Rechnernetze I

SoSe 2020

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

Stand: 11. Mai 2020

Rechnernetze I

SoSe 2020

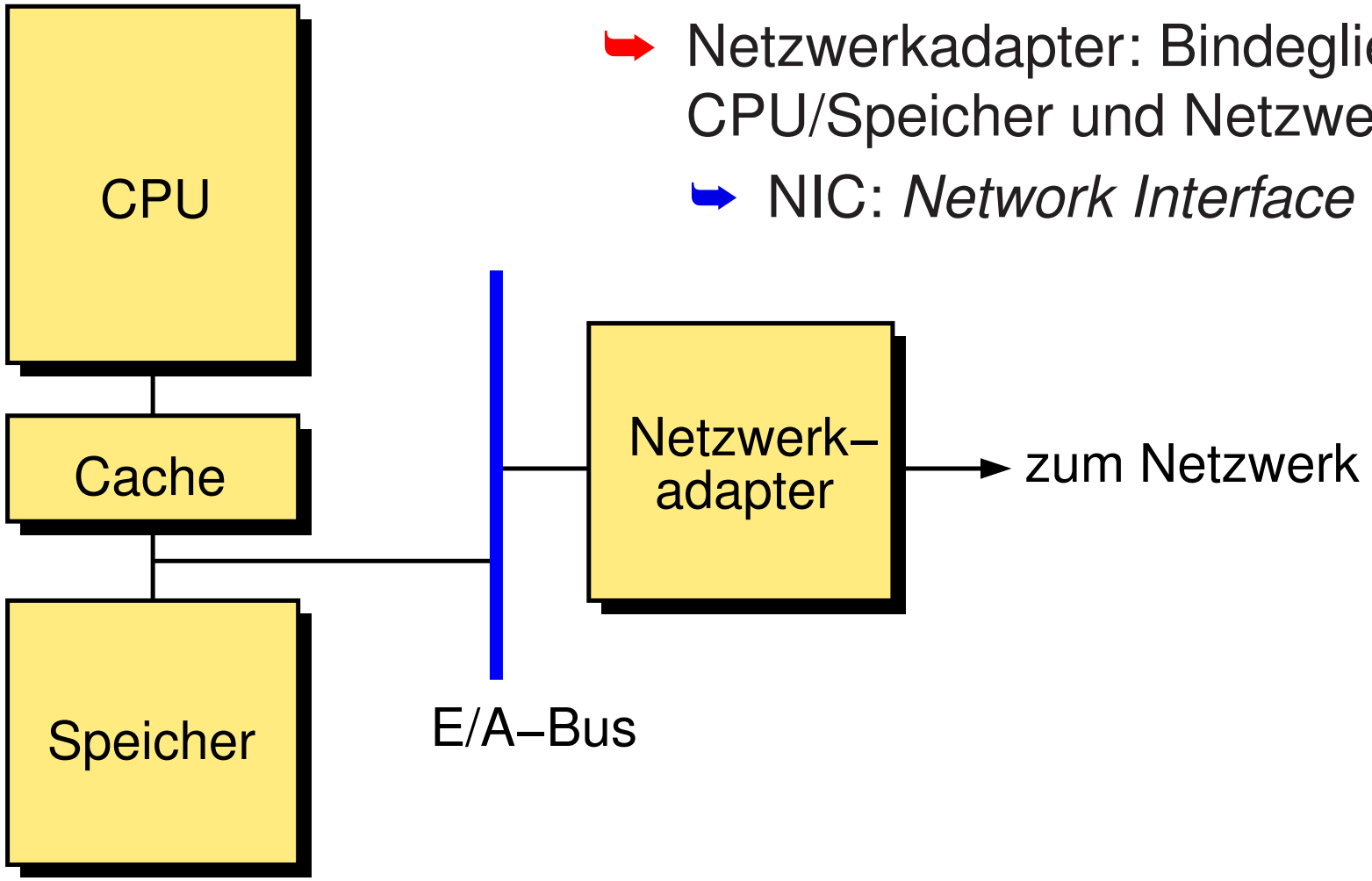
3 Direktverbindungsnetze

Inhalt

- Hardware-Bausteine: Knoten und Verbindungsleitungen
- Modulation
- Codierung
- Framing
- Fehlererkennung und Fehlerkorrektur
- Medienzugriffssteuerung (MAC)
  - Ethernet (CSMA-CD)
  - Token-Ring

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

Aufbau eines Knotens



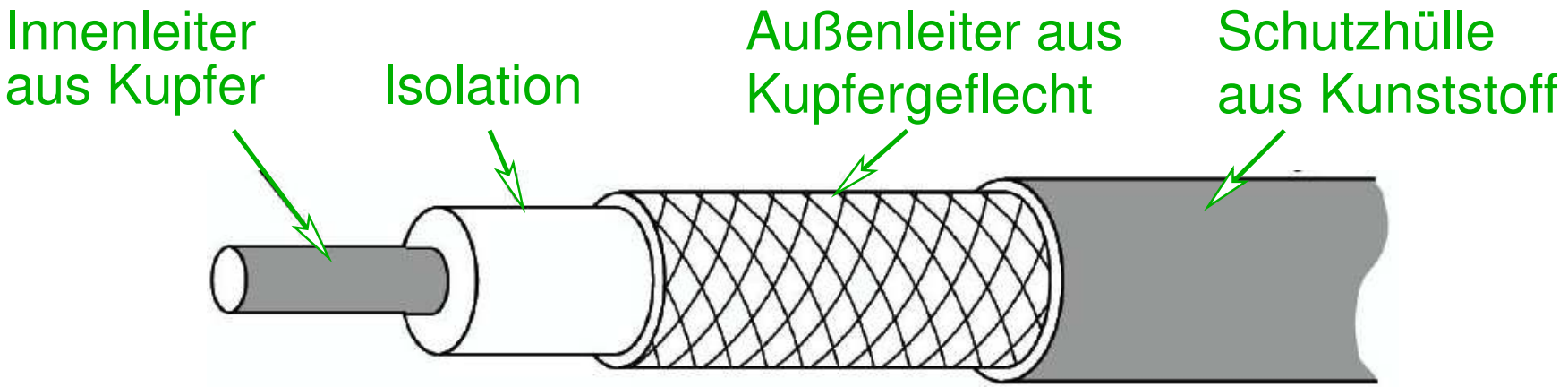
- Netzwerkadapter: Bindeglied zwichen CPU/Speicher und Netzwerk-Schnittstelle
  - NIC: *Network Interface Controller*

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

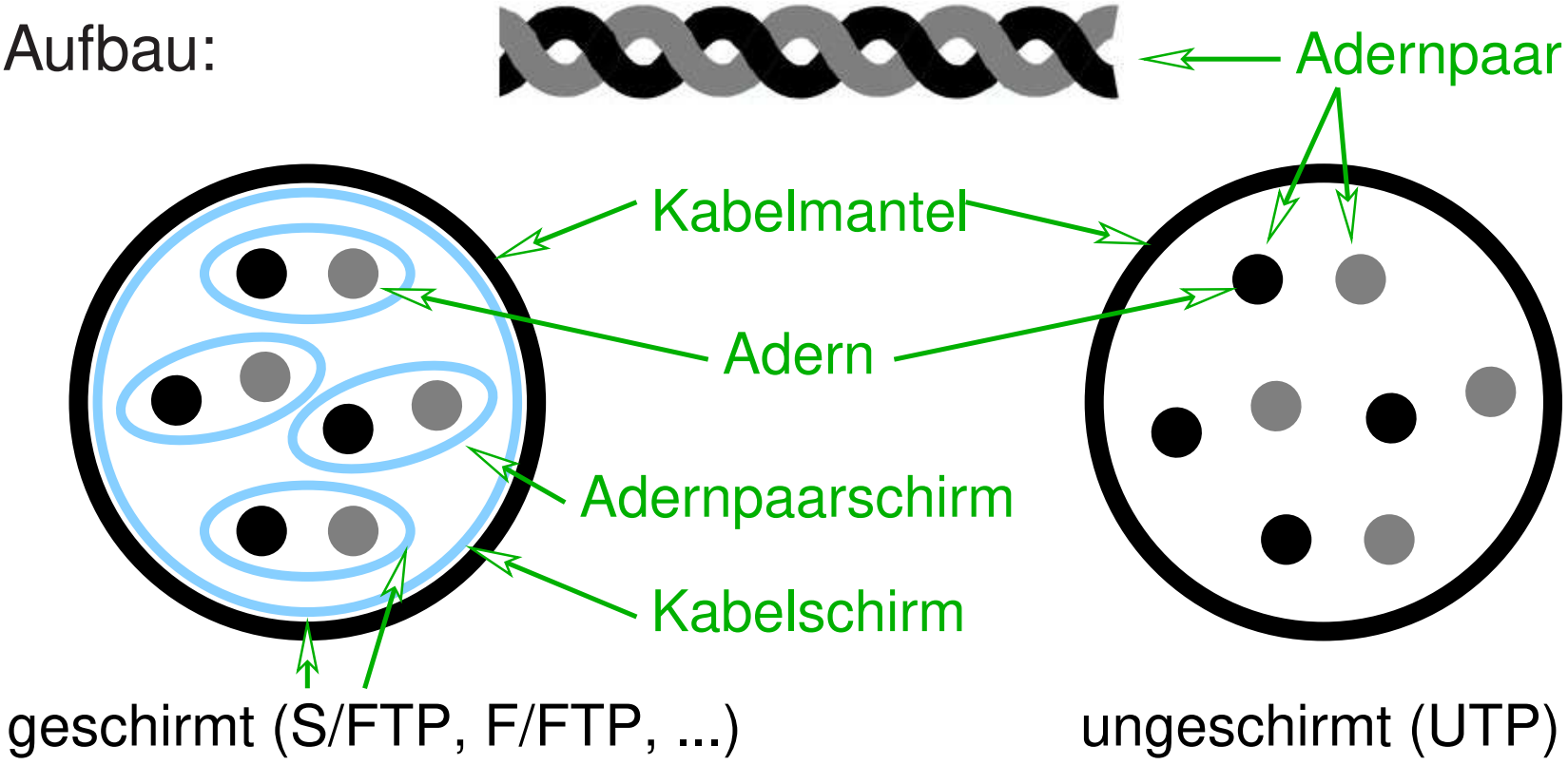
- Aufbau:



- 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

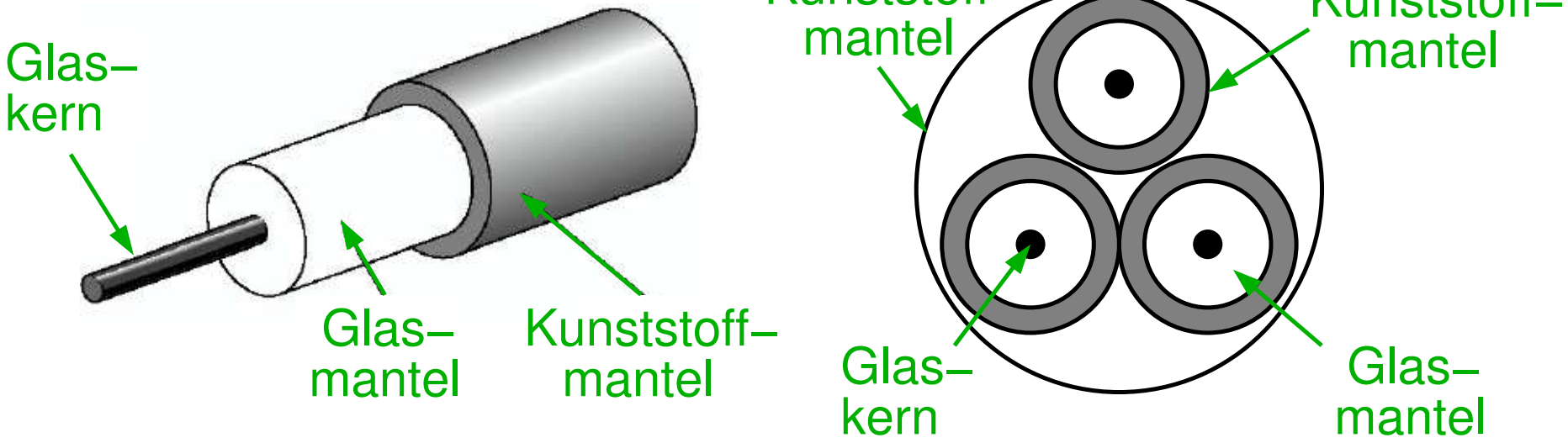
Kupferkabel: Twisted-Pair (verdrilltes) Kabel

- Aufbau:

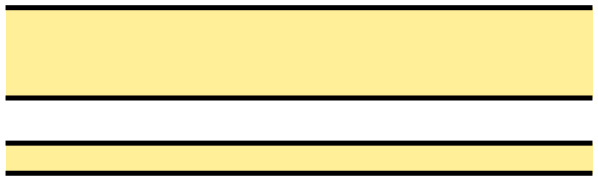


- Geringe Kosten, relativ gute Bandbreite
- Beispiel: Ethernet 100BASE-TX

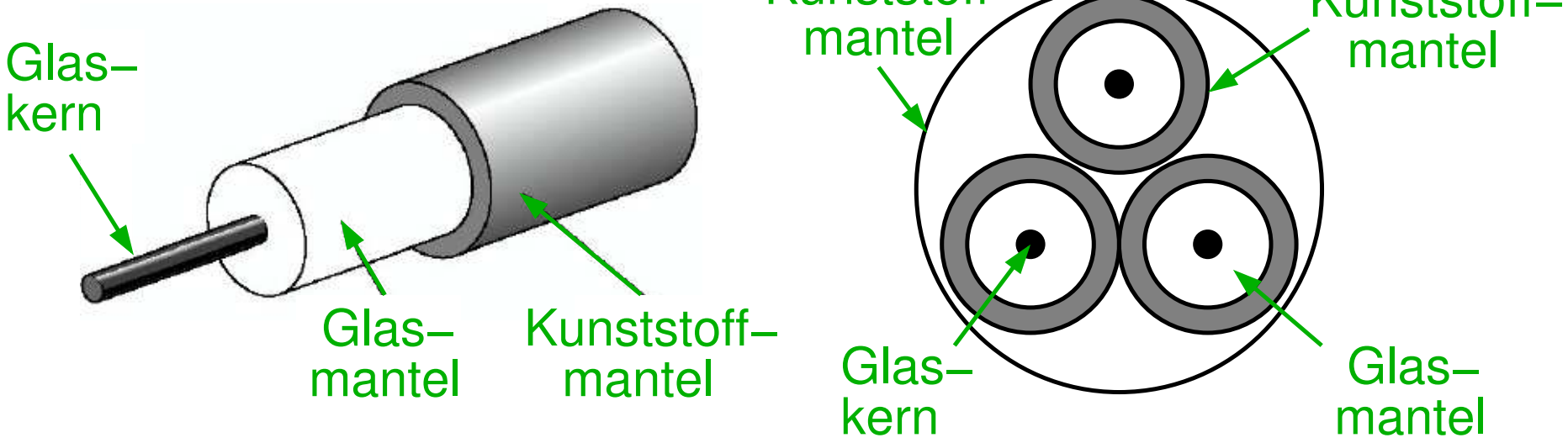
Glasfaserkabel



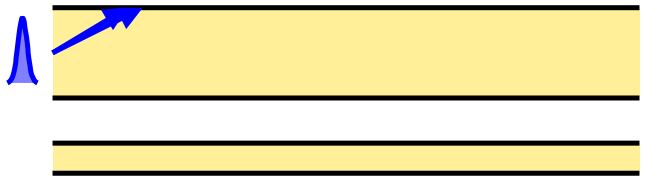
- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



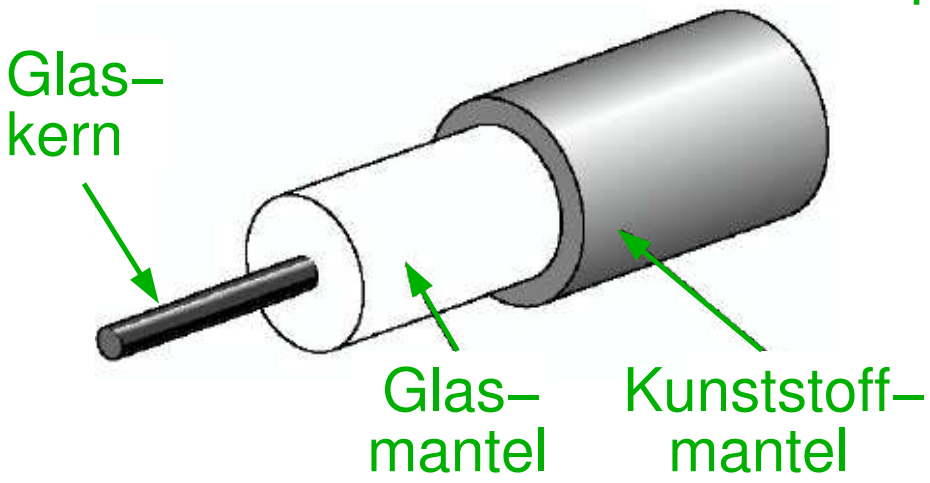
Glasfaserkabel

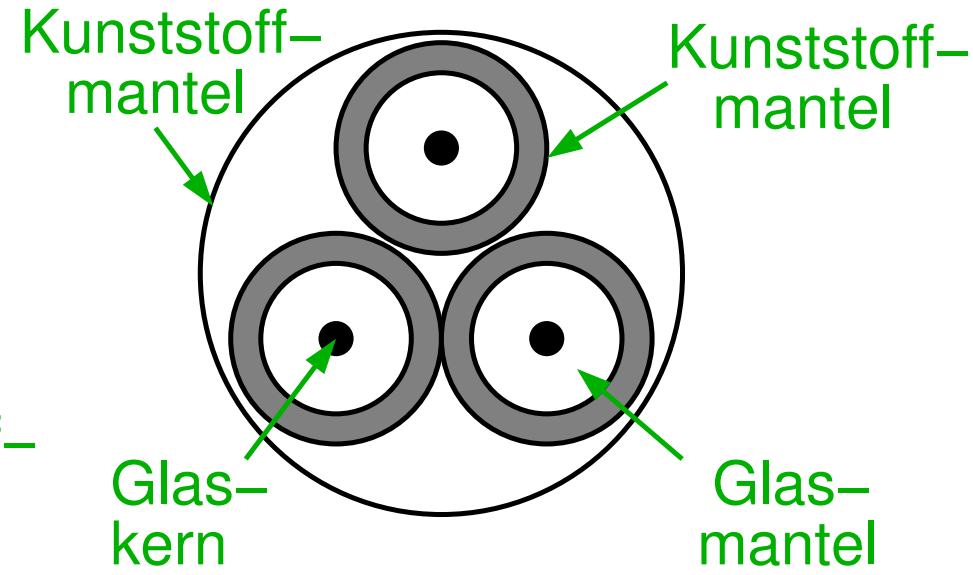


- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)

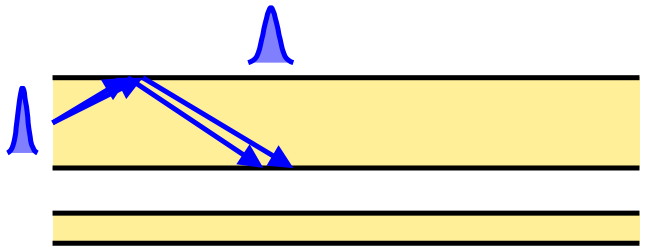


Glasfaserkabel

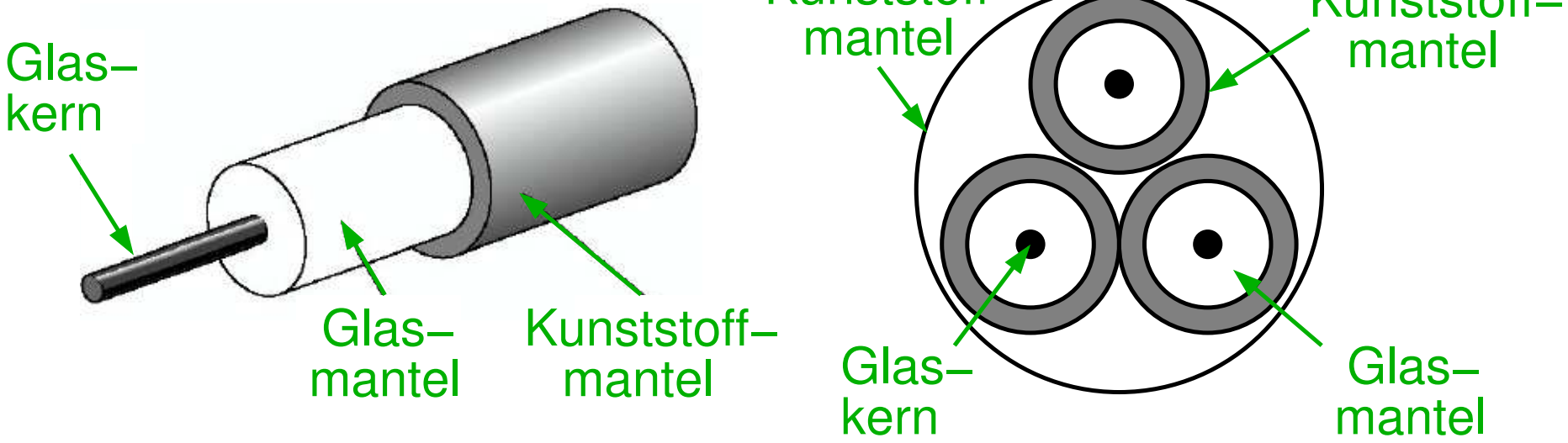




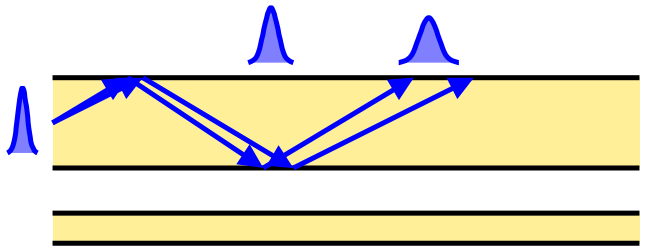
- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)



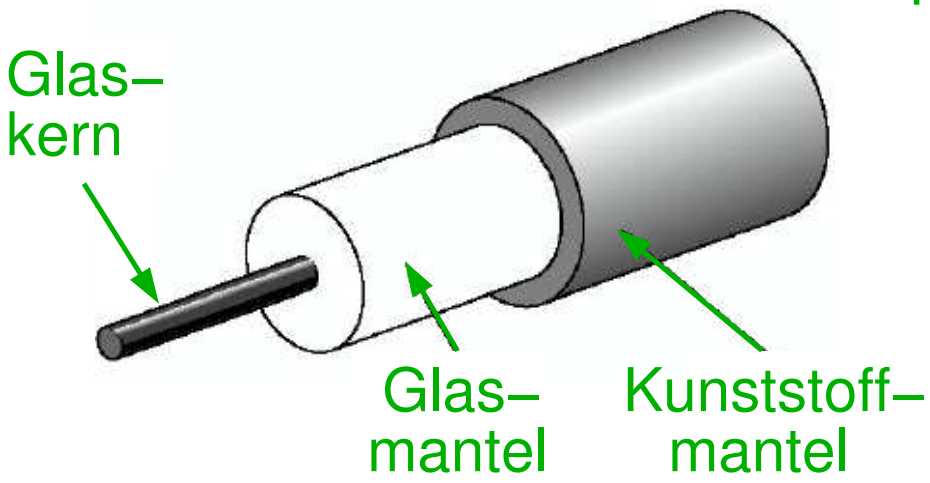
Glasfaserkabel

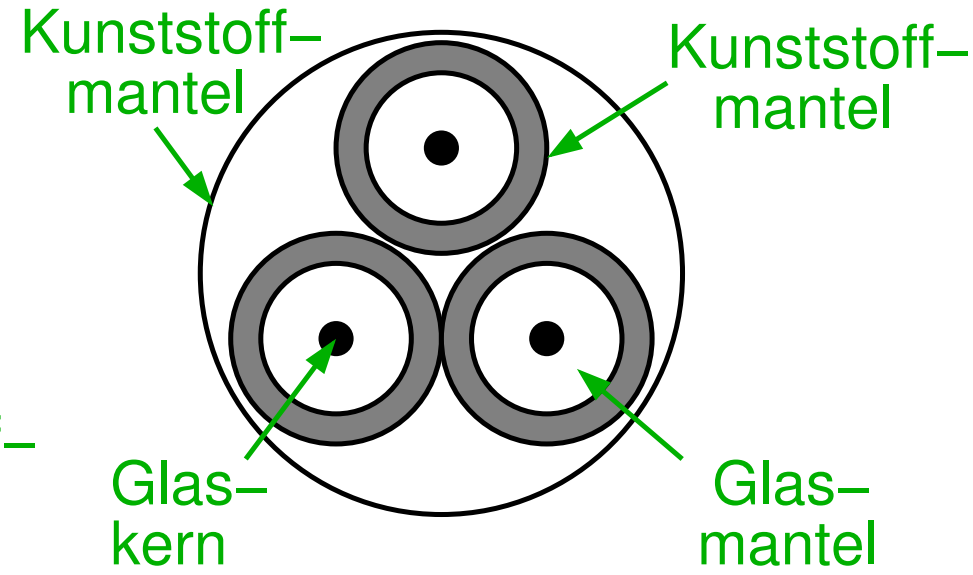


- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)

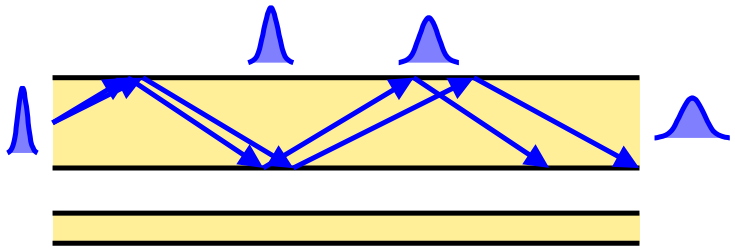


Glasfaserkabel

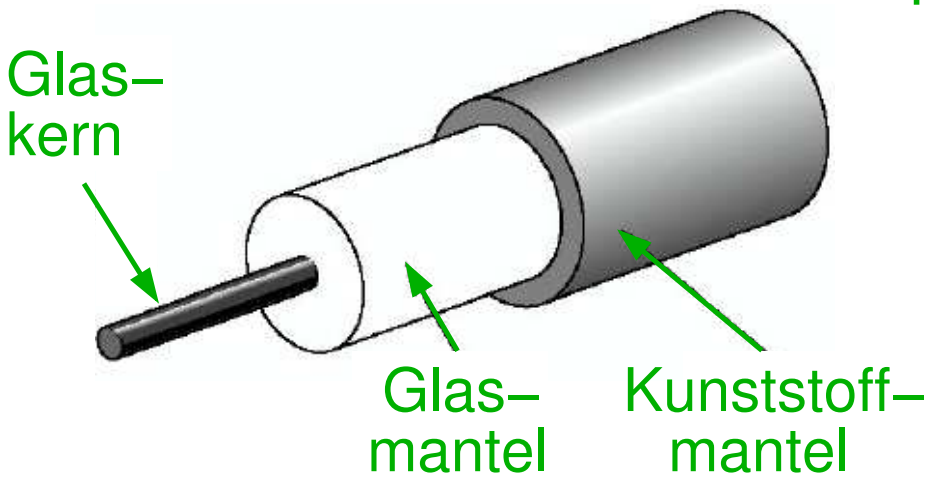


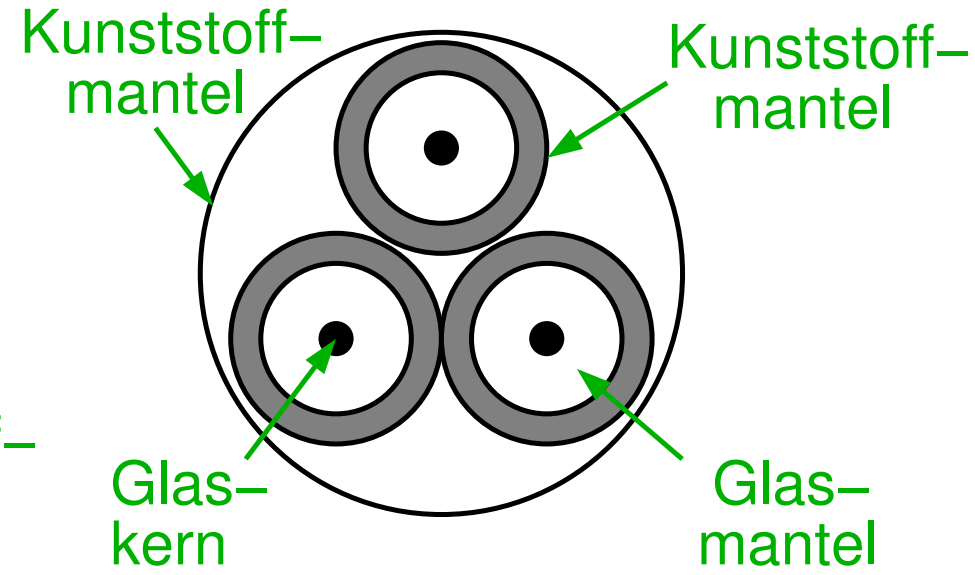


- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)

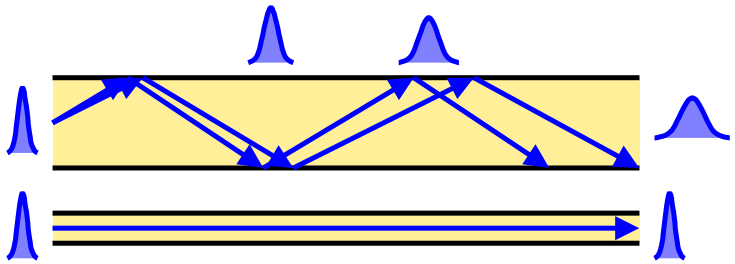


Glasfaserkabel

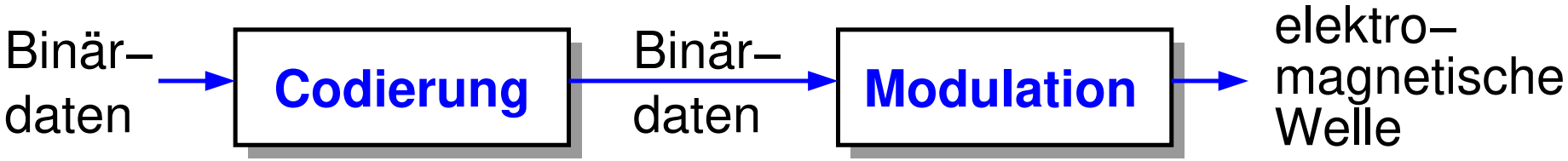




- Führung von Lichtwellen durch Totalreflexion
- Bandbreite im Bereich Gb/s, Länge im Bereich km
- Varianten:
  - Multimode-Faser
  - Monomode-Faser
    - hohe Bandbreite, teuer (Laserdioden)

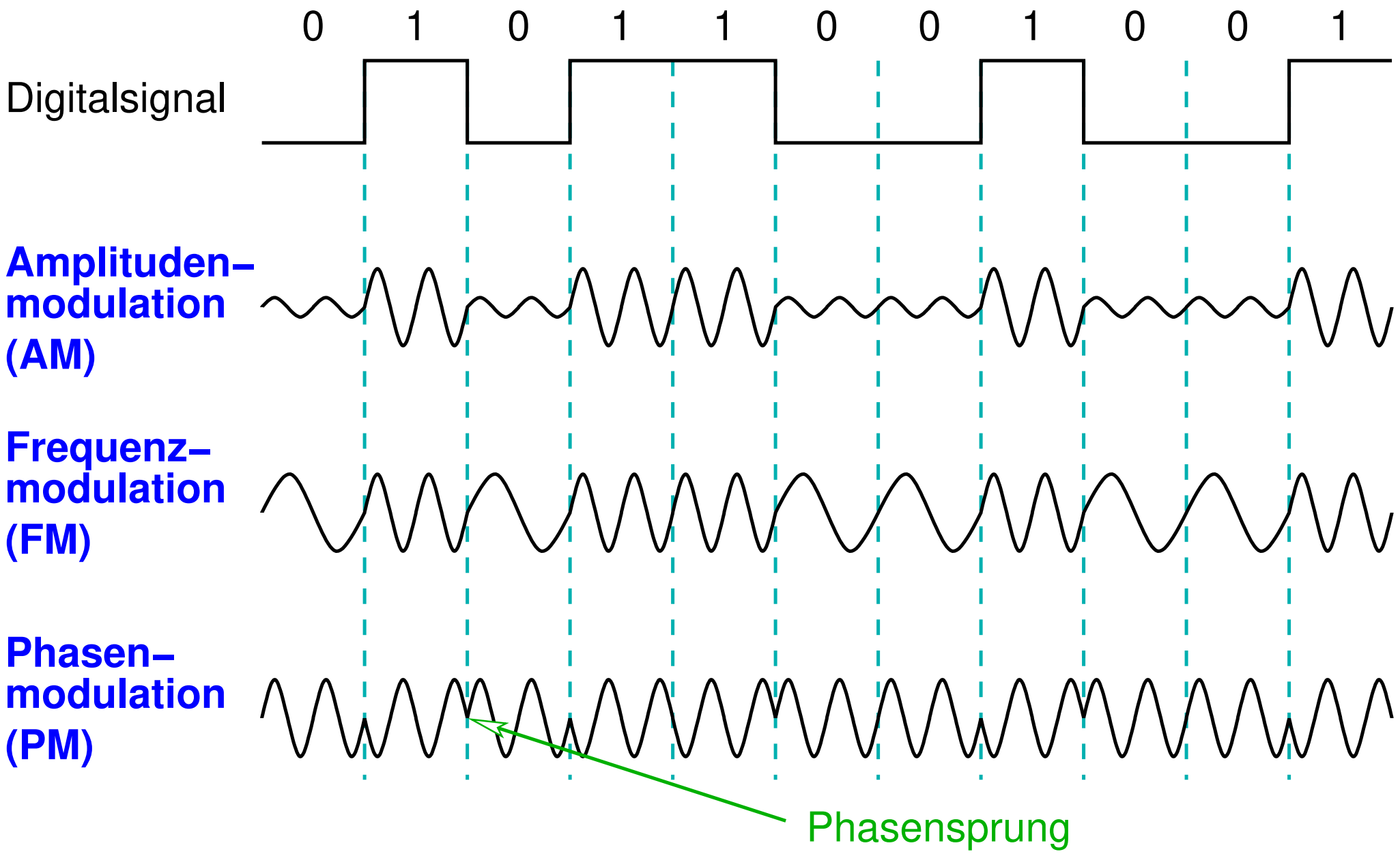


- Zur Übertragung müssen Binärdaten (digitale Signale) in analoge elektrische Signale (elektromagnetische Wellen) umgesetzt werden
- Umsetzung in zwei Schritten:

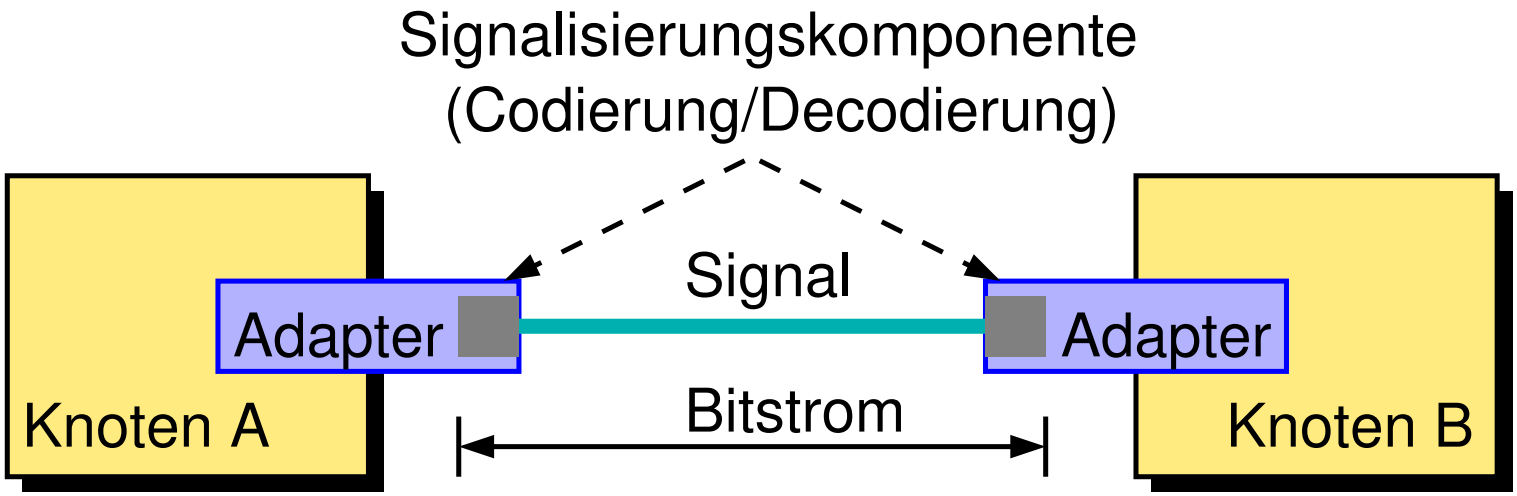


- **Modulation:**
  - Variation von Frequenz, Amplitude und/oder Phase einer Welle
  - zur Überlagerung der (Träger-)Welle mit dem Nutzsignal
    - z.B. bei Funk, Modem, Breitbandkabel, ...
  - (entfällt bei Basisband-Übertragung)

3.2 Modulation ...

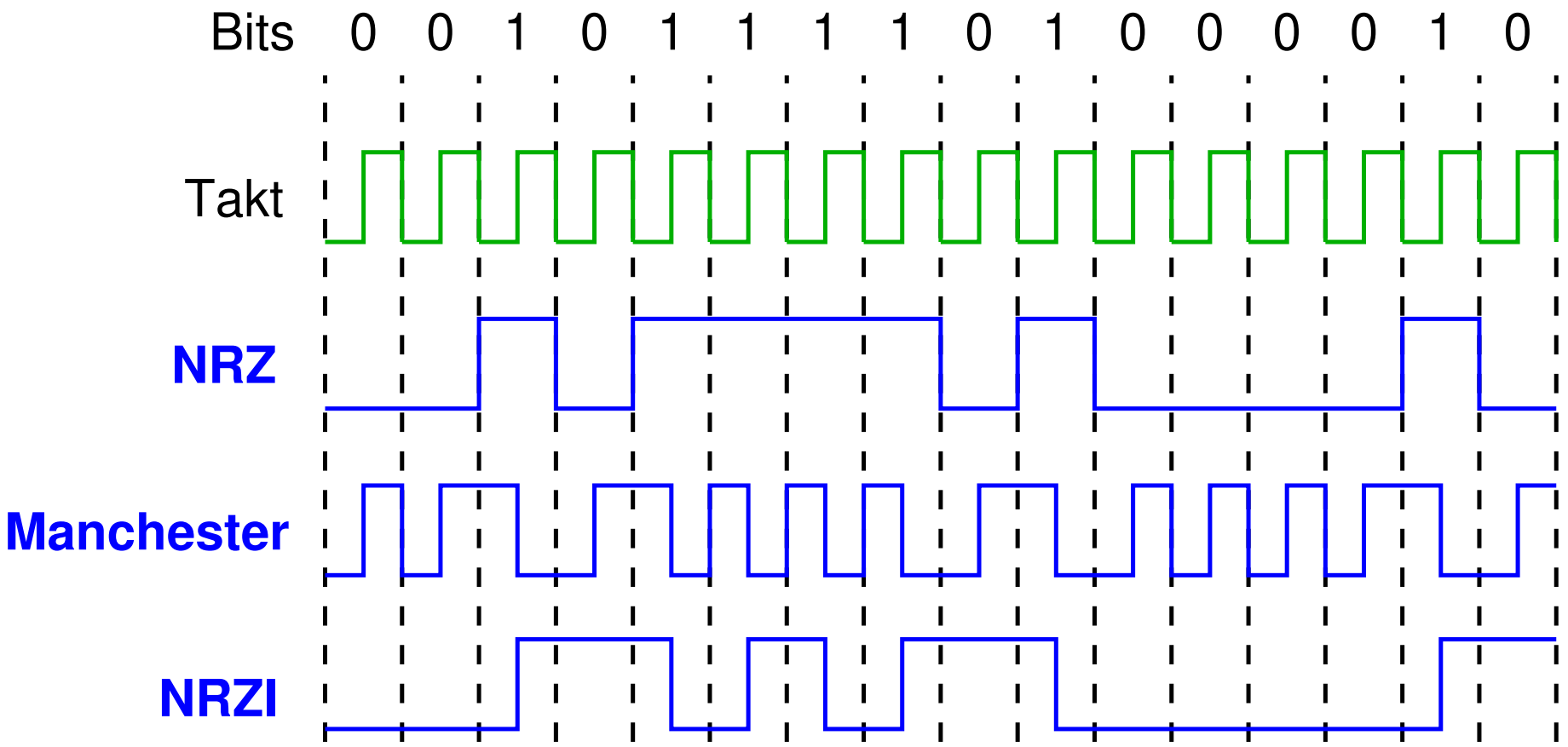


- Übertragung eines Bitstroms zwischen zwei Knoten:



- 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?

- 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

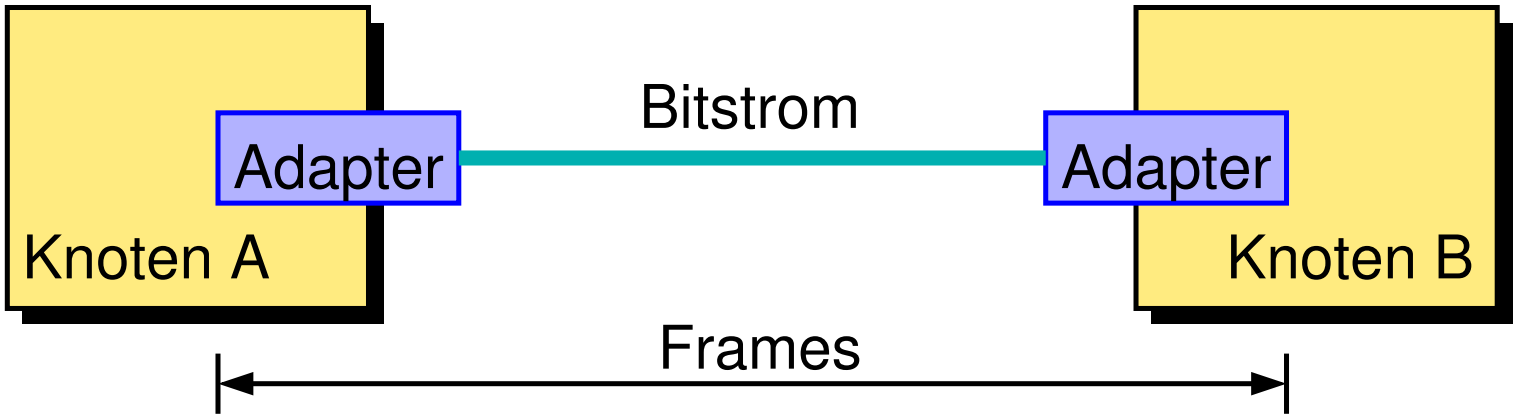
- Signal wird bei jedem **1**-Bit invertiert
- Problem: keine Taktwiederherstellung bei aufeinanderfolgenden Nullen möglich

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

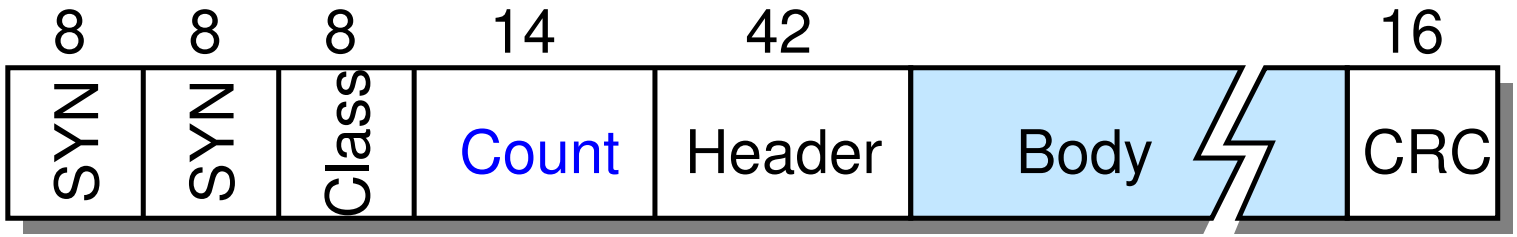
- Wir betrachten nun die Übertragung von Datenblöcken (**Frames**) zwischen Rechnern:



- 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

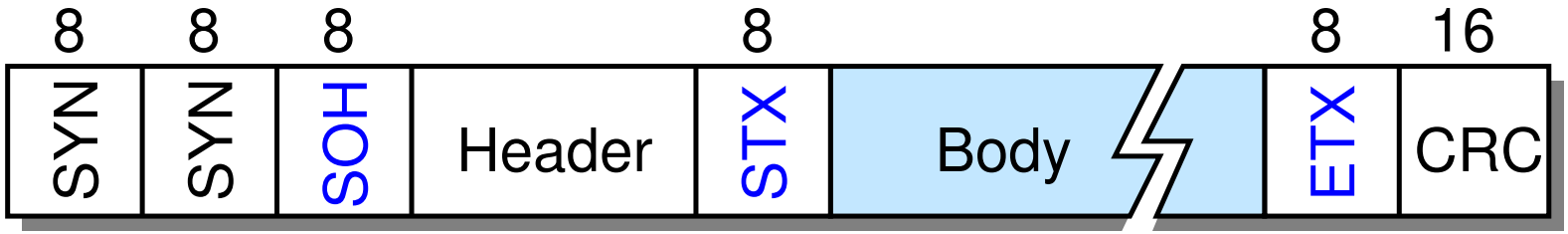
- Frame-Header enthält Länge des Datenteils
- Beispiel: Frame im DDCMP-Protokoll, DECNET:



- 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

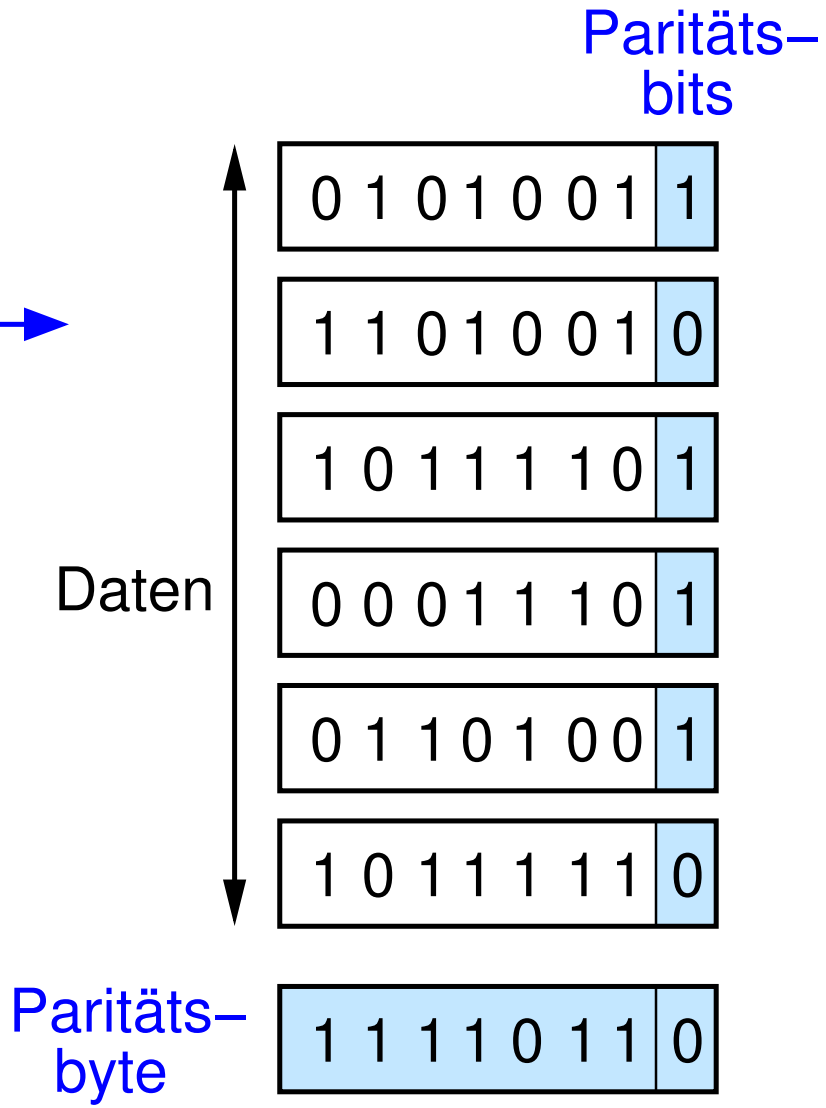
Sentinel-Methode ...

- 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)

- 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 $d = 2$

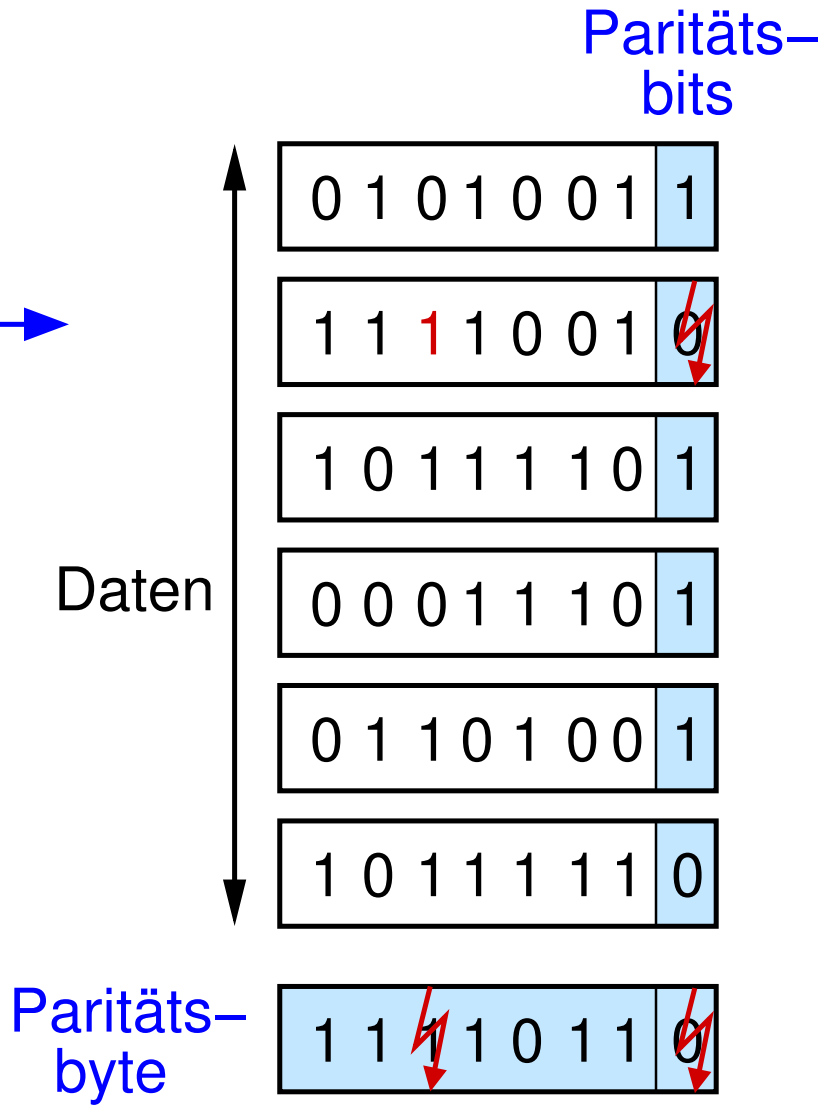
Zweidimensionale Parität

- 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



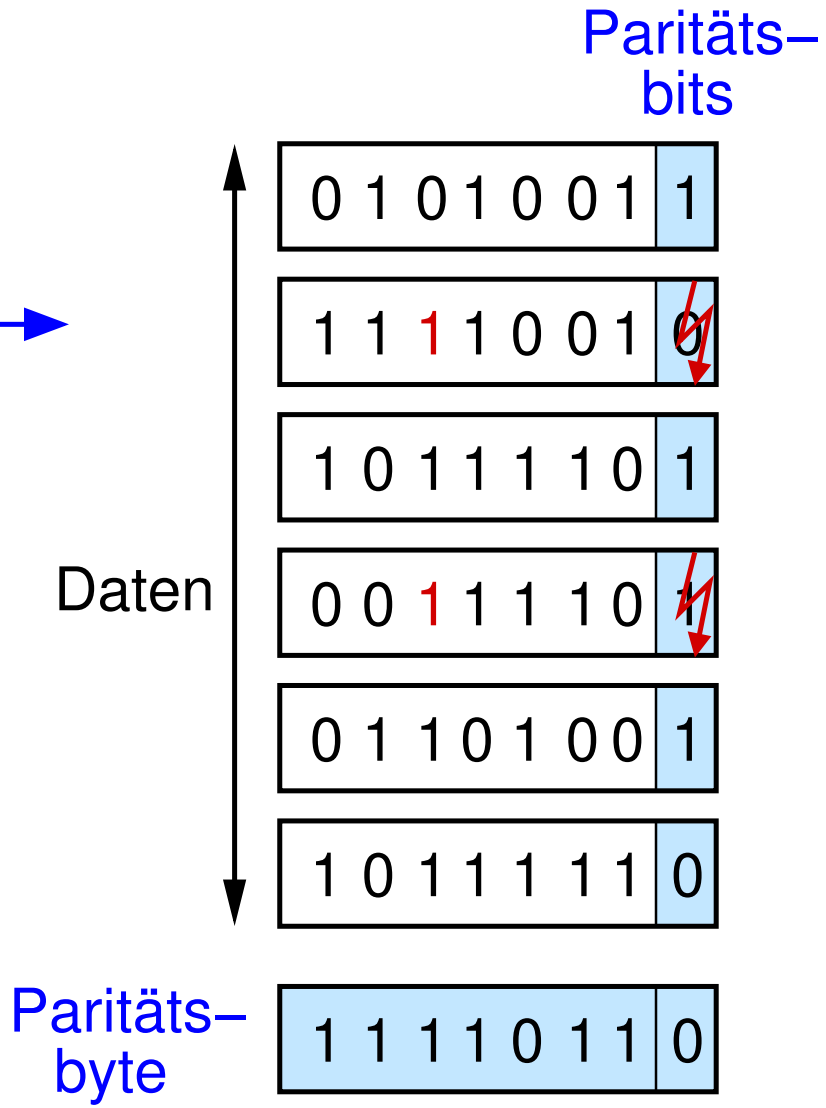
Zweidimensionale Parität

- 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



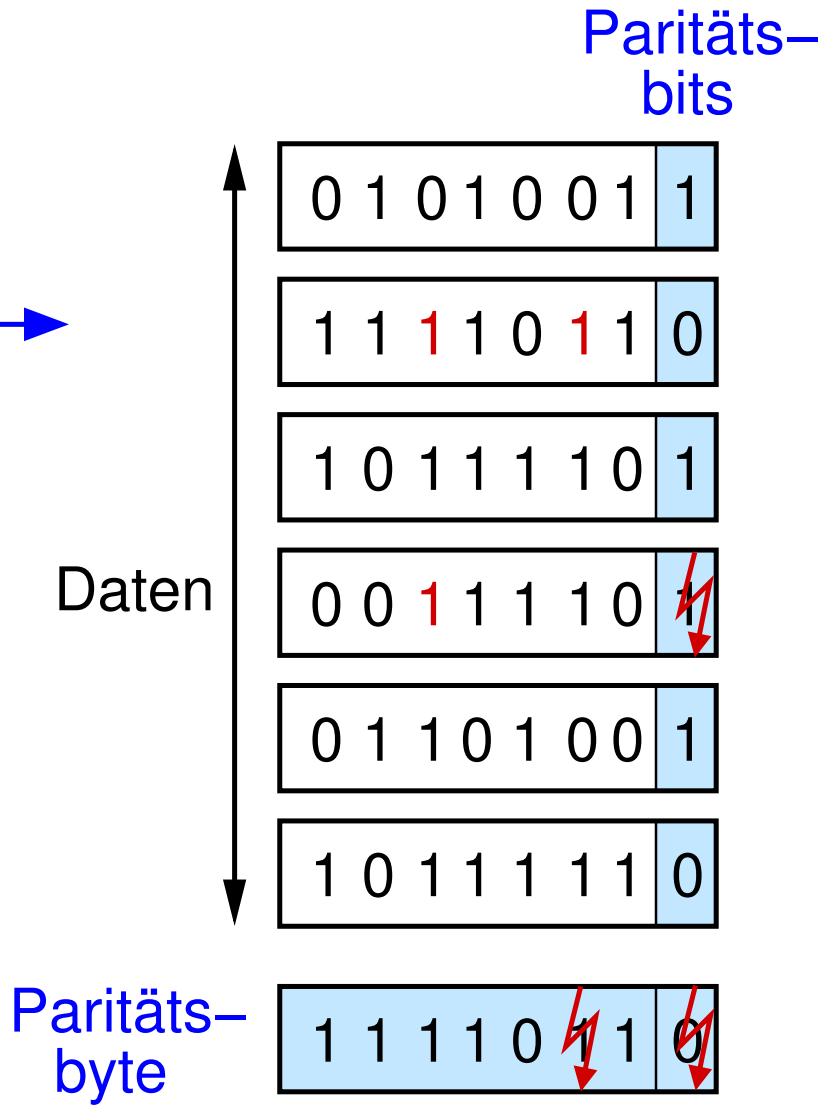
Zweidimensionale Parität

- 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 Warscheinlichkeit 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 Warscheinlichkeit 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

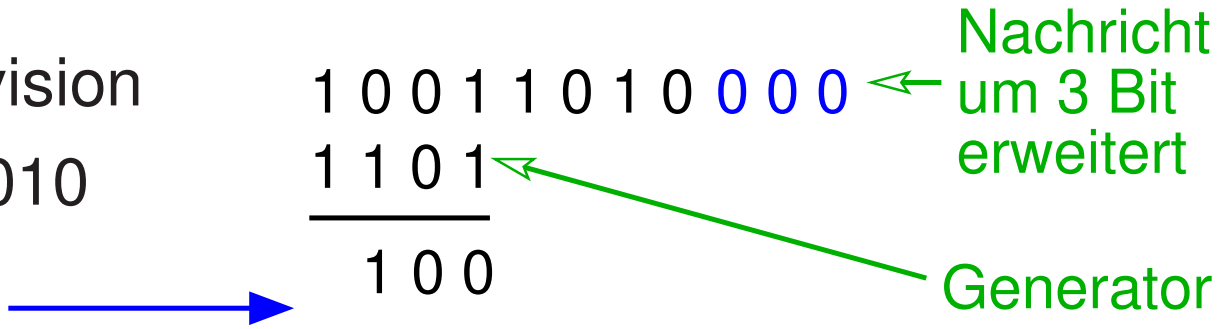
CRC (*Cyclic Redundancy Check*) ...

- 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

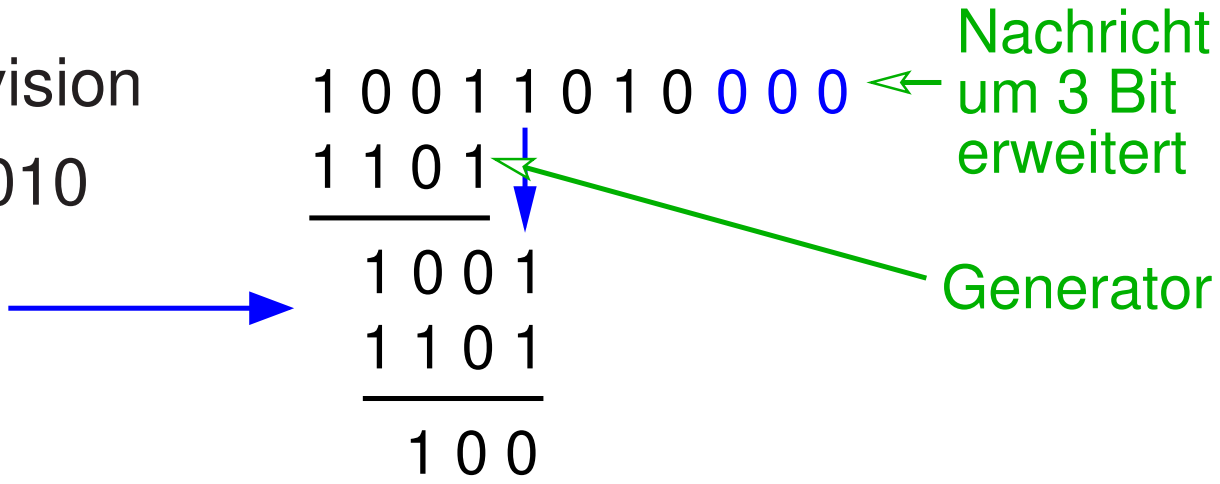
CRC (*Cyclic Redundancy Check*) ...

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



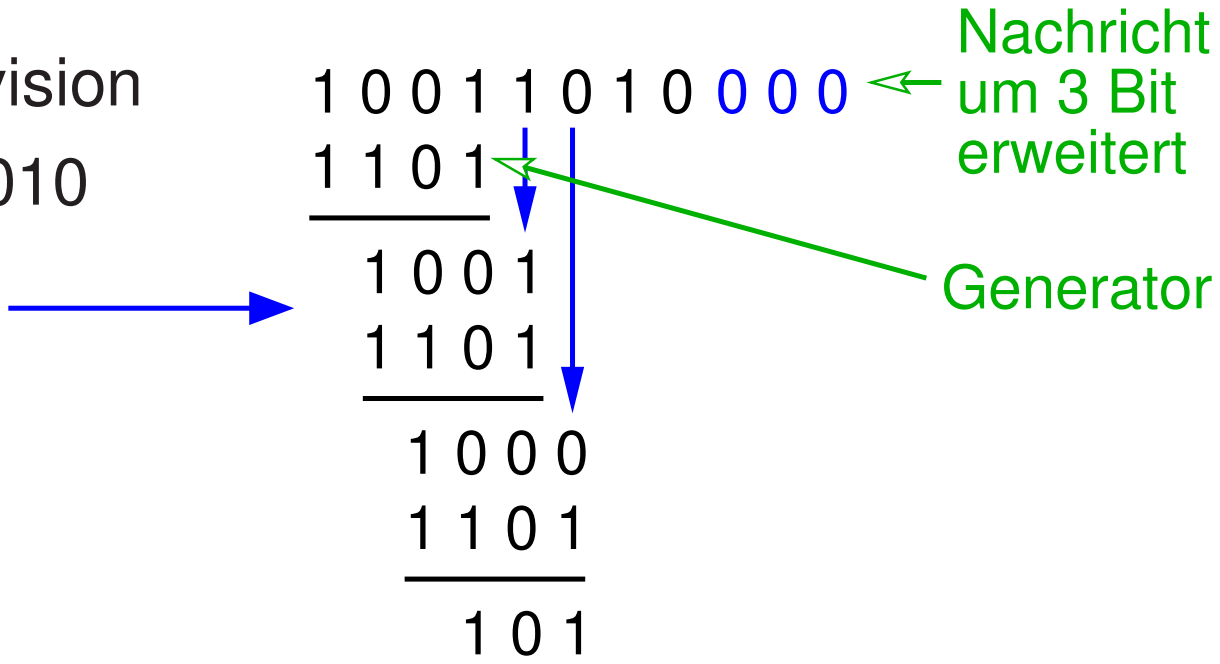
CRC (*Cyclic Redundancy Check*) ...

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



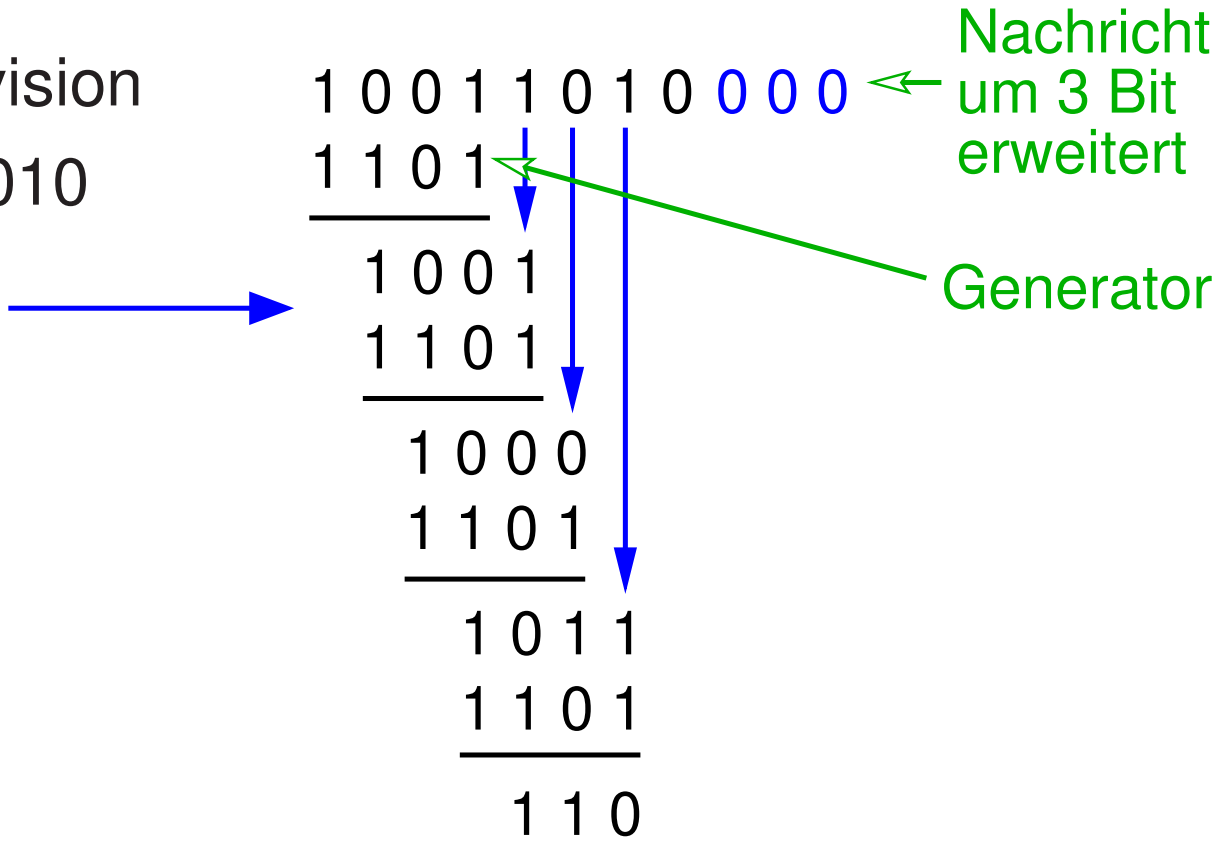
CRC (*Cyclic Redundancy Check*) ...

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



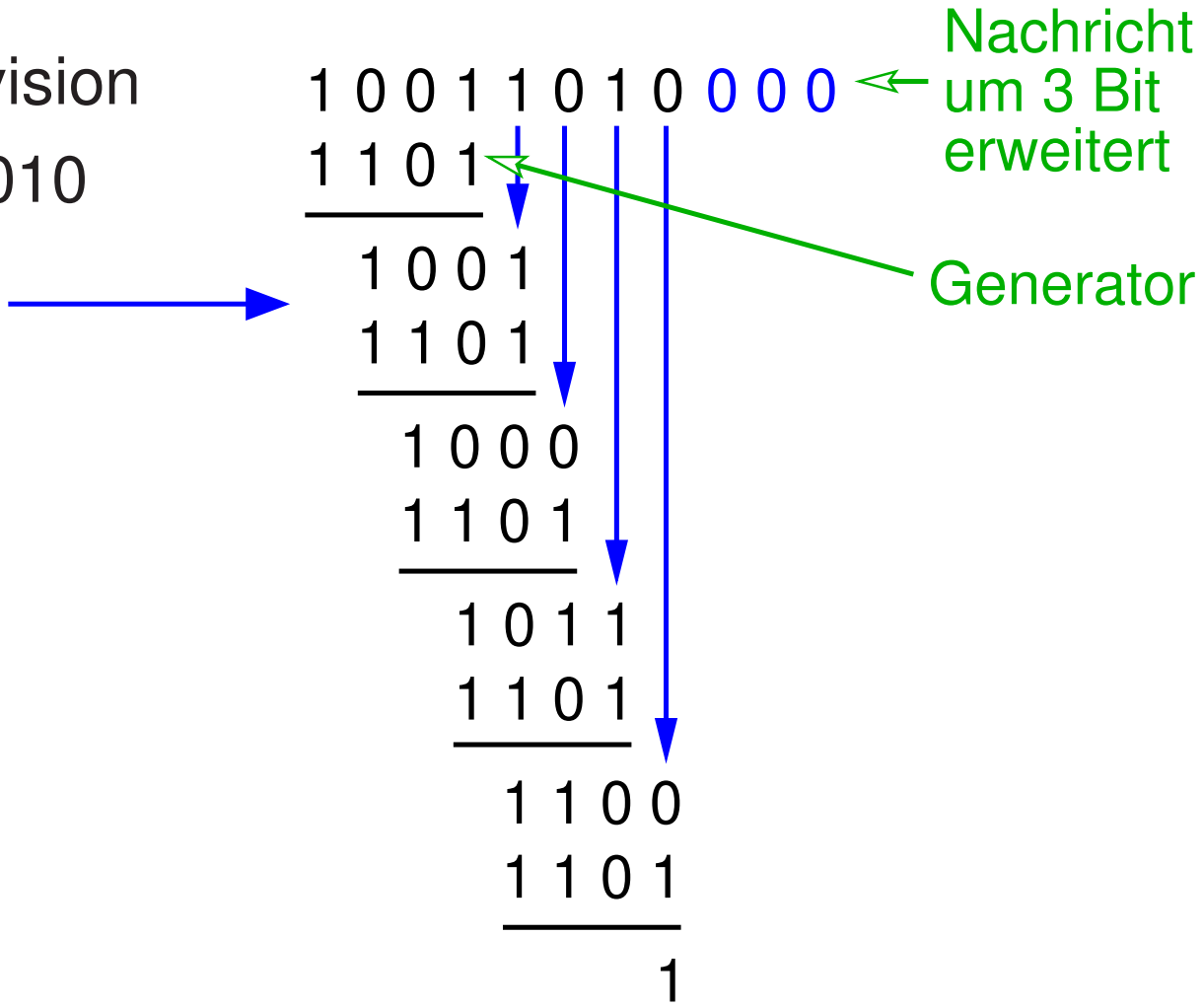
CRC (*Cyclic Redundancy Check*) ...

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



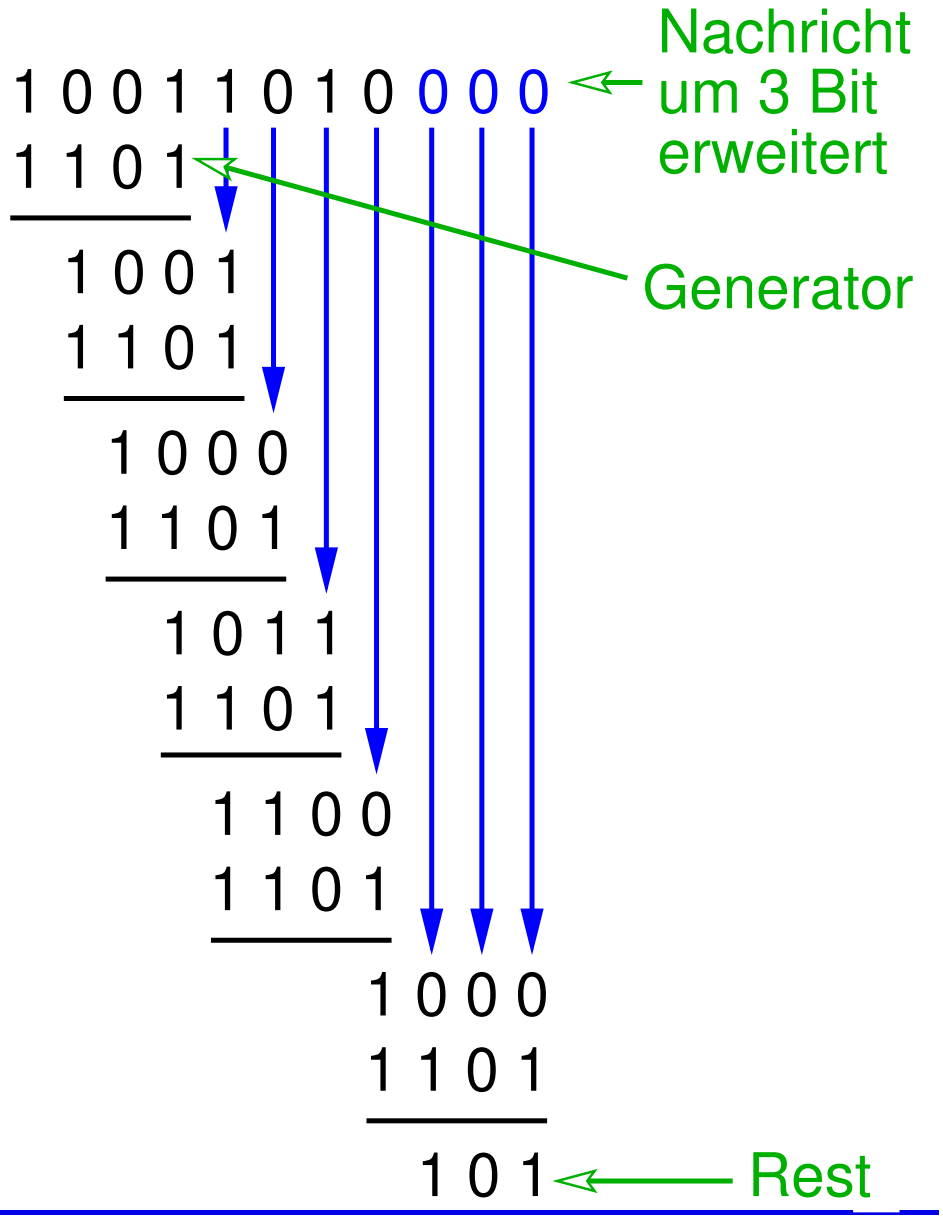
CRC (*Cyclic Redundancy Check*) ...

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



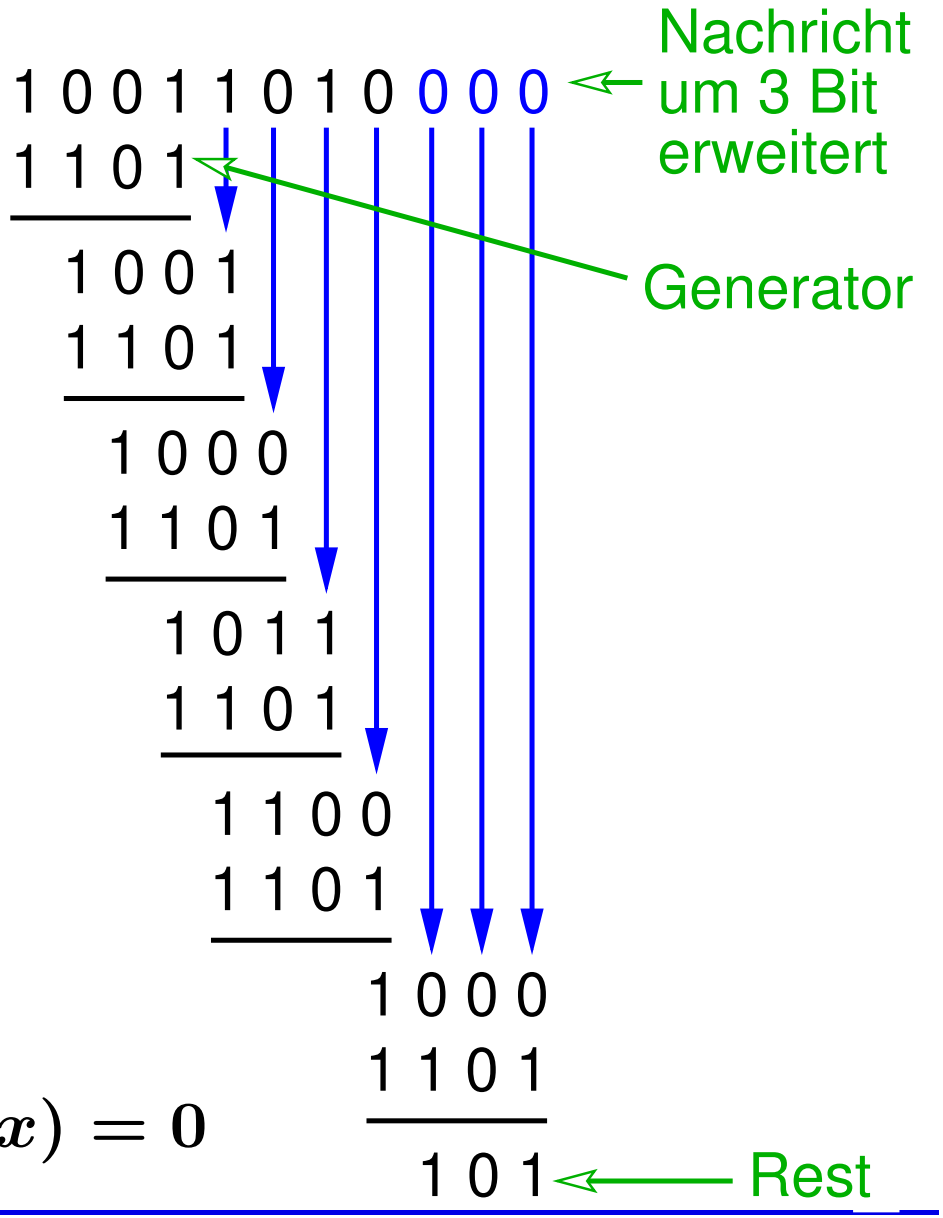
CRC (*Cyclic Redundancy Check*) ...

- 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

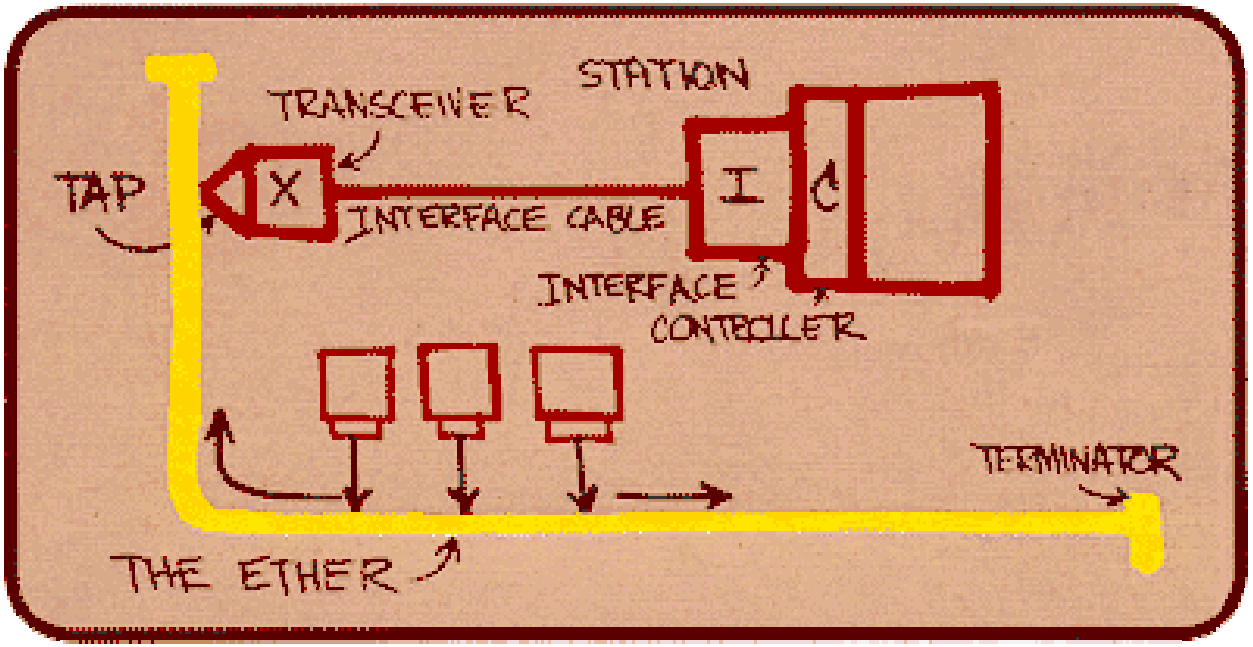
- Gründe für den Einsatz von CRC:
  - Gute Fehlererkennung
  - Sehr effizient in Hardware realisierbar

- 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 Verfahren:
  - CSMA/CD (Ethernet)
  - Tokenbasierte Verfahren (Token-Ring)
  - CSMA/CA (WLAN, CAN-Bus, ☞ **RN_II**)

3.6.1 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

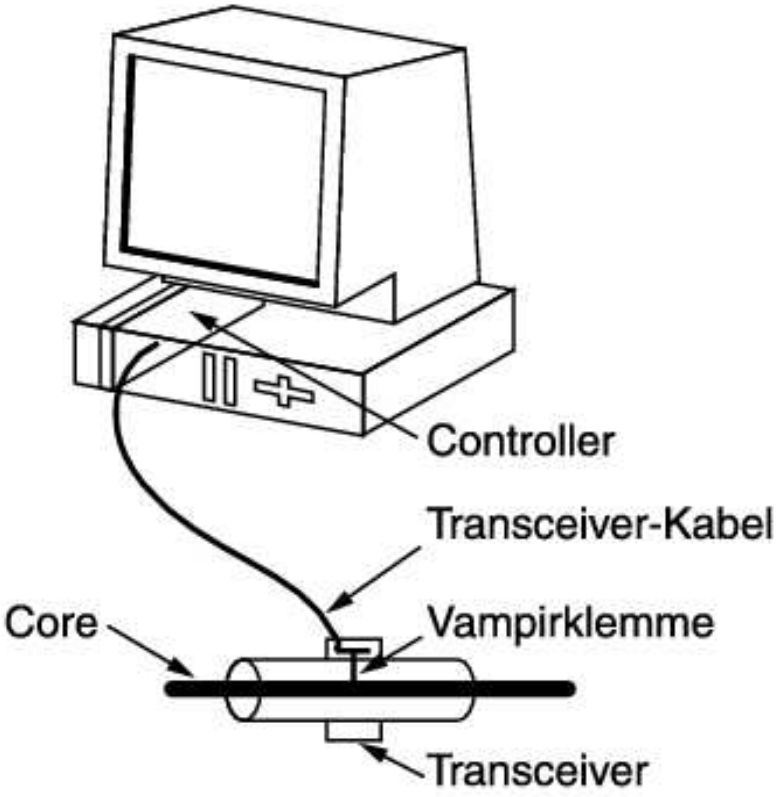


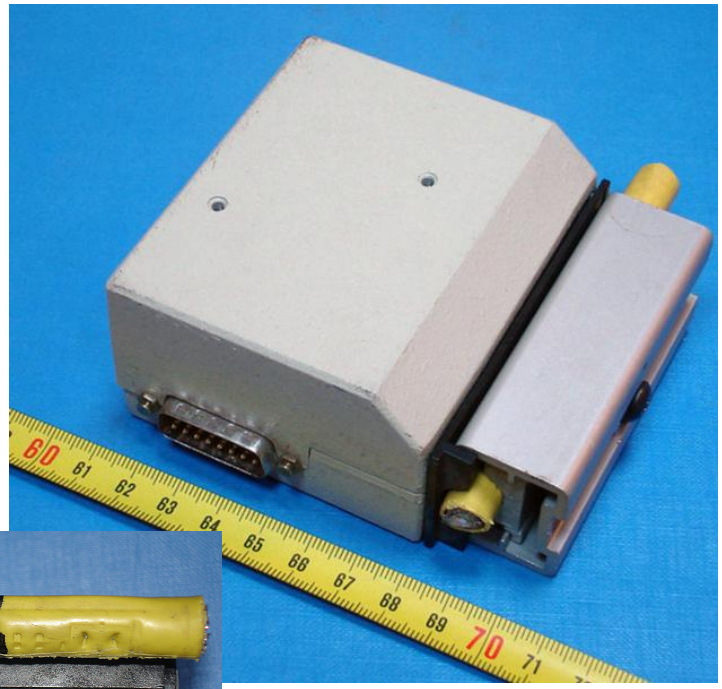
Bob Metcalfe
Xerox, 1976

- Heute: Punkt-zu-Punkt Verbindungen

Verkabelung

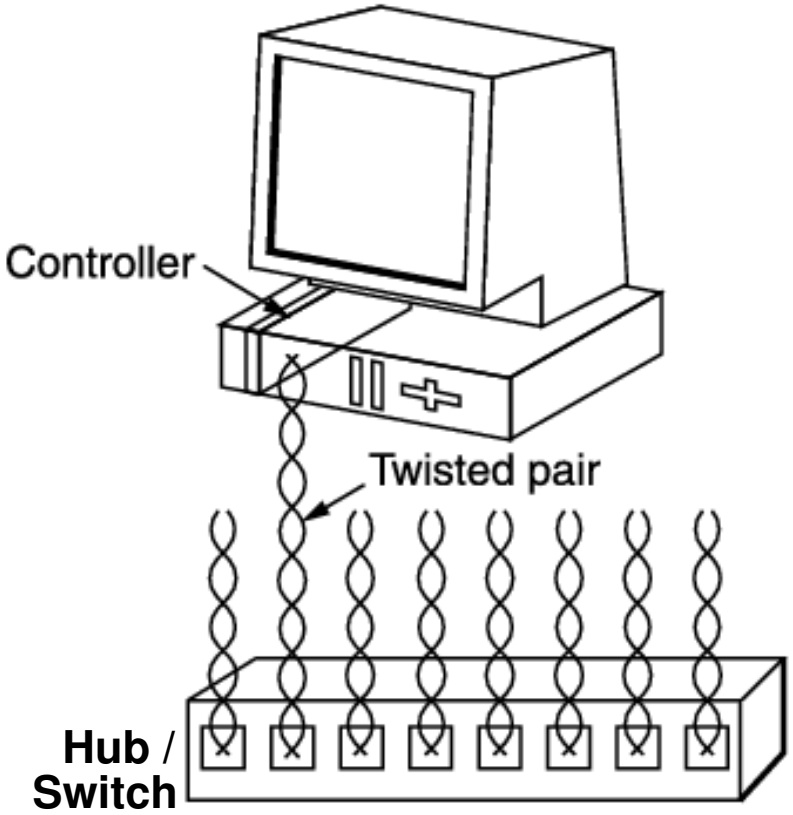
10Base5 (max. 500m)

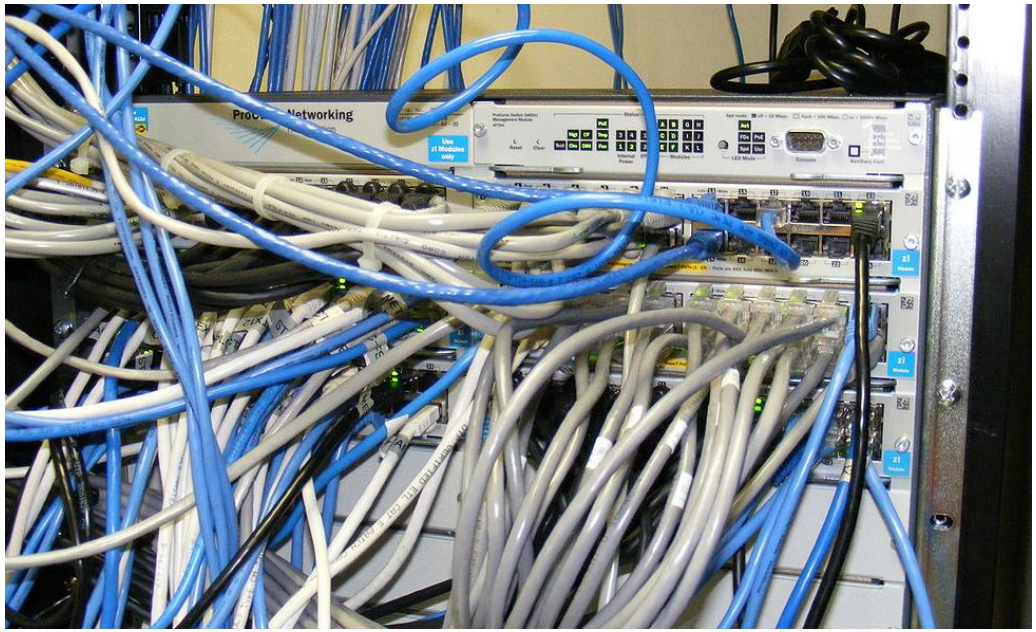




Verkabelung

10BaseT / 100BaseTx (100m)



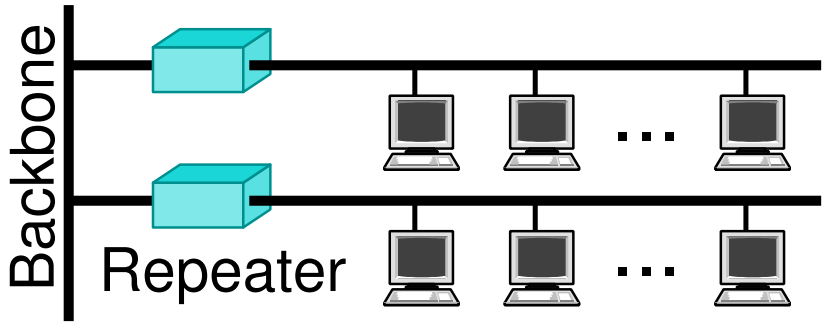


- **Hub**: „Verstärker“ und Verteiler (OSI-Schicht 1)
- **Switch**: Vermittlungsknoten (OSI-Schicht 2)

Physische Eigenschaften

10BASE-5 (10 Mb/s)

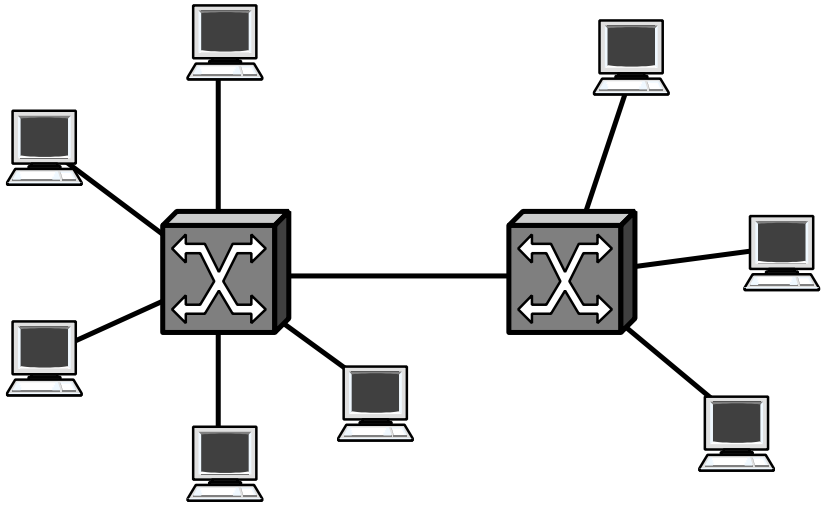
- Segmente aus Koaxialkabel, je max. 500m
- Segmente über **Repeater** (Hub mit 2 Ports) verbindbar



  - max. 4 Repeater zw. zwei Knoten erlaubt
- Manchester-Codierung

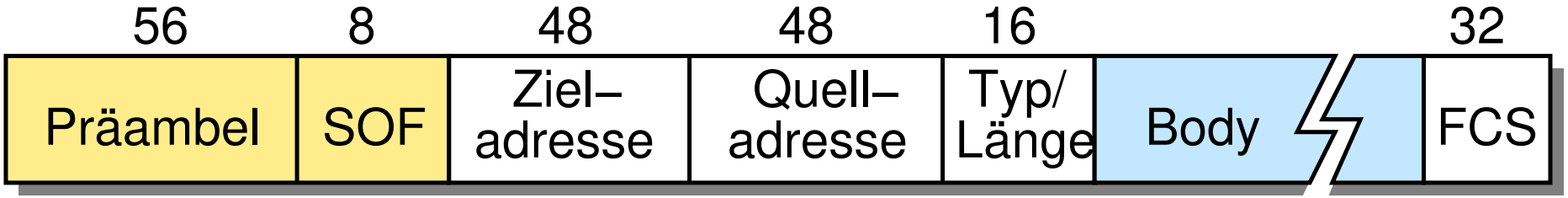
100BASE-TX (100 Mb/s)

- 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

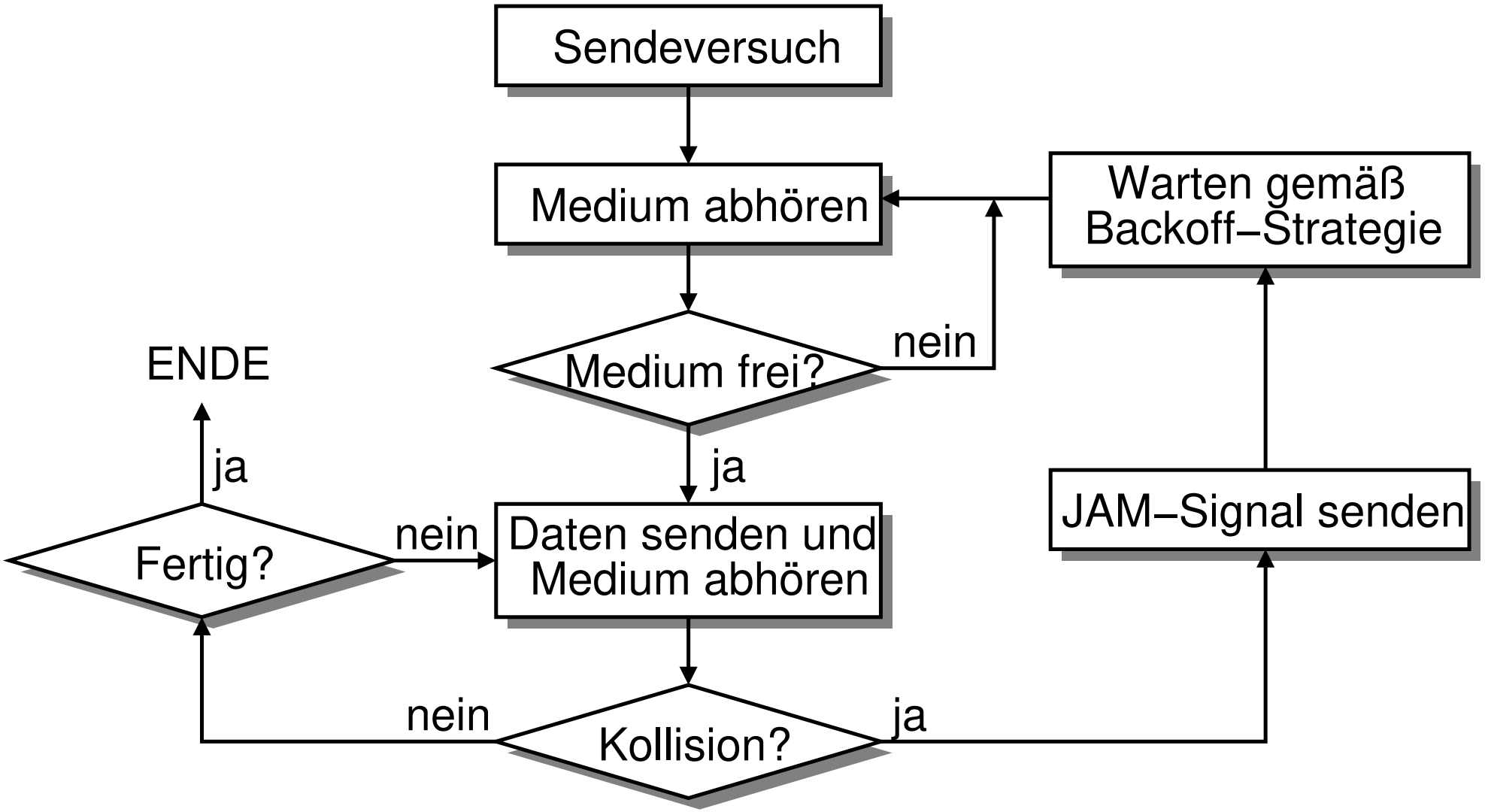
Ethernet-Adressen (MAC-Adressen)

- 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

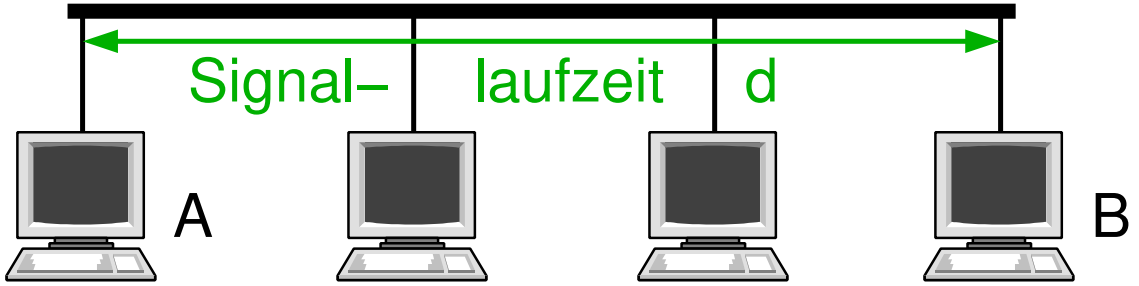
CSMA/CD – Algorithmus



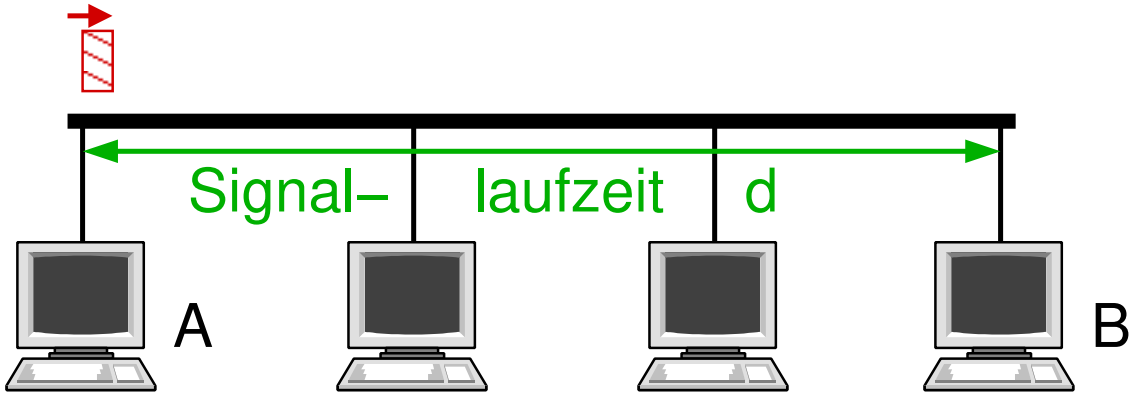
Exponential Backoff-Strategie

- 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: neue Sendeversuche zeitlich entzerrt, Wartezeit an Netzlast angepaßt

Kollisionserkennung: Worst-Case Szenario

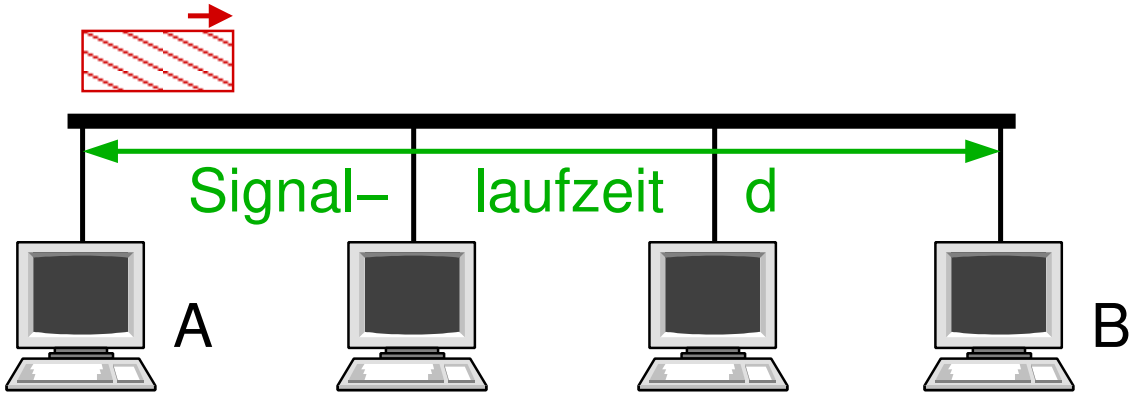


Kollisionserkennung: Worst-Case Szenario



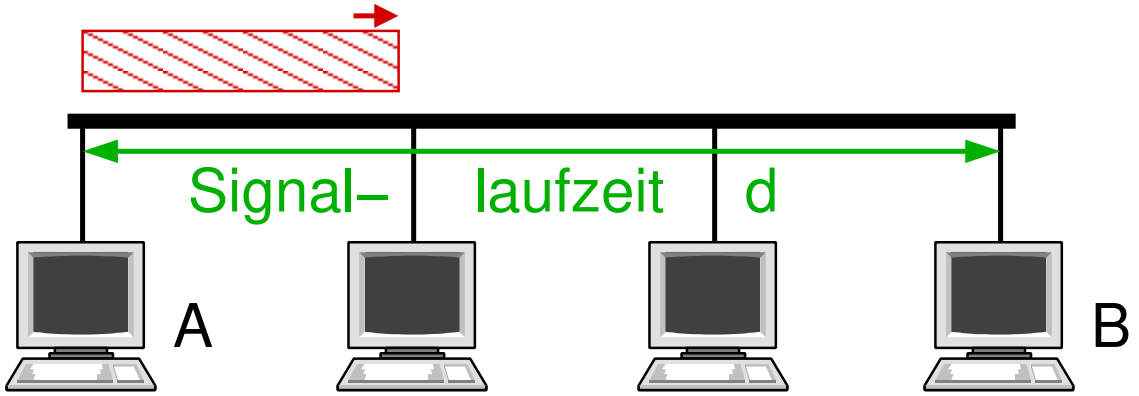
➥ $t = 0$: A beginnt, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



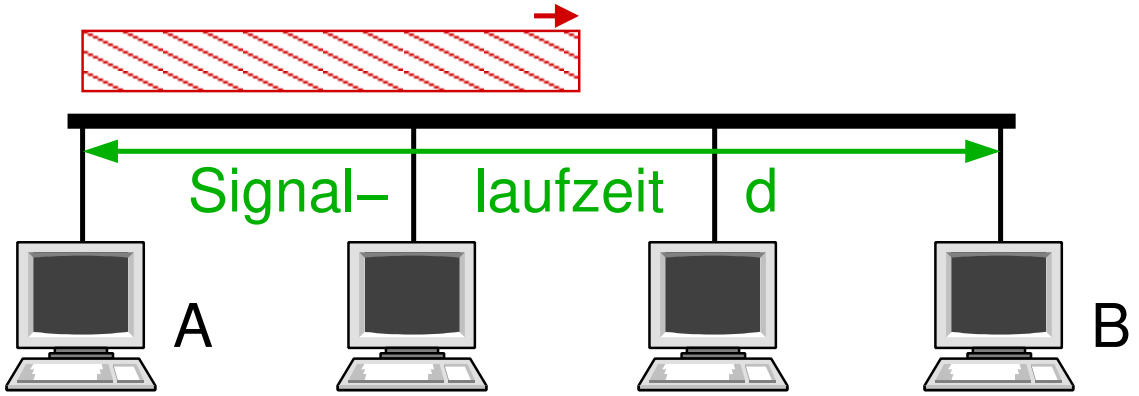
➥ $t = 0$: A beginnt, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



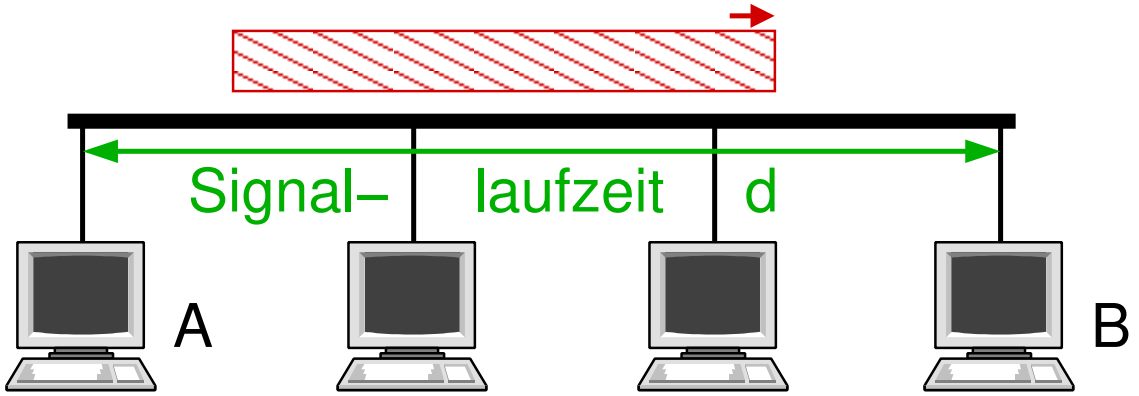
➥ $t = 0$: A beginnt, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



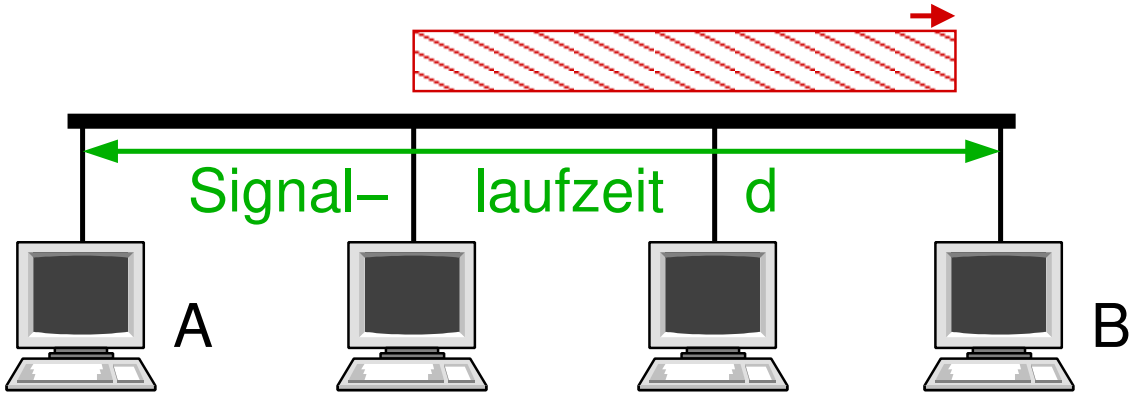
➥ $t = 0$: A beginnt, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



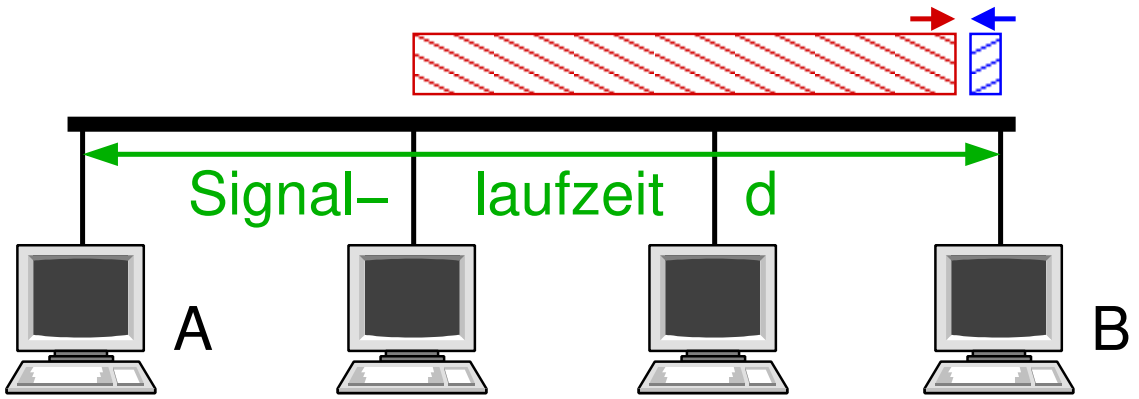
➥ $t = 0$: A beginnt, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



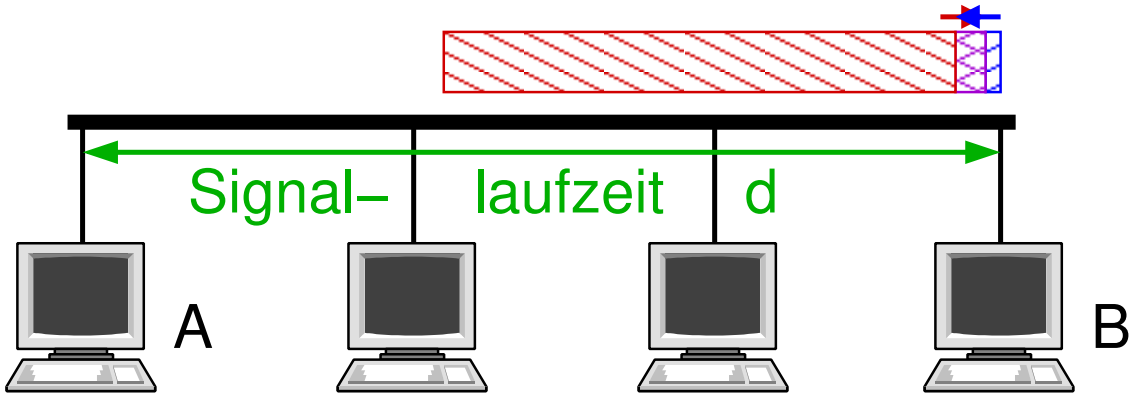
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: Das Medium ist bei B noch frei

Kollisionserkennung: Worst-Case Szenario



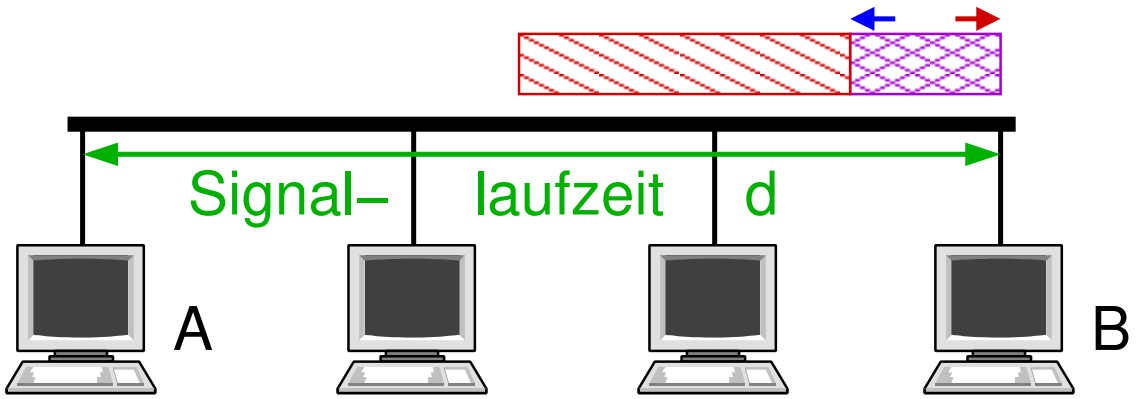
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden

Kollisionserkennung: Worst-Case Szenario



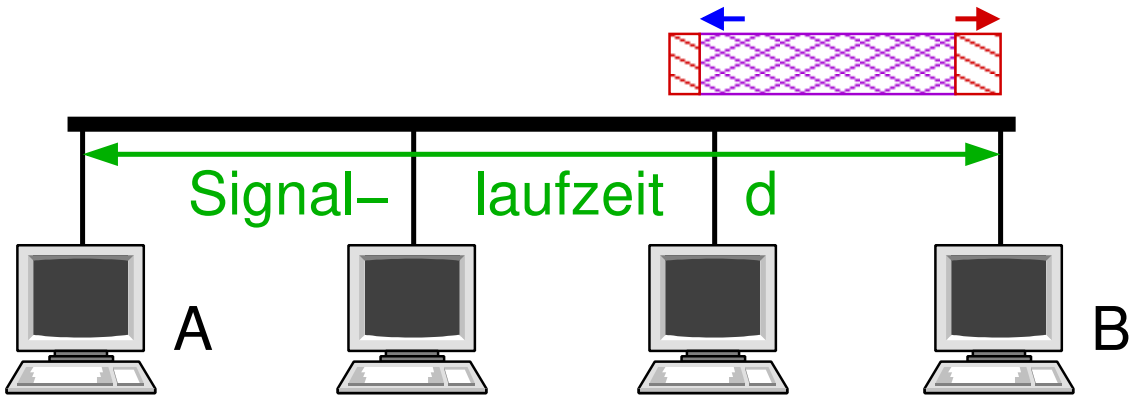
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

Kollisionserkennung: Worst-Case Szenario



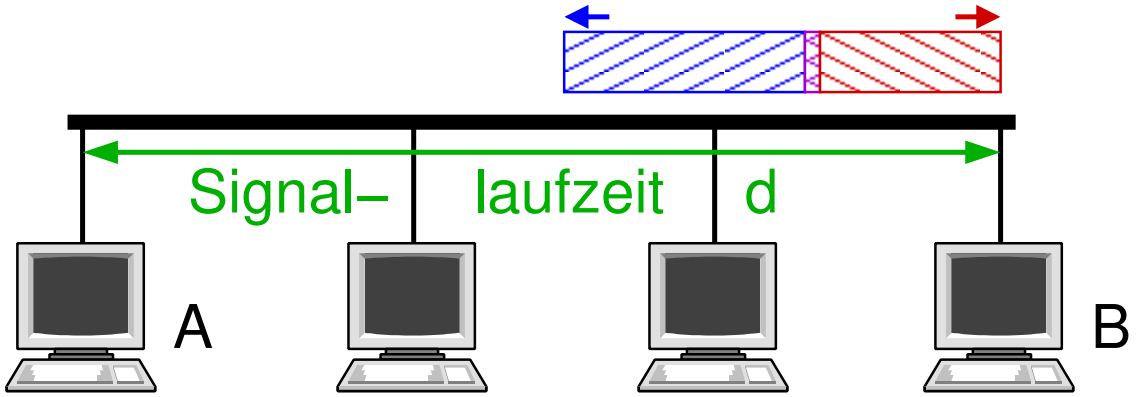
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

Kollisionserkennung: Worst-Case Szenario



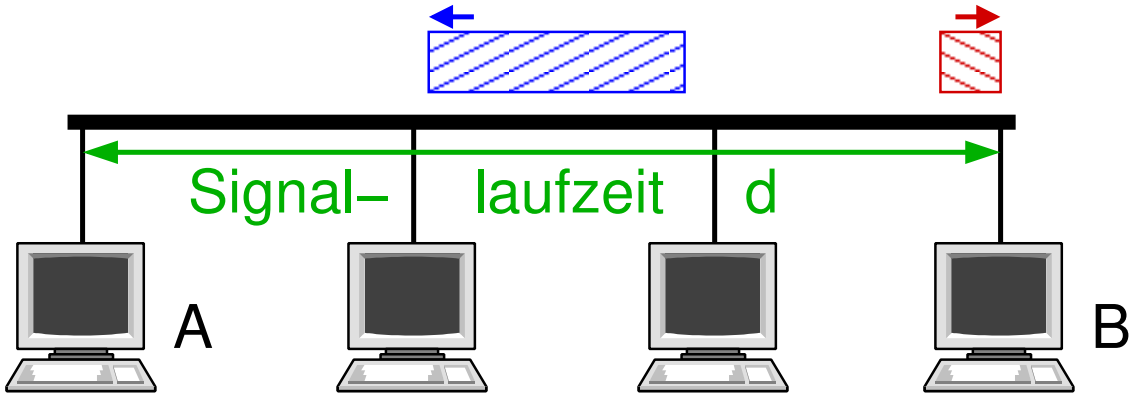
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

Kollisionserkennung: Worst-Case Szenario



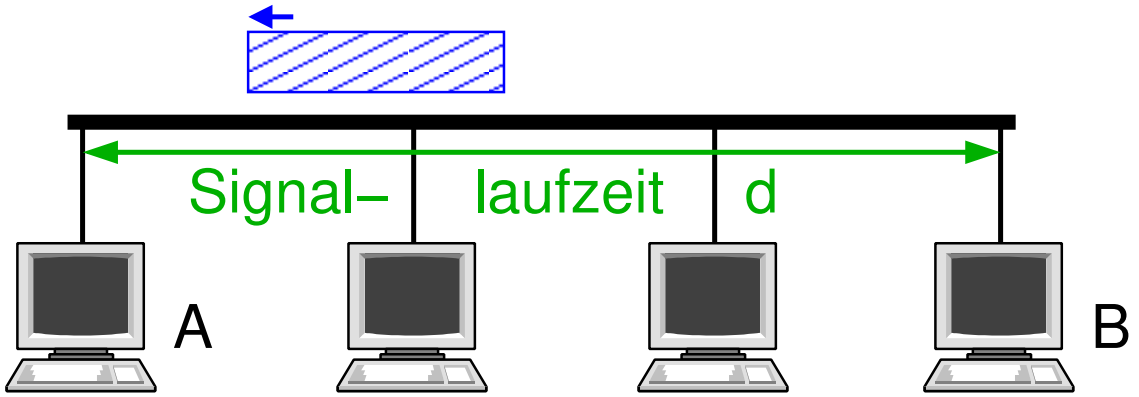
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

Kollisionserkennung: Worst-Case Szenario



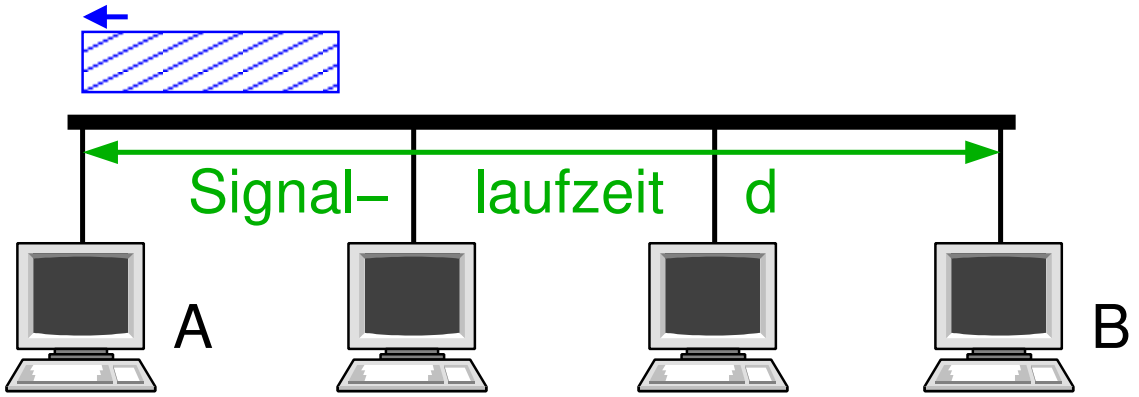
- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!

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

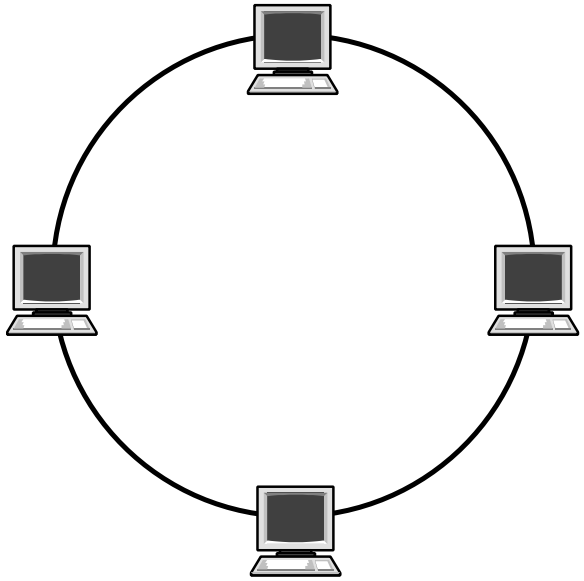


- $t = 0$: A beginnt, einen Frame zu senden
- $t = d - \epsilon$: B beginnt ebenfalls, einen Frame zu senden
- $t = d$: A's Frame kommt bei B an $\Rightarrow$ Kollision!
- $t = 2 \cdot d$: B's (Kollisions-)Frame kommt bei A an
  - wenn A zu diesem Zeitpunkt nicht mehr sendet, erkennt A keine Kollision

Sichere Kollisionserkennung

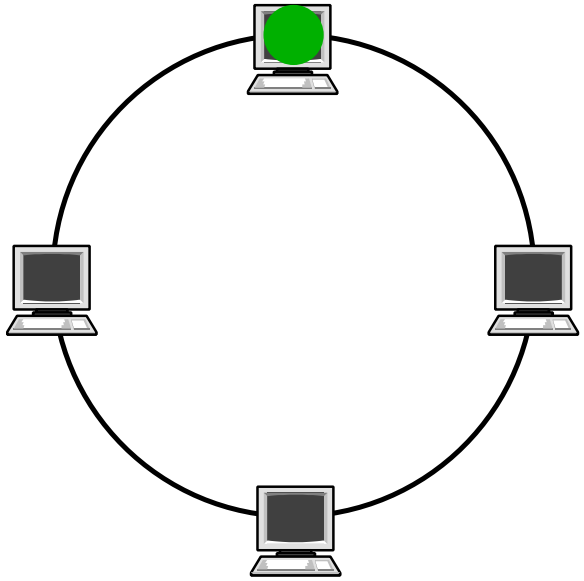
- Um Kollisionen immer erkennen zu können, definiert Ethernet:
  - maximale RTT: 512 Bit-Zeiten
    - 51,2 μs bei 10 Mb/s, 5,12 μs bei 100 Mb/s
    - legt maximale Ausdehnung des Netzes fest, z.B. 200m bei 100BASE-TX mit Hubs
  - minimale Framelänge: 512 Bit (64 Byte), zzgl. Präambel
    - kleinere Frames 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

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

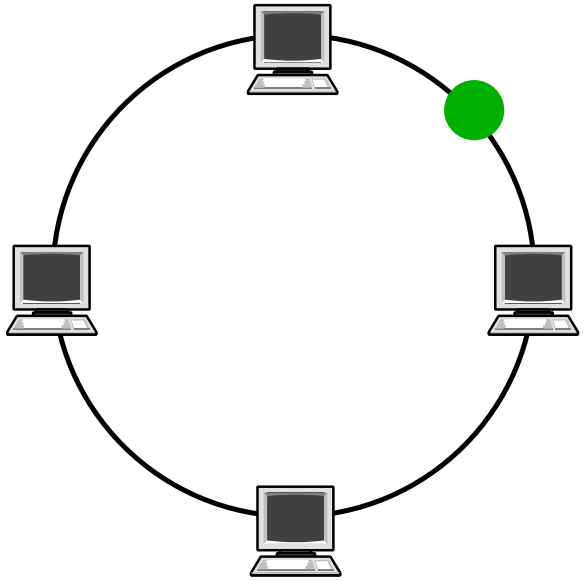


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

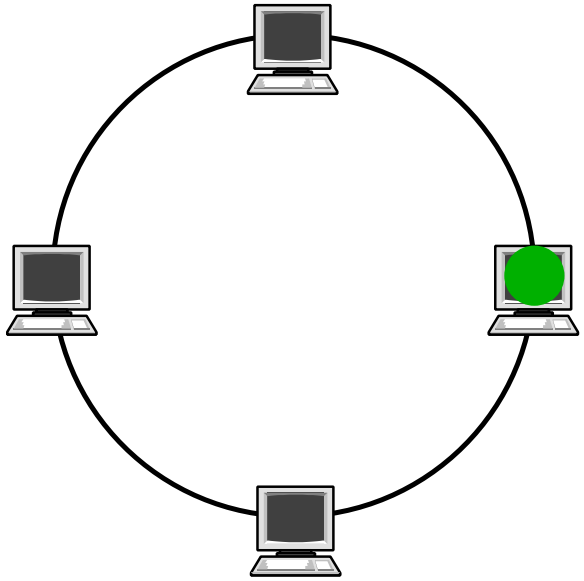


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben



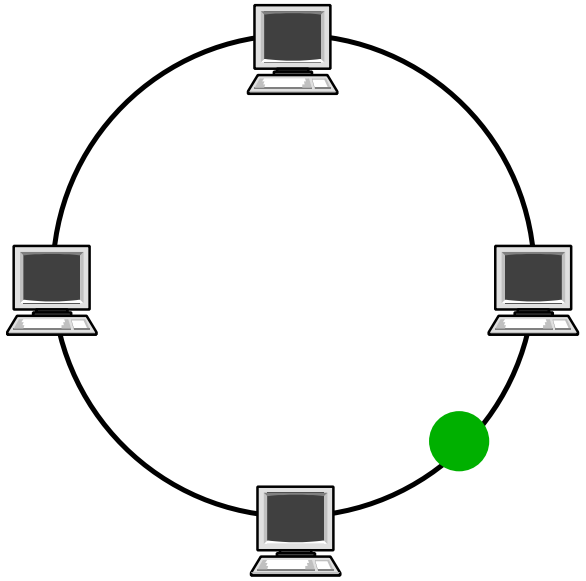
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



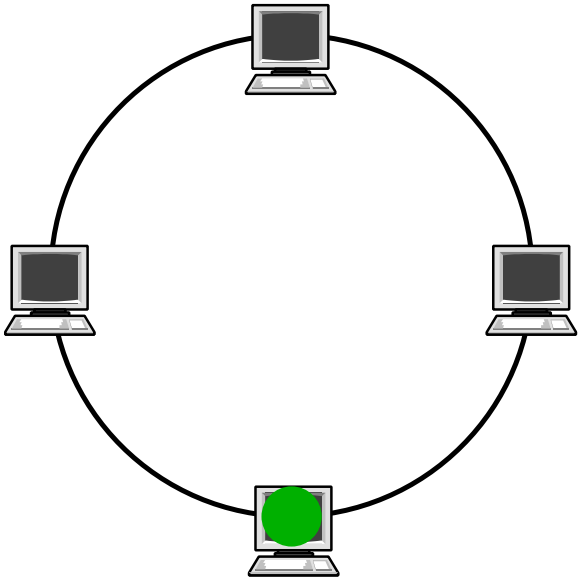
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



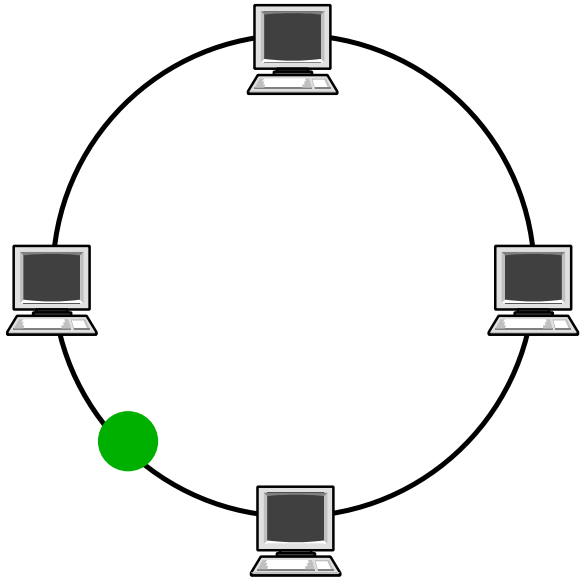
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

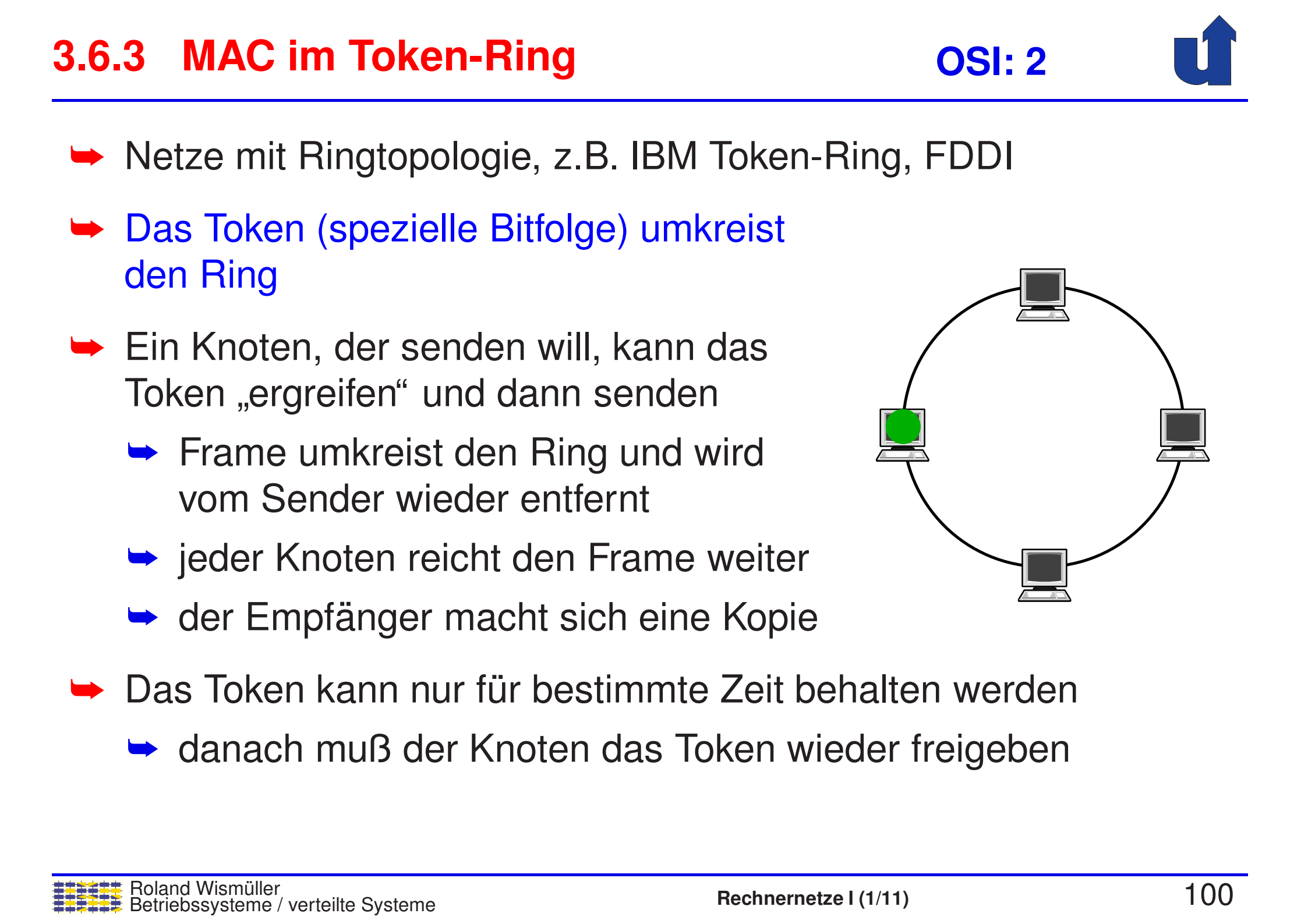


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



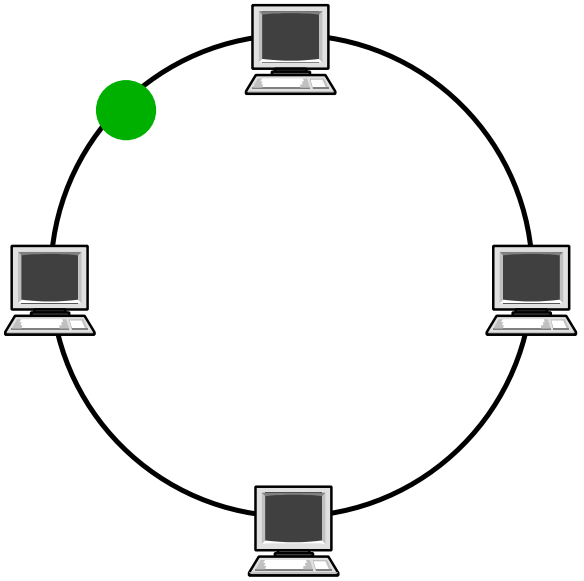
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben



- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

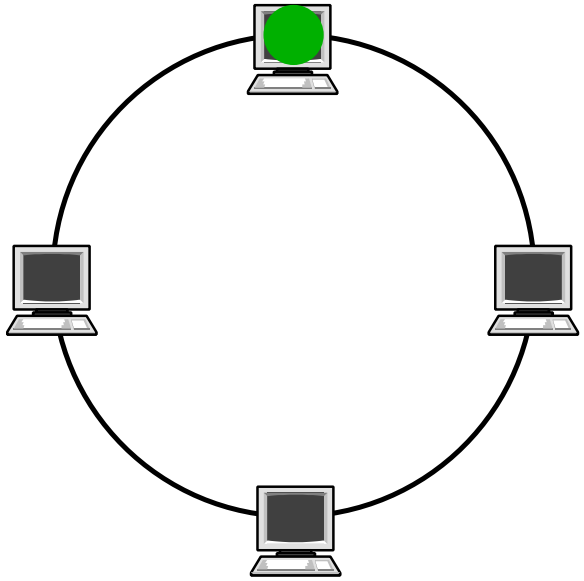
3.6.3 MAC im Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

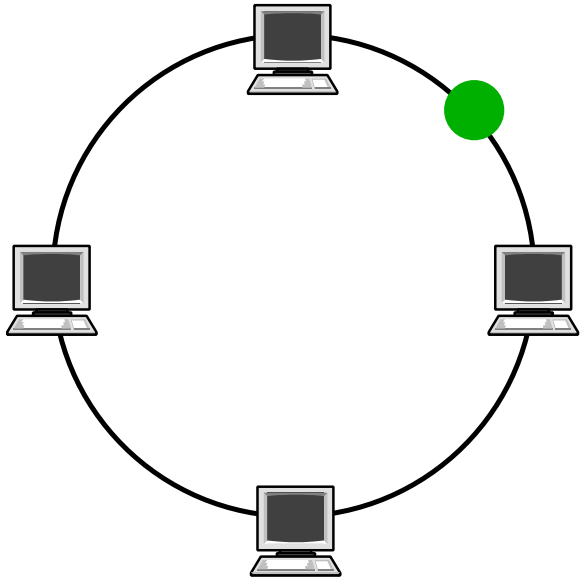


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

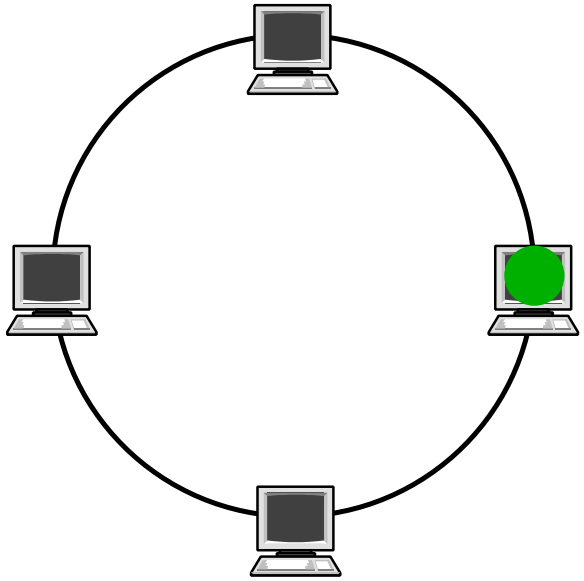


- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben



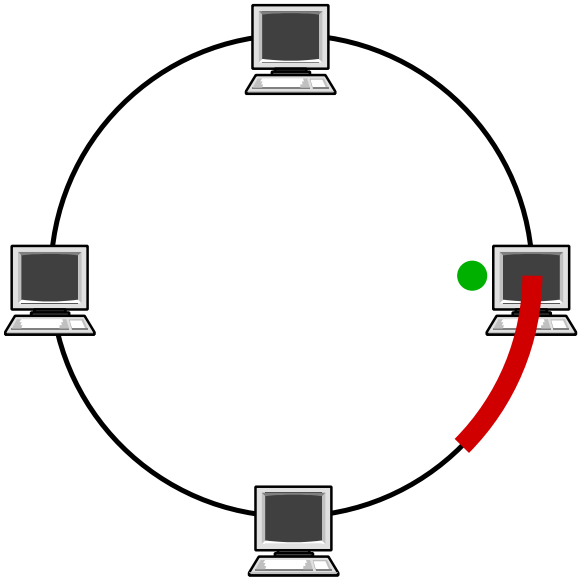
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger macht sich eine Kopie
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

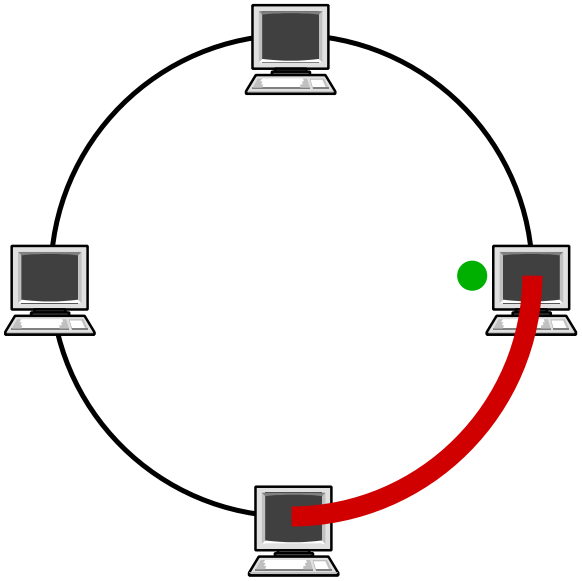
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger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



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

3.6.3 MAC im Token-Ring

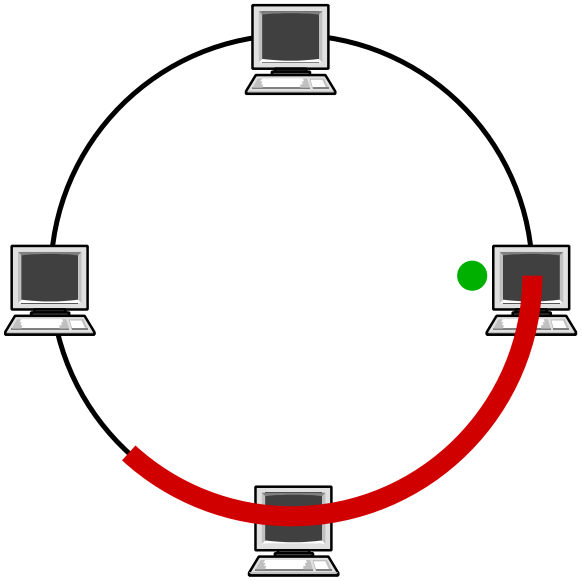
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger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



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

3.6.3 MAC im Token-Ring

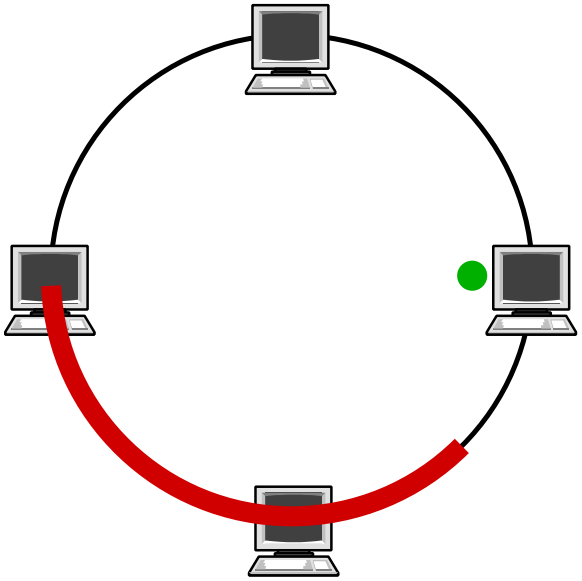
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger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



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

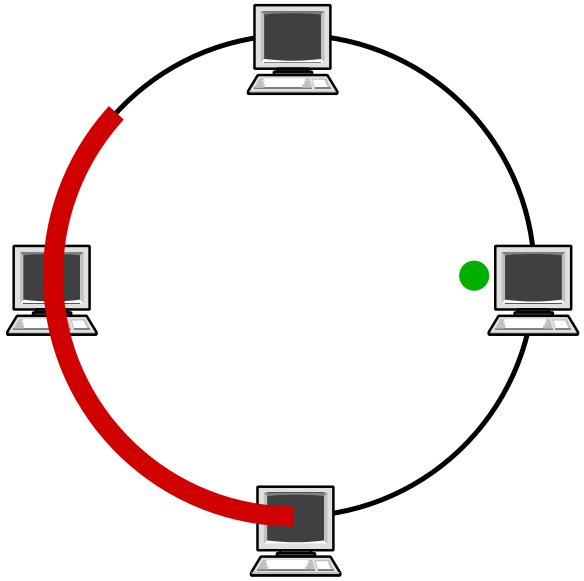
3.6.3 MAC im Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



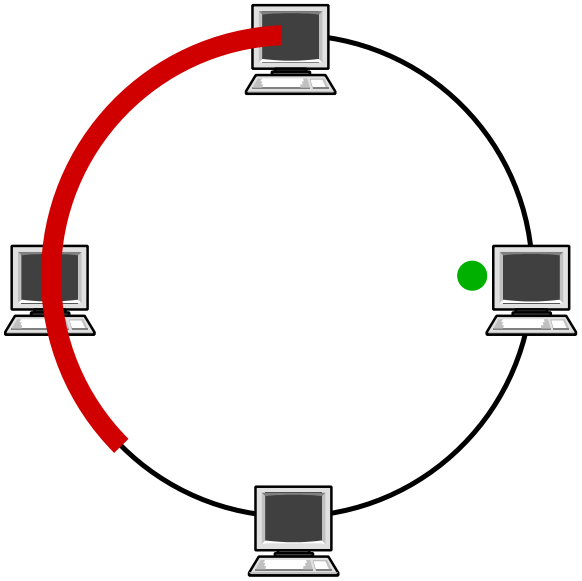
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

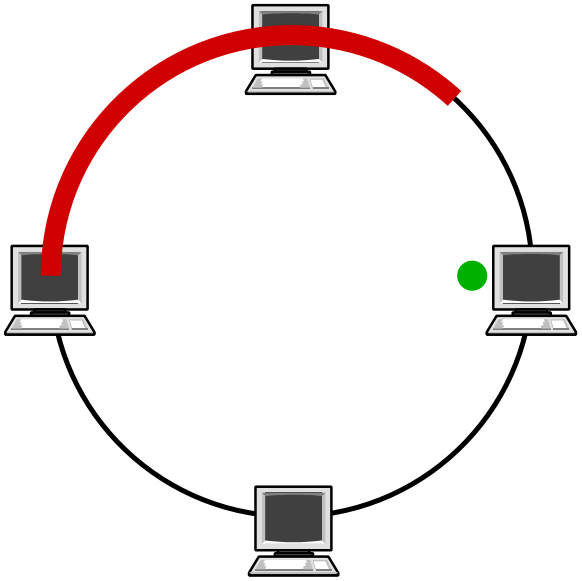
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger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



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

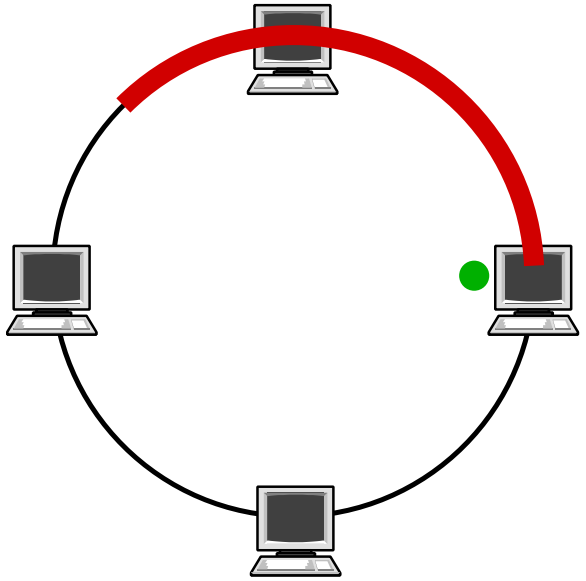
3.6.3 MAC im Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



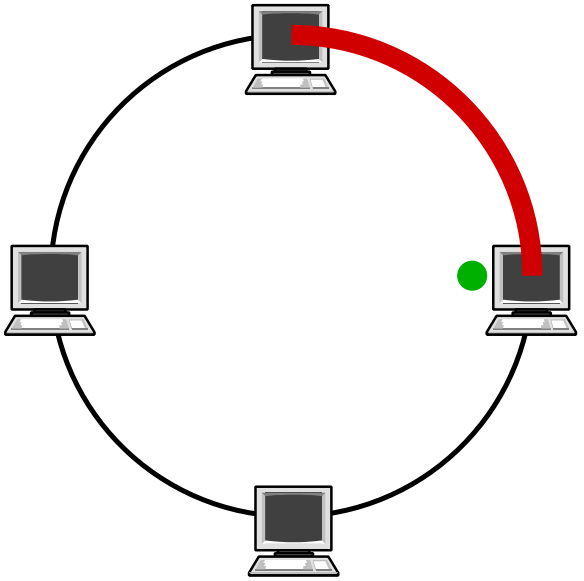
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



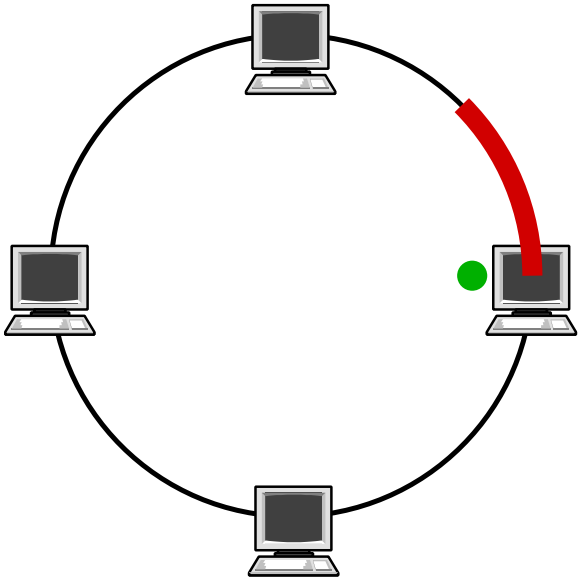
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



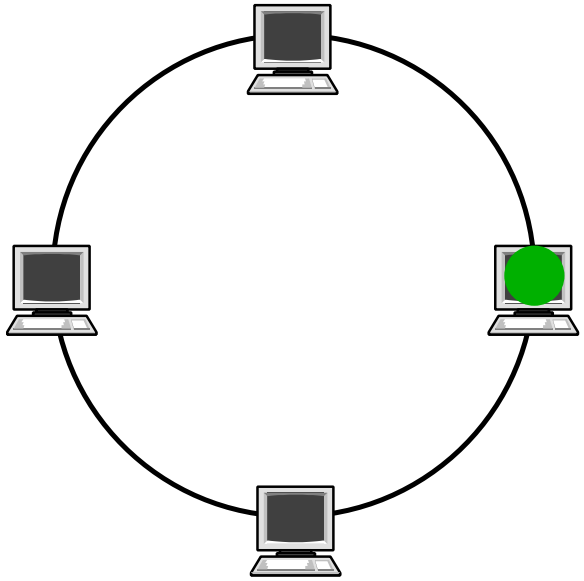
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

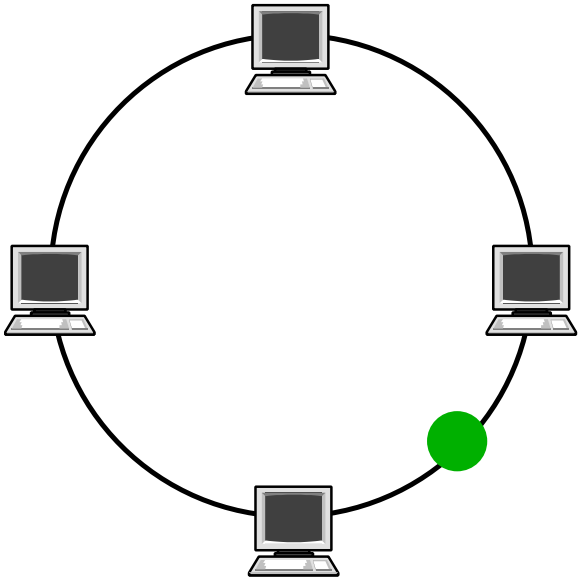
- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist den Ring
- Ein Knoten, der senden will, kann das Token „ergreifen“ und dann senden
  - Frame umkreist den Ring und wird vom Sender wieder entfernt
  - jeder Knoten reicht den Frame weiter
  - der Empfänger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



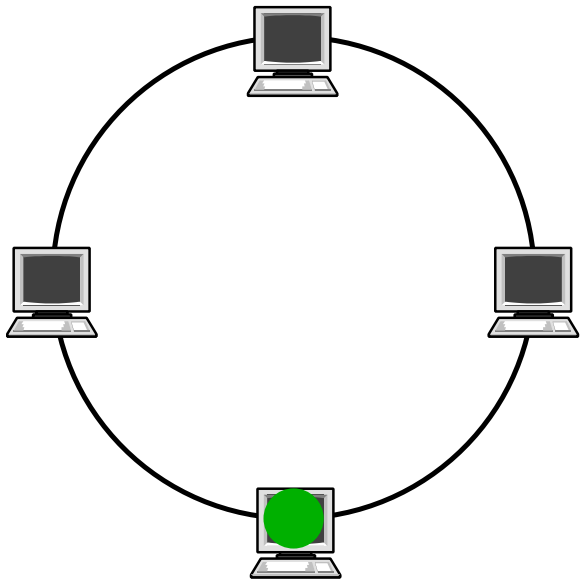
- Das Token kann nur für bestimmte Zeit behalten werden
  - danach muß der Knoten das Token wieder freigeben

3.6.3 MAC im Token-Ring

- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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



- Netze mit Ringtopologie, z.B. IBM Token-Ring, FDDI
- Das Token (spezielle Bitfolge) umkreist 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

Diskussion

- 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

- Daher: Eignung für Realzeitanwendungen

- Mit CSMA/CD sind keine derartigen Garantien möglich

- Andererseits: bei unbelastetem Netz kann ein Knoten bei CSMA/CD immer sofort senden

3.7 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)
  - Ethernet (CSMA-CD)
  - Token-Ring: garantierte maximale Sendeverzögerung

Nächste Lektion:

- Paketvermittlung, LAN-Switching