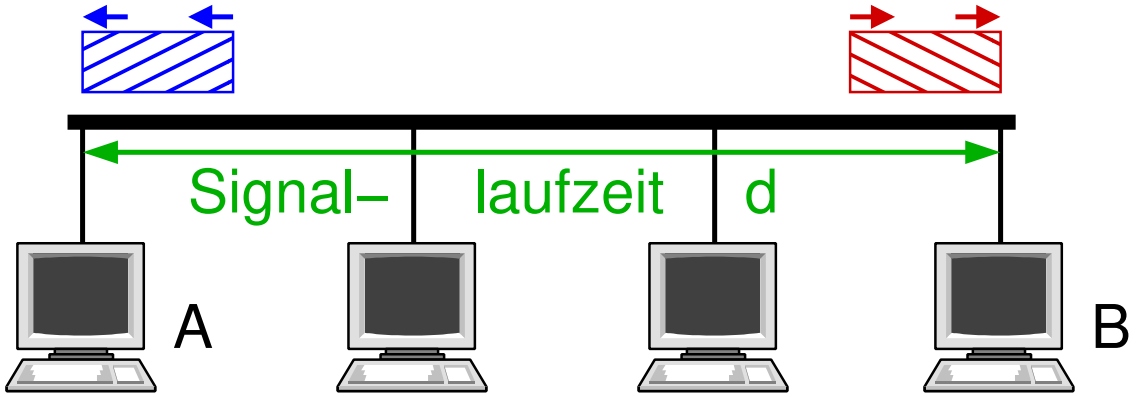
Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

★★

Kollisionserkennung: Worst-Case Szenario



- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!
- $t = 2 \cdot d$: B's (Kollisions-)Frame kommt bei A an
  - wenn A zu diesem Zeitpunkt nicht mehr sendet, erkennt A keine Kollision

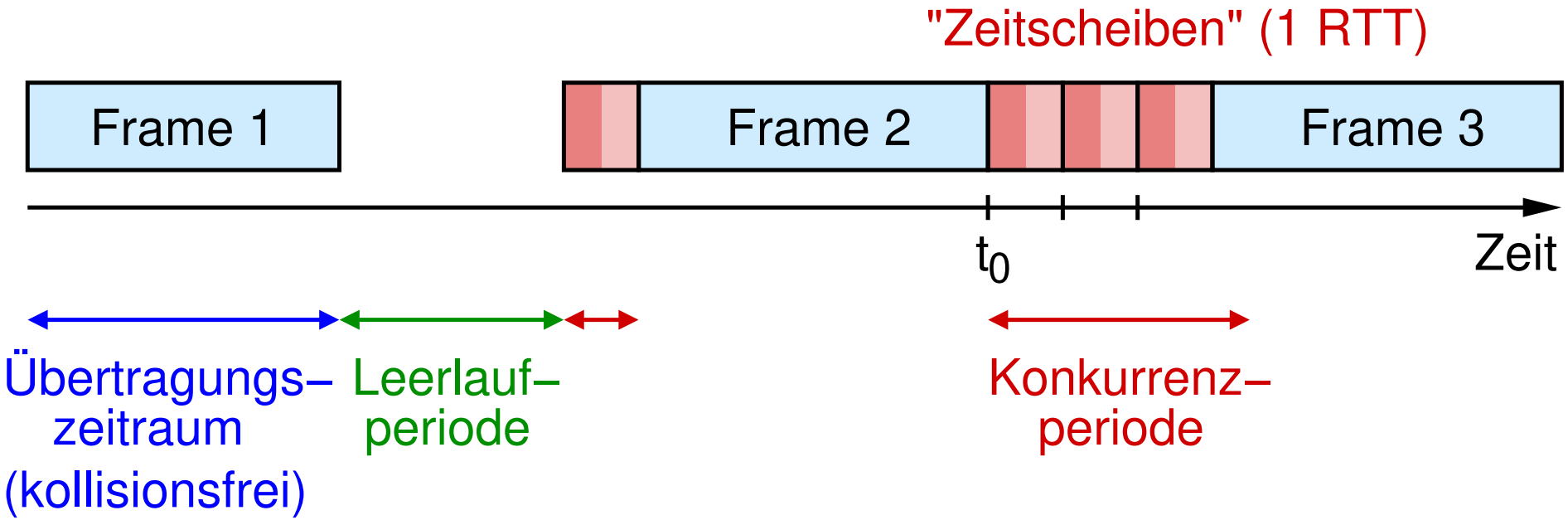
★★

Sichere Kollisionserkennung

- Um Kollisionen immer erkennen zu können, definiert Ethernet:
  - maximale RTT: 512 Bit-Zeiten
    - 51,2 μs bei 10 Mb/s, 5,12 μs bei 100 Mb/s
    - legt maximale Ausdehnung des Netzes fest, z.B. 200m bei 100BASE-TX mit Hubs
  - minimale Framelänge: 512 Bit (64 Byte), zzgl. Präambel
    - kleinere Frames werden vor dem Senden aufgefüllt
- Das stellt im Worst-Case Szenario sicher, daß Station A immer noch sendet, wenn B's Frame bei ihr ankommt
  - damit erkennt auch Station A die Kollision und kann ihren Frame wiederholen

Vorteil der Kollisionserkennung

- Kollisionen können nur innerhalb einer RTT nach Sendebeginn auftreten
  - bei Erkennung einer Kollision: Sendeabbruch
  - Rest der Frame-Übertragungszeit ist dann kollisionsfrei



3.9 Zusammenfassung

- Hardware: Knoten, Leitungen (Kupfer, Glasfaser, Funk)
- Codierung und Modulation
  - Umsetzen des Bitstroms in ein elektrisches Signal
  - wichtig: Taktwiederherstellung
- Framing: Erkennung des Anfangs / Endes eines Datenblocks
- Fehlererkennung (und -korrektur)
- Medienzugriffssteuerung (MAC)
  - Token-Ring: garantierte maximale Sendeverzögerung
  - Ethernet (CSMA-CD)

Nächste Lektion:

- Paketvermittlung, LAN-Switching