9.Vorlesung Grundlagen der Informatik

Dr. Christian Baun

Hochschule Darmstadt
Fachbereich Informatik
christian.baun@h-da.de

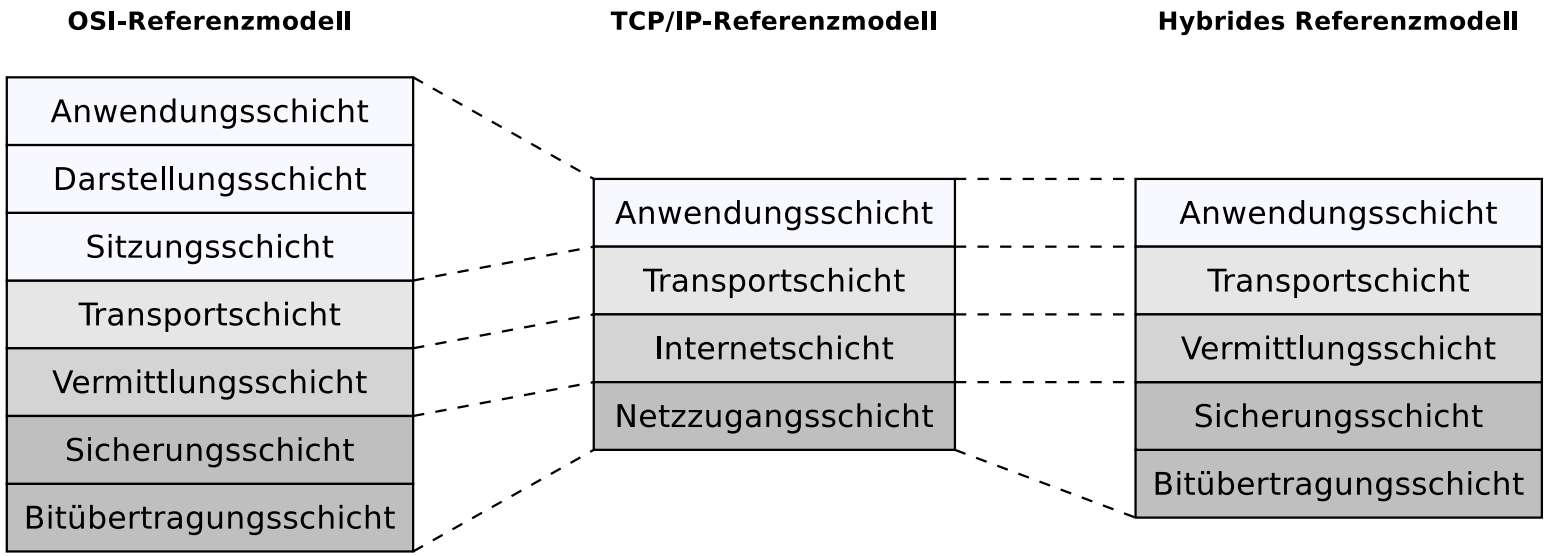
8.12.2011

Wiederholung vom letzten Mal

- Grundlagen der Computervernetzung
  - Netzwerkdienste und Rollen
  - Übertragungsmedien
  - Einteilung der Netzwerke
  - Formen der Datenübertragung
  - Richtungsabhängigkeit der Datenübertragung
  - Topologien von Computernetzwerken
  - Frequenz
  - Datensignal
  - Fourierreihe
  - Bandbreite
  - Zugriffsverfahren
- Kommunikation in Netzwerken
  - Protokolle und Protokollschichten
  - TCP/IP-Referenzmodell
  - Hybrides Referenzmodell
  - OSI-Referenzmodell

Heute

- Bitübertragungsschicht
  - Kodierung von Daten
- Sicherungsschicht
  - Adressierung (MAC-Adressen)
  - Fehlererkennung (Zweidimensionale Parität, Zyklische Redundanzprüfung)
- Vermittlungsschicht
  - Link-State-Routing-Protokoll (Dijkstra-Algorithmus)



Kodierung von Daten

- Die effiziente Kodierung von Daten ist nicht erst seit dem Aufkommen von Computernetzwerken wichtig
- Beispiel: Morsekode bzw. Morsealphabet nach Samuel Morse von 1838

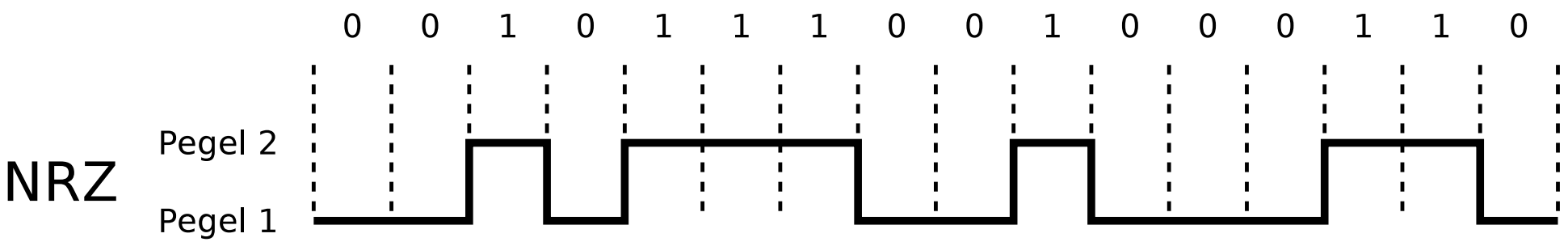
A	· —	M	— —	Y	— · — —
B	— · · ·	N	— ·	Z	— — · ·
C	— · — ·	O	— — —	1	· — — — —
D	— · ·	P	· — — ·	2	· · — — —
E	·	Q	— — · —	3	· · · — —
F	· · — ·	R	· — ·	4	· · · · —
G	— — ·	S	· · ·	5	· · · · ·
H	· · · ·	T	—	6	— · · · ·
I	· ·	U	· · —	7	— — · · ·
J	· — — —	V	· · · —	8	— — — · ·
K	— · —	W	· — —	9	— — — — ·
L	· — · ·	X	— · · —	0	— — — — —

Kodierung von Daten in Netzwerken

- In Netzwerken sind folgende Aktionen notwendig
  1. Umwandlung von Binärdaten bzw. Binärzahlen in Signale (Kodierung)
  2. Übertragung der Signale vom Sender zum Empfänger
  3. Rückwandlung der Signale in Bits (Dekodierung)
- Die Kodierung von Bits in Signale ist auf verschiedene Arten möglich
  - **Non-Return to Zero** (NRZ) und **Non-Return to Zero Invert** (NRZI)
  - **Return-to-Zero** (RZ-Kodierung)
  - **Manchesterkodierung**
  - **4B5B, 6B6B, 8B10B**
- Die einfachste Form der Darstellung von logischer 0 und 1 ist mit verschiedenen Spannungsniveaus möglich
  - Es könnte z.B. eine 0 durch 0 Volt und eine 1 durch 5 Volt kodiert werden
  - Diese Kodierung bezeichnet man als Non-Return to Zero (NRZ)

Non-Return to Zero (NRZ)

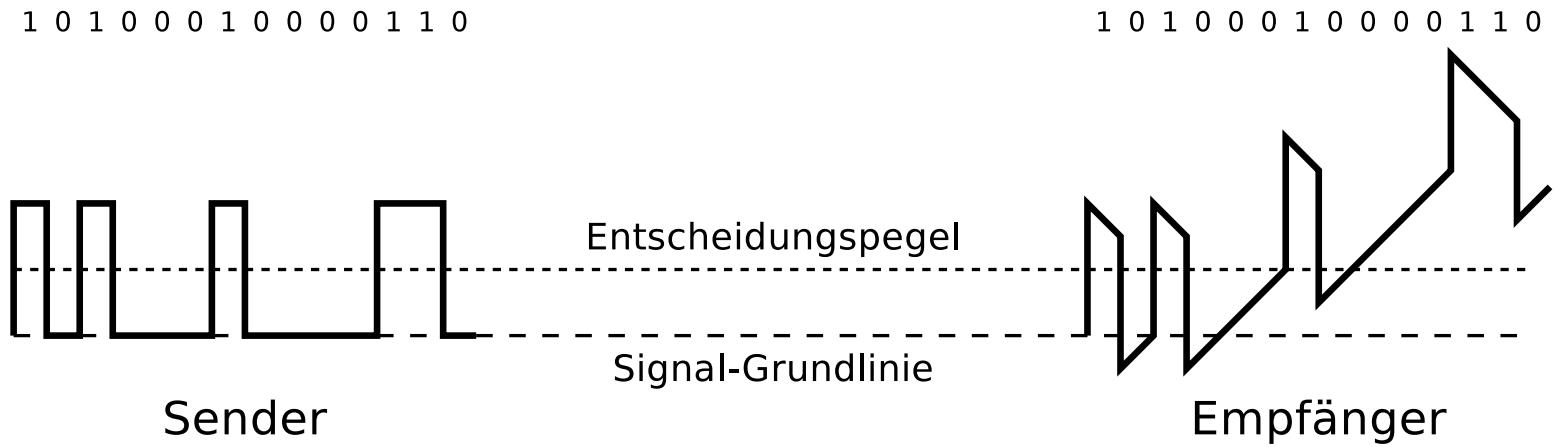
- Wird auch Non-Return to Zero-Level (NRZ-L) genannt
- Vorgehensweise
  - Datenwert 0 ist low-Signal (Pegel 1)
  - Datenwert 1 ist high-Signal (Pegel 2)



- Beim Übertragen einer längeren Serie von Nullen oder einer Serie von Einsen gibt es keine Pegeländerung
- Das führt zu 2 Problemen
  1. Verschiebung des Durchschnitts (Baseline Wander)
  2. Taktwiederherstellung (Clock Recovery)

Non-Return to Zero (NRZ) – Baseline Wander

- Problem: Verschiebung des Durchschnitts (**Baseline Wander**) bei NRZ
- Empfänger unterscheidet low- und high-Signale anhand des Durchschnitts einer bestimmten Anzahl zuletzt empfangener Signale
  - Signale weit unter dem Durchschnitt, interpretiert der Empfänger als 0
  - Signale deutlich über dem Durchschnitt, interpretiert der Empfänger als 1
- Beim Übertragen längerer Serien von Nullen oder Einsen kann sich der Durchschnitt soweit verschieben, dass es schwierig wird eine signifikante Änderung im Signal zu erkennen



Quelle: Handbuch Netzwerktechnologien. Cisco Systems (2001)

Non-Return to Zero (NRZ) – Taktwiederherstellung

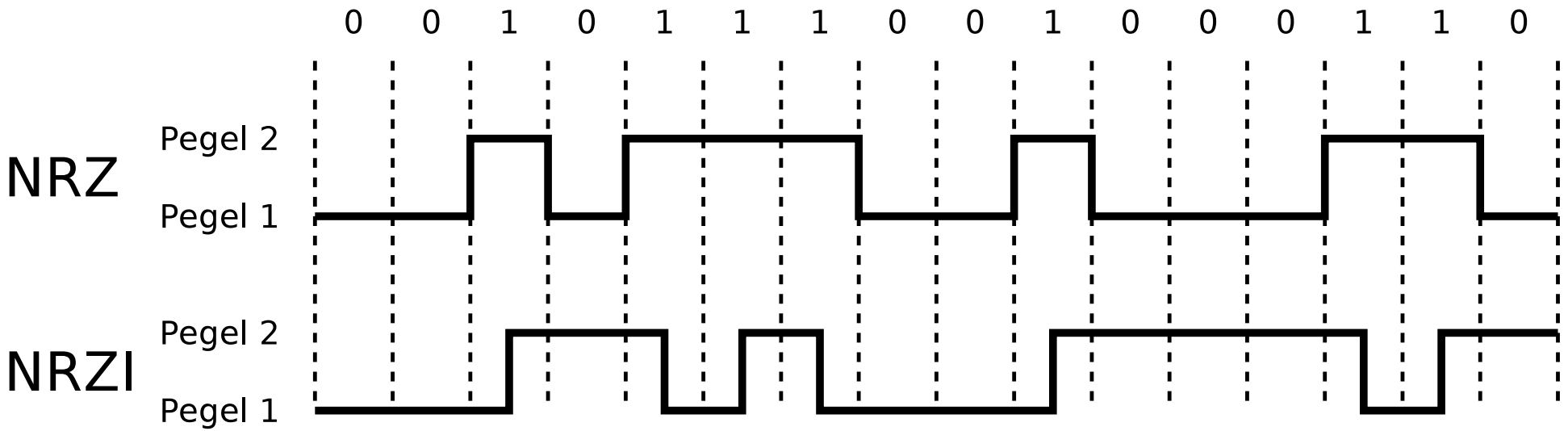
- Problem: **Taktwiederherstellung (Clock Recovery)** bei NRZ
- Die Prozesse für Kodierung und Dekodierung meist auf unterschiedlichen Rechnern, müssen aber vom gleichen Takt gesteuert werden

Man kann sich den lokalen Takt als internes Signal vorstellen, das von low nach high wechselt. Ein low/high-Paar ist ein Taktzyklus.

- In jedem Taktzyklus überträgt der Sender ein Bit und der Empfänger empfängt ein Bit
- Driften die Uhren von Sender und Empfänger auseinander, könnte sich der Empfänger bei einer langen Folge von Nullen oder Einsen verzählen
- Theoretische Lösungsmöglichkeit: Eine getrennte Leitung zum Empfänger, die den Takt überträgt
  - Nicht praktikabel, da sich so der Verkabelungsaufwand verdoppelt
- Lösungsmöglichkeit: NRZI-Kodierung

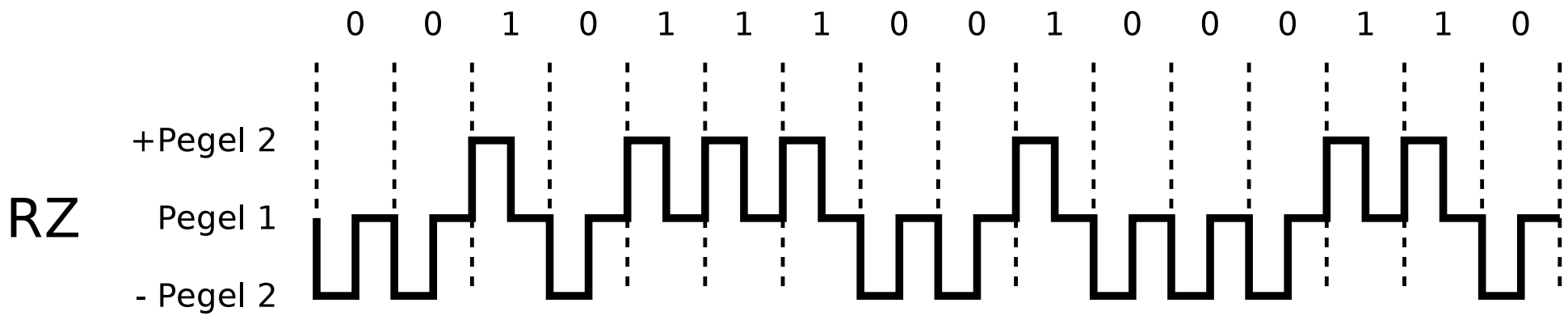
Non-Return to Zero Invert (NRZI)

- Variante von Non-Return to Zero (NRZ)
- Um eine 1 zu senden, findet zu Beginn des Takts ein Pegelsprung statt
- Um eine 0 zu senden, bleibt der Pegel einen ganzen Takt unverändert
- Das Problem aufeinanderfolgender Einsen ist gelöst
- Das Problem aufeinanderfolgender Nullen besteht immer noch



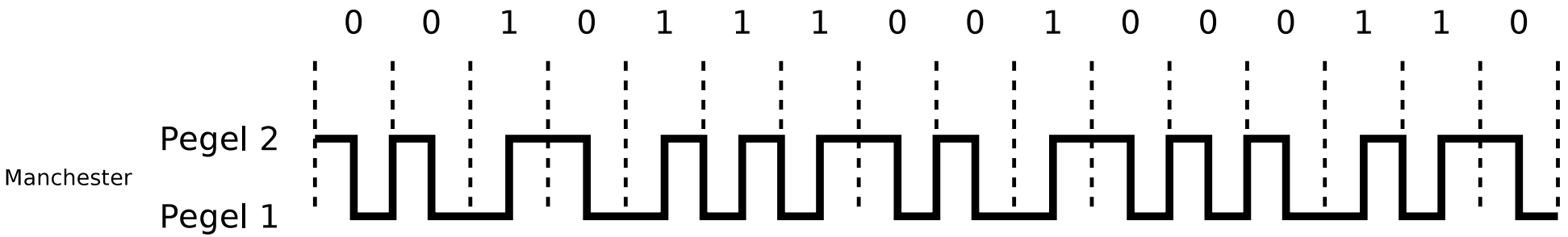
Return-to-Zero (RZ-Kodierung)

- Return-to-Zero $\Longrightarrow$ *Rückkehr zur Null*
- Weiterentwicklung der NRZ-Kodierung
- Verwendet 3 Pegelwerte (Spannungsniveaus): +1, 0 und -1
- Beim Senden einer 1 kehrt man nach dem halben Takt zum Pegel 0 zurück
- Beim Senden einer 0 wird der Pegel -1 für einen halben Takt übertragen und danach zum Pegel 0 zurückgekehrt
- Vorteil: Es gibt bei jedem Bit eine Pegeländerung
  - Ermöglicht dem Empfänger die Taktrückgewinnung (Synchronisierung)
- Nachteil: Doppelt so hohe Bandbreite gegenüber NRZ-Kodierung nötig



Manchesterkodierung

- Arbeitet mit 2 Pegeln (0 und 1)
- Selbstsynchronisierend
- Innerhalb jeder Bitzelle kommt es zum Pegelwechsel
  - Eine Bitzelle ist der für die Übertragung eines Bits reservierte Zeitraum
- Sprung von 1 nach 0 (fallende Flanke) entspricht einer 0
- Sprung von 0 nach 1 (steigende Flanke) entspricht einer 1
- Folgen 2 identische Bits aufeinander, wird am Ende der Bitzelle auf das Anfangsniveau zurückgesprungen
- Wird bei 10-MBit/s Ethernet verwendet



Manchesterkodierung

- Der Beginn einer Übertragung (also die erste Bitzelle) wird durch eine spezielle Bitfolge (die Präambel) gekennzeichnet
- Es gibt stets Pegelwechsel zur Taktrückgewinnung
  $\Longrightarrow$ Synchronisierung ist problemlos möglich
- Der Durchschnitt kann sich nicht verschieben
  $\Longrightarrow$ Baseline Wander ist kein Problem
- Nachteil: 1 Bit übertragen erfordert im Schnitt 1,5 Pegelwechsel

Da die Anzahl der Pegelwechsel ein limitierender Faktor des Übertragungsmediums ist, zieht man andere Kodierungen der Manchesterkodierung vor

- Bei Manchesterkodierung ist die Bitrate die Hälfte der Baudrate
  - Die Effizienz der Kodierung ist somit nur 50% im Vergleich zu NRZ

Bitrate und Baudrate

- **Bitrate**: Anzahl der Nutzdaten (in Bits) pro Zeit
- **Baudrate**: Rate, in der sich Signale ändern pro Zeit

Zusammenfassung

- Alle bislang vorgestellten Kodierungen haben Nachteile
  1. Verschiebung des Durchschnitts (Baseline Wander)
     - Problem aufeinanderfolgender Nullen und Einsen bei NRZ
     - Problem aufeinanderfolgender Nullen bei NRZI
  2. Taktwiederherstellung (Clock Recovery)
     - Schlecht bei NRZ
     - Nur teilweise gelöst bei NRZI
  3. Mangelhafte Effizienz
     - Bei der (differentiellen) Manchesterkodierung und bei RZ ist bei jedem Bit ist eine Pegeländerung (Taktwiederherstellung) garantiert, aber dafür ist die Effizienz schlecht
- Lösungsmöglichkeit: Die Eingabe in einer Form kodieren, die Effizienz verspricht, Taktwiederherstellung garantiert und die Verschiebung des Durchschnitts vermeidet
  $\Longrightarrow$ 4B5B, 5B6B, B8B10...
  - Danach ist eine Kodierung mit NRZ oder NRZI problemlos

4B5B-Kodierung

- 4 Nutzdatenbits werden auf 5 Codebits abgebildet
- Wird u.a. bei Fast Ethernet 100BASE-TX und Glasfaserverbindungen nach dem FDDI-Standard verwendet
- Wegen des zusätzlichen Bits zur Kodierung wird die kodierte Bitrate um den Faktor 5/4 gegenüber der Nutzdatenbitrate gesteigert
  - Die Effizienz der 4B5B-Kodierung ist 80%
- Mit 5 Bits sind 32 Kodierungen möglich
  - Nur 16 Kodierungen werden für Daten verwendet (0–9 und A–F)
  - Die Übrigen 16 Kodierungen werden teilweise für Steuerzwecke verwendet
- Jede 5-Bit-Kodierung hat maximal eine führende Null
  - Es gibt höchstens 3 Nullen in Folge
  - Übertragung der Bits erfolgt mittels NRZI-Kodierung
    - Bei NRZI ist das Problem der aufeinanderfolgenden Einsen schon gelöst
    - Darum kümmert sich 4B5B nur um das Problem aufeinanderfolgender Nullen

4B5B-Kodierung (Tabelle)

Bezeichnung	4B	5B	Funktion
0	0000	11110	0 Hexadezimal (Nutzdaten)
1	0001	01001	1 Hexadezimal (Nutzdaten)
2	0010	10100	2 Hexadezimal (Nutzdaten)
3	0011	10101	3 Hexadezimal (Nutzdaten)
4	0100	01010	4 Hexadezimal (Nutzdaten)
5	0101	01011	5 Hexadezimal (Nutzdaten)
6	0110	01110	6 Hexadezimal (Nutzdaten)
7	0111	01111	7 Hexadezimal (Nutzdaten)
8	1000	10010	8 Hexadezimal (Nutzdaten)
9	1001	10011	9 Hexadezimal (Nutzdaten)
A	1010	10110	A Hexadezimal (Nutzdaten)
B	1011	10111	B Hexadezimal (Nutzdaten)
C	1100	11010	C Hexadezimal (Nutzdaten)
D	1101	11011	D Hexadezimal (Nutzdaten)
E	1110	11100	E Hexadezimal (Nutzdaten)
F	1111	11101	F Hexadezimal (Nutzdaten)
Q	—	00000	Quiet (Leitung ist tot) $\Longrightarrow$ Signalverlust
I	—	11111	Idle (Leitung ist untätig) $\Longrightarrow$ Pause
J	—	11000	Start (Teil 1)
K	—	10001	Start (Teil 2)
T	—	01101	Ende (Teil 1)
R	—	00111	Ende (Teil 2) $\Longrightarrow$ Reset
S	—	11001	Set
H	—	00100	Halt (Übertragungsfehler)

- Die in der Tabelle fehlenden 5-Bit-Kombinationen sind ungültig, da sie mehr als eine führende oder zwei aufeinanderfolgende Nullen besitzen
- Bei Fast Ethernet 100BASE-TX beginnt ein Datenrahmen mit einem JK und endet mit einem TR

5B6B-Kodierung

- 5 Nutzdatenbits werden auf 6 Codebits abgebildet
- Wird u.a. bei Fast Ethernet 100Base-VG verwendet
- Von den 32 möglichen 5-Bit-Wörtern haben 20 die identische Anzahl an Einsen und Nullen $\Longrightarrow$ neutrale Ungleichheit
- Für die verbleibenden zwölf 5-Bit-Wörter existiert je eine Variante mit 2 Einsen und 4 Nullen und eine mit 4 Einsen und 2 Nullen
  $\Longrightarrow$ positive oder negative Ungleichheit
- Sobald bei der Kodierung das erste 6-Bit-Wort ohne neutrale Ungleichheit verarbeitet werden soll, wird auf die Variante mit der positiven Ungleichheit zurückgegriffen
  - Bei der Kodierung des nächsten 6-Bit-Worts ohne neutrale Ungleichheit wird auf die Variante mit der negativen Ungleichheit zurückgegriffen
  - Die Varianten mit positiver oder negativer Ungleichheit wechseln sich in der Folge ab
- Vorteil: Höhere Baudrate gegenüber 4B5B
- Übertragung der Bits erfolgt mittels NRZ-Kodierung

5B6B-Kodierung (Tabelle)

5B	6B neutral	6B positiv	6B negativ	5B	6B neutral	6B positiv	6B negativ
00000		001100	110011	10000		000101	111010
00001	101100			10001	100101		
00010		100010	101110	10010		001001	110110
00011	001101			10011	010110		
00100		001010	110101	10100	111000		
00101	010101			10101		011000	100111
00110	001110			10110	011001		
00111	001011			10111		100001	011110
01000	000111			11000	110001		
01001	100011			11001	101010		
01010	100110			11010		010100	101011
01011		000110	111001	11011	110100		
01100		101000	010111	11100	011100		
01101	011010			11101	010011		
01110		100100	011011	11110		010010	101101
01111	101001			11111	110010		

8B10B-Kodierung

- 8-Bit-Blöcke werden in 10-Bit-Blöcke umgewandelt
  - Große Ähnlichkeit zur 4B5B-Kodierung. 80% Effizienz
- Wird u.a. bei Gigabit-Ethernet 1000Base-CX, -SX, -LX, FibreChannel, InfiniBand, DisplayPort und USB 3.0 verwendet
- Jede 8B10B-Kodierung ist derart aufgebaut, dass entweder...
  - 5 mal die 0 und 5 mal die 1 vorkommt $\Longrightarrow$ neutrale Ungleichheit
  - 4 mal die 0 und 6 mal die 1 vorkommt $\Longrightarrow$ positive Ungleichheit
  - 6 mal die 0 und 4 mal die 1 vorkommt $\Longrightarrow$ negative Ungleichheit
- Baseline Wander ist kein Problem
  - Einige der 256 möglichen 8-Bit-Wörter können auf zwei verschiedene Arten kodiert werden
  - So können vorherige Ungleichheiten ausgeglichen werden
- Jede 8-Bit-Kodierung enthält mindestens 3 Pegelsprünge auf und nach spätestens fünf Takten wechselt der Pegel
  - Damit wird beim Empfänger die Taktwiederherstellung erleichtert
- Übertragung der Bits erfolgt mittels NRZ-Kodierung

Zusammenfassung

Kodierung	Pegel-werte	Pegel-wechsel	Gleich-verteilt	Selbstsynchro-nisierend	Effizienz	Direkt Übertragbar
NRZ	2	bei Wechseln	nein	nein		ja
NRZI	2	bei 1	nein	nein		ja
RZ	3	immer	nein	ja		ja
Manchester	2	immer	nein	ja	50%	ja
4B5B	2	—	nein	ja	80%	nach NRZI
5B6B	2	—	ja	ja	80%	nach NRZ
8B10B	2	—	ja	ja	80%	nach NRZ

Adressierung auf der Sicherungsschicht

- Jeder Knoten (Hosts und Router) hat eine Adresse auf der Ebene der Sicherungsschicht
  - Bei den Adressen handelt es sich um die **MAC-Adressen**
    - MAC = Media Access Control
- Die MAC-Adressen der Sicherungsschicht sind nicht zu verwechseln mit den (IP-)Adressen der Vermittlungsschicht
- Mit dem Protokoll **Address Resolution Protocol** (ARP) kann man IPv4-Adressen in MAC-Adressen übersetzen
  - Bei IPv6 wird das **Neighbor Discovery Protocol** (NDP) verwendet, dessen Funktionalität identisch ist und das ähnlich arbeitet

MAC-Adressen

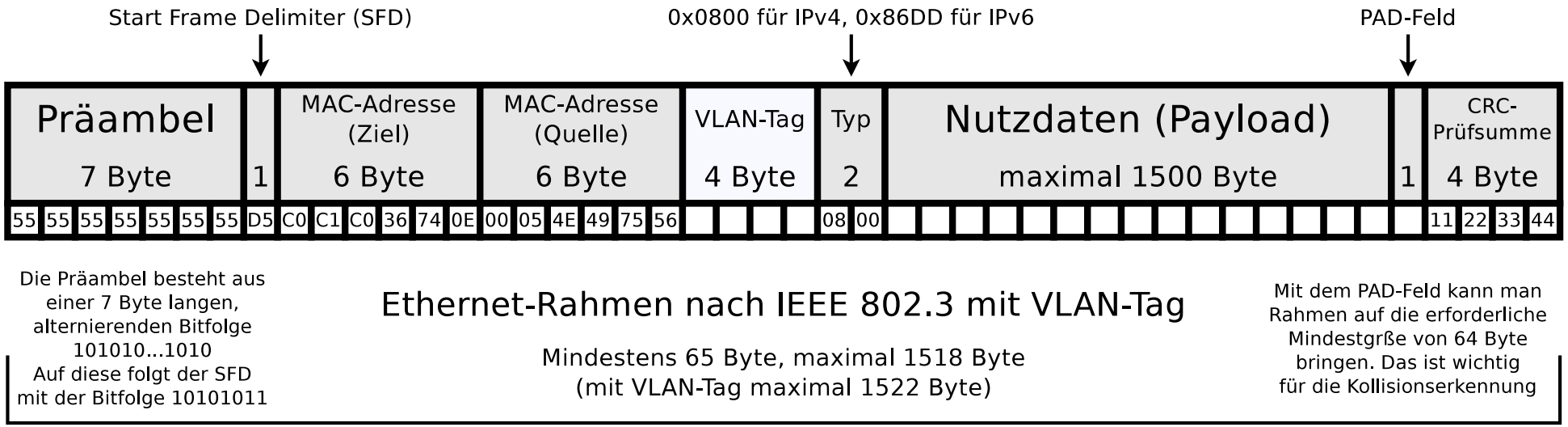
- Nennt man auch **physische Adressen**
- Länge: 48 Bit (6 Byte)
  - Damit sind 2^{48} Adressen möglich
- Werden normalerweise in hexadezimaler Schreibweise wiedergegeben
  - Üblich ist eine byteweise Schreibweise, wobei die einzelnen Bytes durch Bindestriche oder Doppelpunkte voneinander getrennt werden
  - Beispiel: `00:16:41:52:df:d7`
- Sollen dauerhaft einem Gerät zugewiesen sein und eindeutig sein
  - Es ist aber auch möglich, die MAC-Adresse softwaremäßig zu ändern

MAC-Broadcast-Adresse

- Möchte ein Gerät einen Rahmen explizit an alle anderen Geräte im LAN senden, fügt er im Rahmen in das Feld der Ziel-Adresse die MAC-Broadcast-Adresse ein
- Bei der Broadcast-Adresse haben alle 48 Bit den Wert 1
  - Hexadezimale Schreibweise: `FF-FF-FF-FF-FF-FF`
- Rahmen, die im Zielfeld die Broadcast-Adresse tragen, werden nicht in ein anderes LAN übertragen

Aufbau von Rahmen in der Sicherungsschicht

- Die Abbildung zeigt den Aufbau von Ethernet-Datenrahmen nach dem heute üblichen Standard IEEE 802.3 (mit 802.1Q VLAN-Tag)



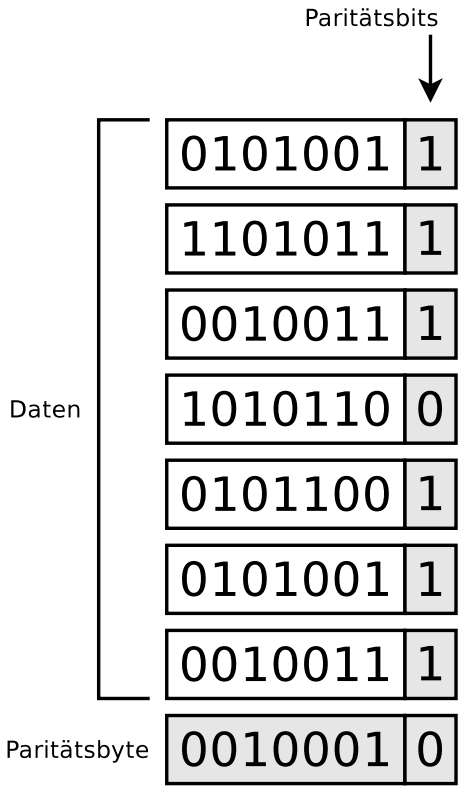
Geräte mit MAC-Adressen

- Alle Netzwerkgeräte, die auf Schicht 2 explizit adressiert werden sollen, brauchen eine MAC-Adresse
  - Leitet ein Netzwerkgerät (z.B. **Repeater** oder **Hub**) Netzwerkpakete nur weiter, ist es auf der Sicherungsschicht nicht sichtbar und braucht somit auch keine MAC-Adresse
- **Bridges** und **Switche** untersuchen die Pakete der Sicherungsschicht, um das Netzwerk physisch in mehrere Kollisionsdomänen aufzuteilen, nehmen aber selbst nicht aktiv an der Kommunikation teil
  - Sie brauchen also für die Basisfunktionalität keine MAC-Adresse
  - Ein Switch braucht dann eine MAC-Adresse, wenn er selbst über das Rechnernetz administriert wird oder Monitoring-Dienste anbietet (z.B. über HTTP)
  - Bridges und Switche brauchten auch dann eine MAC-Adresse, wenn sie den Spanning Tree Algorithmus zur Vermeidung von Schleifen in redundant ausgelegten Rechnernetzen verwenden

Fehlererkennung

- Zur Fehlererkennung beim Empfänger wird jedem Rahmen vom Sender eine Prüfsumme angefügt
  - So können fehlerhafte Rahmen vom Empfänger erkannt und entweder verworfen oder sogar korrigiert werden
  - Ein erneutes Anfordern verworfener Rahmen sieht die Sicherungsschicht nicht vor
- Möglichkeiten der Fehlererkennung:
  - Zweidimensionale Parität
  - Cyclic Redundancy Check (CRC)

Zweidimensionale Parität



- Eindimensionale Parität
  - Zu jedem 7-Bit-Abschnitt wird ein zusätzliches **Paritätsbit** addiert, um die Anzahl der Einsen im Byte auszugleichen
    - Gerade Parität setzt das achte Bit (Paritätsbit) bei Bedarf auf 1 oder 0, um eine gerade Anzahl von Einsen im Byte zu erwirken
    - Ungerade Parität setzt das achte Bit (Paritätsbit) bei Bedarf auf 1 oder 0, um eine ungerade Anzahl von Einsen im Byte zu erwirken
- Zweidimensionale Parität
  - Neben den Paritätsbits in jedem Byte existiert für jeden Rahmen noch ein zusätzliches **Paritätsbyte**
  - Der Inhalt des Paritätsbyte ist das Ergebnis der Paritätsberechnung quer über jedes Byte des Rahmens

Prüfsummenverfahren – Cyclic Redundancy Check (CRC)

- Zyklische Redundanzprüfung (CRC) basiert darauf, das man Bitfolgen als Polynome mit den Koeffizienten 0 und 1 schreiben kann
- Ein Rahmen mit k Bits wird als Polynom vom Grad $k-1$ betrachtet
  - Das werthöchste Bit ist der Koeffizient von x^{k-1}
  - Das nächste Bit ist der Koeffizient für x^{k-2}
  - ...
- Beispiel: Die Bitfolge 10011010 wird dargestellt als:

$$\begin{aligned} M(x) &= 1 * x^7 + 0 * x^6 + 0 * x^5 + 1 * x^4 + 1 * x^3 + 0 * x^2 + 1 * x^1 + 0 * x^0 \\ &= x^7 + x^4 + x^3 + x^1 \end{aligned}$$

- Das Senden und Empfangen von Nachrichten kann man sich als Austausch von Polynomen vorstellen

Polynome in der Mathematik

Ein Polynom ist eine Summe von Vielfachen von Potenzen mit natürlichzahligen Exponenten einer Variablen

Prüfsummenverfahren (Vorgehensweise)

- Sender und Empfänger müssen ein **Generatorpolynom** bzw. **Divisor-Polynom** $C(x)$ vereinbaren
- $C(x)$ ist ein Polynom vom Grad k
  - Sei z.B. $C(x) = x^3 + x^2 + 1 = 1101$, dann ist $k = 3$
  - Das Generatorpolynom ist also vom Grad 3

Der Grad des Generatorpolynoms entspricht der Anzahl der Bits minus eins

- Das Generatorpolynom wird vom Protokoll festgelegt
  - Die Auswahl des Generatorpolynoms legt fest, welche Fehlerarten erkannt werden
  - Es existieren einige Generatorpolynome, die sich für verschiedene Netzwerkumgebungen eignen
- Soll für einen Rahmen die CRC-Prüfsumme berechnet werden, werden n Nullen an den Rahmen angehängt
  - n entspricht dem Grad des Generatorpolynoms

Prüfsummenverfahren (Beispiel) – Quelle: Wikipedia

Generatorpolynom:	110101
Rahmen (Nutzdaten):	11011
Rahmen mit Anhang:	1101100000
Übertragener Rahmen (Codepolynom):	1101100101

- Das Generatorpolynom hat r Stellen, also werden $r - 1 = n$ Nullen ergänzt. Hier ist $r = 6$
- Der Rahmen mit Anhang wird von links nur mit XOR durch das Generatorpolynom dividiert
  - `1 XOR 1 = 0, 1 XOR 0 = 1, 0 XOR 1 = 1, 0 XOR 0 = 0`
  - Der Rest (Restpolynom) ist die **Prüfsumme**
- Die Prüfsumme wird an die Nutzdaten angehängt
  - Der Rest muss aus n Bits bestehen
  - n ist der Grad des Generatorpolynoms
- Ergebnis: 00101 wird an den Rahmen angehängt
- Übertragener Rahmen (Codepolynom): 1101100101

```
1101100000
110101||||
------vvvv
0000110000
    110101
    ------
       101 (Rest)
```

Prüfsummenverfahren (Beispiel) – Quelle: Wikipedia

Übertragener Rahmen (Codepolynom):	1101100101
Generatorpolynom:	110101

- Der Empfänger der Nachricht kann überprüfen, ob sie korrekt angekommen ist
- Mit Division (ausschließlich mit XOR) durch das Generatorpolynom werden fehlerhafte Übertragungen erkannt
  - Bei der Division mit XOR immer mit der ersten gemeinsamen 1 anfangen!
- Der Rest der Division ist gleich null
  - Also ist kein Fehler aufgetreten

```
1101100101
110101||||
------vvvv
    110101
    110101
    ------
    000000
```

Prüfsummenverfahren (Beispiel) – Quelle: Wikipedia

Übertragener Rahmen (Codepolynom):	1001100101
Generatorpolynom:	110101
Korrekte Übertragung:	1101100101

- Der Rest der Division ist ungleich null
  - Also ist ein Fehler während der Übertragung aufgetreten

```
1001100101
110101|||
------v|||
 100110|||
 110101|||
 ------v||
  100111||
  110101||
  ------v|
   100100|
   110101|
   ------v
    100011
    110101
    ------
     10110
```

Auswahl gängiger CRC-Generatorpolynome

Name	$C(x)$
CRC-5	$x^5 + x^2 + 1$
CRC-8	$x^8 + x^2 + x^1 + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + 1$
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-24	$x^{24} + x^{23} + x^{18} + x^{18} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x^1 + 1$
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
CRC-64	$x^{64} + x^4 + x^3 + x^1 + 1$

- USB nutzt CRC-5
- ISDN nutzt CRC-8
- Ethernet nutzt CRC-32
- X.25-kompatible WANs über das Telefonnetz nutzen CRC-CCITT
- ATM nutzt CRC-8, CRC-10 und CRC-32

Asynchronous Transfer Mode (ATM)

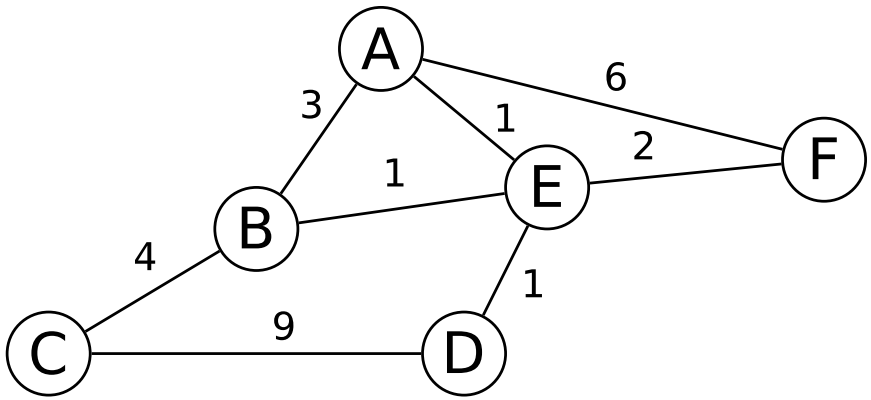
ATM ist eine Übertragungsart, bei der der Datenverkehr in kleine Pakete mit fester Länge kodiert wird. ATM wird in lokalen Hochleistungsnetzen und in Backbones genutzt.

Vermittlung und Weiterleitung

- Netzwerke wie das Internet mit globalen Ausmaßen benötigen die Weiterleitung der Pakete durch **Switche** (Vermittler)
  - **Ein Switch ist Router und Bridge in einem**
    - Er arbeitet auf der Sicherungsschicht und der Vermittlungsschicht
- Switch = Gerät mit mehreren Ein-/Ausgängen von/zu Hosts
  - Ein Switch ermöglicht die Sterntopologie
  - Ziel: Vergrößerung des Netzwerks
- Ein Switch ist für die **Weiterleitung (Forwarding)** zuständig
  - Dabei wird die Zieladresse eines Paketes in der lokalen Weiterleitungstabelle nachgesehen
  - Anschließend wird das Paket in die Richtung gesendet, die von der Weiterleitungstabelle vorgegeben ist
- Die Weiterleitungstabelle können nicht manuell gepflegt werden
- **Routing**...
  - ist der Prozess, bei dem die Weiterleitungstabellen erstellt werden
  - findet im Hintergrund statt
  - wird mit verteilten Algorithmen realisiert

Netzwerk als Graph

- Routing ist ein graphentheoretisches Problem
- Das Netzwerk wird als Graph aufgefasst
  - Knoten können Netzwerke, Hosts oder Switche sein
  - Kanten sind Netzverbindungen mit Kostengewichtung
- In unserem Beispiel sind die Knoten Switche
- Aufgabe beim Routing:
  - Finde den billigsten Weg von Quelle zum Ziel
  - Die Kosten eines Weges entsprechen der Summe der Kosten aller Kanten, aus denen sich der Weg zusammensetzt



Netzwerk als Graph

- Routing wird durch Routing-Protokolle realisiert, die zwischen den Knoten ausgeführt werden
- Diese Protokolle bieten eine dynamische Lösung für das Problem, den **Weg mit den niedrigsten Kosten zu ermitteln**
  - Der Ausfall einzelner Verbindungen oder Knoten und die Änderung von Kantenkosten werden dabei berücksichtigt
- Gute Routing-Protokolle sind **verteilte Algorithmen**
  - Grund: **Bessere Skalierbarkeit**
- Häufigstes Routing-Protokoll: **Link-State-Verfahren** (Dijkstra-Algorithmus)

Link-State-Routing-Protokoll

- Wird von Routern verwendet, um eine **lokale Datenbank** mit Topologie-Informationen aufzubauen
  - Mit Hilfe dieser Datenbank werden die Pakete im Netzwerk weitergeleitet
- Konzept:
  - Jeder Knoten kann den Zustand der Verbindung zu seinen Nachbarn und die Kosten dahin ermitteln
  - Die Information, die ein Knoten hat, wird **an alle anderen** verteilt
  - Jeder Knoten kann sich daraufhin eine komplette Übersicht über das Netzwerk erstellen
    - Komplexe Datenbank mit Topologie-Informationen
  - Regelmäßige Link-State-Aktualisierungen durch *Flooding* (Fluten)
- Arbeitsweise:
  - Zuverlässiges Verbreiten der Link-State-Informationen an alle Knoten
  - Berechnen von Routen aus der Summe der Daten
    - Shortest Path First
  - Link-State-Informationen haben eine Lebensdauer (TTL)
    - Alte Link-State-Informationen sollen nicht endlos im Netzwerk zirkulieren

Link-State-Routing-Protokoll

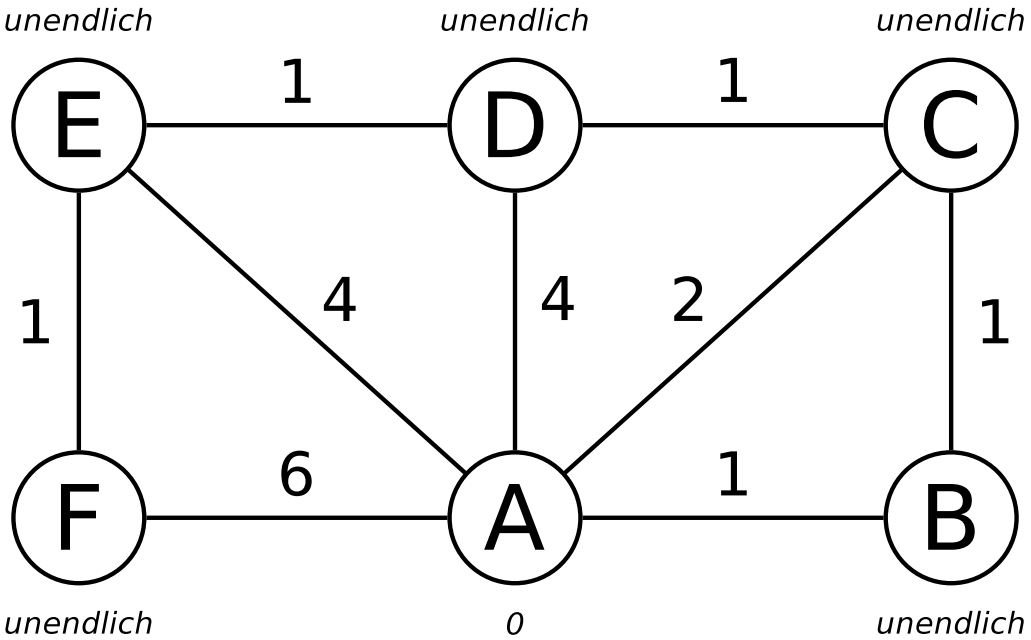
- Vorteil: Reagiert rasch auf Topologieänderungen und Knotenausfälle
- Nachteil: Skalierbarkeit
  - Alle Knoten speichern lokal Informationen über alle Knoten im Netzwerk
- Der Link-State-Algorithmus ist die praktische Umsetzung des **Dijkstra-Algorithmus**
- OSPF (Open Shortest Path First Protocol) ist das häufigst eingesetzte Link-State-Protokoll

Dijkstra-Algorithmus

- Berechnung des kürzesten Weges zwischen einem Startknoten und allen anderen Knoten in einem kantengewichteten Graphen
  - Kantengewichte dürfen nicht negativ sein
  - Ist man nur am Weg zu einem bestimmten Knoten interessiert, kann man in Schritt 2 abbrechen, wenn der gesuchte Knoten der aktive ist

1. Weise allen Knoten die Eigenschaften **Distanz** und **Vorgänger** zu
   - Initialisiere die Distanz im Startknoten mit 0 und in allen anderen Knoten mit ∞
2. Solange es noch nicht besuchte Knoten gibt, wähle darunter denjenigen mit minimaler Distanz aus
   - Speichere, dass dieser Knoten schon besucht wurde
   - Berechne für alle noch nicht besuchten Nachbarknoten die Summe des jeweiligen Kantengewichtes und der Distanz im aktuellen Knoten
   - Ist dieser Wert für einen Knoten kleiner als die dort gespeicherte Distanz, aktualisiere sie und setze den aktuellen Knoten als Vorgänger

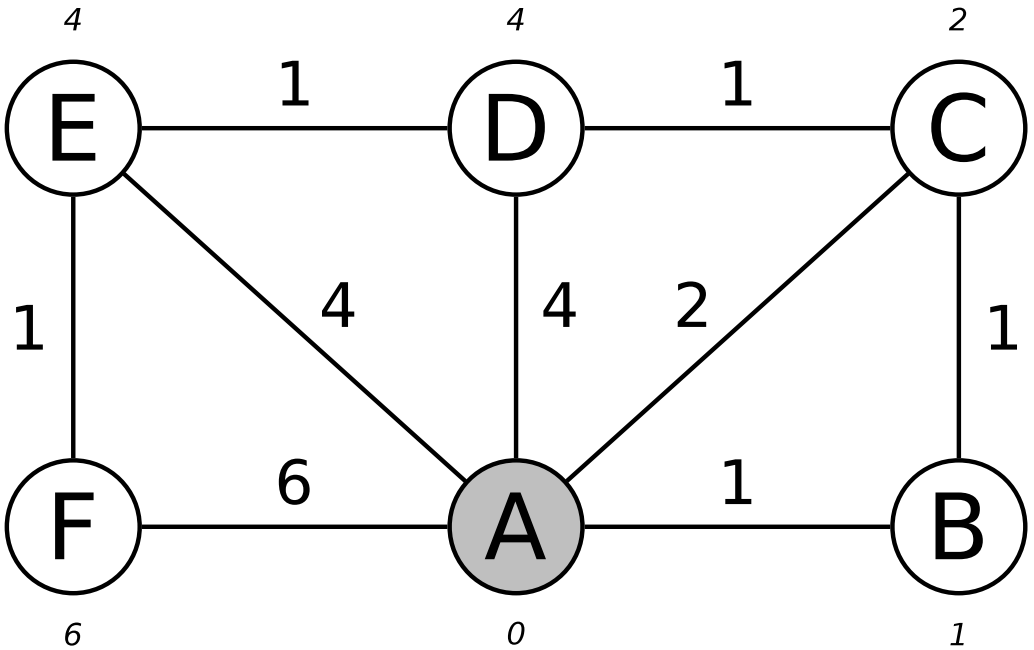
Dijkstra-Algorithmus – Beispiel 1



Distanzwerte	
$d_A = 0$	
$d_B = \infty$	
$d_C = \infty$	
$d_D = \infty$	
$d_E = \infty$	
$d_F = \infty$	

- Schritt 1: Initialisiere mit 0 und ∞
  - Sei A der Startknoten
  - A hat die minimale Distanz
- Besuchte Knoten $= \{\}$
- Quellbaum $= \{\}$

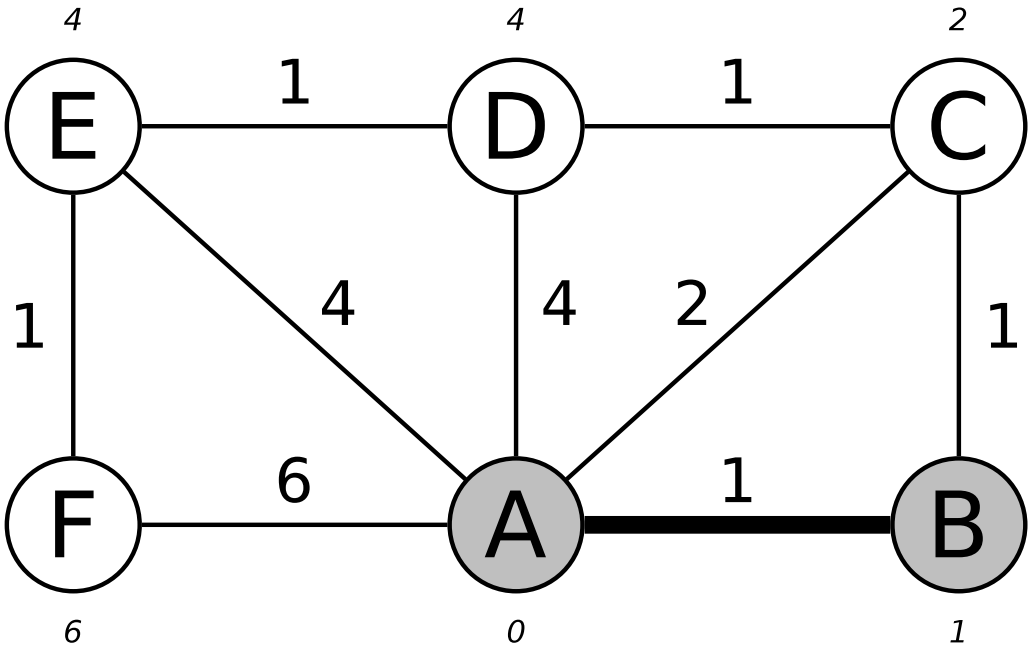
Dijkstra-Algorithmus – Beispiel 2



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	⟵ minimale Distanz
$d_C = 2$	
$d_D = 4$	
$d_E = 4$	
$d_F = 6$	

- Schritt 2: Summe der Kantengewichte berechnen
  - B hat die minimale Distanz
- Besuchte Knoten $= \{A\}$
- Quellbaum $= \{A\}$

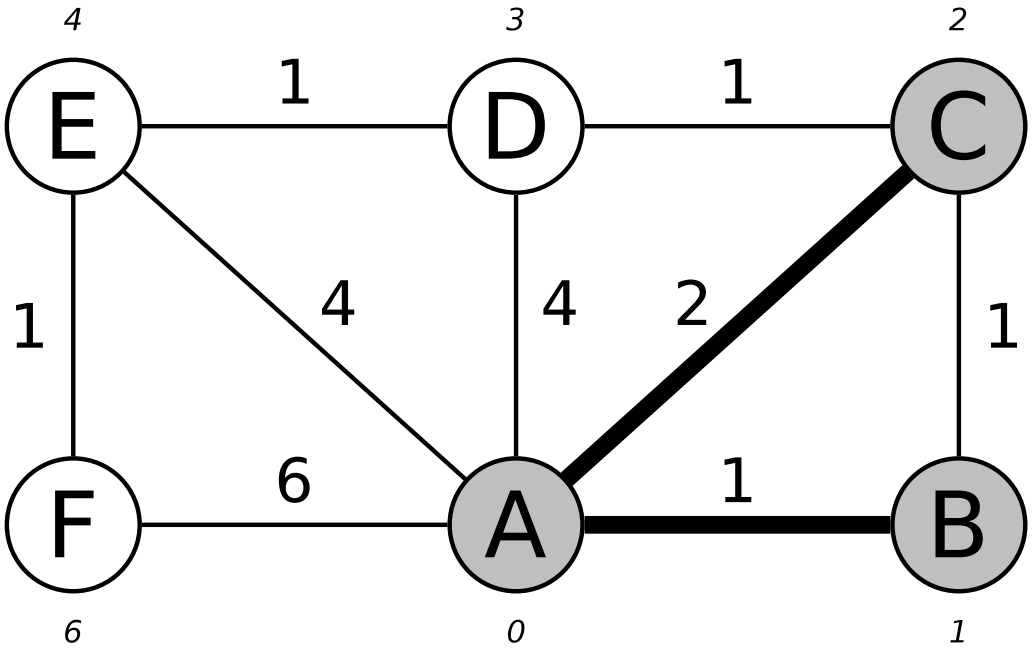
Dijkstra-Algorithmus – Beispiel 3



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	$\longleftarrow$ minimale Distanz
$d_D = 4$	
$d_E = 4$	
$d_F = 6$	

- Schritt 3: Knoten B besuchen
  - Keine Veränderung zu C
  - C hat die minimale Distanz
- Besuchte Knoten = {A, B}
- Quellbaum = {A, A$\longrightarrow$B}

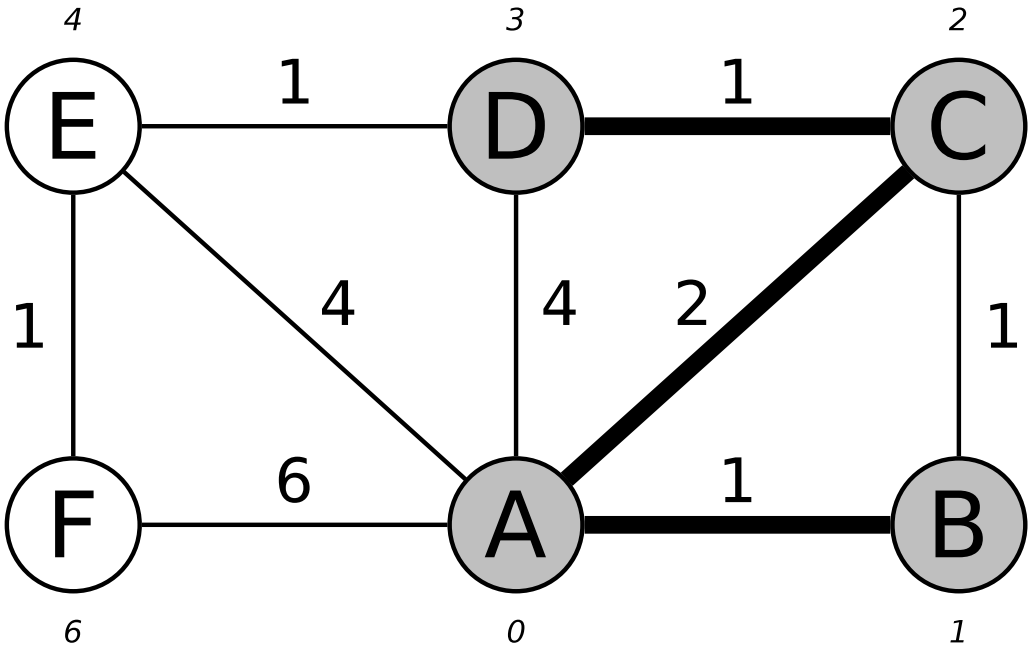
Dijkstra-Algorithmus – Beispiel 4



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = \mathbf{3}$	$\longleftarrow$ minimale Distanz
$d_E = 4$	
$d_F = 6$	

- Schritt 4: Knoten C besuchen
  - Keine Veränderung zu B
  - Veränderung zu D (Weg über C ist kürzer als der direkte Weg)
  - D hat die minimale Distanz
- Besuchte Knoten = {A, B, C}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C}

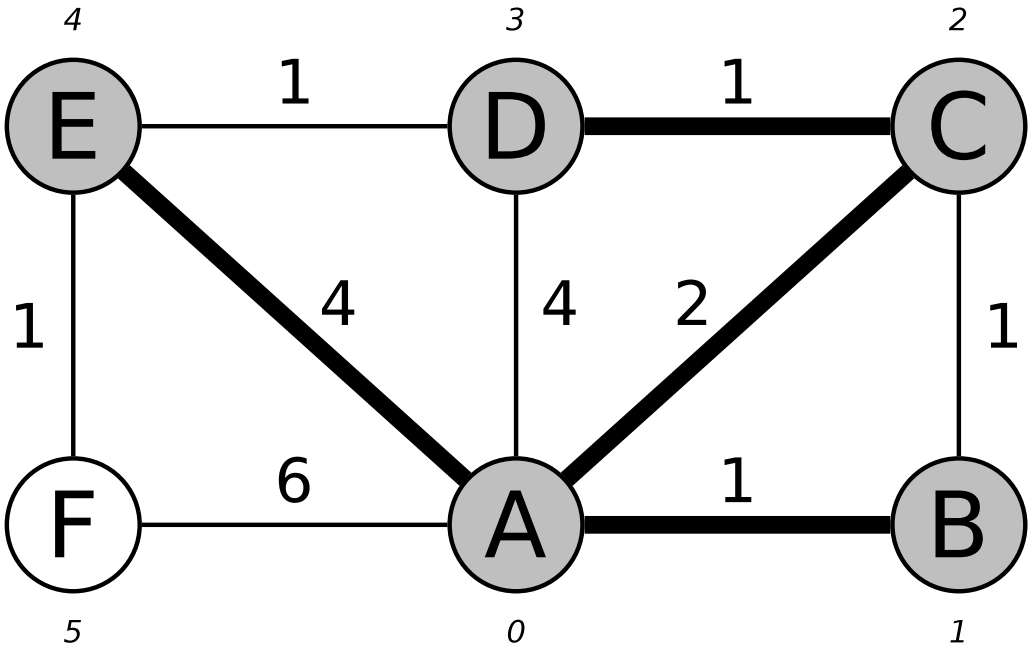
Dijkstra-Algorithmus – Beispiel 5



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = 3$	besucht
$d_E = 4$	$\longleftarrow$ minimale Distanz
$d_F = 6$	

- Schritt 5: Knoten D besuchen
  - Keine Veränderung zu C
  - Keine Veränderung zu E
  - E hat die minimale Distanz
- Besuchte Knoten = {A, B, C, D}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C, C$\longrightarrow$D}

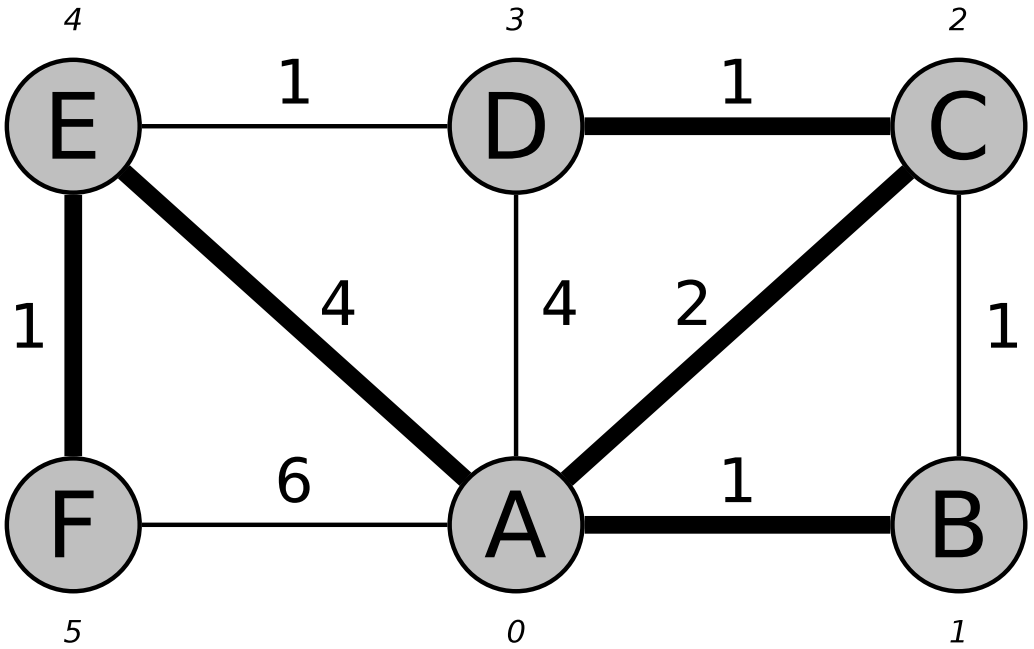
Dijkstra-Algorithmus – Beispiel 6



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = 3$	besucht
$d_E = 4$	besucht
$d_F = \mathbf{5}$	$\longleftarrow$ minimale Distanz

- Schritt 6: Knoten E besuchen
  - Keine Veränderung zu D
  - Veränderung zu F (Weg über E ist kürzer als der direkte Weg)
  - F hat die minimale Distanz
- Besuchte Knoten = {A, B, C, D, E}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C, C$\longrightarrow$D, A$\longrightarrow$E}

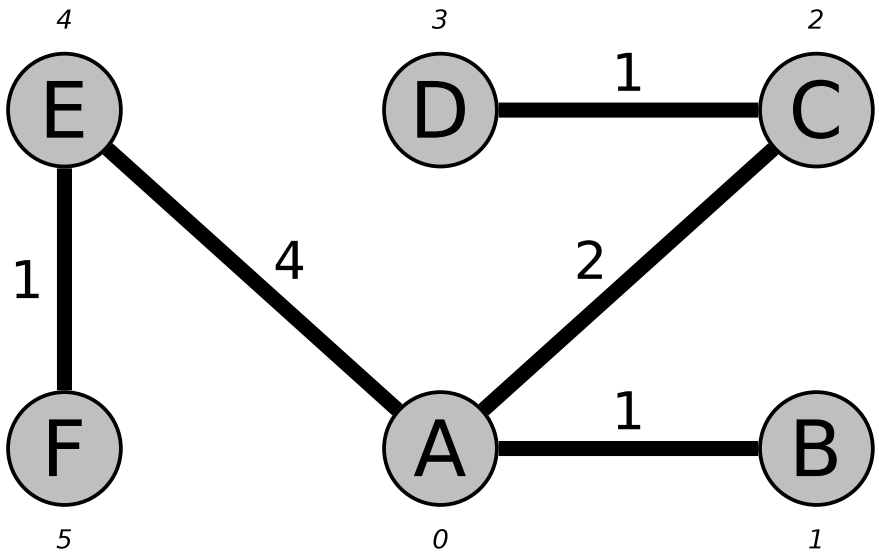
Dijkstra-Algorithmus – Beispiel 7



Distanzwerte	
$d_A = 0$	besucht
$d_B = 1$	besucht
$d_C = 2$	besucht
$d_D = 3$	besucht
$d_E = 4$	besucht
$d_F = 5$	besucht

- Schritt 7: Knoten F besuchen
  - Keine Veränderung zu E
- Besuchte Knoten = {A, B, C, D, E, F}
- Quellbaum = {A, A$\longrightarrow$B, A$\longrightarrow$C, C$\longrightarrow$D, A$\longrightarrow$E, E$\longrightarrow$F}

Dijkstra-Algorithmus – Beispiel (Ergebnis)



- Ergebnis

Nächste Vorlesung

Nächste Vorlesung:

15.12.2011