

Systeme II

3. Die Datensicherungsschicht

Christian Schindelhauer

Technische Fakultät

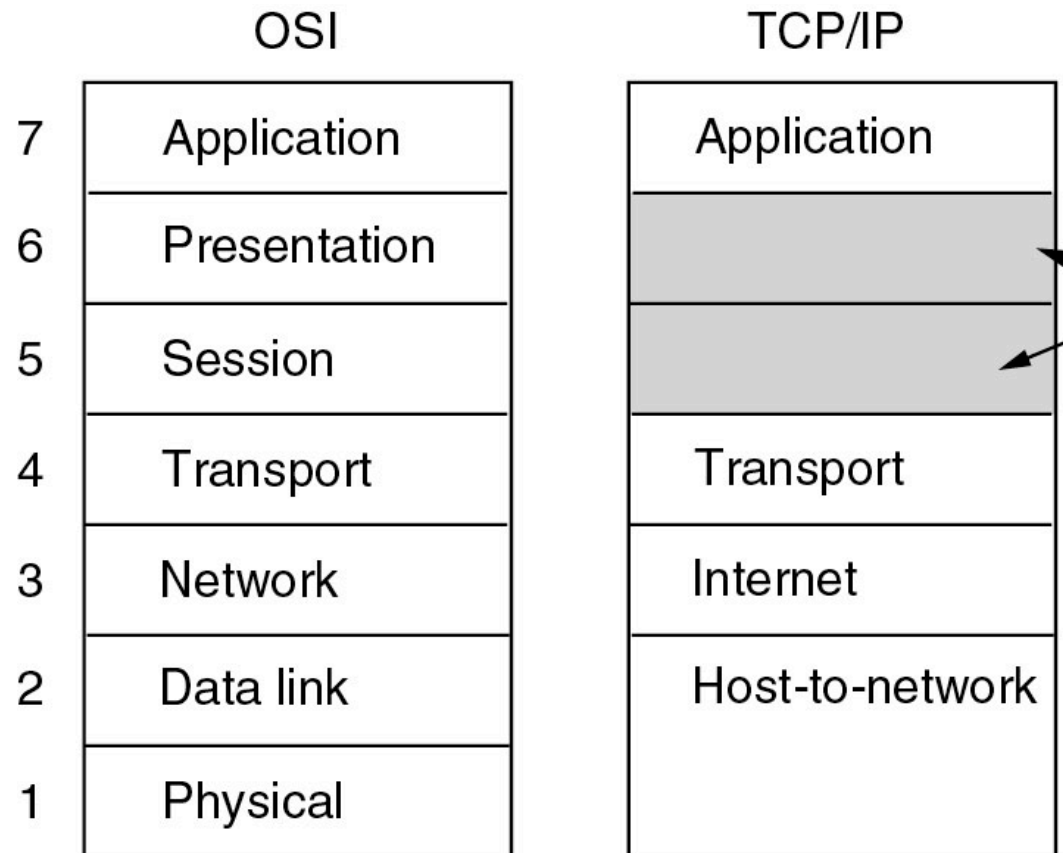
Rechnernetze und Telematik

Albert-Ludwigs-Universität Freiburg

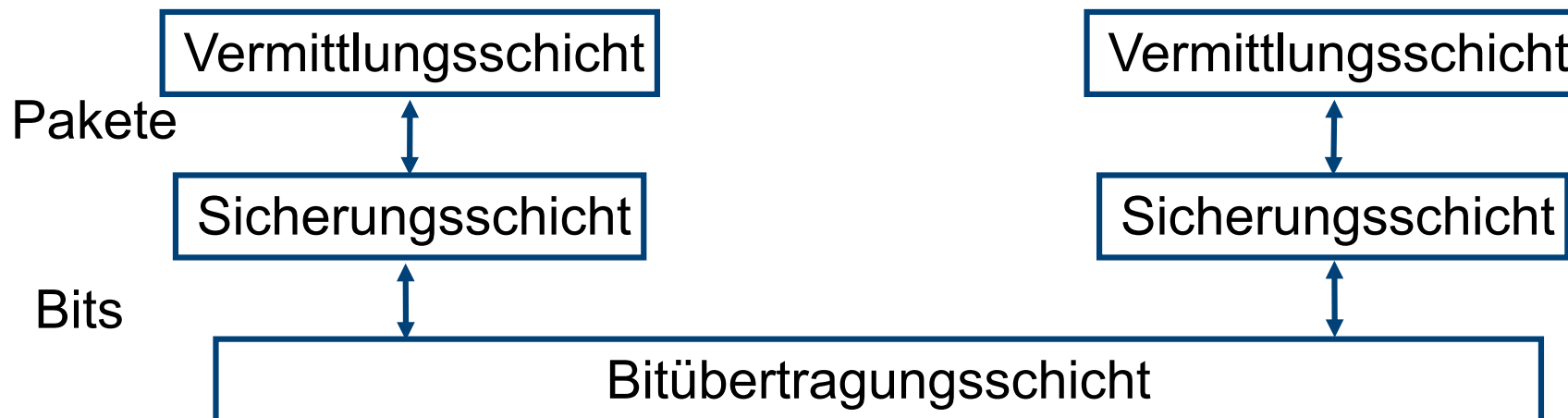
Version 06.05.2013

- Aufgaben der Sicherungsschicht (Data Link Layer)

- Dienste für die Vermittlungsschicht
- Frames
- Fehlerkontrolle
- Flusskontrolle



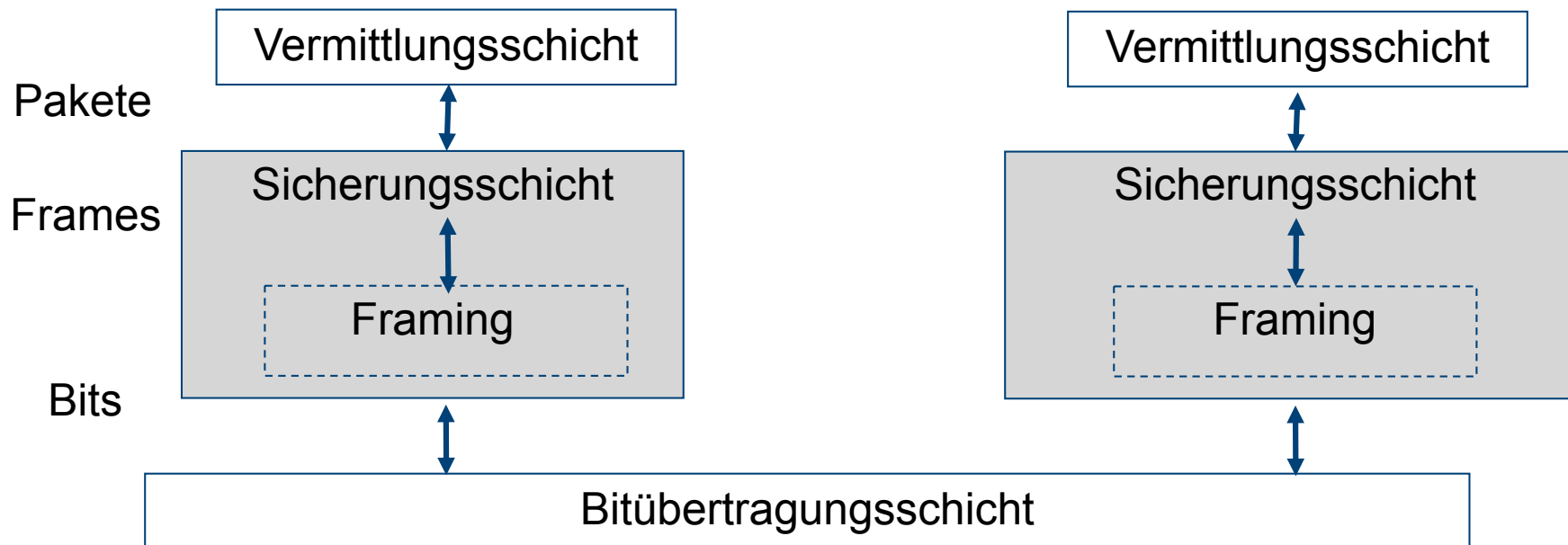
- Situation der Sicherungsschicht
 - Die Bitübertragungsschicht überträgt Bits
 - Aber unstrukturiert und möglicherweise fehlerbehaftet
- Die Vermittlungsschicht erwartet von der Sicherungsschicht
 - Fehlerfreie Übermittlung
 - Übermittlung von strukturierten Daten
 - Datenpakete oder Datenströme
 - Störungslosen Datenfluss



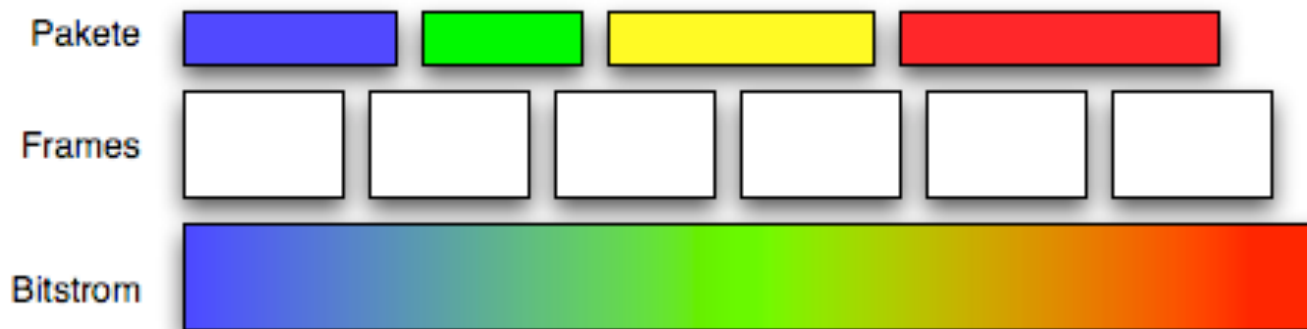
- **Verlässlicher Dienst?**
 - Das ausgelieferte und das empfangene Paket müssen identisch sein
 - Alle Pakete sollen (irgendwann) ankommen
 - Pakete sollen in der richtigen Reihenfolge ankommen
 - Fehlerkontrolle ist möglicherweise notwendig
- **Verbindungsorientiert?**
 - Ist die Punkt-zu-Punktverbindung in einem größerem Kontext?
 - Reservierung der Verbindung notwendig?
- **Pakete oder Datenströme (Bitströme)?**

- Beispiel
 - Verbindungsloser und verlässlicher Dienst wird durch die Vermittlungsschicht gefordert
 - Sicherungsschicht verwendet intern verbindungsorientierten Dienst mit Fehlerkontrolle
- Andere Kombinationen sind möglich

- Der Bitstrom der Bitübertragungsschicht wird in kleinere “Frames” unterteilt
 - Notwendig zur Fehlerkontrolle
 - Frames sind Pakete der Sicherungsschicht
- Frame-Unterteilung (Fragmentierung) und Defragmentierung sind notwendig
 - Falls die Pakete der Vermittlungsschicht größer sind als die Frames



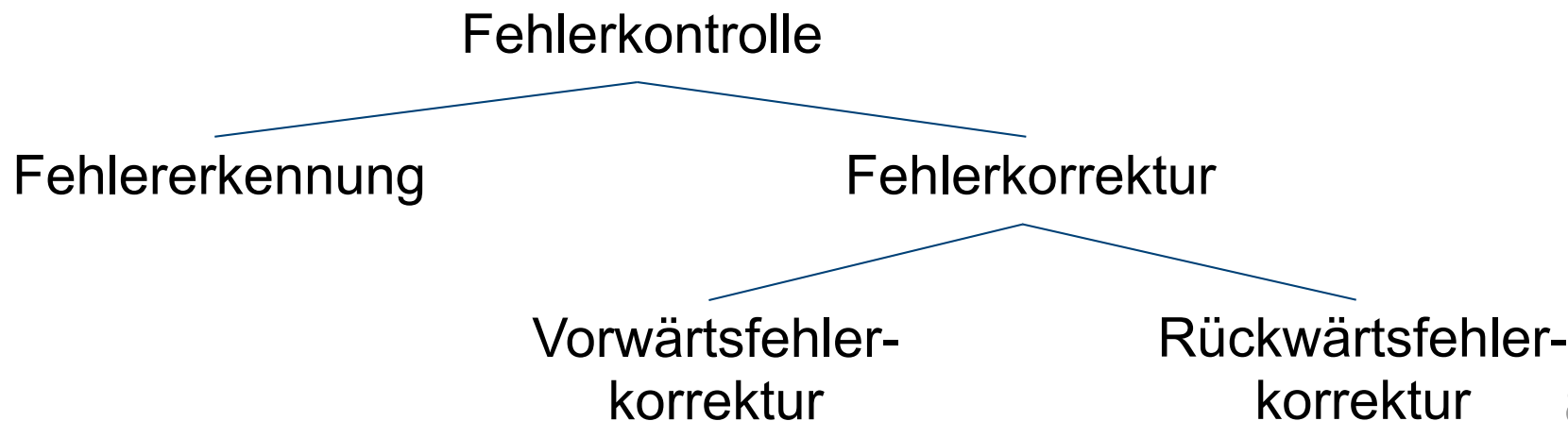
- Die Sicherungsschicht zwischen der Bitübertragungsschicht mit Bitstrom und der Vermittlungsschicht mit Paketen



- Pakete werden in Framegröße fragmentiert



- Zumeist gefordert von der Vermittlungsschicht
 - Mit Hilfe der Frames
- Fehlererkennung
 - Gibt es fehlerhaft übertragene Bits?
- Fehlerkorrektur
 - Behebung von Bitfehlern
 - Vorwärtsfehlerkorrektur (Forward Error Correction)
 - Verwendung von redundanter Kodierung, die es ermöglicht Fehler ohne zusätzliche Übertragungen zu beheben
 - Rückwärtsfehlerkorrektur (Backward Error Correction)
 - Nach Erkennen eines Fehlers, wird durch weitere Kommunikation der Fehler behoben



- Nutzen von Verbindungen
 - Kontrolle des Verbindungsstatus
 - Korrektheit des Protokolls
 - Fehlerkontrolle
 - Verschiedene Fehlerkontrollverfahren vertrauen auf gemeinsamen Kontext von Sender und Empfänger
- Aufbau und Terminierung von Verbindungen
 - “Virtuelle Verbindungen”
 - Es werden keine Schalter umgelegt
 - Interpretation des Bitstroms
 - Kontrollinformationen in Frames
 - Besonders wichtig bei drahtlosen Medien
- Das Problem wird im Rahmen der Transportschicht ausführlich diskutiert
 - Vgl. Sitzungsschicht vom OSI-Modell

- Problem: Schneller Sender und langsamer Empfänger
 - Der Sender lässt den Empfangspuffer des Empfängers überlaufen
 - Übertragungsbandweite wird durch sinnlosen Mehrfachversand (nach Fehlerkontrolle) verschwendet
- Anpassung der Frame-Sende-Rate an dem Empfänger notwendig

Langsamer Empfänger



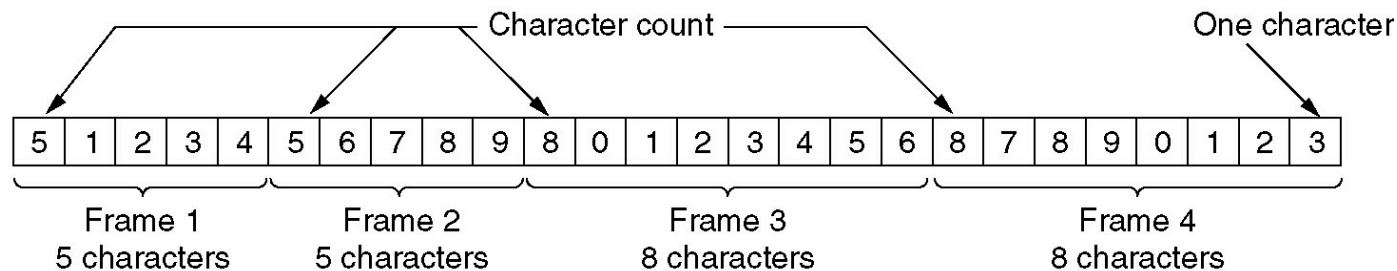
Schneller Sender

- Übertragener Bitstrom 0110010101110101110010100010101010101010101100010

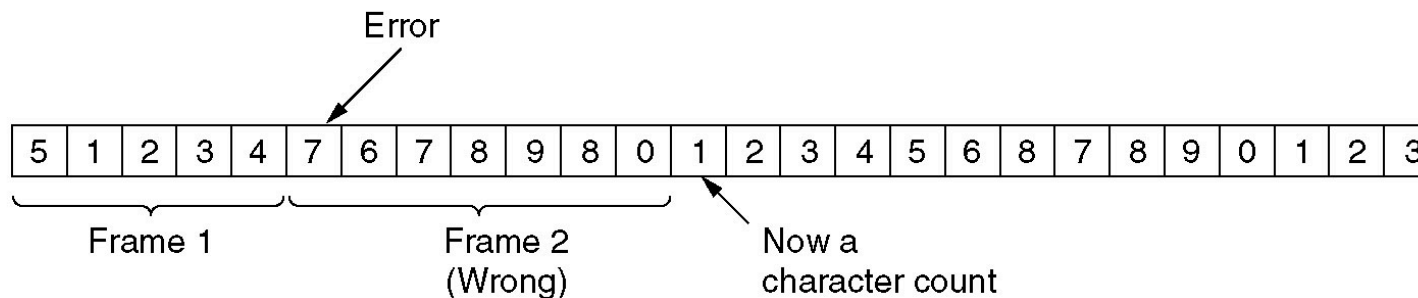
Frame-Ende?

Frame-Grenzen durch Paketlängen?

- Idee: Ankündigung der Bitanzahl im Frame-Header

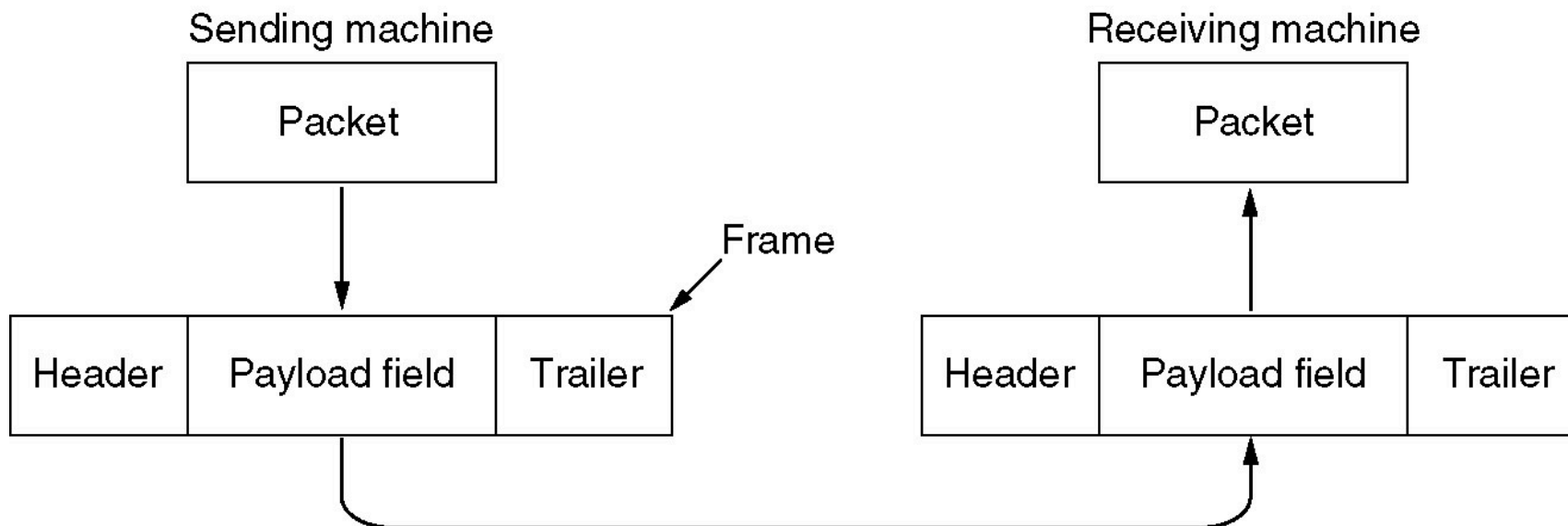


- Problem: Was, wenn die Frame-Länge fehlerhaft übertragen wird?
 - Der Empfänger kommt aus dem Takt und interpretiert neue, sinnlose Frames
 - Variable Frame-Größen mit Längeninformation sind daher kein gutes Konzept



■ Header und Trailer

- Zumeist verwendet man Header am Anfang des Frames, mitunter auch Trailer am Ende des Frames
- signalisieren den Frame-Beginn und das Frame-Ende
- tragen Kontrollinformationen
 - z.B. Sender, Empfänger, Frametypen, Fehlerkontrollinformation

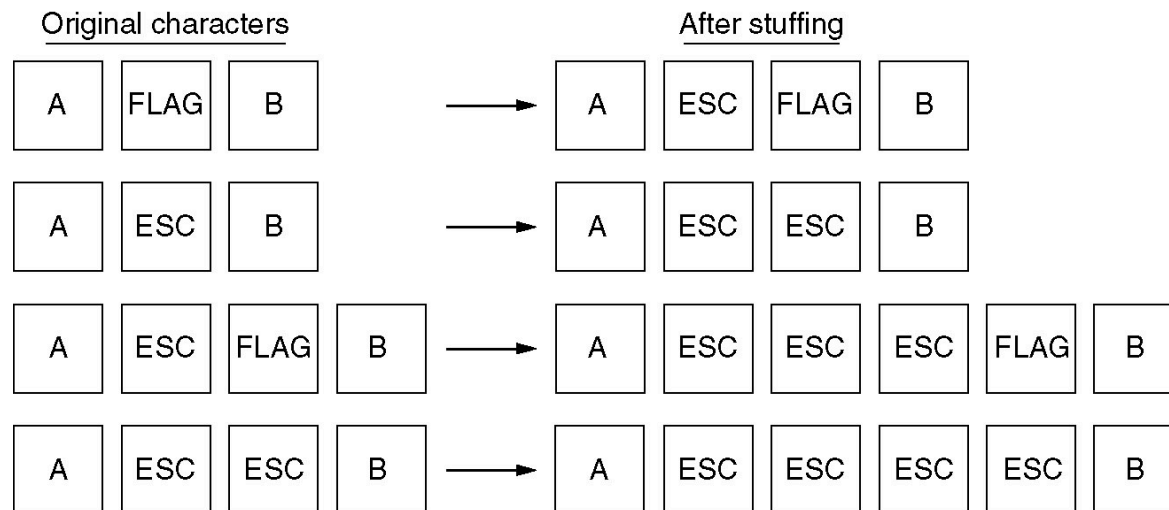


Flag Bytes und Bytestopfen

- Besondere “Flag Bytes” markieren Anfang und Ende eines Frames



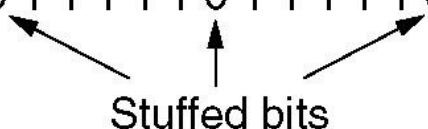
- Falls diese Marker in den Nutzdaten vorkommen
 - Als Nutzdatenbyte mit Sonderzeichen (Escape) markieren
 - Bytestopfen (byte stuffing)
 - Falls Sonderzeichen und “Flag-Byte” erscheinen, dito,
 - etc., etc.



- Bytestopfen verwendet das Byte als elementare Einheit
 - Das Verfahren funktioniert aber auch auf Bitebene
- Flag Bits und Bitstopfen (bit stuffing)
 - Statt flag byte wird eine Bit-Folge verwendet
 - z.B.: 01111110
 - Bitstopfen
 - Wenn der Sender eine Folge von fünf 1er senden möchte, wird automatisch eine 0 in den Bitstrom eingefügt
 - Außer bei den Flag Bits
- Der Empfänger entfernt eine 0 nach fünf 1ern

Originale Nutzdate (a) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

Nach dem Bitstopfen (b) 0 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 0 0 1 0



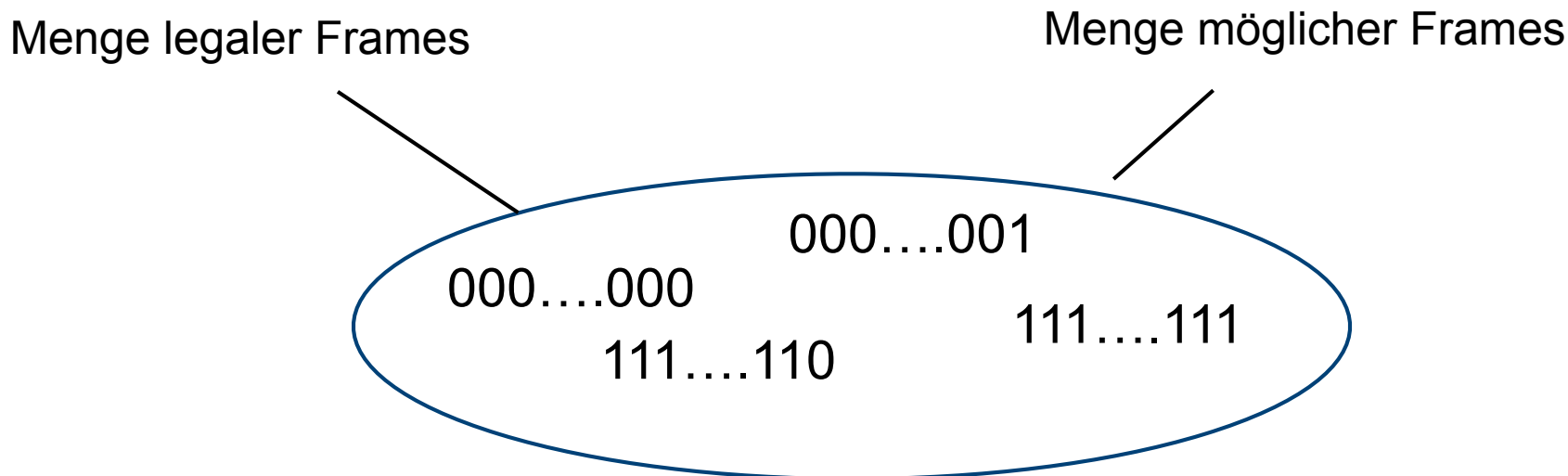
Stuffed bits

Nach der "Entstopfung" (c) 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0

- Möglicher Spielraum bei Bitübertragungsschicht bei der Kodierung von Bits auf Signale
 - Nicht alle möglichen Kombination werden zur Kodierung verwendet
 - Zum Beispiel: Manchester-Kodierung hat nur tief/hoch und hoch/tief-Übergang
- Durch “Verletzung” der Kodierungsregeln kann man Start und Ende des Rahmens signalisieren
 - Beispiel: Manchester – Hinzunahme von hoch/hoch oder tief/tief
 - Selbsttaktung von Manchester gefährdet?
- Einfache und robuste Methode
 - z.B. verwendet in Ethernet
 - Kosten? Effiziente Verwendung der Bandbreite?

- Aufgaben
 - Erkennung von Fehlern (fehlerhafte Bits) in einem Frame
 - Korrektur von Fehlern in einem Frame
- Jede Kombination dieser Aufgaben kommt vor
 - Erkennung ohne Korrektur
 - Löschen eines Frames ohne weiter Benachrichtigung (drop a frame)
 - Höhere Schichten müssen sich um das Problem kümmern
 - Korrektur ohne Erkennung
 - Es werden bestmöglich Bitfehler beseitigt, möglicherweise sind aber noch Fehler vorhanden
 - Sinnvoll, falls Anwendung Fehler tolerieren kann
 - Beispiel: Tonübertragung
 - Prinzipiell gerechtfertigt, weil immer eine positive Restfehlerwahrscheinlichkeit bleibt

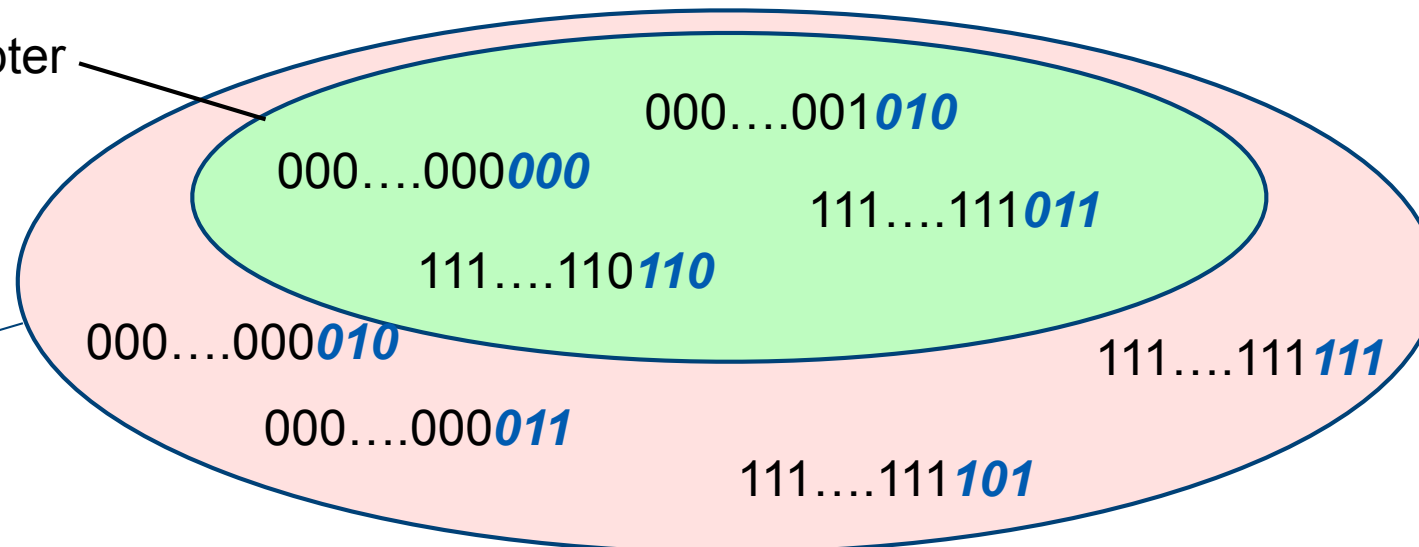
- Redundanz ist eine Voraussetzung für Fehlerkontrolle
- Ohne Redundanz
 - Ein Frame der Länge m kann 2^m mögliche Daten repräsentieren
 - Jede davon ist erlaubt
- Ein fehlerhaftes Bit ergibt einen neuen Dateninhalt



- Kernidee:
 - Einige der möglichen Nachrichten sind verboten
 - Um dann 2^m legale Frames darzustellen
 - werden mehr als 2^m mögliche Frames benötigt
 - Also werden mehr als m Bits in einem Frame benötigt
 - Der Frame hat also Länge $n > m$
 - $r = n - m$ sind die redundanten Bits
 - z.B. Im Header oder Trailer
- Nur die Einschränkung auf erlaubte und verbotene (legal/illegal) Frames ermöglicht die Fehlerkontrolle

Menge erlaubter
Frames

Menge
aller
Frames

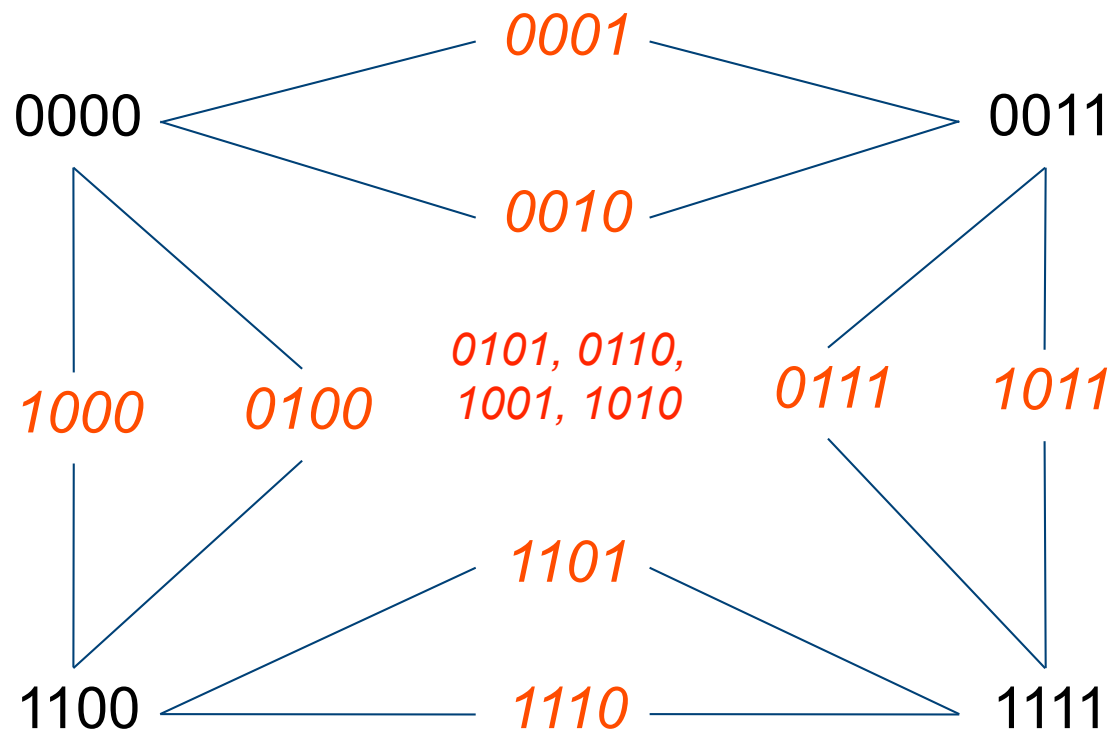


- Eine einfache Regel um ein redundantes Bit zu erzeugen
(d.h. $n=m+1$)
- Parität
 - Odd parity
 - Eine Eins wird hinzugefügt, so dass die Anzahl der 1er in der Nachricht ungerade wird (ansonsten eine Null)
 - Even parity
 - Eine Eins wird hinzugefügt, so dass die Anzahl der 1er in der Nachricht gerade wird (ansonsten wird eine Null hinzugefügt)
- Beispiel:
 - Originalnachricht ohne Redundanz: 01101011001
 - Odd parity: 011010110011
 - Even parity: 011010110010

- Der Sender sendet nur erlaubte Frames
- In der Bitübertragungsschicht könnten Bits verfälscht werden
- Hoffnung:
 - Legale Frames werden nur in illegale Nachrichten verfälscht
 - Und niemals ein legaler Frame in einen anderen Legalen
- Notwendige Annahme
 - In der Bitübertragungsschicht werden nur eine bestimmte Anzahl von Bits verändert
 - z.B. k Bits pro Frame
 - Die legalen Nachrichten sind verschieden genug, um diese Frame-Fehlerrate zu erkennen

Veränderung der Frames durch Bitfehler

- Angenommen die folgenden Frames sind erlaubt: 0000, 0011, 1100, 1111



Kanten verbinden Frames, die sich nur in einem Bit unterscheiden

Ein einfacher Bitfehler kann erlaubte Frames nicht in einen anderen erlaubten Frame umformen!

uvxy – erlaubt *abcd* – verboten

- Der “Abstand” der erlaubten Nachrichten zueinander war immer zwei Bits
- Definition: Hamming-Distanz
 - Seien $x = x_1, \dots, x_n$ und $y = y_1, \dots, y_n$ Nachrichten
 - Dann sei $d(x,y)$ = die Anzahl der 1er Bits in $x \text{ XOR } y$
- Intuitiver: die Anzahl der Positionen, in denen sich x und y unterscheiden

- Die Hamming-Distanz ist eine Metrik

- Symmetrie

- $d(x,y) = d(y,x)$

- Dreiecksungleichung:

- $d(x,y) \leq d(x,z) + d(z,y)$

- Identität

- $d(x,x) = 0$ und
 $d(x,y) = 0$ gdw. $x = y$

- Beispiel:

- $x =$ 0011010111

- $y =$ 0110100101

- $x \text{ XOR } y =$ 0101110010

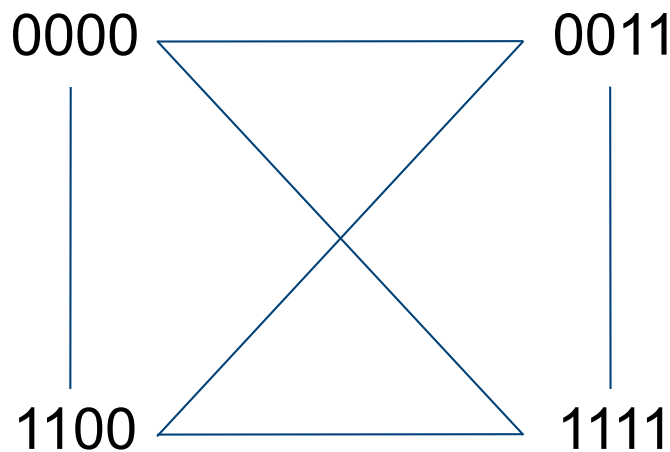
- $d(x,y) = 5$

- Die Hamming-Distanz einer Menge von (gleich langen) Bit-Strings S ist:

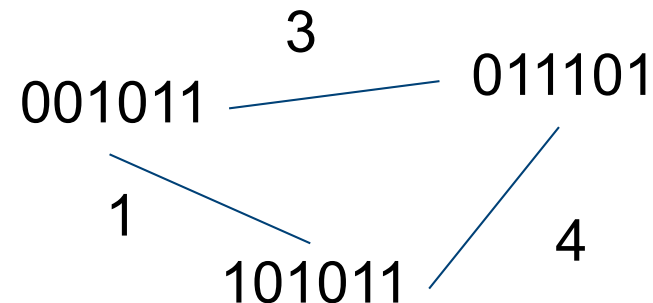
$$d(S) = \min_{x,y \in S, x \neq y} d(x, y)$$

- d.h. der kleinste Abstand zweier verschiedener Wörter in S

Beispiel:



Alle Abstände sind 2



Ein Abstand ist 1!

- 1. Fall $d(S) = 1$
 - Keine Fehlerkorrektur
 - Legale Frames unterscheiden sich in nur einem Bit
- 2. Fall $d(S) = 2$
 - Dann gibt es nur $x, y \in S$ mit $d(x, y) = 2$
 - Somit ist jedes u mit $d(x, u) = 1$ illegal,
 - wie auch jedes u mit $d(y, u) = 1$



- 1-Bit-Fehler
 - können immer erkannt werden
 - aber nicht korrigiert werden

■ 3. Fall $d(S) = 3$

- Dann gibt es nur $x, y \in S$ mit $d(x, y) = 3$
- Jedes u mit $d(x, u) = 1$ illegal und $d(y, u) > 1$



- Falls u empfangen wird, sind folgende Fälle denkbar:
 - x wurde gesendet und mit 1 Bit-Fehler empfangen
 - y wurde gesendet und mit 2 Bit-Fehlern empfangen
 - Etwas anderes wurde gesendet und mit mindestens 2 Bit-Fehlern empfangen
- Es ist also wahrscheinlicher, dass x gesendet wurde, statt y

- Um d Bit-Fehler zu erkennen ist eine Hamming-Distanz von $d+1$ in der Menge der legalen Frames notwendig
- Um d Bit-Fehler zu korrigieren, ist eine Hamming-Distanz von $2d+1$ in der Menge der legalen Frames notwendig

- Die Menge der legalen Frames $S \in \{0,1\}^n$ wird **das Code-Buch** oder einfach Kodierung genannt.
 - Die Rate R eines Codes S ist definiert als
 - Die Rate charakterisiert die Effizienz des Codes

$$R_S = \frac{\log |S|}{n}$$

- Die Distanz δ des Codes S ist definiert als
 - charakterisiert die Fehlerkorrektur oder Fehlererkennungsmöglichkeiten

$$\delta_S = \frac{d(S)}{n}$$

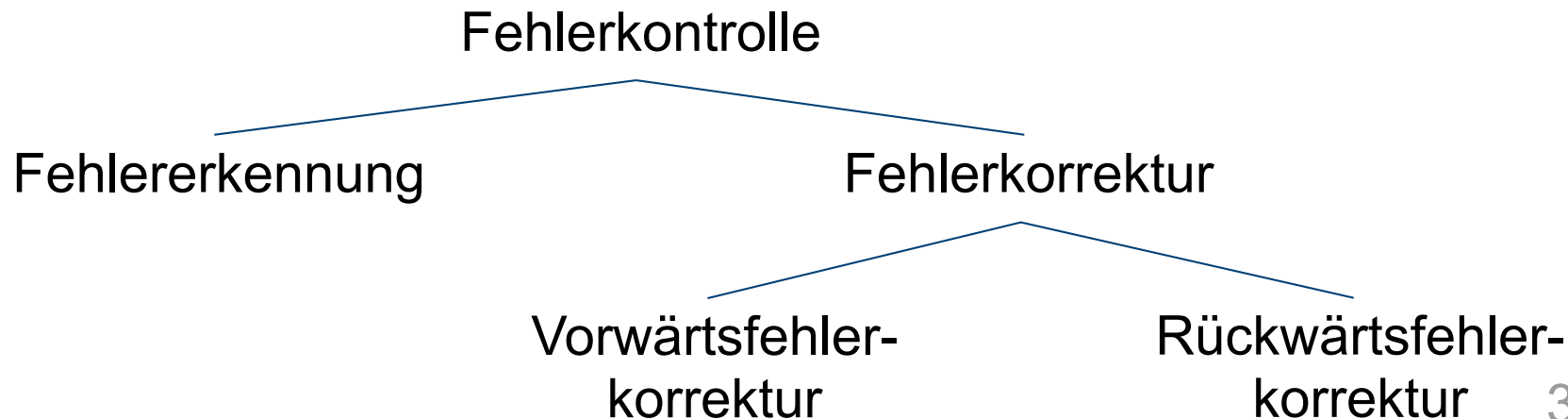
- Gute Codes haben hohe Raten und hohe Distanz
 - Beides lässt sich nicht zugleich optimieren

- Block-Codes kodieren k Bits Originaldaten in n kodierte Bits
 - Zusätzlich werden n-k Symbole hinzugefügt
 - Binäre Block-Codes können höchstens bis zu t Fehler in einem Code-Wort der Länge n mit k Originalbits erkennen, wobei (Gilbert-Varshamov-Schranke):

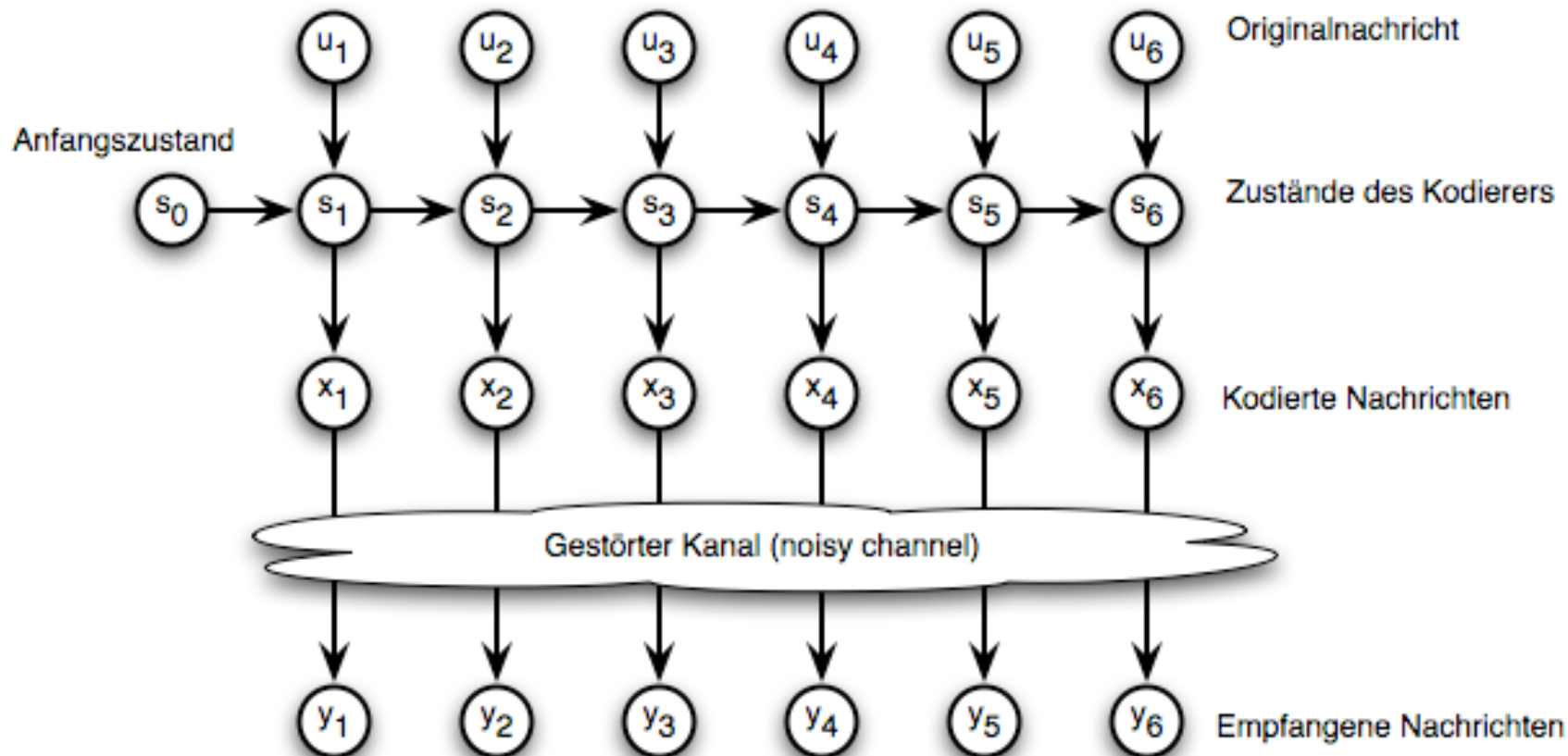
$$2^{n-k} \geq \sum_{i=0}^t \binom{n}{i}$$

- Das ist eine theoretische obere Schranke
- Beispiele
 - Bose Chaudhuri Hocquenghem (BCH) Codes
 - basierend auf Polynomen über endlichen Körpern (Galois-Körpern)
 - Reed Solomon Codes
 - Spezialfall nichtbinärer BCH-Codes

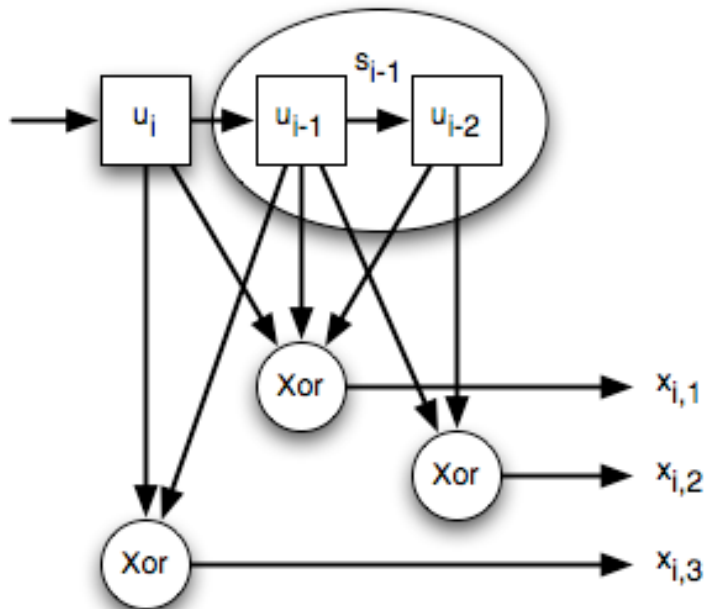
- Zumeist gefordert von der Vermittlungsschicht
 - Mit Hilfe der Frames
- Fehlererkennung
 - Gibt es fehlerhaft übertragene Bits?
- Fehlerkorrektur
 - Behebung von Bitfehlern
 - Vorwärtsfehlerkorrektur (Forward Error Correction)
 - Verwendung von redundanter Kodierung, die es ermöglicht Fehler ohne zusätzliche Übertragungen zu beheben
 - Rückwärtsfehlerkorrektur (Backward Error Correction)
 - Nach Erkennen eines Fehlers, wird durch weitere Kommunikation der Fehler behoben



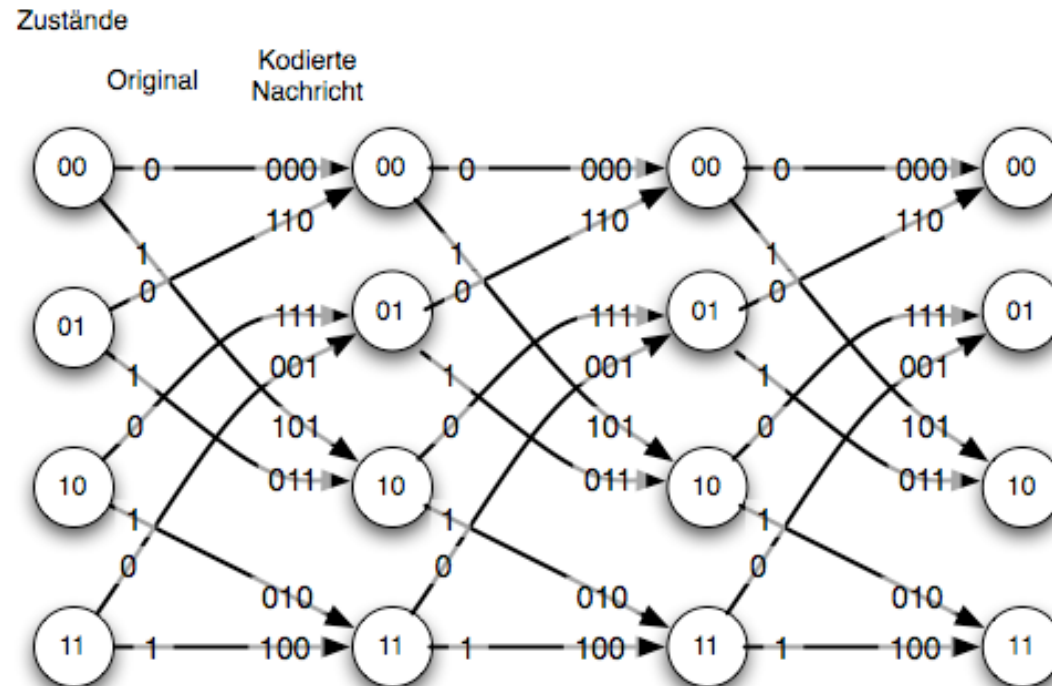
- Faltungs-Codes (Convolutional Codes)
 - Daten und Fehlerredundanz werden vermischt.
 - k Bits werden auf n Bits abgebildet
 - Die Ausgabe hängt von den k letzten Bits und dem internen Zustand ab.



Faltungs-Kodierer



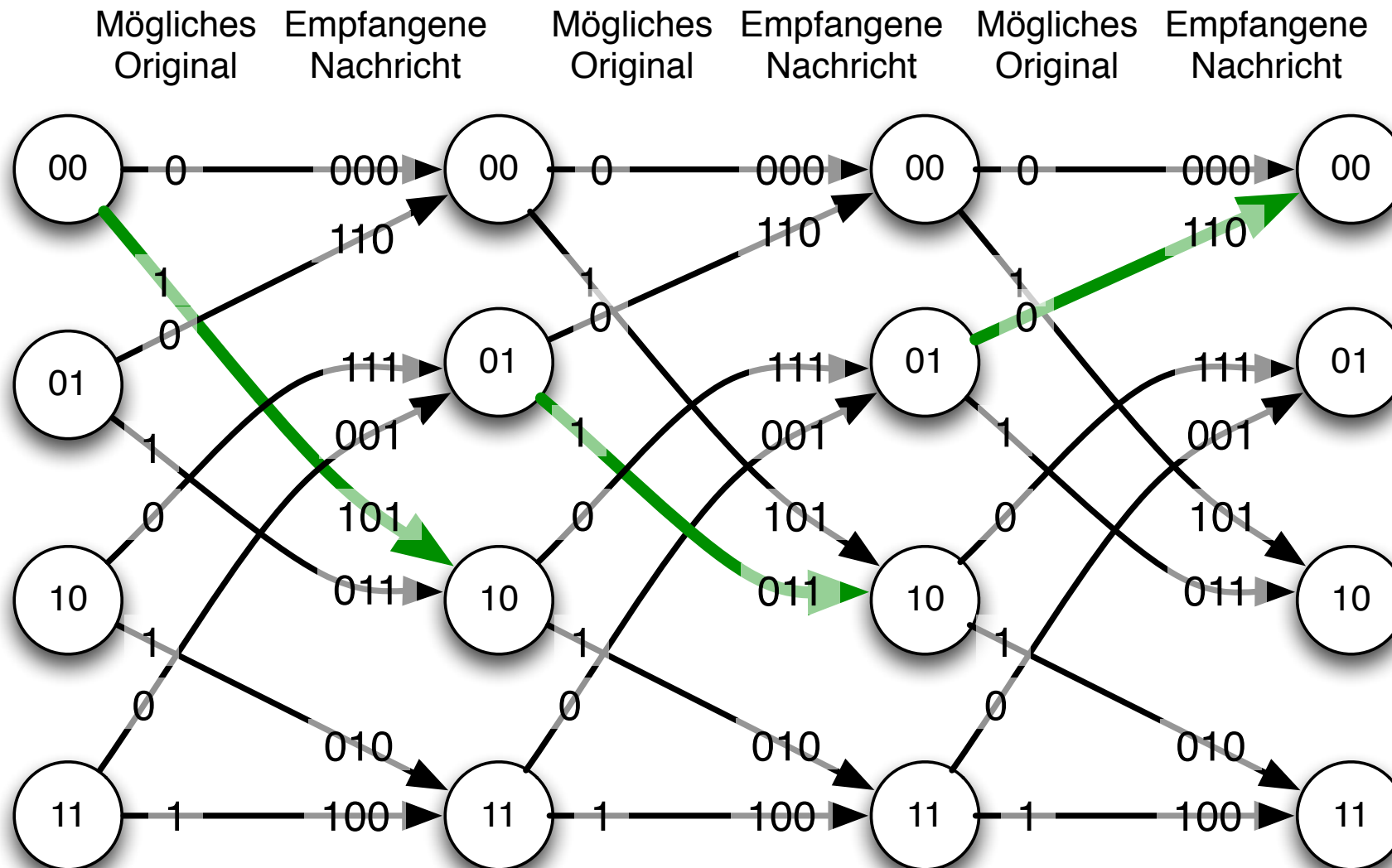
Trellis-Diagramm



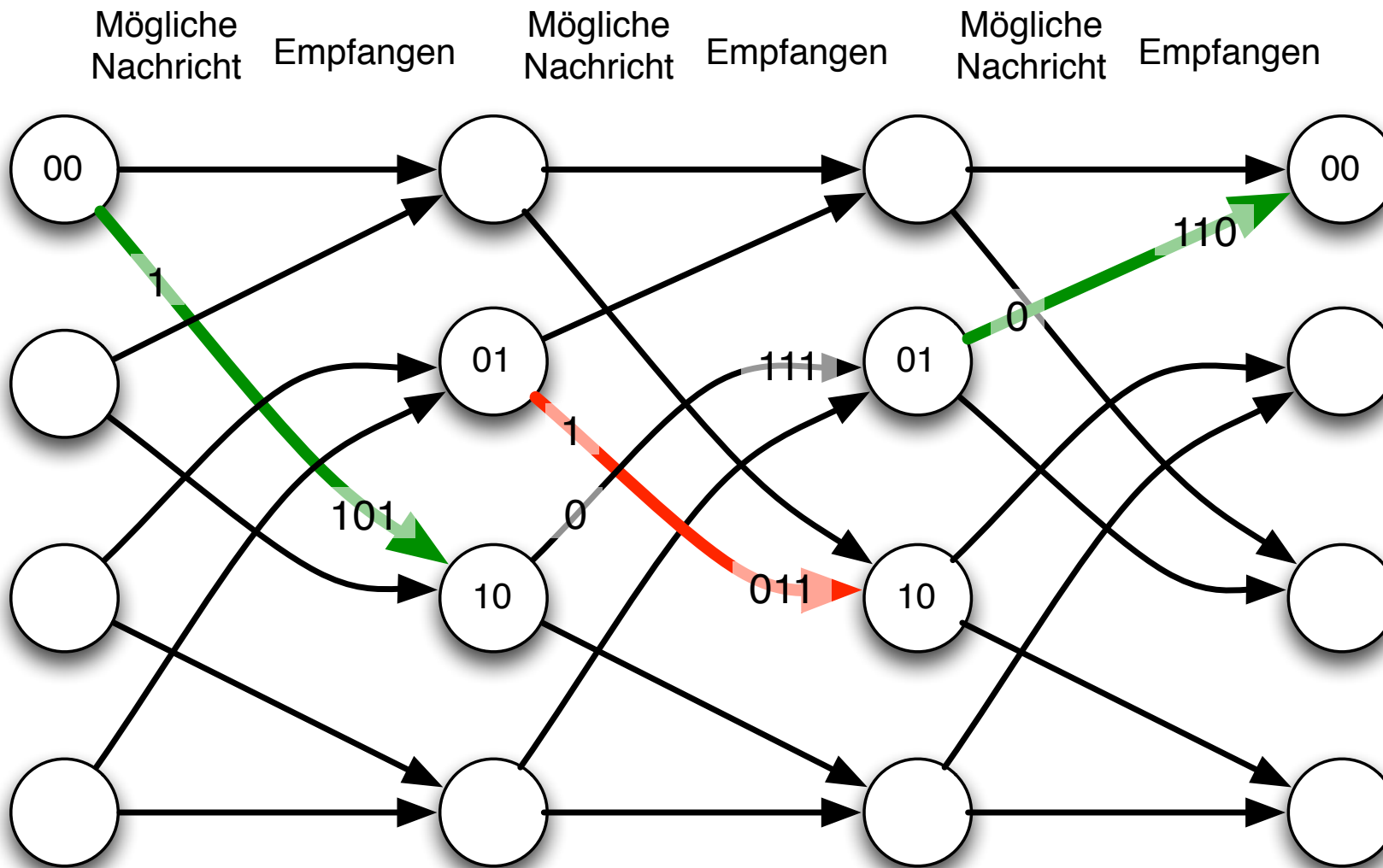
- Dynamische Programmierung
- Zwei notwendige Voraussetzungen für Dekodierung
 - (für den Empfänger) unbekannte Folge von Zuständen
 - beobachtete Folge von empfangenen Bits (möglicherweise mit Fehler)
- Der Algorithmus von Viterbi bestimmt die wahrscheinlichste Folge von Zuständen, welches die empfangenen Bits erklärt
 - Hardware-Implementation möglich

Dekodierung (I)

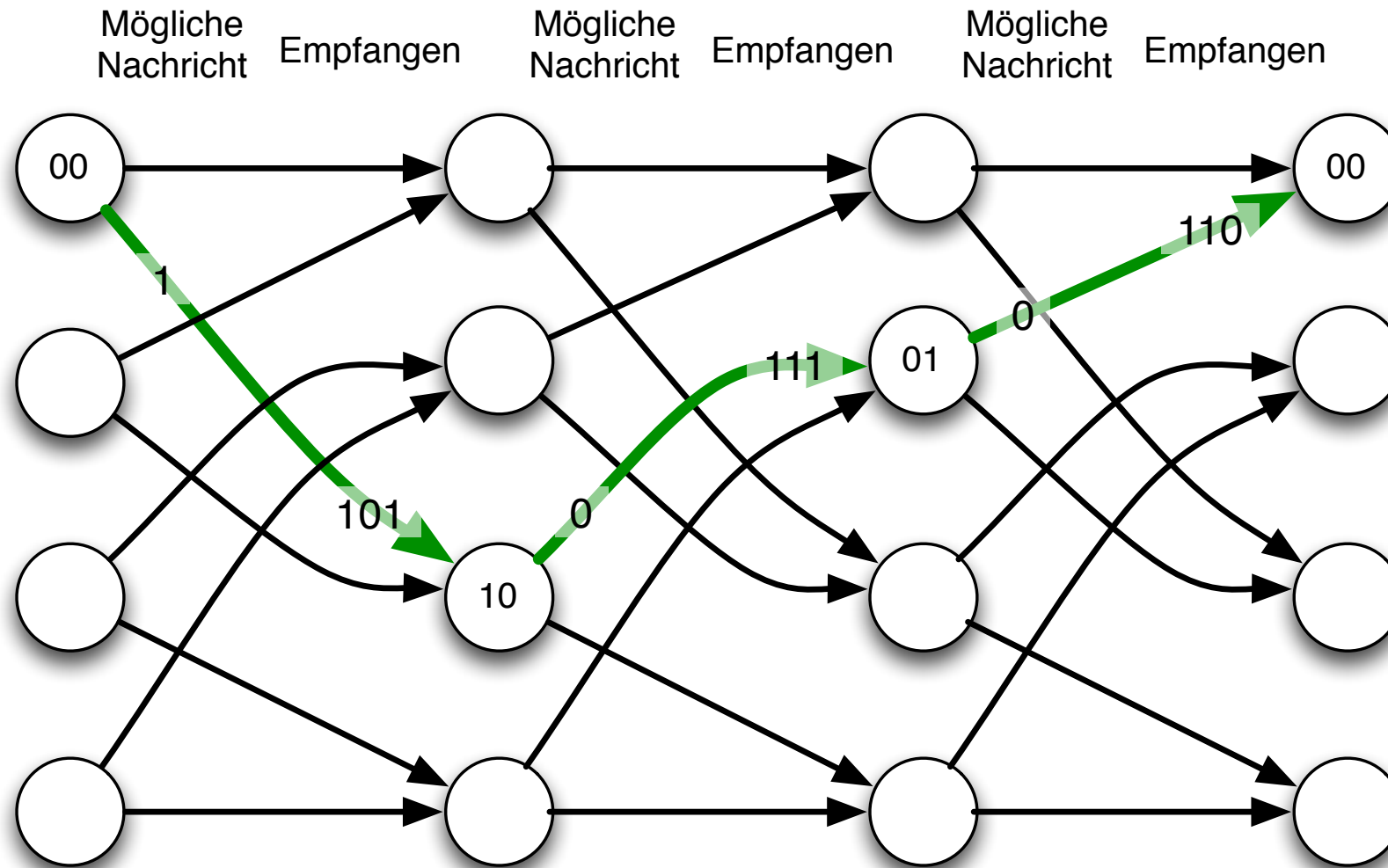
Zustände



Dekodierung (II)

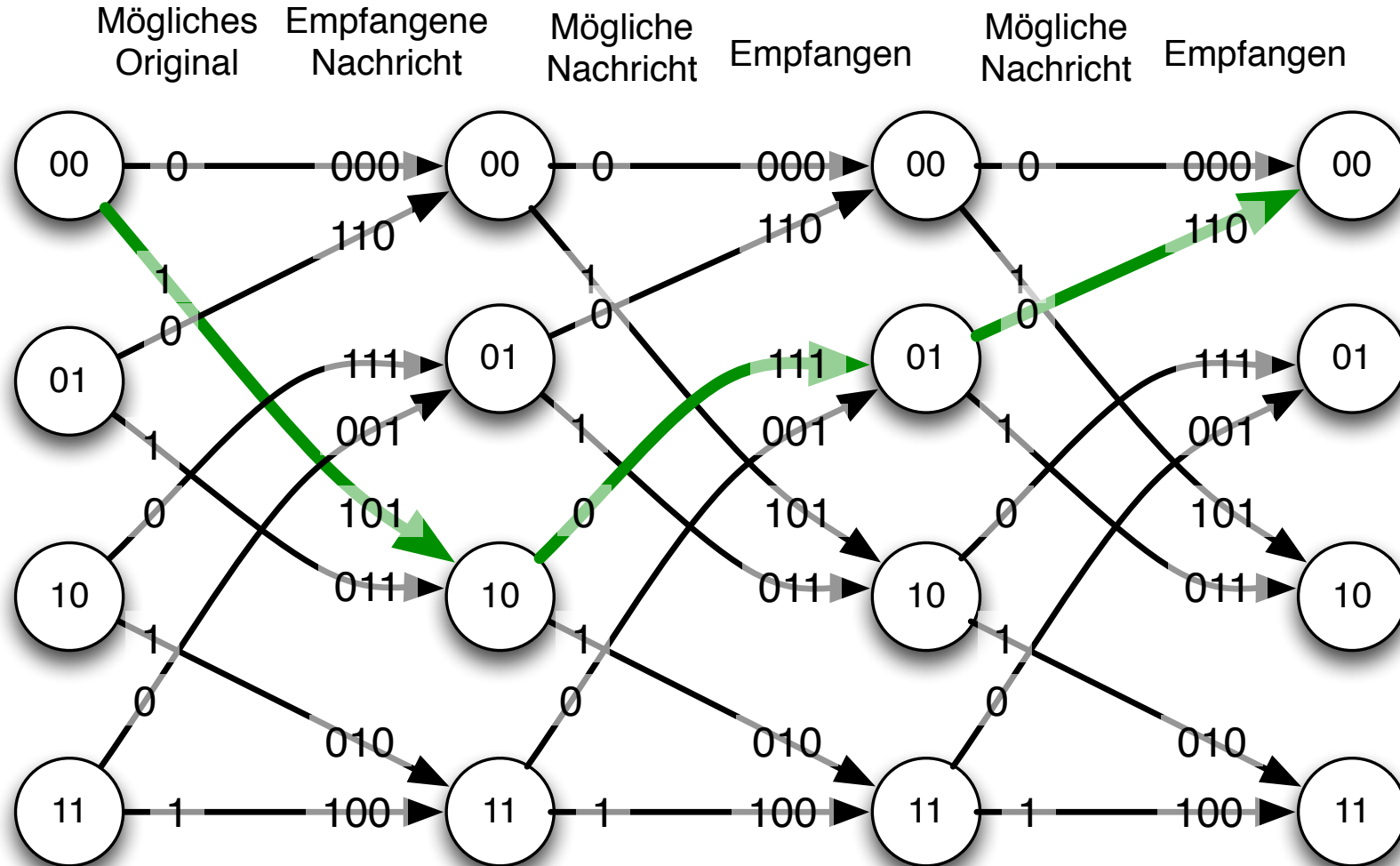


Dekodierung (III)

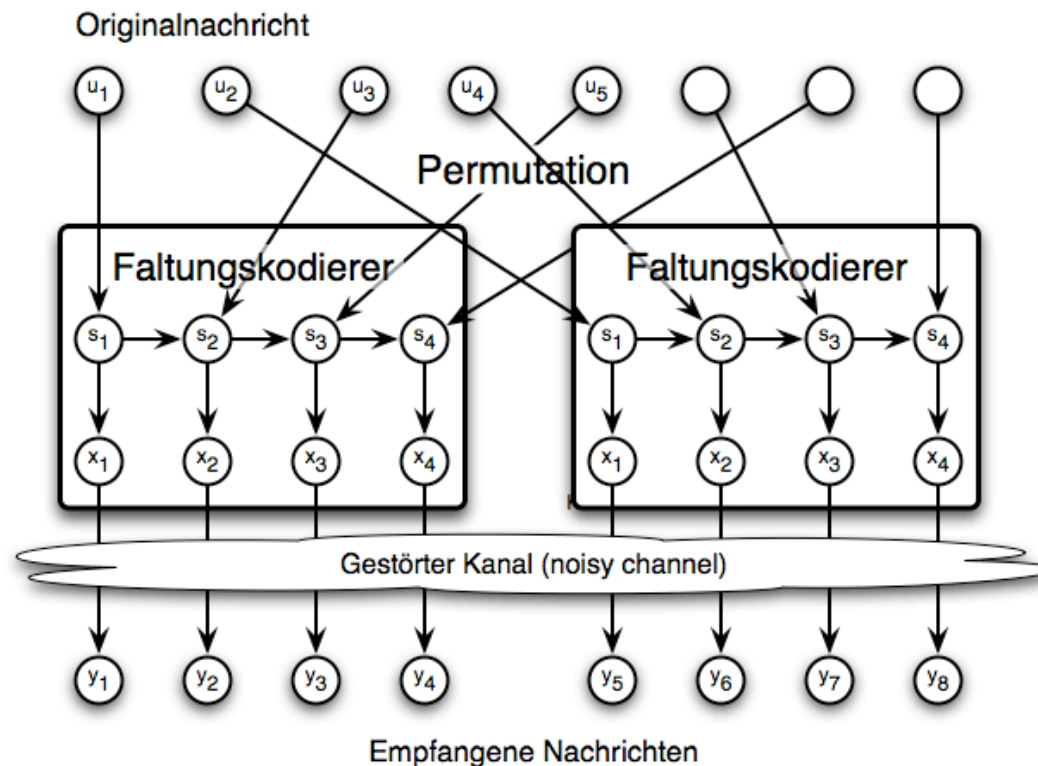


Dekodierung (IV)

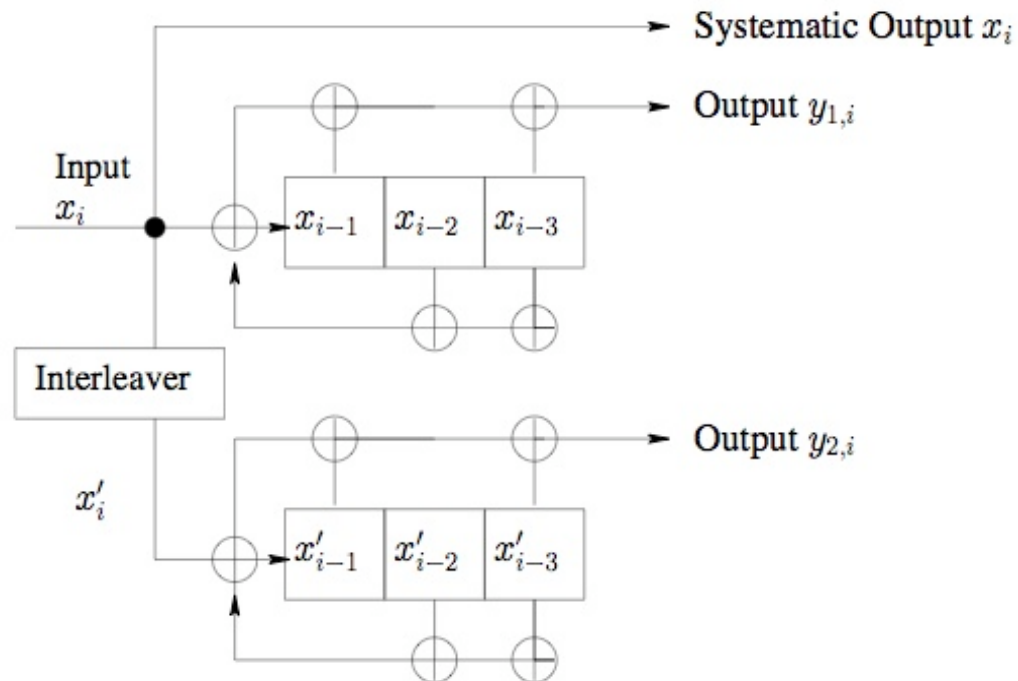
Zustände



- Turbo-Codes sind wesentlich effizienter als Faltungs-Codes
 - bestehen aus zwei Faltungs-Codes welche abwechselnd mit der Eingabe versorgt werden.
 - Die Eingabe wird durch eine Permutation (Interleaver) im zweiten Faltungs-Code umsortiert



- Beispiel:
 - UMTS Turbo-Kodierer
- Dekodierung von Turbo-Codes ist effizienter möglich als bei Faltungscode
- Kompensation von Bursts



- Fehler treten oftmals gehäuft auf (Bursts)
 - z.B.: Daten: 0 1 2 3 4 5 6 7 8 9 A B C D E F
 - mit Fehler: 0 1 2 3 ? ? ? ? ? 9 A B C D E F
- Dann scheitern klassische Kodierer ohne Interleavers
 - Nach Fehlerkorrektur (zwei Zeichen in Folge reparierbar):
 0 1 2 3 4 5 ? 7 8 9 A B C D E F
- Interleaver:
 - Permutation der Eingabekodierung:

0 1 2 3
 4 5 6 7
 8 9 A B
 C D E F
 - z.B. Row-column Interleaver:

0 4 8 C 1 5 9 D 2 6 A E 3 7 B F
 - mit Fehler:

0 4 8 C ? ? ? ? ? 6 A E 3 7 B F
 - Rückpermuiert:

0 ? ? 3 4 ? 6 7 8 ? A B C D ? F
 - nach FEC:

0 1 2 3 4 5 6 7 8 9 A B C D E F

- Effiziente Fehlererkennung: Cyclic Redundancy Check (CRC)
- Praktisch häufig verwendeter Code
 - Hoher Fehlererkennungsrate
 - Effizient in Hardware umsetzbar
- Beruh auf Polynomarithmetik im Restklassenring $\mathbb{Z}_2$
 - Zeichenketten sind Polynome
 - Bits sind Koeffizienten des Polynoms

$$\begin{array}{r}
 3,50 \\
 7,90 \\
 \hline
 x^2 + x + 0 = x(x+1)
 \end{array}$$

$$\begin{array}{r}
 \begin{array}{cc}
 \text{bits} & \text{bits} \\
 \downarrow & \downarrow \\
 3,50 & 1
 \end{array} \\
 = \\
 + \quad 5 \cdot 10 \quad \swarrow x \\
 + \quad 3 \cdot 100 \quad \swarrow x^2
 \end{array}$$

$$A + B = x$$

$$B = x - A$$

- Rechnen modulo 2:
- Regeln:
 - Addition modulo 2 = Xor = Subtraktion modulo 2
 - Multiplikation modulo 2 = And

A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A - B
0	0	0
0	1	1
1	0	1
1	1	0

A	B	A · B
0	0	0
0	1	0
1	0	0
1	1	1

- Beispiel: $0 + \underbrace{(1 \cdot 0)}_0 + 1 + \underbrace{(1 \cdot 1)}_1 = 0$

$$\begin{array}{r}
 101 \cdot 1011 \\
 \hline
 1011 \\
 \Sigma \quad 101100 + \\
 \hline
 100111
 \end{array}$$

- Betrachte Polynome über den Restklassenring $\mathbb{Z}_2$
 - $p(x) = a_n x^n + \dots + a_1 x^1 + a_0$
 - Koeffizienten a_i und ~~Variable x~~ sind aus $\{0,1\}$
 - Berechnung erfolgt modulo 2
- Addition, Subtraktion, Multiplikation, Division von Polynomen wie gehabt

$$\begin{array}{r}
 x^2 + x + 1 \\
 \quad \quad x \\
 \hline
 x^2 \quad \quad + 1
 \end{array}$$

$$\begin{array}{r}
 1 \quad 1 \quad 1 \\
 \Sigma \quad \quad 1 \quad 0 \\
 \hline
 1 \quad 0 \quad 1
 \end{array}$$

Kein Übertrag!

- Idee:
 - Betrachte Bitstring der Länge n als Variablen eines Polynoms

- Bit string: $b_n b_{n-1} \dots b_1 b_0$

Polynom: $b_n x^n + \dots + b_1 x^1 + b_0$

- Bitstring mit $(n+1)$ Bits entspricht Polynom des Grads n

- Beispiel

- $A \text{ xor } B = A(x) + B(x)$ *mod 2*

$$101000 = 101 \cdot 1000$$

$$x^5 + x^3 = (x^2 + 1) \cdot x^3$$

- Wenn man A um k Stellen nach links verschiebt, entspricht das

- $B(x) = A(x) x^k$

- Mit diesem Isomorphismus kann man Bitstrings dividieren

Polynome zur Erzeugung von Redundanz: CRC

- Definiere ein Generatorpolynom $G(x)$ von Grad g
 - Dem Empfänger und Sender bekannt
 - Wir erzeugen g redundante Bits
- Gegeben:
 - Frame (Nachricht) M , als Polynom $M(x)$
- Sender
 - Berechne den Rest der Division $r(x) = x^g M(x) \bmod G(x)$
 - Übertrage $T(x) = x^g M(x) + r(x)$
 - Beachte: $x^g M(x)$ + $r(x)$ ist ein Vielfaches von $G(x)$
- Empfänger
 - Empfängt $m(x)$
 - Berechnet den Rest: $m(x) \bmod G(x)$

$$x^4 + x + 1$$

$$\underline{110011}$$

$$10111010 \quad \underline{111}$$

$$10111010 \bigg| 0000$$

$$r(x) = 1100 \\ = x^3 + x^2$$

$$10111010 \bigg| 1100$$

$$r(x) + x^8 M(x) \bmod G(x) = r(x) + r(x) \\ = 0$$

CRC Übertragung und Empfang

M 101110101100
T 101111001100

110000 = E(x)

G(x) · 1
10
11

10011

- Keine Fehler:
 - T(x) wird korrekt empfangen
- Bitfehler: T(x) hat veränderte Bits
 - Äquivalent zur Addition eines Fehlerpolynoms E(x)
 - Beim Empfänger kommt T(x) + E(x) an

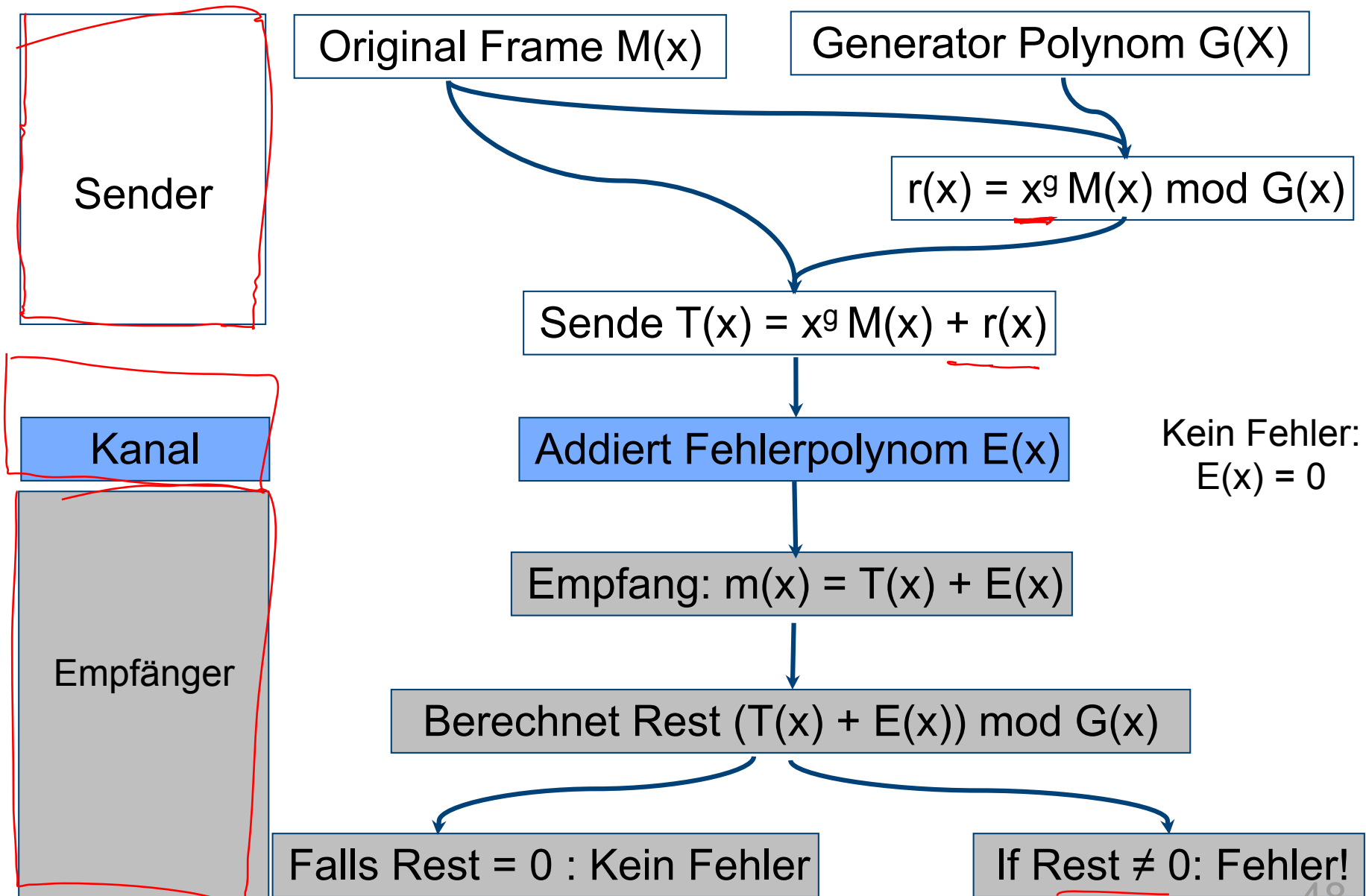
Empfänger

- Empfangen: m(x)
- Berechnet Rest m(x) mod G(x)
- Kein Fehler: m(x) = T(x),
 - dann ist der Rest 0

- Bit errors: m(x) mod G(x) = (T(x) + E(x)) mod G(x)
= T(x) mod G(x) + E(x) mod G(x)

0

Fehlerindikator



Der Generator bestimmt die CRC-Eigenschaften

$$x^5 + 1$$

- Bit-Fehler werden nur übersehen, falls $E(x)$ ein Vielfaches von $G(x)$ ist
- Die Wahl von $G(x)$ ist trickreich:
- Einzel-Bit-Fehler: $E(x) = x^i$ für Fehler an Position i
 - $G(x)$ hat mindestens zwei Summenterme, dann ist $E(x)$ kein Vielfaches von $G(x)$ ist
- Zwei-Bit-Fehler: $E(x) = x^i + x^j = x^j(x^{i-j} + 1)$ für $i > j$
 - $G(x)$ darf nicht $(x^k + 1)$ teilen für alle k bis zur maximalen Frame-Länge
- Ungerade Anzahl von Fehlern:
 - $E(x)$ hat nicht $(x+1)$ als Faktor
 - Gute Idee (?): Wähle $(x+1)$ als Faktor von $G(x)$
 - Dann ist $E(x)$ kein Vielfaches von $G(x)$
- Bei guter Wahl von $G(x)$:
 - kann jede Folge von r Fehlern erfolgreich erkannt werden
- Häufig:
 - $G(x)$ wird als irreduzibles Polynom gewählt, dass heißt es ist kein Vielfache eines anderen (kleineren) Polynoms

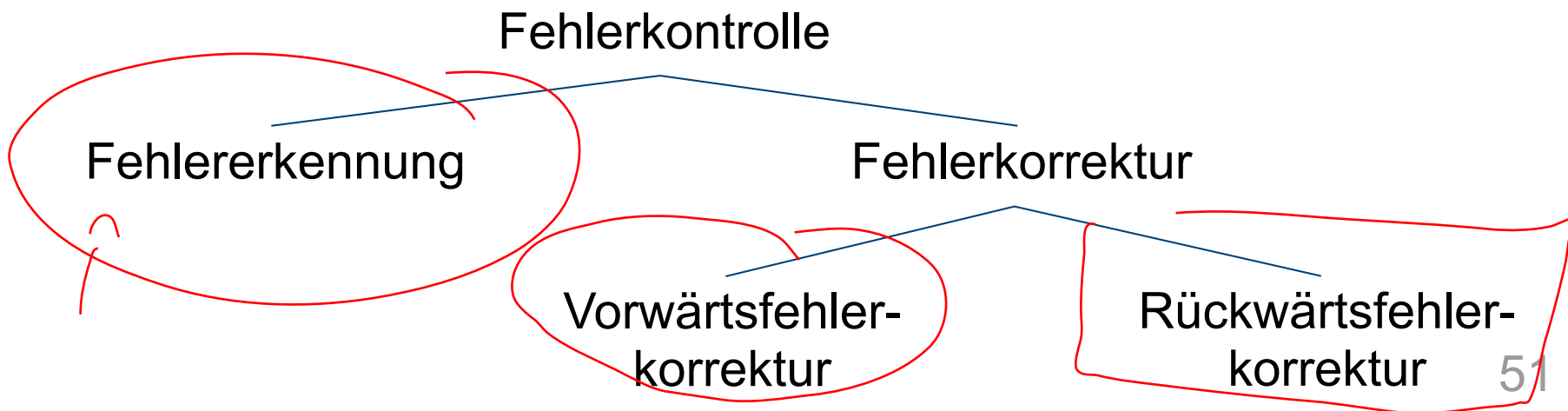
$$x^2 + 1$$

$$x^2 + 1$$

$$x^2 + 1$$

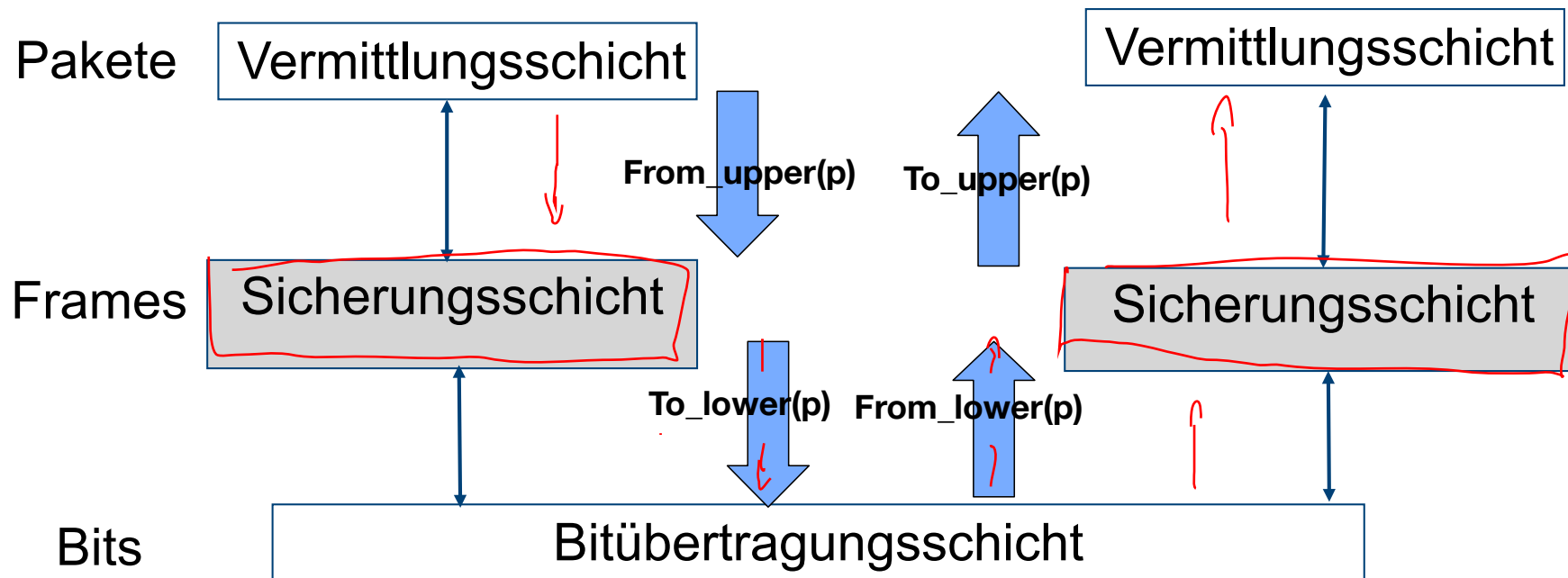
- Verwendetes irreduzibles Polynom gemäß IEEE 802:
 - $x^{32} + x^{23} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Achtung:
 - Fehler sind immer noch möglich
 - Insbesondere wenn der Bitfehler ein Vielfaches von G(x) ist.
- Implementation:
 - Für jedes Polynom x^i wird $r(x,i) = x^i \bmod G(x)$ berechnet
 - Ergebnis von $B(x) \bmod G(x)$ ergibt sich aus
 - $b_0 r(x,0) + b_1 r(x,1) + b_2 r(x,2) + \dots + b_{k-1} r(x,k-1)$
 - Einfache Xor-Operation

- Zumeist gefordert von der Vermittlungsschicht
 - Mit Hilfe der Frames
- Fehlererkennung
 - Gibt es fehlerhaft übertragene Bits?
- Fehlerkorrektur
 - Behebung von Bitfehlern
 - Vorwärtsfehlerkorrektur (Forward Error Correction)
 - Verwendung von redundanter Kodierung, die es ermöglicht Fehler ohne zusätzliche Übertragungen zu beheben
 - Rückwärtsfehlerkorrektur (Backward Error Correction)
 - Nach Erkennen eines Fehlers, wird durch weitere Kommunikation der Fehler behoben



Rückwärtsfehlerkorrektur

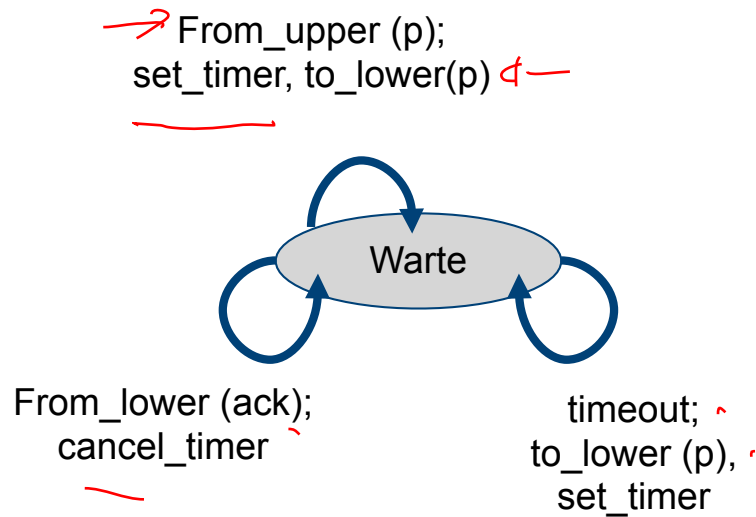
- Bei Fehlererkennung muss der Frame nochmal geschickt werden
- Wie ist das Zusammenspiel zwischen Sender und Empfänger?



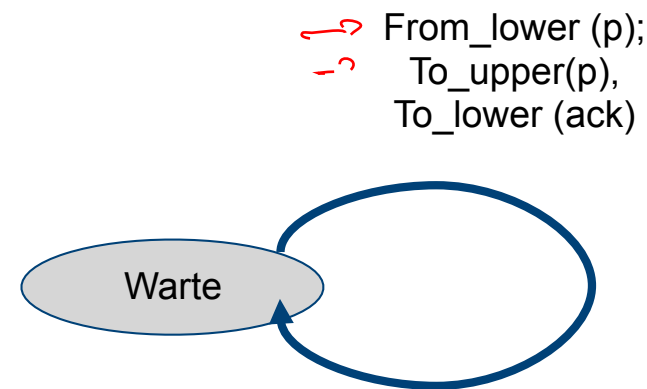
to_lower, from_lower beinhalten CRC
oder (bei Bedarf) Vorwärtsfehlerkorrektur

- Empfänger bestätigt Pakete dem Sender
- Der Sender wartet für eine bestimmte Zeit auf die Bestätigung (acknowledgment)
- Falls die Zeit abgelaufen ist, wird das Paket wieder versendet
- Erster Lösungsansatz

Sender



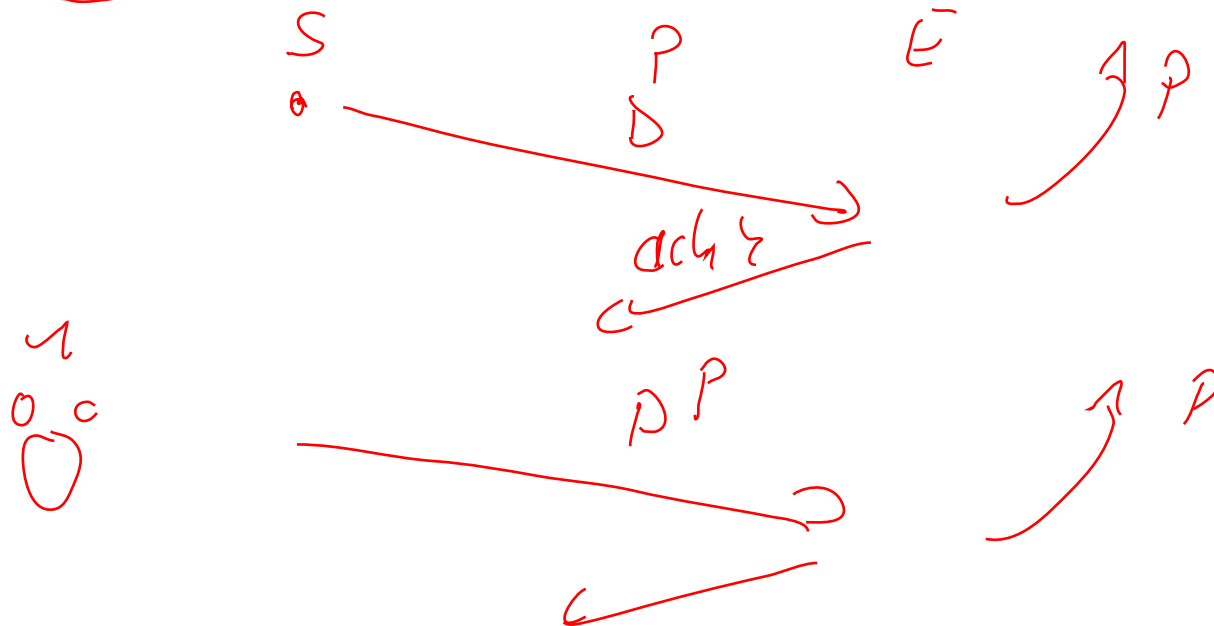
Empfänger



■ Probleme

- Sender ist schneller als Empfänger

- Was passiert, wenn Bestätigungen verloren gehen?



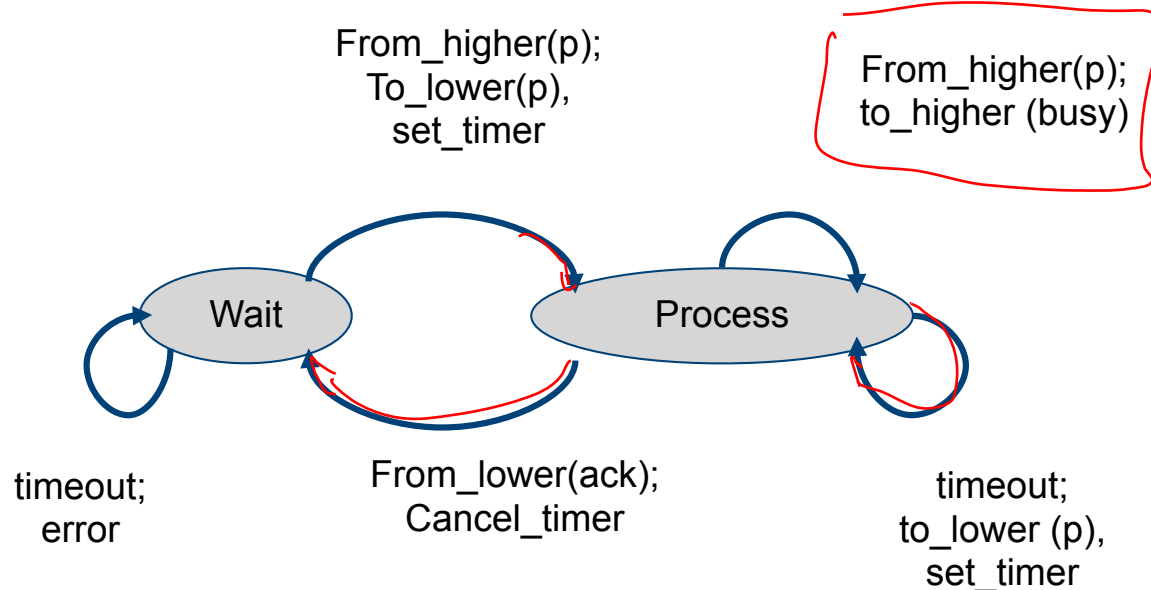
2. Versuch

■ Lösung des ersten Problems

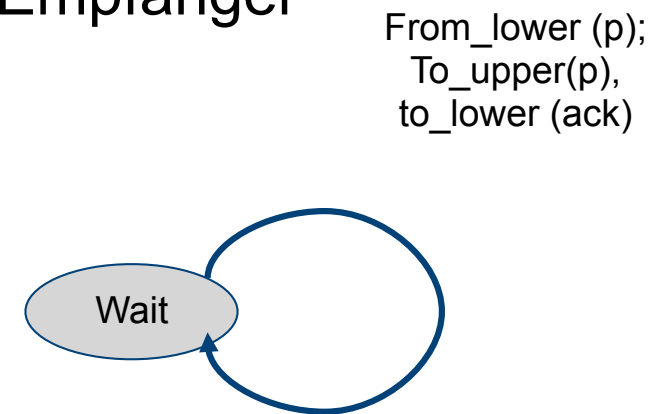
- Ein Paket nach dem anderen

■

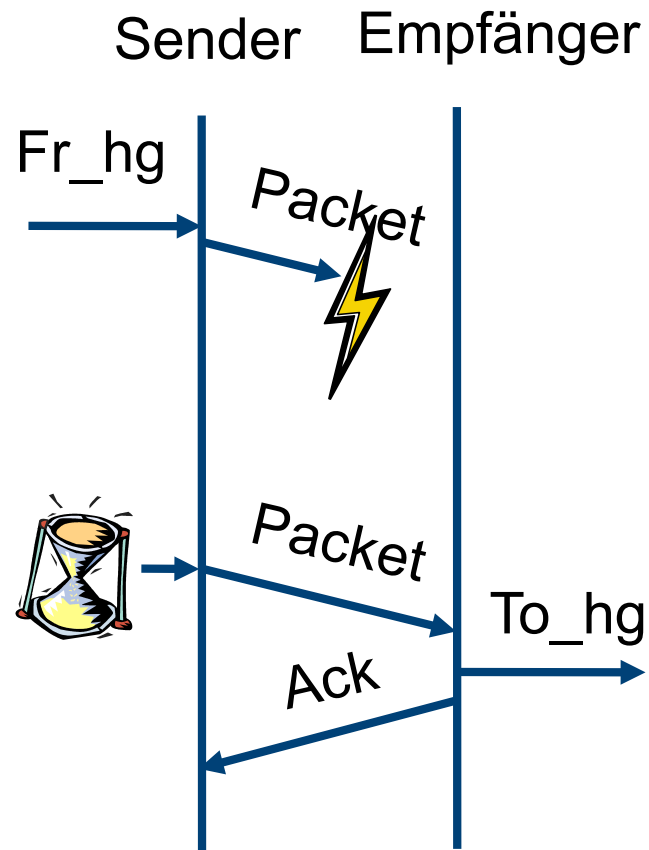
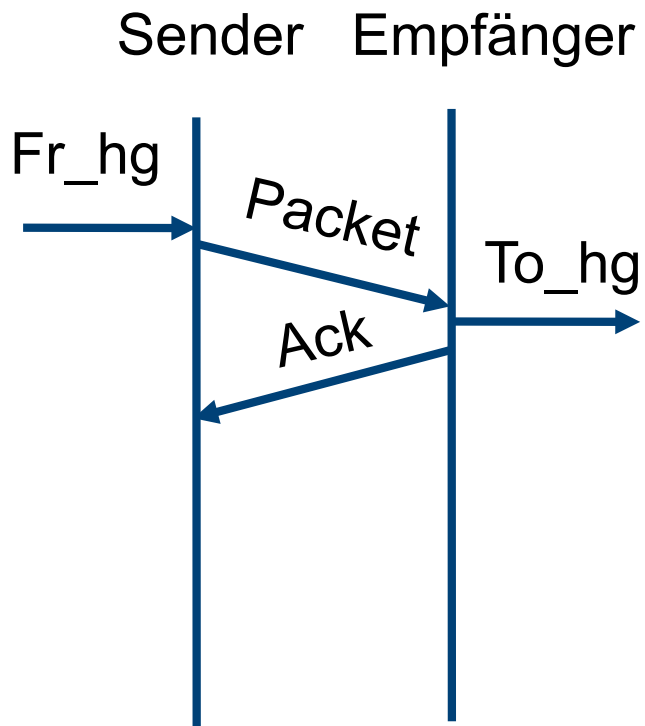
- Sender



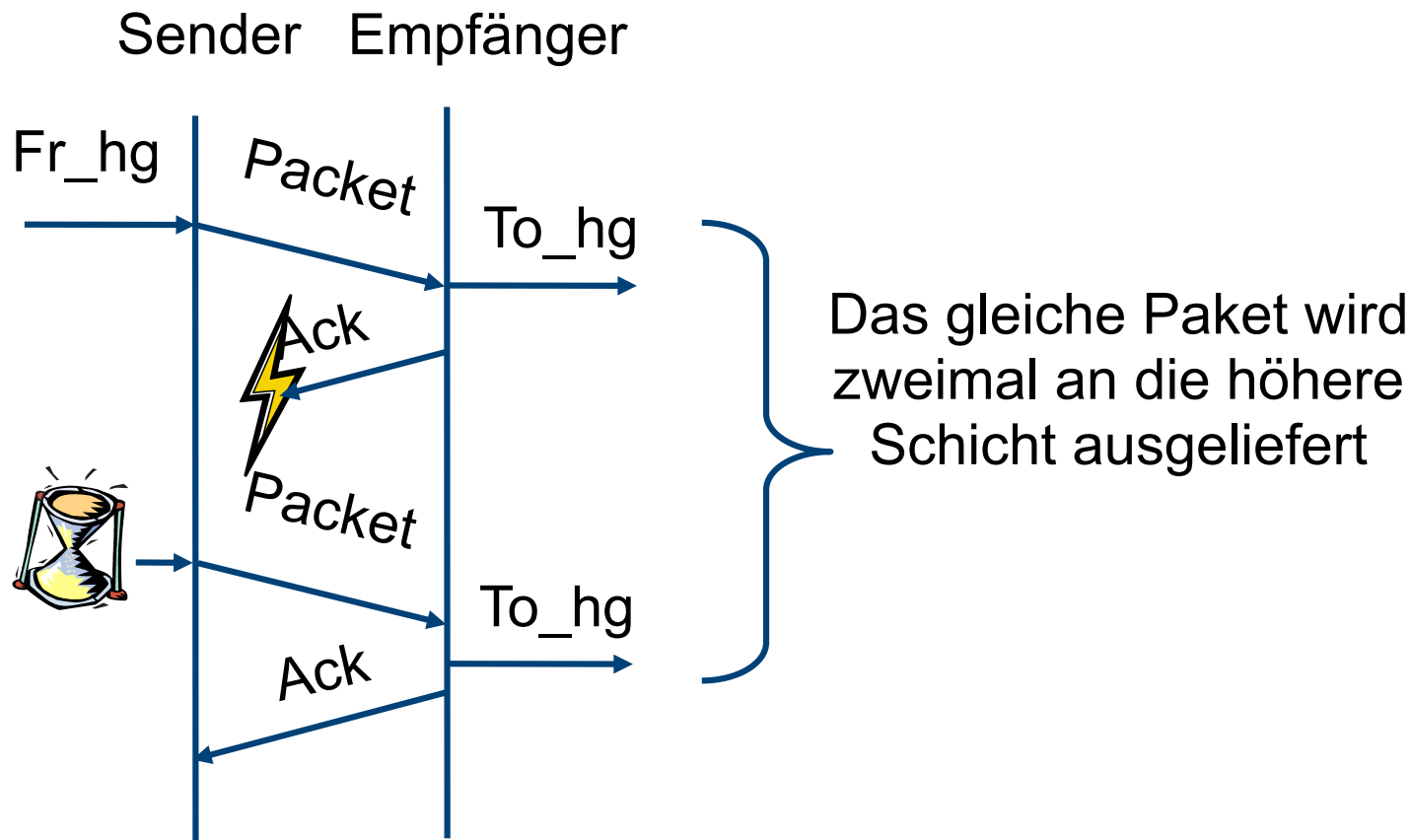
Empfänger



- Protokoll etabliert elementare Flusskontrolle

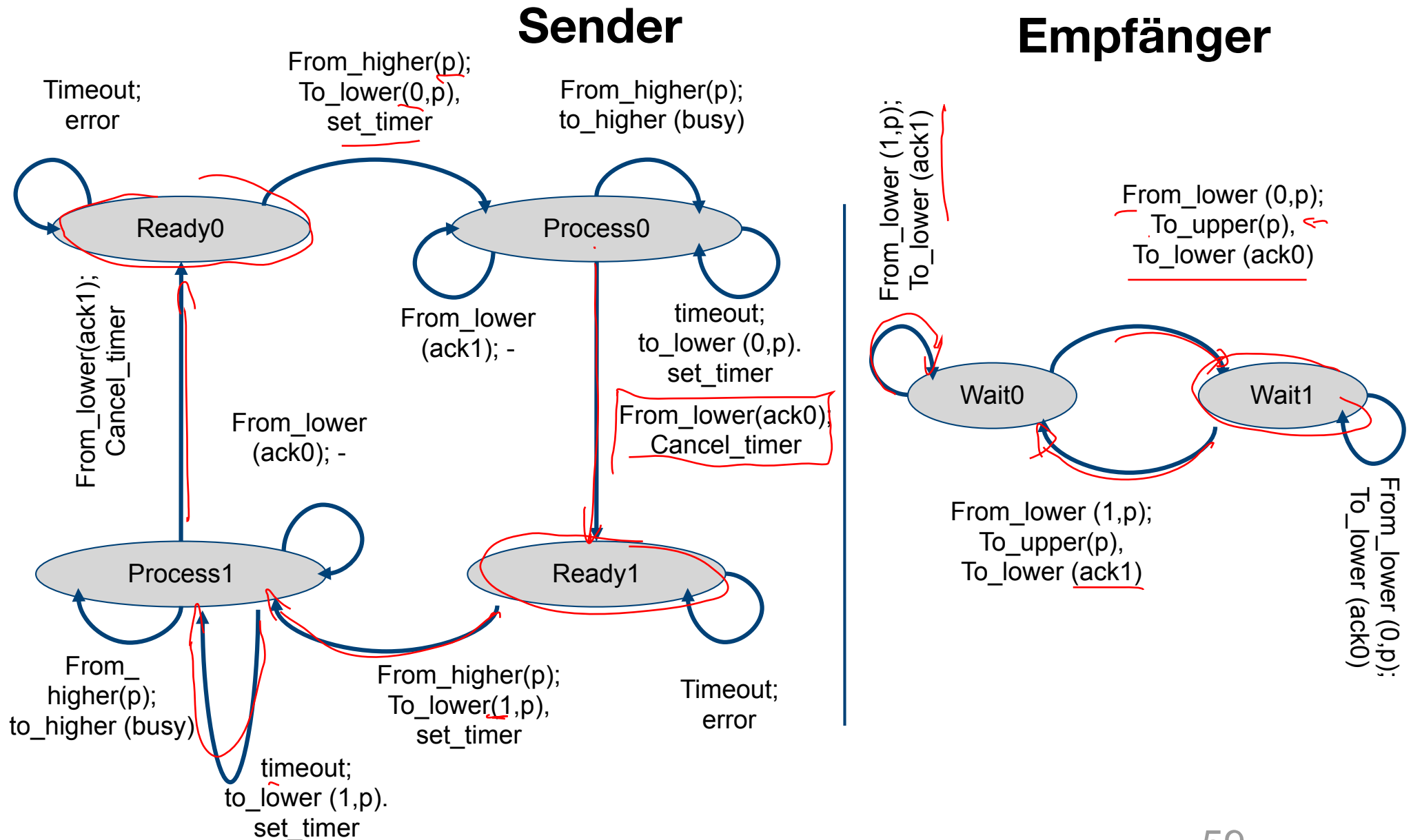


- 2. Fall: Verlust von Bestätigung



- Sender kann nicht zwischen verlorenem Paket und verlorener Bestätigung unterscheiden
 - Paket muss neu versendet werden
- Empfänger kann nicht zwischen Paket und redundanter Kopie eines alten Pakets unterscheiden
 - Zusätzliche Information ist notwendig
- Idee:
 - Einführung einer Sequenznummer in jedes Paket, um den Empfänger Identifikation zu ermöglichen
 - Sequenznummer ist im Header jedes Pakets
 - Hier: nur 0 oder 1
- Notwendig in Paket und Bestätigung
 - In der Bestätigung wird die Sequenznummer des letzten korrekt empfangenen Pakets mitgeteilt
 - (reine Konvention)

3. Versuch: Bestätigung und Sequenznummern



3. Version

Alternating Bit Protocol

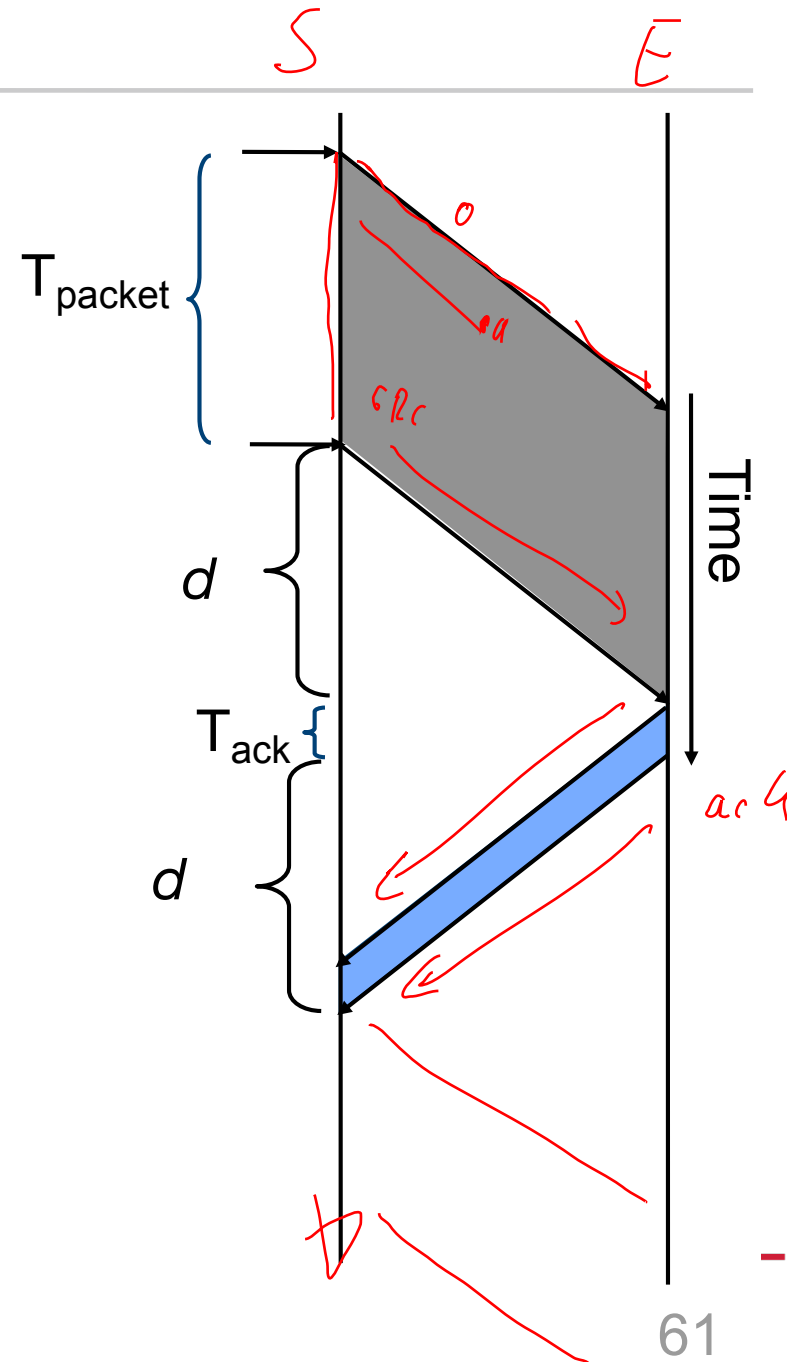
- Die 3. Version ist eine korrekte Implementation eines verlässlichen Protokolls über einen gestörten Kanal
 - Alternating Bit Protokoll
 - aus der Klasse der Automatic Repeat reQuest (ARQ) Protokolle
 - beinhaltet auch eine einfache Form der Flusskontrolle
- Zwei Aufgaben einer Bestätigung
 - Bestätigung, dass Paket angekommen ist
 - Erlaubnis ein neues Paket zu schicken

Alternating Bit Protocol – Effizienz

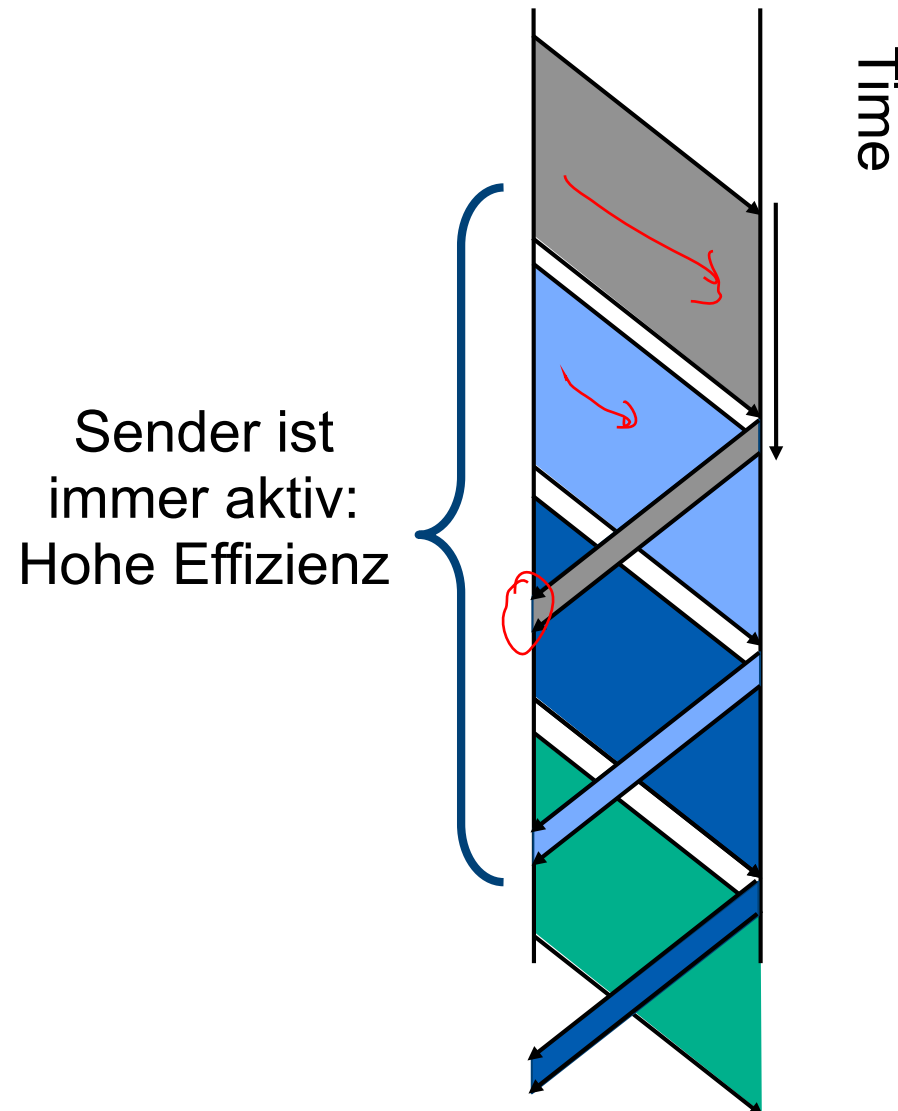
■ Effizienz η

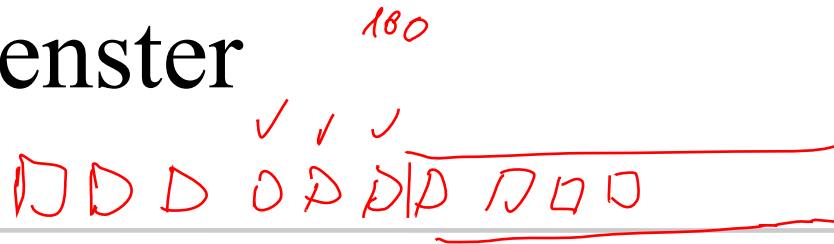
- Definiert als das Verhältnis zwischen
 - der Zeit um zu senden
 - und der Zeit bis neue Information gesendet werden kann
 - (auf fehlerfreien Kanal)
- $\eta = \frac{T_{\text{packet}}}{T_{\text{packet}} + d + T_{\text{ack}} + d}$

- Bei großen Delay ist das Alternating Bit Protocol nicht effizient



- Durchgehendes Senden von Paketen erhöht Effizienz
 - Mehr “ausstehende” nicht bestätigte Pakete erhöhen die Effizienz
 - “Pipeline” von Paketen
- Nicht mit nur 1-Bit-Sequenznummer möglich

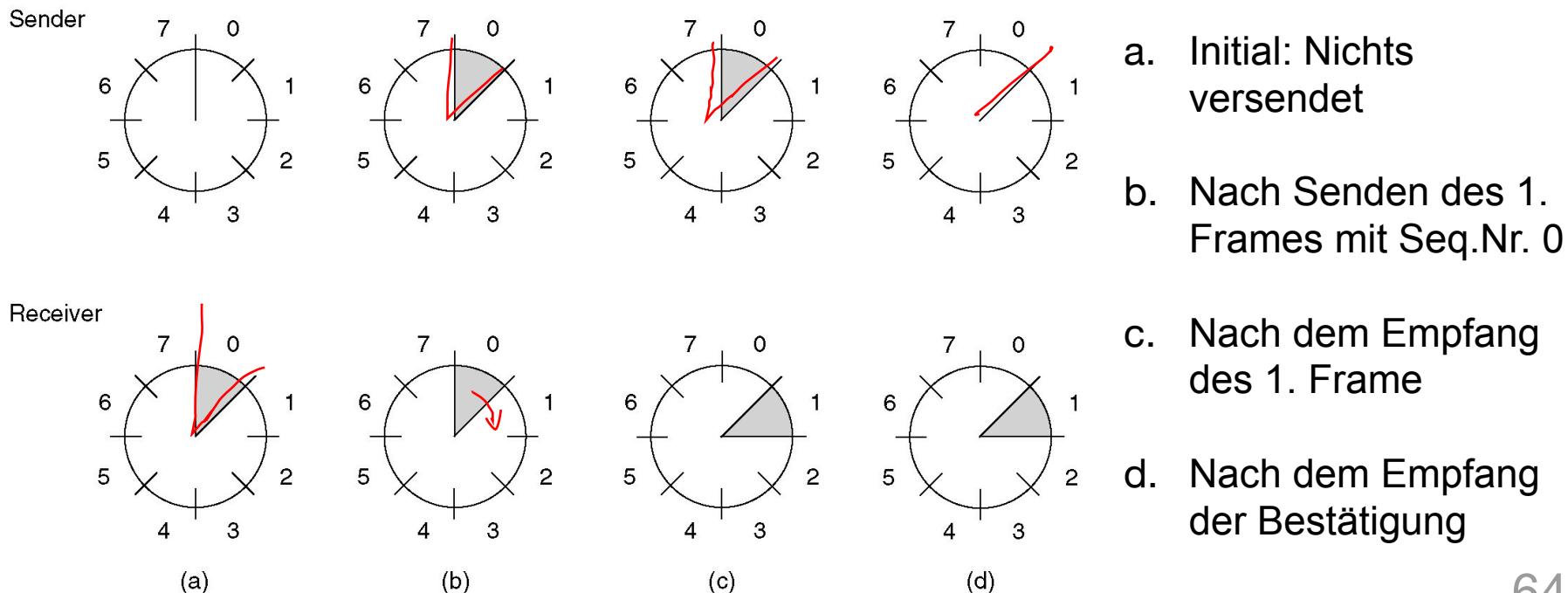




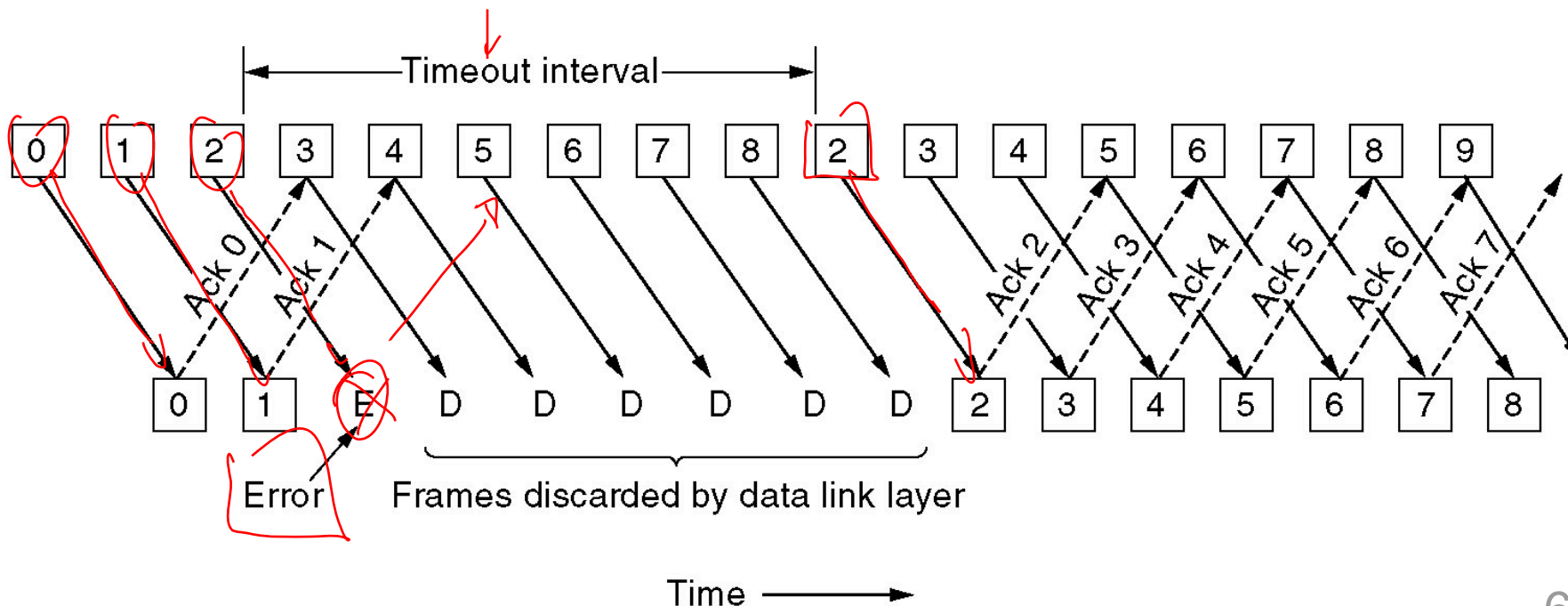
- Der Raum für Sequenznummern wird vergrößert
 - auf n Bits oder 2^n Sequenznummern
- Nicht alle davon können gleichzeitig verwendet werden
 - auch bei Alternating Bit Protocol nicht möglich
- “Gleitende Fenster” (sliding windows) bei Sender und Empfänger behandeln dieses Problem
 - Sender: Sende-Fenster ✓
 - Folge von Sequenznummer, die zu einer bestimmten Zeit gesendet werden können ✓
 - Empfänger: Empfangsfenster ✓
 - Folge von Sequenznummer, die er zu einer bestimmten Zeit zu akzeptieren bereit ist ✓
 - Größe der Fenster können fest sein oder mit der Zeit verändert werden
 - Fenstergröße entspricht Flusskontrolle

Beispiel

- “Sliding Window”-Beispiel für $n=3$ und fester Fenstergröße = 1
- Der Sender zeigt die momentan unbestätigten Sequenznummern an
 - Falls die maximale Anzahl nicht bestätigter Frames bekannt ist, dann ist das das Sende-Fenster



- Annahme:
 - Sicherungsschicht muss alle Frames korrekt in der richtigen Reihenfolge verschicken
 - Sender "pipelined" Paket zur Erhöhung der Effizienz
- Bei Paketverlust:
 - werden alle folgenden Pakete ebenfalls fallen gelassen



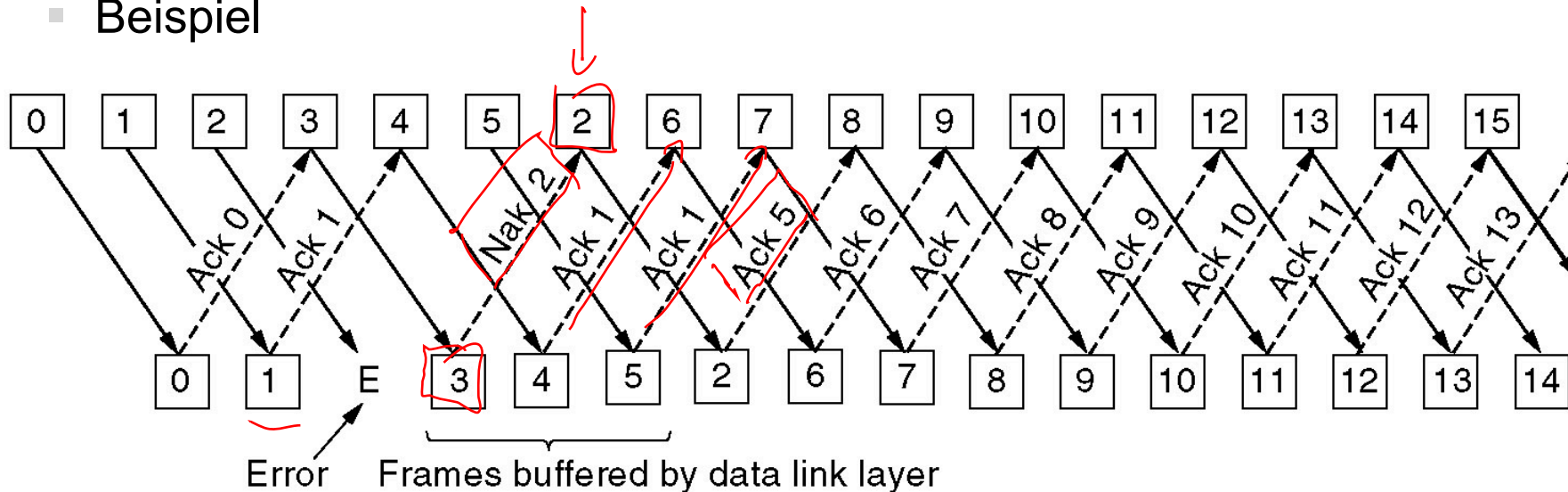
- Mit Empfangsfenster der Größe 1 können die Frames, die einem verlorenen Frame folgen, nicht durch den Empfänger bearbeitet werden
 - Sie können einfach nicht bestätigt werden, da nur eine Bestätigung für das letzte korrekt empfangene Paket verschickt wird
- Der Sender wird einen “Time-Out” erhalten
 - Alle in der Zwischenzeit versandten Frames müssen wieder geschickt werden
 - “Go-back N” Frames!
- Kritik
 - - Unnötige Verschwendung des Mediums
 - - Spart aber Overhead beim Empfänger

Selektierte Wiederholung

■ Angenommen

- der Empfänger kann die Pakete puffern, welche in der Zwischenzeit angekommen sind
- d.h. das Empfangsfenster ist größer als 1

■ Beispiel



- Der Empfänger informiert dem Sender fehlende Pakete mit negativer Bestätigung
- Der Sender verschickt die fehlenden Frames selektiv
- Sobald der fehlende Frame ankommt, werden alle (in der korrekten Reihenfolge) der Vermittlungsschicht übergeben

Duplex-Betrieb und Huckepack

→ Simplex

- Senden von Informationen in einer Richtung

→ Duplex

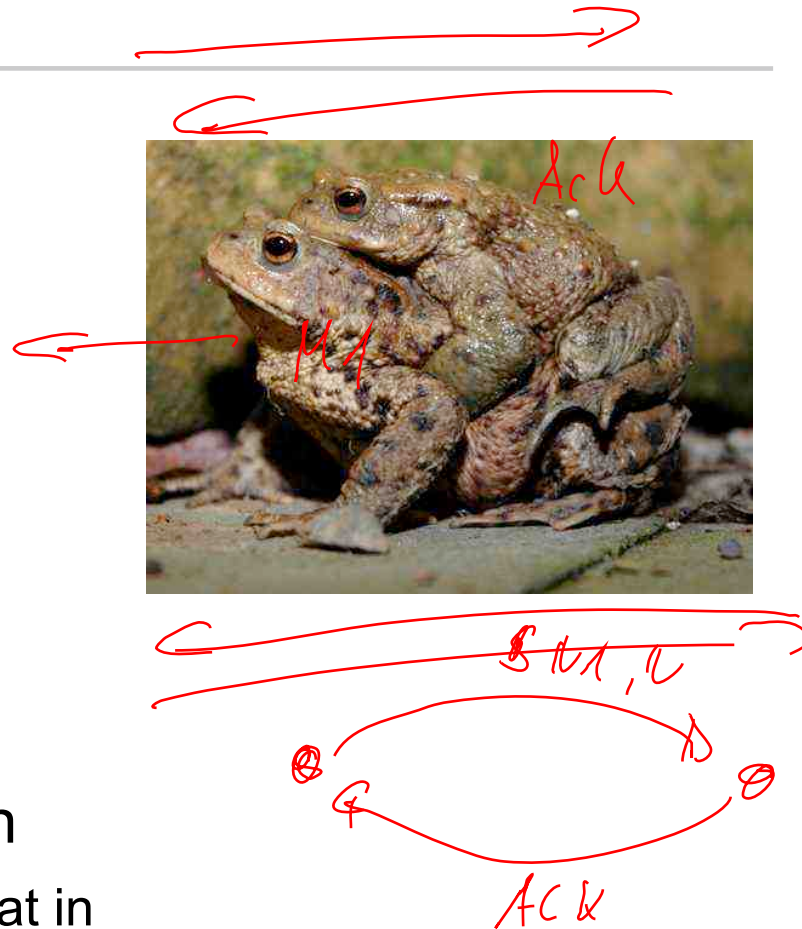
- Senden von Informationen in beide Richtungen

■ Bis jetzt:

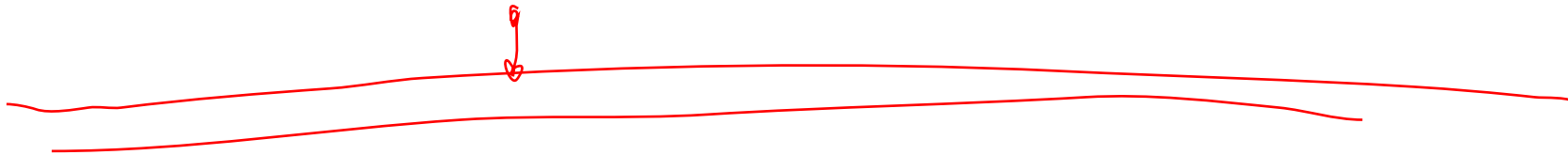
- Simplex in der Vermittlungsschicht
- Duplex in der Sicherungsschicht

■ Duplex in den höheren Schichten

- Nachrichten und Datenpakete separat in jeder Richtung
- Oder Rucksack-Technik
 - Die Bestätigung wird im Header eines entgegen kommenden Frames gepackt



- Die Bitübertragung kann erst stattfinden, wenn das Medium reserviert wurde
 - Funkfrequenz bei drahtloser Verbindung (z.B. W-LAN 802.11, GSM, GPRSM)
 - ⊗ Zeitraum bei einem Kabel mit mehreren Rechnern (z.B. Ethernet)
- Aufgabe der Sicherungsschicht
 - Koordination zu komplex für die “einfache” Bitübertragungsschicht



○ Statisches Multiplexen *Band*

→ ■ Dynamische Kanalbelegung

→ - Kollisionsbasierte Protokolle

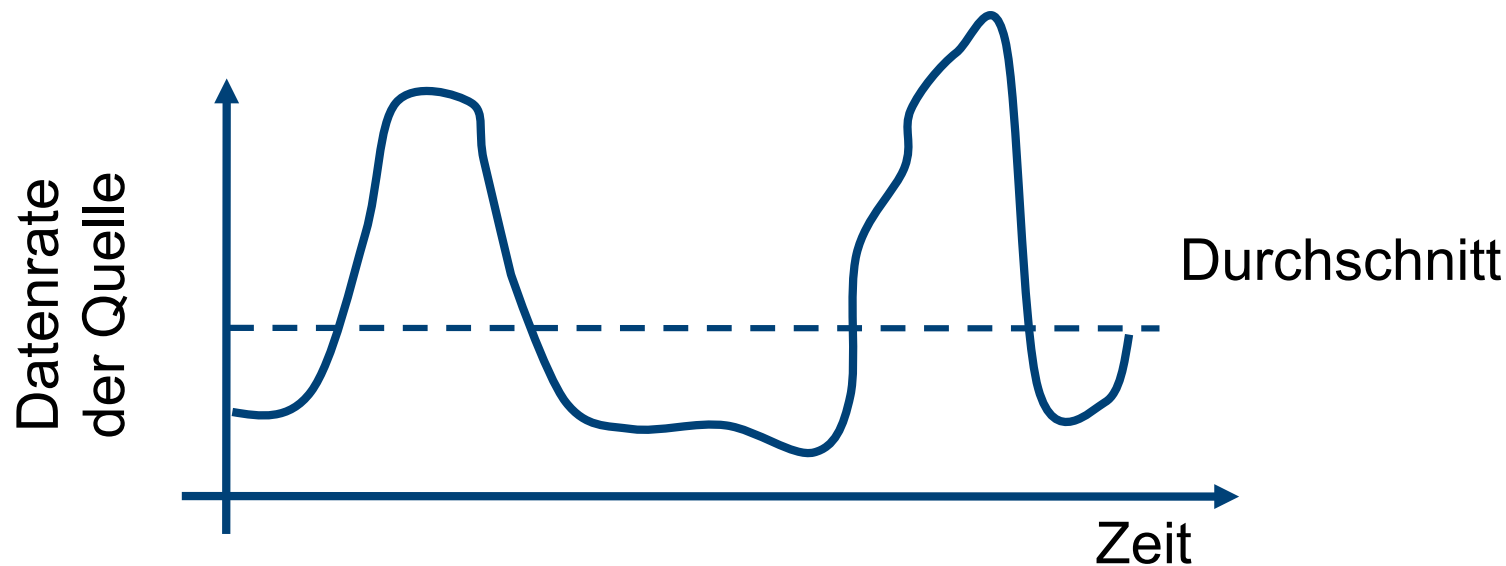
→ - Kollisionsfreie Protokolle (contention-free)

→ - Protokolle mit beschränktem Wettbewerb (limited contention)

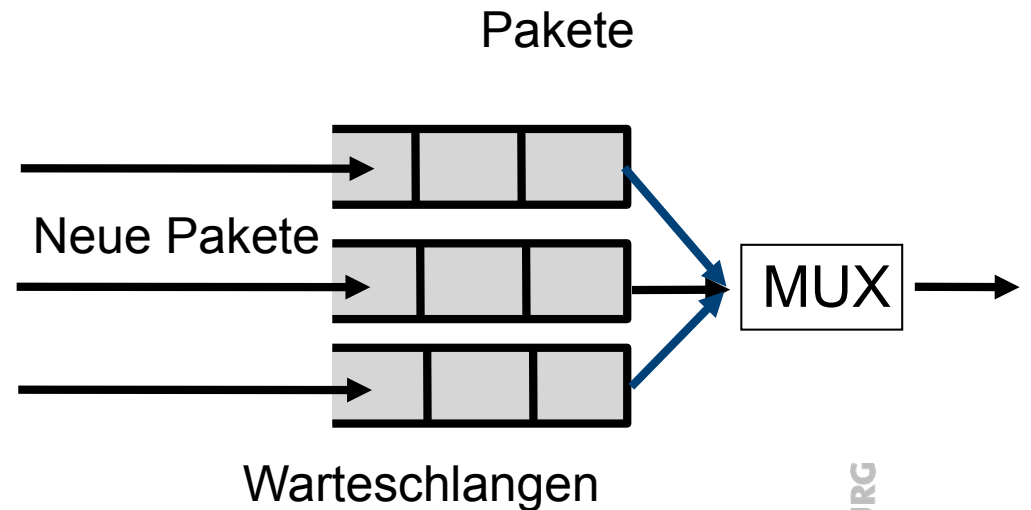
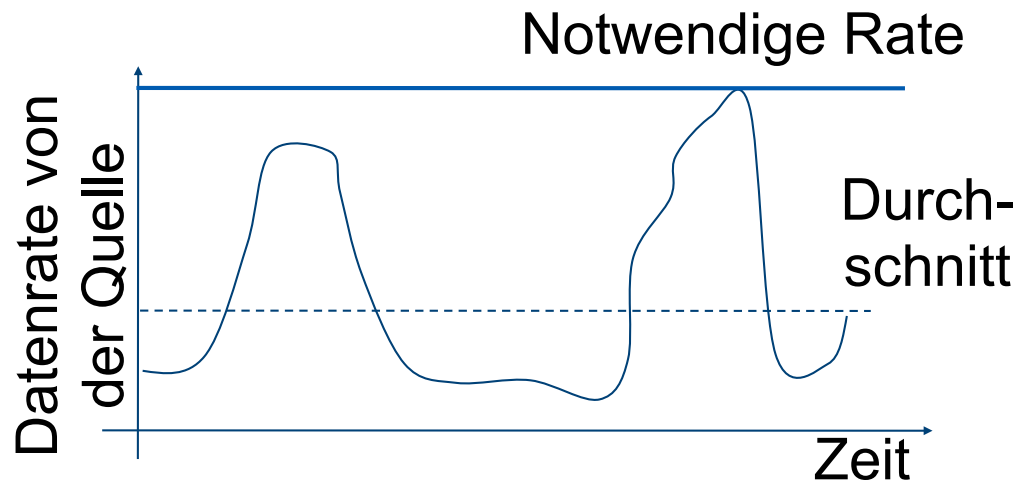
■ Fallbeispiel: Ethernet

- Gegeben sei eine einzelne Leitung (Ressource)
- Mehreren Kommunikationsverbindungen werden feste Zeiträume/Kanäle (slots/channels) zugewiesen
 - Oder: Feste Frequenzbänder werden ihnen zugewiesen
- Feste Datenraten und entsprechenden Anteilen am Kanal
 - Quellen lasten die Leitung aus

- Problem: Verkehrsspitzen (bursty traffic)
 - Definition: Großer Unterschied zwischen Spitze und Durchschnitt
 - In Rechnernetzwerken: Spitze/Durchschnitt = 1000/1 nicht ungewöhnlich



- Leitung für statisches Multiplexen:
- entweder
 - Genügend große Kapazität um mit dem Peak fertig zu werden
 - Verschwendung, da die Durchschnittsrate den Kanal nicht auslasten wird
- oder
 - Ausgelegt für Durchschnittsrate
 - Versehen mit Warteschlangen (queue)
 - Vergrößerung der Verzögerung (delay) der Pakete



- Vergleich der Verzögerung
- Ausgangsfall:
 - Kein Multiplexing
 - Einfacher Datenquelle mit Durchschnittsrate ρ (bits/s) und der Leitungskapazität C bits/s
 - Sei T die Verzögerung
- Multiplex-Fall
 - Die Datenquelle wird in N Quellen unterteilt mit der selben Datenrate
 - Statischer Multiplex über die selbe Leitung
 - Dann ergibt sich (im wesentlichen) die Verzögerung: $N T$
- Schluss: Statisches Multiplexen vergrößert den Delay eines Pakets in der Regel um den Faktor N
 - Grund: Bei einer Verkehrsspitze sind $n-1$ Kanäle leer

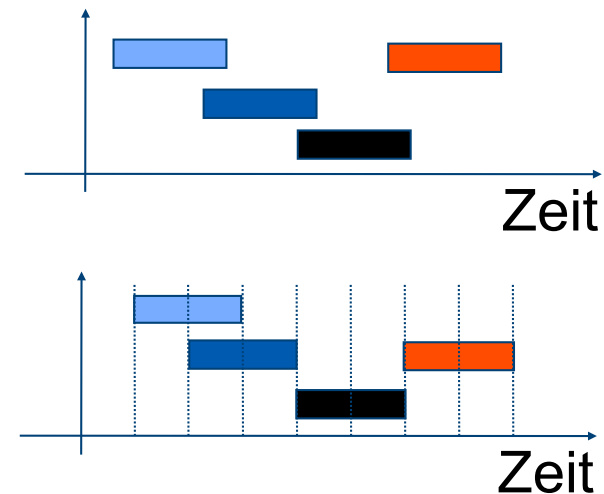
- Statisches Multiplexen
- Dynamische Kanalbelegung
 - Kollisionsbasierte Protokolle
 - Kollisionsfreie Protokolle (contention-free)
 - Protokolle mit beschränktem Wettbewerb (limited contention)
- Fallbeispiel: Ethernet

- Statisches Multiplexing ist nicht geeignet für Datenverbindung mit Spitzen
- Alternative: Zuweisung des Slots/Kanals an die Verbindung mit dem größten Bedarf
 - Dynamische Medium-Belegung
 - statt fester
- Der Mediumzugriff wird organisiert:
 - Mediumszugriff-Protokoll (Medium Access Control protocol - MAC)

- Stationsmodell (terminal model)
 - N unabhängige Stationen möchten eine Leitung/ Ressource teilen
 - Mögliches Lastmodell:
 - Wahrscheinlichkeit, dass ein Paket im Intervall der Länge Δt erzeugt wird ist $\lambda \Delta t$ für eine Konstante λ
- Eine Leitung/Kanal
 - für alle Stationen
 - Keine weitere Verbindungen möglich
- Collision assumption
 - Nur ein einfacher Frame kann auf dem Kanal übertragen werden
 - Zwei (oder mehr) sich zeitlich überschneidende Frames kollidieren und werden gelöscht
 - Noch nicht einmal Teile kommen an

■ Zeitmodelle

- Kontinuierlich
 - Übertragungen können jeder Zeit beginnen (keine zentrale Uhr)
- Diskret (Slotted time)
 - Die Zeitachse ist in Abschnitte (slots) unterteilt
 - Übertragungen können nur an Abschnittsgrenzen starten
 - Slots können leer (idle), erfolgreich (mit Übertragung) sein oder eine Kollision beinhalten

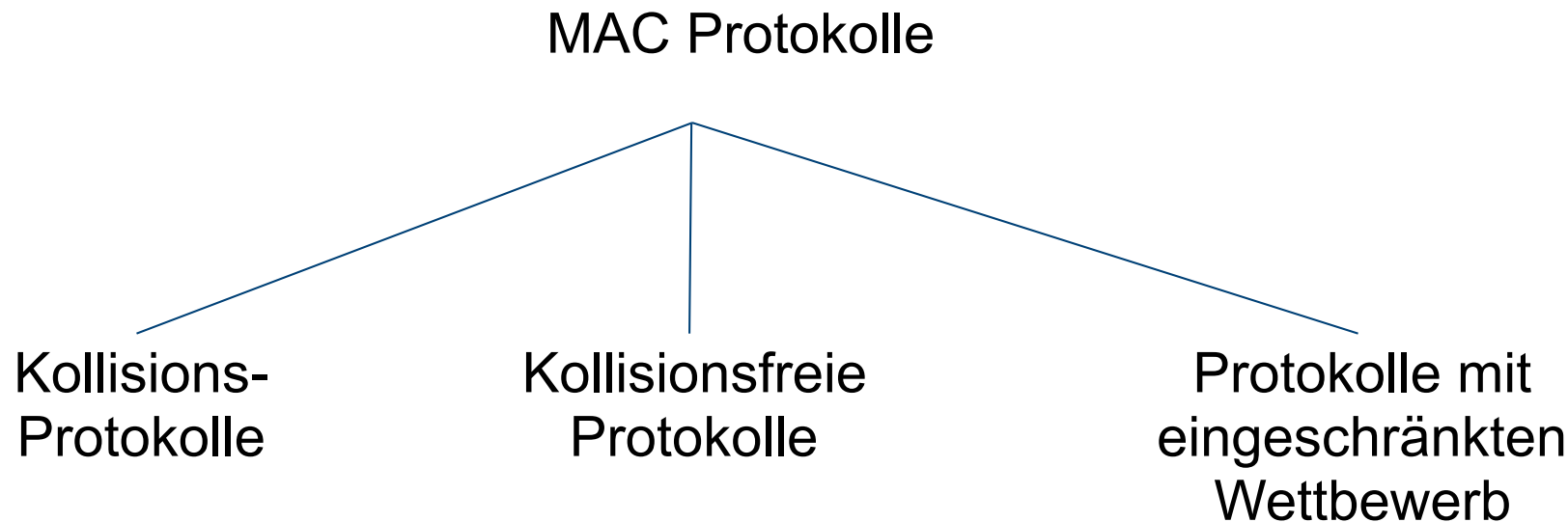


■ Träger-Messung (Carrier Sensing)

- Stationen können erkennen ob der Kanal momentan von anderen Stationen verwendet wird
 - Nicht notwendigerweise zuverlässig

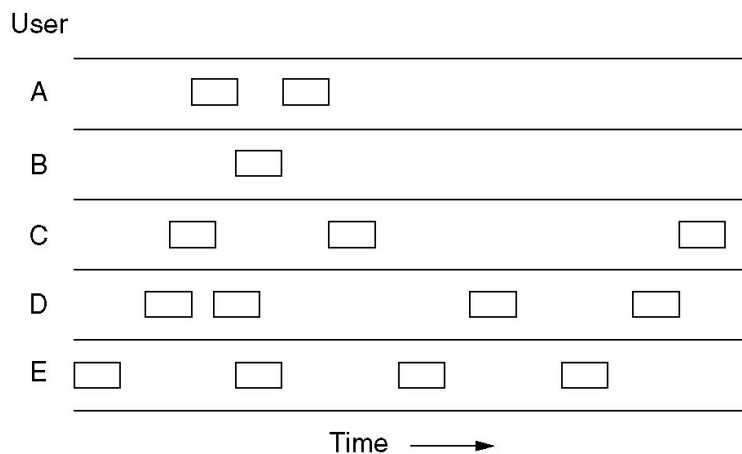
- Methoden zur Bewertung der Effizienz einer Kanaluweisung
- Durchsatz (throughput)
 - Anzahl Pakete pro Zeiteinheit
 - Besonders bei großer Last wichtig
- Verzögerung (delay)
 - Zeit für den Transport eines Pakets
 - Muss bei geringer Last gut sein
- Gerechtigkeit (fairness)
 - Gleichbehandlung aller Stationen
 - Fairer Anteil am Durchsatz und bei Delay

- Unterscheidung: Erlaubt das Protokoll Kollisionen?
 - Als Systemscheidung
 - Die unbedingte Kollisionsvermeidung kann zu Effizienzeinbußen führen

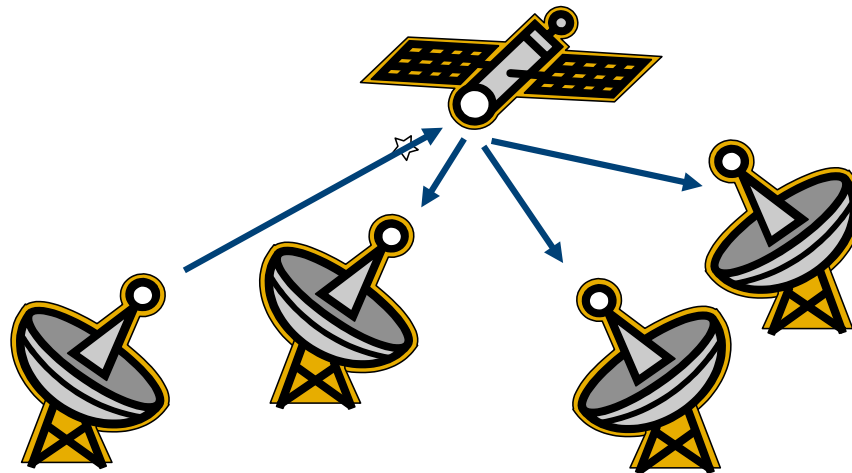


System mit Kollisionen: **Contention System**

- Algorithmus
 - Sobald ein Paket vorhanden ist, wird es gesendet
- Ursprung
 - 1985 by Abrahamson et al., University of Hawaii
 - Ziel: Verwendung in Satelliten-Verbindung



Pakete werden zu beliebigen Zeiten übertragen



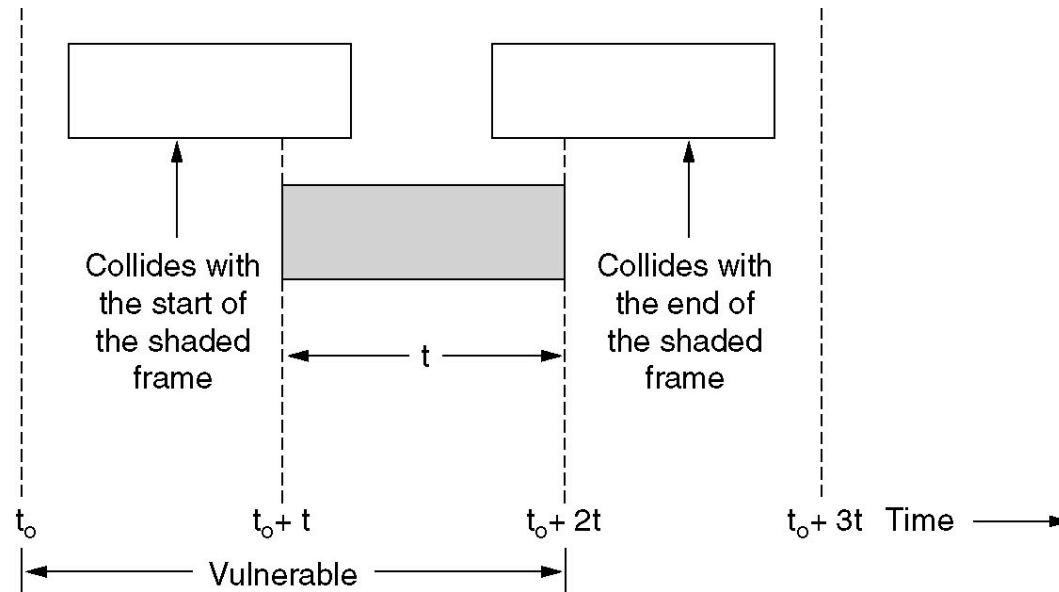
- Vorteile
 - Einfach
 - Keine Koordination notwendig
- Nachteile
 - Kollisionen
 - Sender überprüft den Kanalzustand nicht
 - Sender hat keine direkte Methode den Sende-Erfolg zu erfahren
 - Bestätigungen sind notwendig
 - Diese können auch kollidieren

- Betrachte Poisson-Prozess zur Erzeugung von Paketen
 - Entsteht durch “unendlich” viele Stationen, die sich gleich verhalten
 - Zeit zwischen zwei Sende-Versuchen ist exponentiell verteilt
 - Sei G der Erwartungswert der Übertragungsversuche pro Paketlänge
 - Alle Pakete haben gleiche Länge
 - Dann gilt

$$P[k \text{ Versuche}] = \frac{G^k}{k!} e^{-G}$$

- Um eine erfolgreiche Übertragung zu erhalten, darf keine Kollision mit einem anderen Paket erfolgen
- Wie lautet die Wahrscheinlichkeit für eine solche Übertragung?

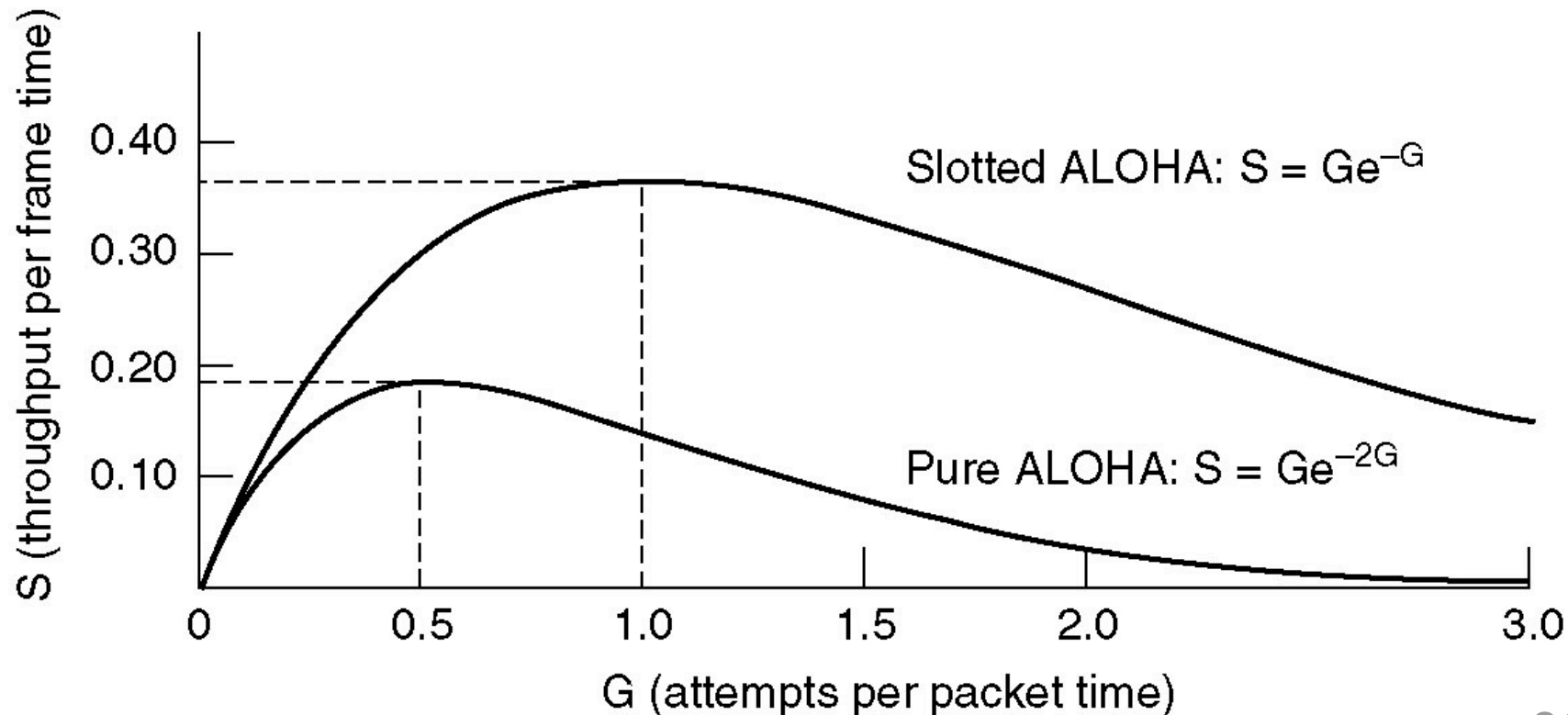
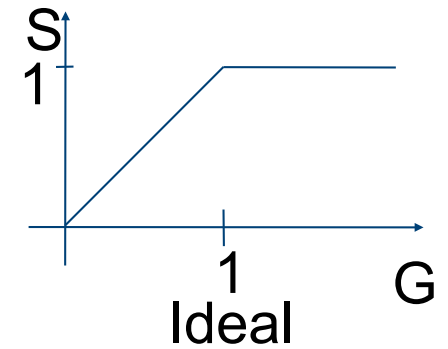
- Ein Paket X wird gestört, wenn
 - ein Paket kurz vor X startet
 - wenn ein Paket kurz vor dem Ende von X startet
- Das Paket wird erfolgreich übertragen, wenn in einem Zeitraum von zwei Paketen kein (anderes) Paket übertragen wird



- ALOHAs Problem:
 - Lange Verwundbarkeit eines Pakets
- Reduktion durch Verwendung von Zeitscheiben (Slots)
 - Synchronisation wird vorausgesetzt
- Ergebnis:
 - Verwundbarkeit wird halbiert
 - Durchsatz wird verdoppelt
 - $S(G) = Ge^{-G}$
 - Optimal für $G=1$, $S=1/e$

Durchsatz in Abhängigkeit der Last

- Für (slotted) ALOHA ist eine geschlossene Darstellung in Abhängigkeit von G möglich
- Kein gutes Protokoll
 - Durchsatz bricht zusammen, wenn die Last zunimmt



- CSMA-Problem:

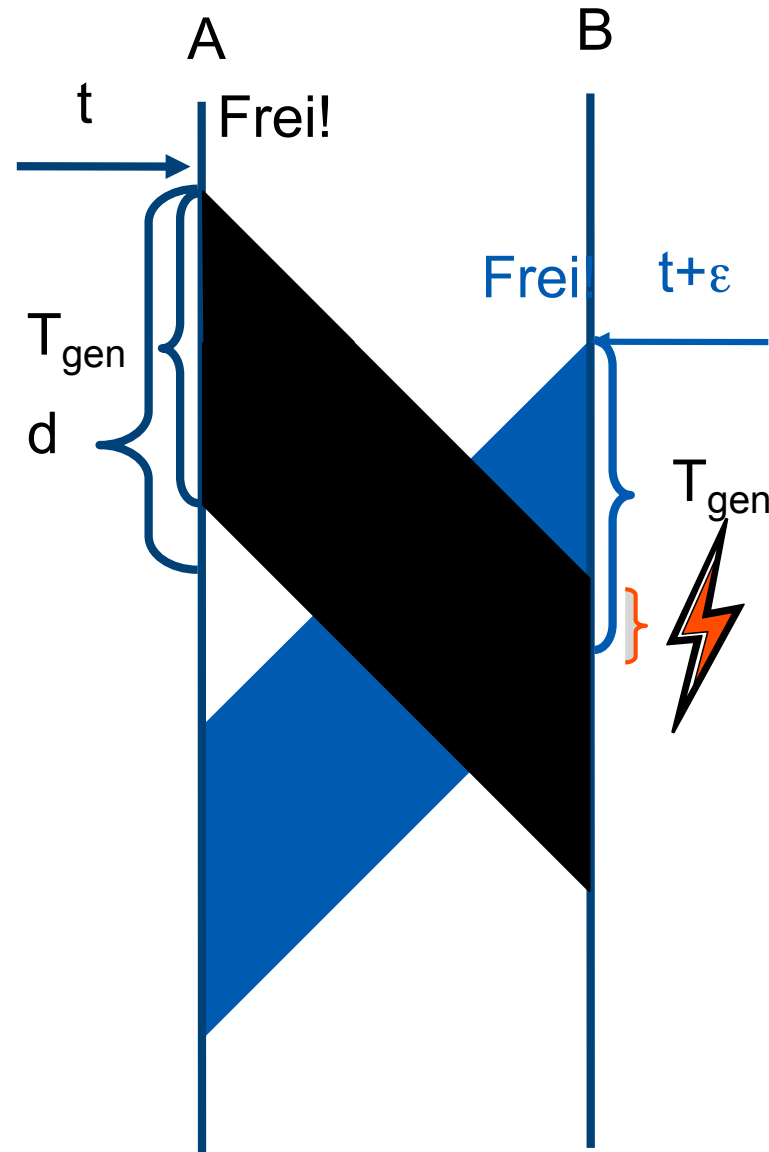
- Übertragungszeit d
(propagation delay)

- Zwei Stationen

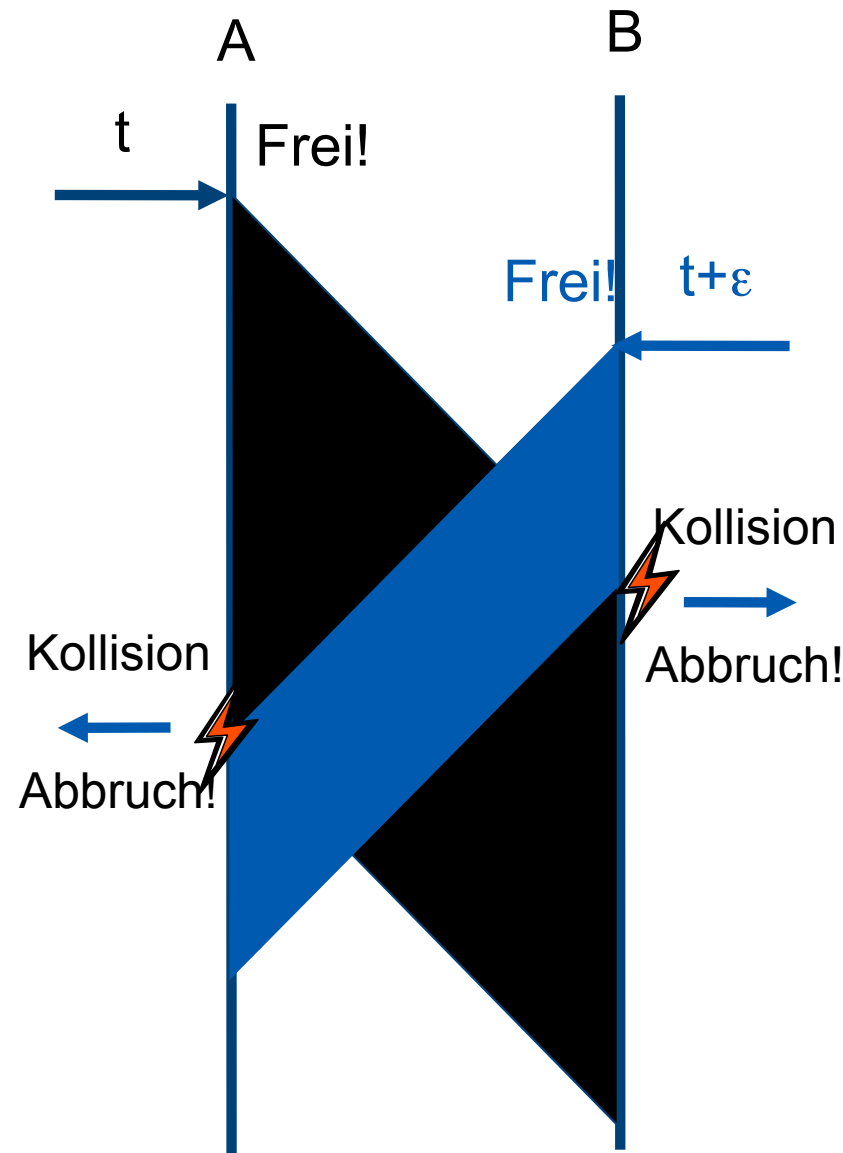
- starten Senden zu den Zeitpunkten t und $t+\epsilon$ mit $\epsilon < d$
- sehen jeweils einen freien Kanal

- Zweite Station

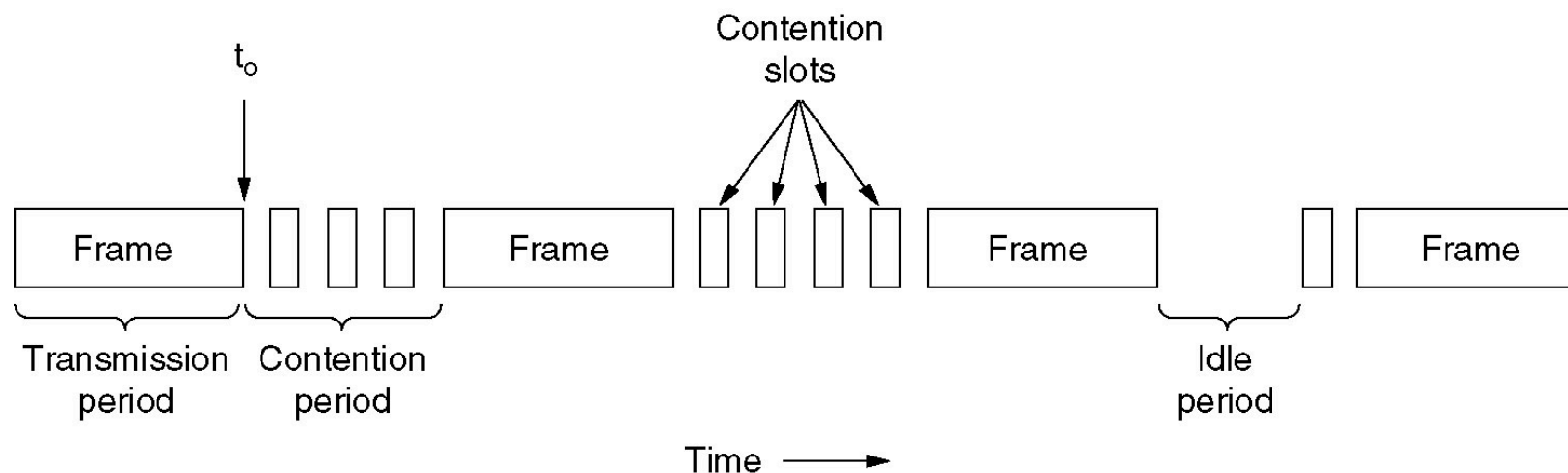
- verursacht eine Kollision



- Falls Kollisionserkennung (collision detection) möglich ist,
 - dann beendet der spätere Sender seine Übertragung
 - Zeitverschwendung wird reduziert, da mindestens eine Nachricht (die erste) übertragen wird
- Fähigkeit der Kollisionserkennung hängt von der Bitübertragungsschicht ab
- CSMA/CD – Carrier Sense Multiple Access/Collision Detection
- Collision Detection
 - setzt gleichzeitiges Abhören des Kanals nach Kollisionen voraus
 - Ist das was auf dem Kanal geschieht, identisch zu der eigenen Nachricht?



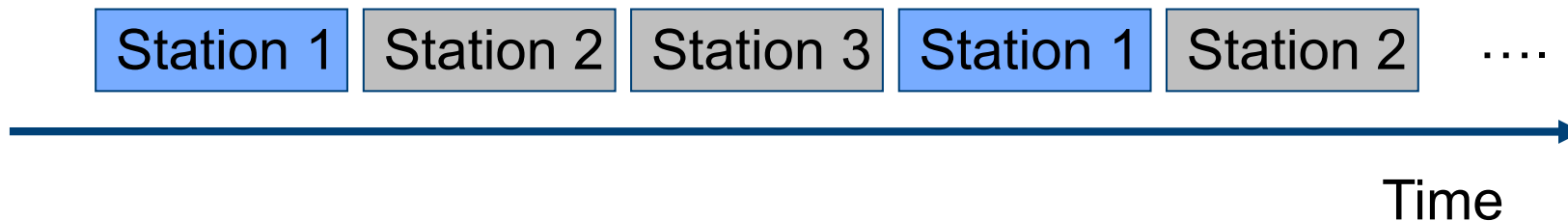
- Leer-Phase (IDLE)
 - Keine Station sendet einen Frame
 - Wettbewerbsphase (Contention Period)
 - Kollisionen entstehen, Übertragungen werden abgebrochen
 - Übertragungsphase (Transmission Period)
 - Keine Kollision, effektiver Teil des Protokolls
- Es gibt nur Wettbewerbs-, Übertragungsphasen und Leer-Phasen



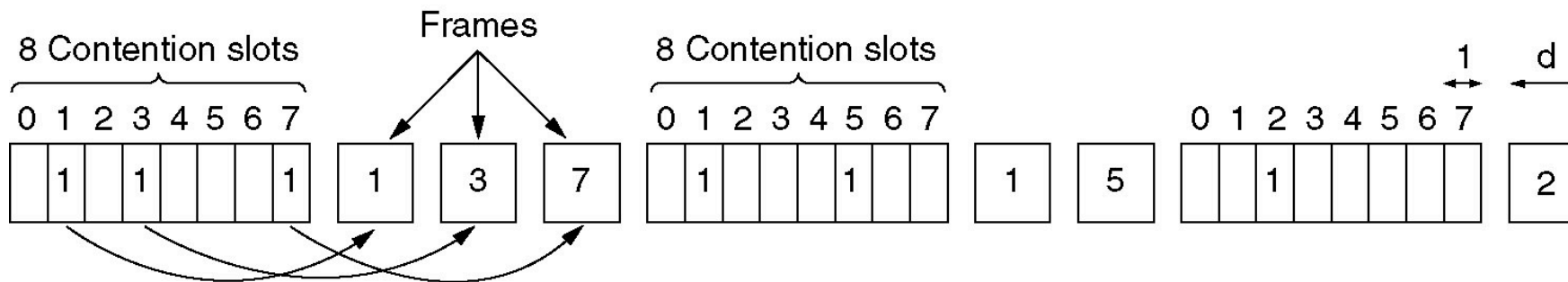
- Nach der Kollision:
- Algorithmus binary exponential backoff
 - $k := 2$
 - Solange Kollision beim letzten Senden
 - Wähle t gleichwahrscheinlich zufällig aus $\{0, \dots, k-1\}$
 - Warte t Zeit-Slots
 - Sende Nachricht (Abbruch bei Collision Detection)
 - $k := 2k$
- Algorithmus
 - passt Wartezeit dynamisch an die Anzahl beteiligter Stationen an
 - sorgt für gleichmäßige Auslastung des Kanals
 - ist fair (auf lange Sicht)

- Statisches Multiplexen
- Dynamische Kanalbelegung
 - Kollisionsbasierte Protokolle
 - Kollisionsfreie Protokolle (contention-free)
 - Protokolle mit beschränktem Wettbewerb (limited contention)
- Fallbeispiel: Ethernet

- Einfaches Beispiel: Statisches Zeit-Multiplexen (TDMA)
 - Jeder Station wird ein fester Zeit-Slot in einem sich wiederholenden Zeitschema zugewiesen
- Nachteile bekannt und diskutiert
- Gibt es dynamische kollisionsfreie Protokoll?



- Probleme von TDMA
 - Wenn eine Station nichts zu senden hat, dann wird der Kanal nicht genutzt
- Reservierungssystem: Bit-map protocol
 - Kurze statische Reservierung-Slots zur Ankündigung
 - Müssen von jeder Station empfangen werden



- Verhalten bei geringer Last
 - Falls keine Pakete verschickt werden, wird der (leere) Wettbewerbs-Slot wiederholt
 - Eine Station muss auf seinen Wettbewerbs-Slot warten
 - Erzeugt gewisse Verzögerung (delay)
- Verhalten bei hoher Last
 - Datenpakete dominieren die Kanalbelegung
 - Datenpakete sind länger als die Contention-Slots
 - Overhead ist vernachlässigbar
 - Guter und stabiler Durchsatz
- Bitmap ist ein Carrier-Sense Protokoll!

- Statisches Multiplexen
- Dynamische Kanalbelegung
 - Kollisionsbasierte Protokolle
 - Kollisionsfreie Protokolle (contention-free)
 - Protokolle mit beschränktem Wettbewerb (limited contention)
- Fallbeispiel: Ethernet

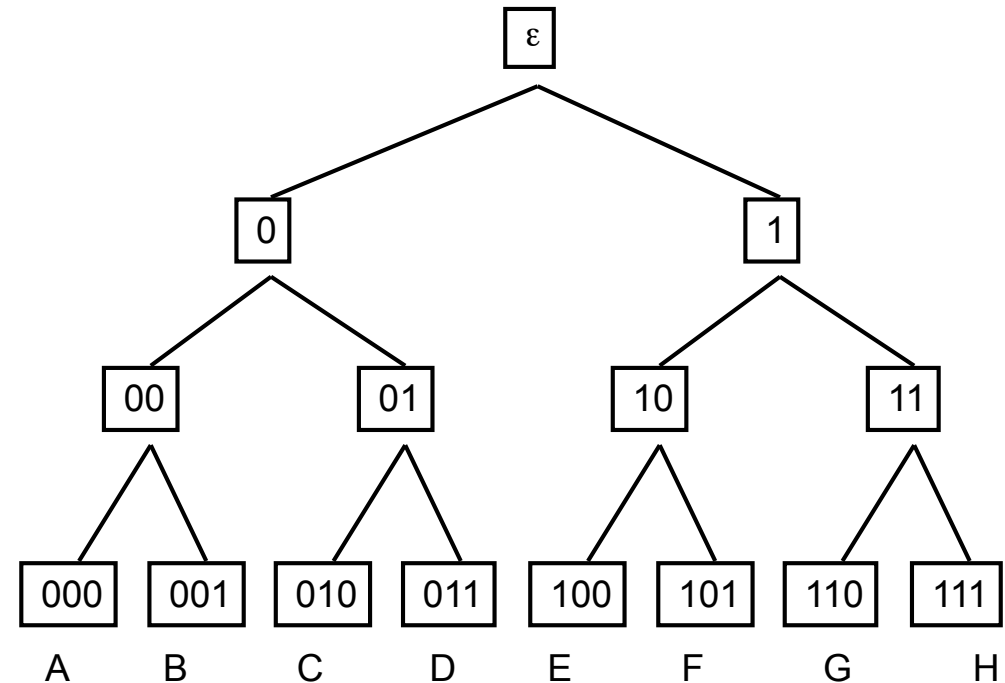
■ Ziel

- geringe Verzögerung bei kleiner Last
 - wie Kollisionsprotokolle
- hoher Durchsatz bei großer Last
 - wie kollisionsfreie Protokolle

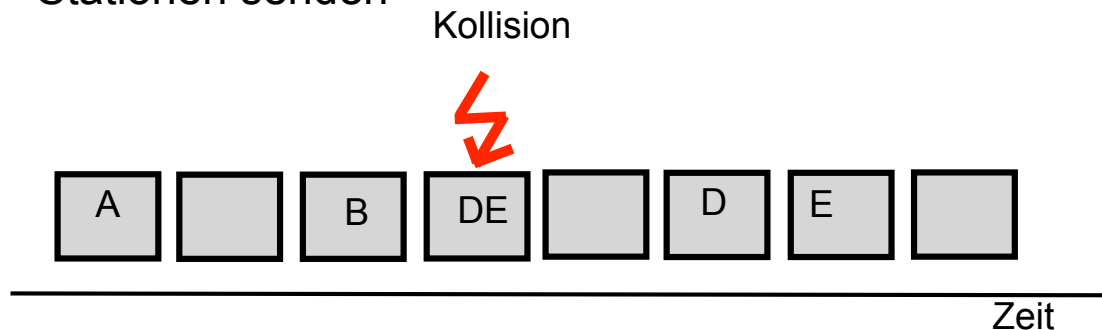
■ Idee

- Anpassung des Wettbewerb-Slots (contention slot) an die Anzahl der teilnehmenden Stationen
- Mehrere Stationen müssen sich dann diese Slots teilen

- Adaptives Baumprotokoll (adaptive tree walk)
- Ausgangspunkt:
 - Binäre, eindeutige Präsentation aller Knoten (ID)
 - Dargestellt in einem Baum
 - Synchronisiertes Protokoll
 - Drei Typen können unterschieden werden:
 - Keine Station sendet
 - Genau eine Station sendet
 - Kollision: mindestens zwei Stationen senden

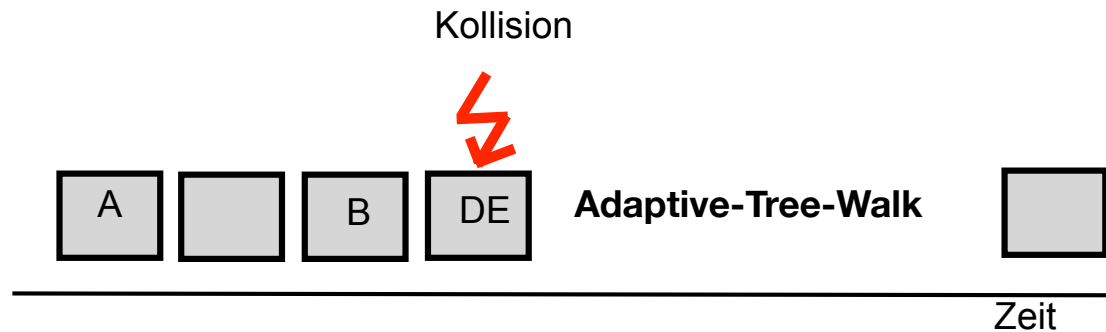


Stationen

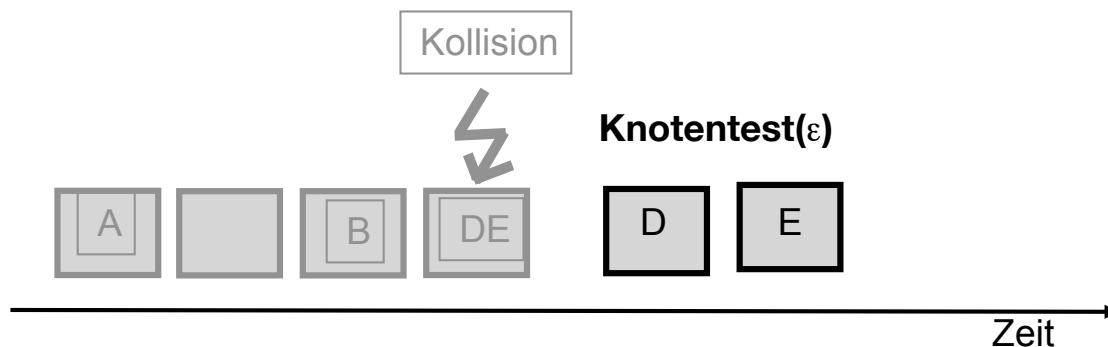
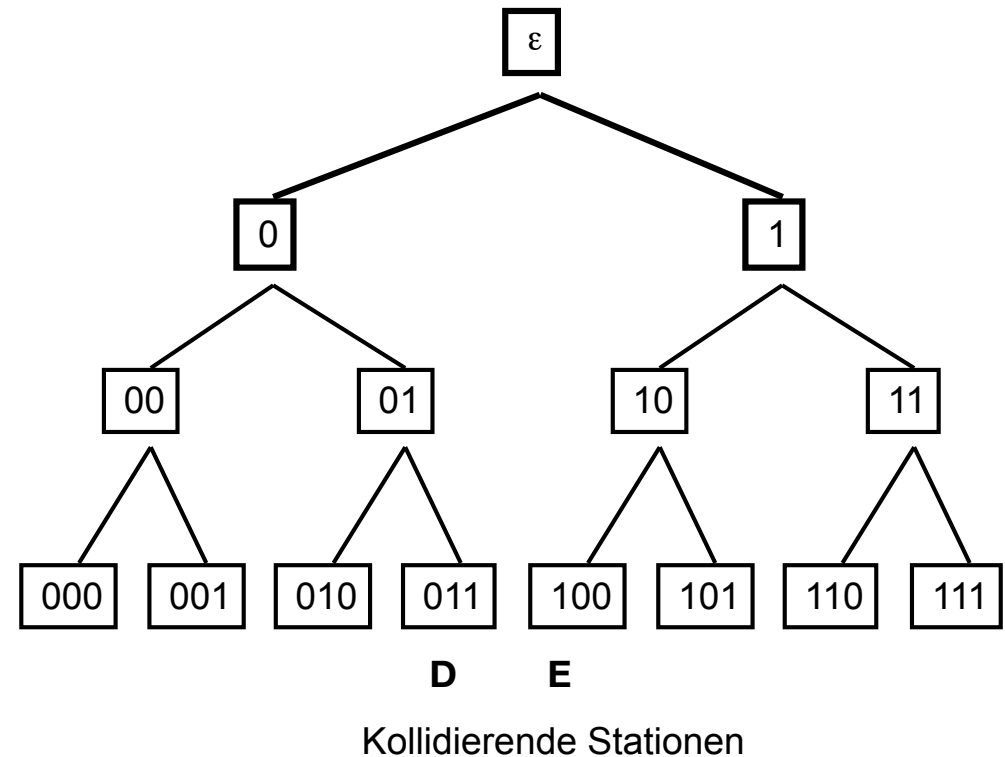


■ Basis-Algorithmus

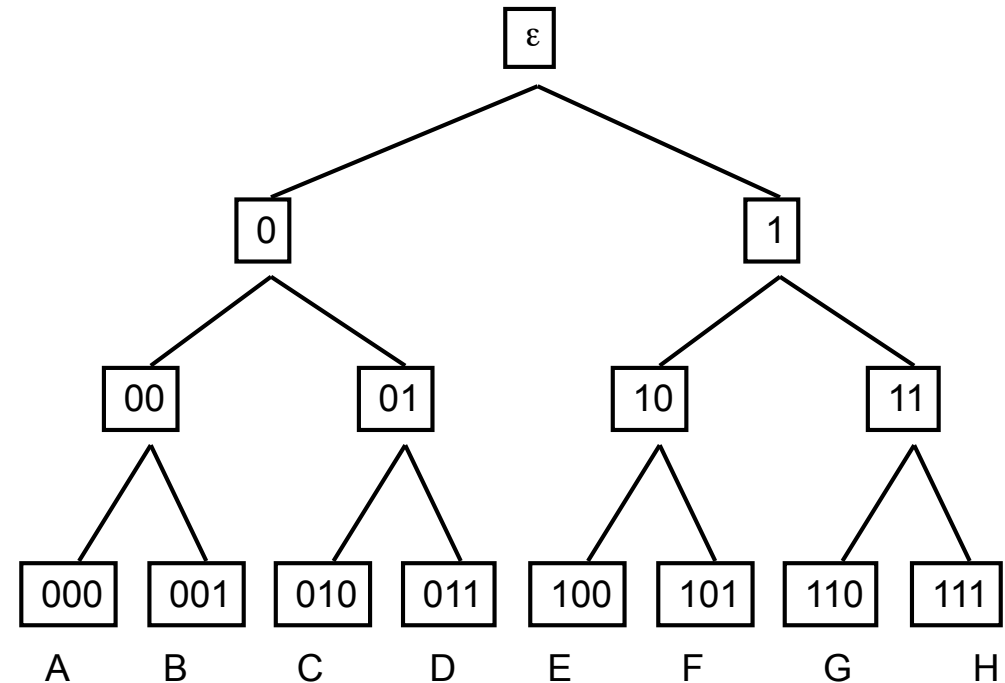
- Jeder Algorithmus sendet sofort (slotted Aloha)
- Falls eine Kollision auftritt,
 - akzeptiert keine Station mehr neue Paket aus der Vermittlungsschicht
 - Führe Adaptive-Tree-Walk(ϵ) aus



- Algorithmus Knoten-Test
 - für Knoten u des Baums und
 - kollidierende Menge S von Station
- Knoten-Test(u)
 - Betrachte zwei Slots pro Knoten des Baums
 - Im ersten Slot senden alle Knoten aus S , die mit ID $u0$ anfangen
 - Im zweiten Slot senden alle Knoten aus S , die mit ID $u1$ anfangen



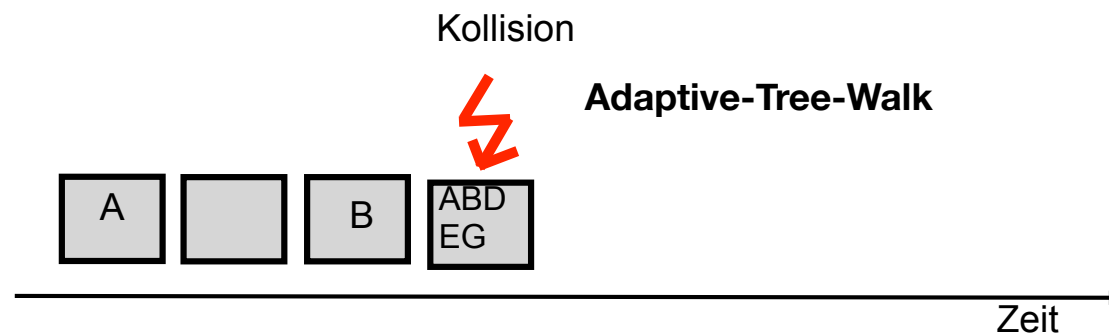
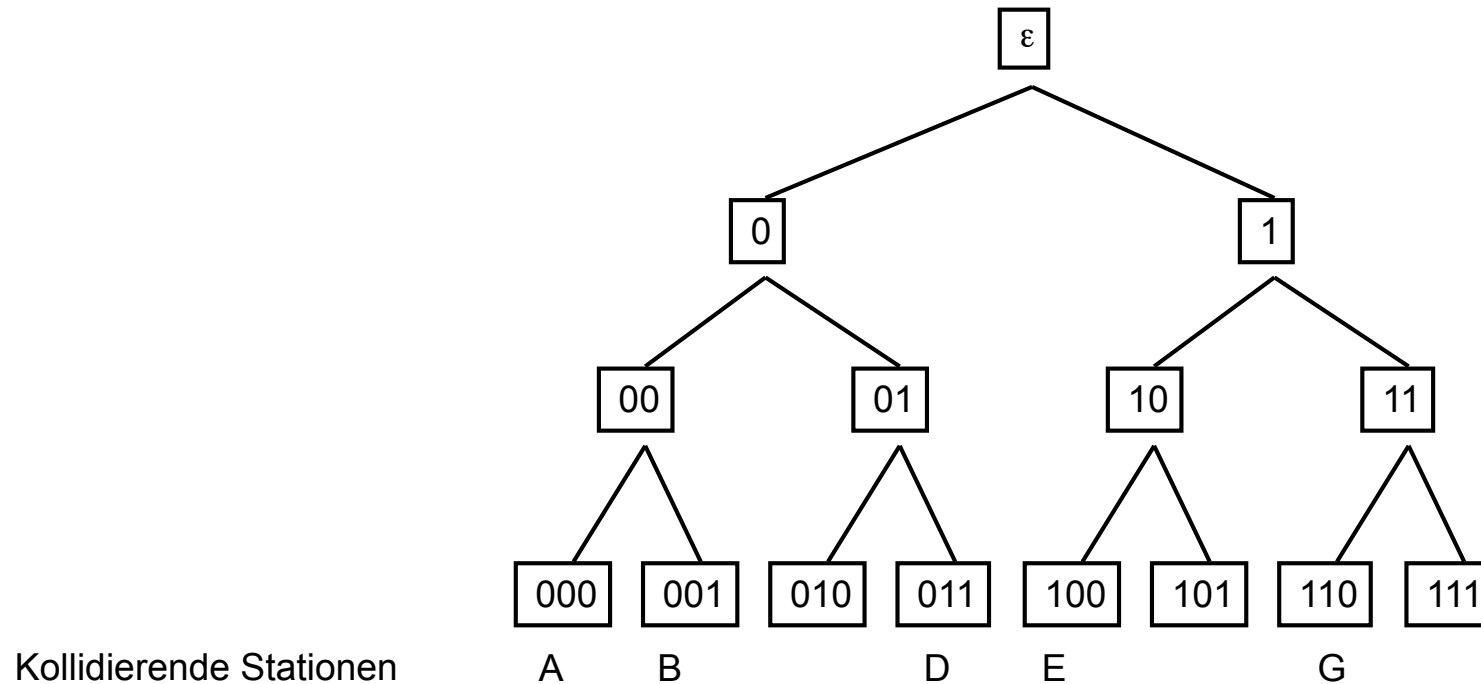
- Algorithmus Knoten-Test
 - für Knoten u des Baums und
 - kollidierende Menge S von Station
- Knoten-Test(u)
 - Betrachte zwei Slots pro Knoten des Baums
 - Im ersten Slot senden alle Knoten aus S , die mit ID $u0$ anfangen
 - Im zweiten Slot senden alle Knoten aus S , die mit ID $u1$ anfangen
- Adaptive Tree Walk(x)
 - Führe Knoten-Test(x) aus
 - Falls Kollision im ersten Slot,
 - führe Adaptive-Tree-Walk($x0$) aus
 - Falls Kollision im zweiten Slot,
 - Führe Adaptive-Tree-Walk($x1$) aus



Stationen

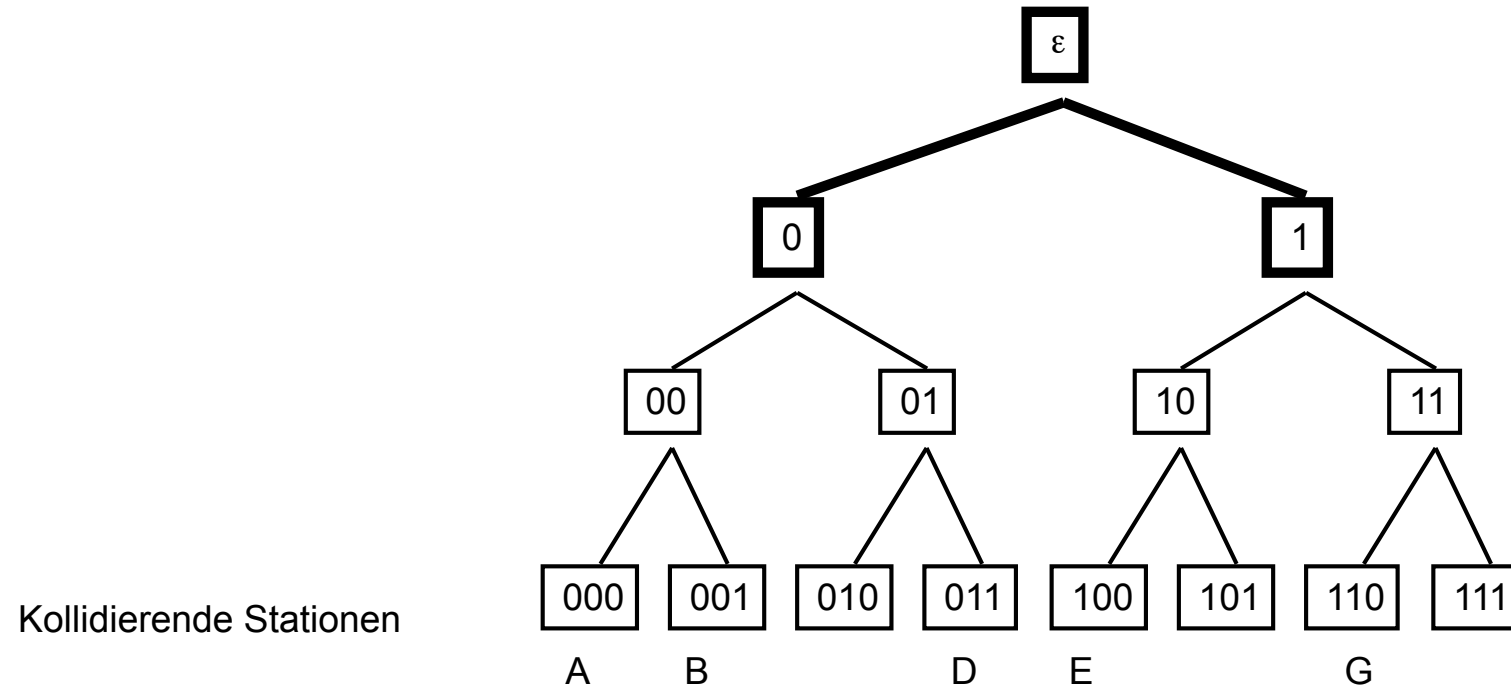
Adaptives Baumprotokoll

Beispiel (1)

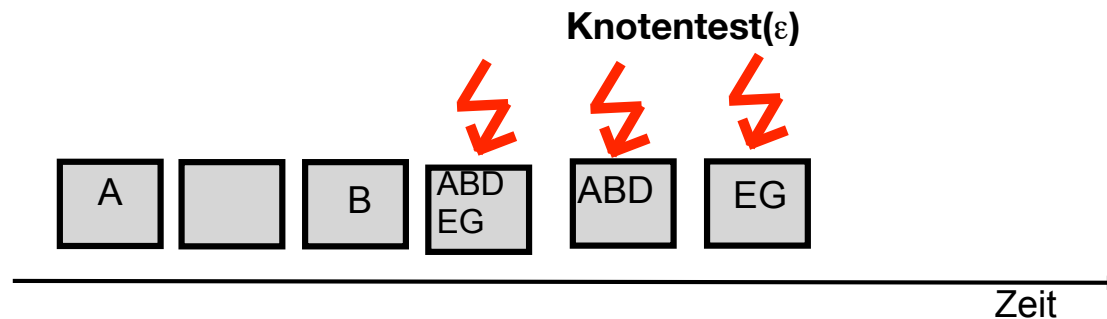


Adaptives Baumprotokoll

Beispiel (2)

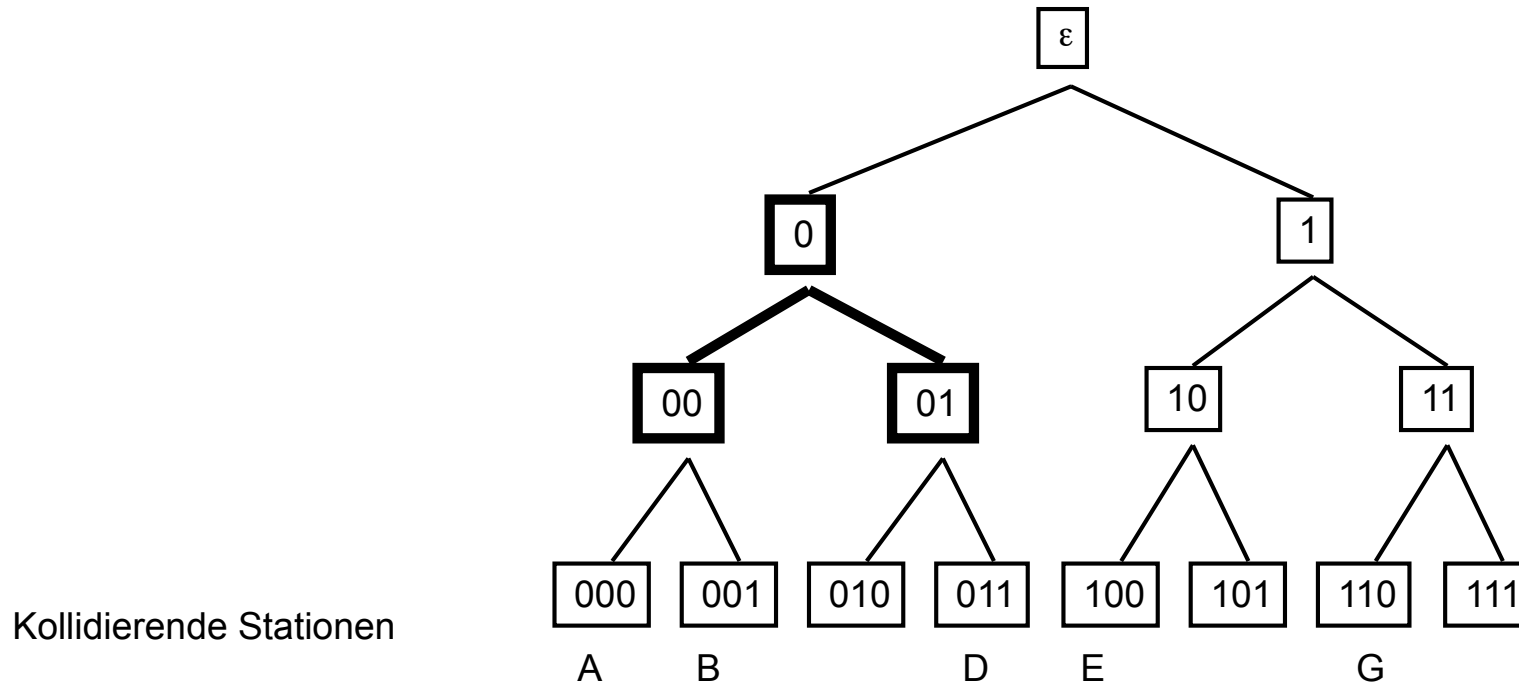


Adaptive-Tree-Walk

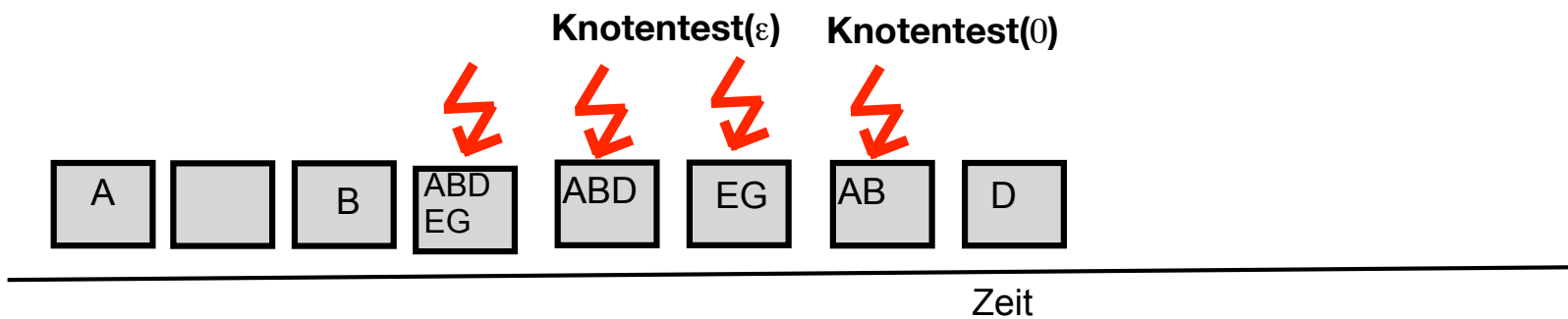


Adaptives Baumprotokoll

Beispiel (3)

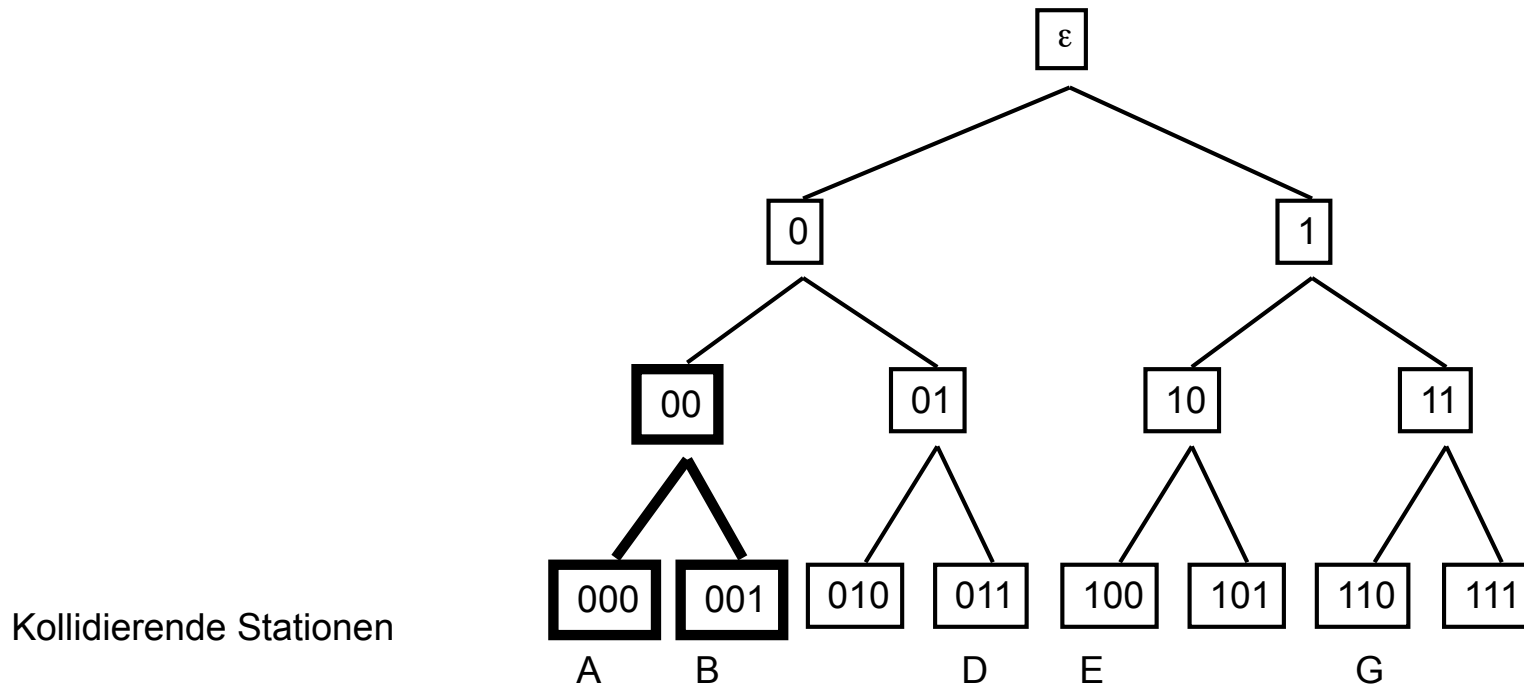


Adaptive-Tree-Walk

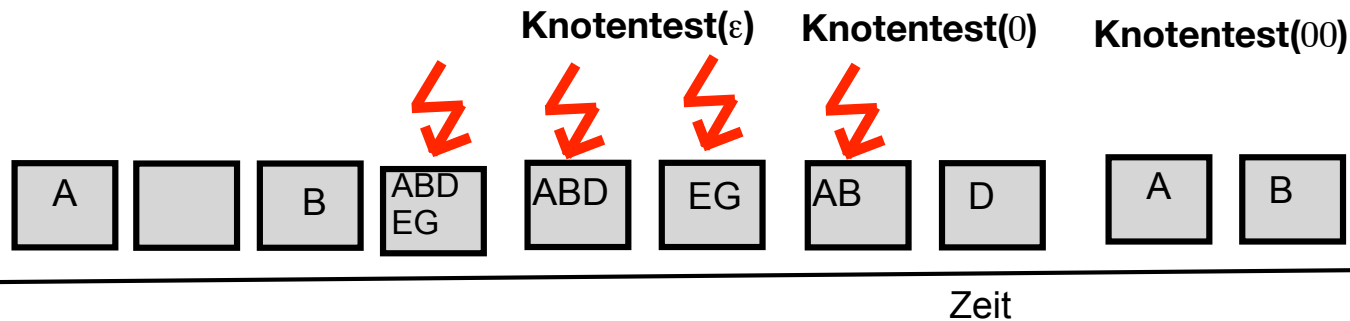


Adaptives Baumprotokoll

Beispiel (4)

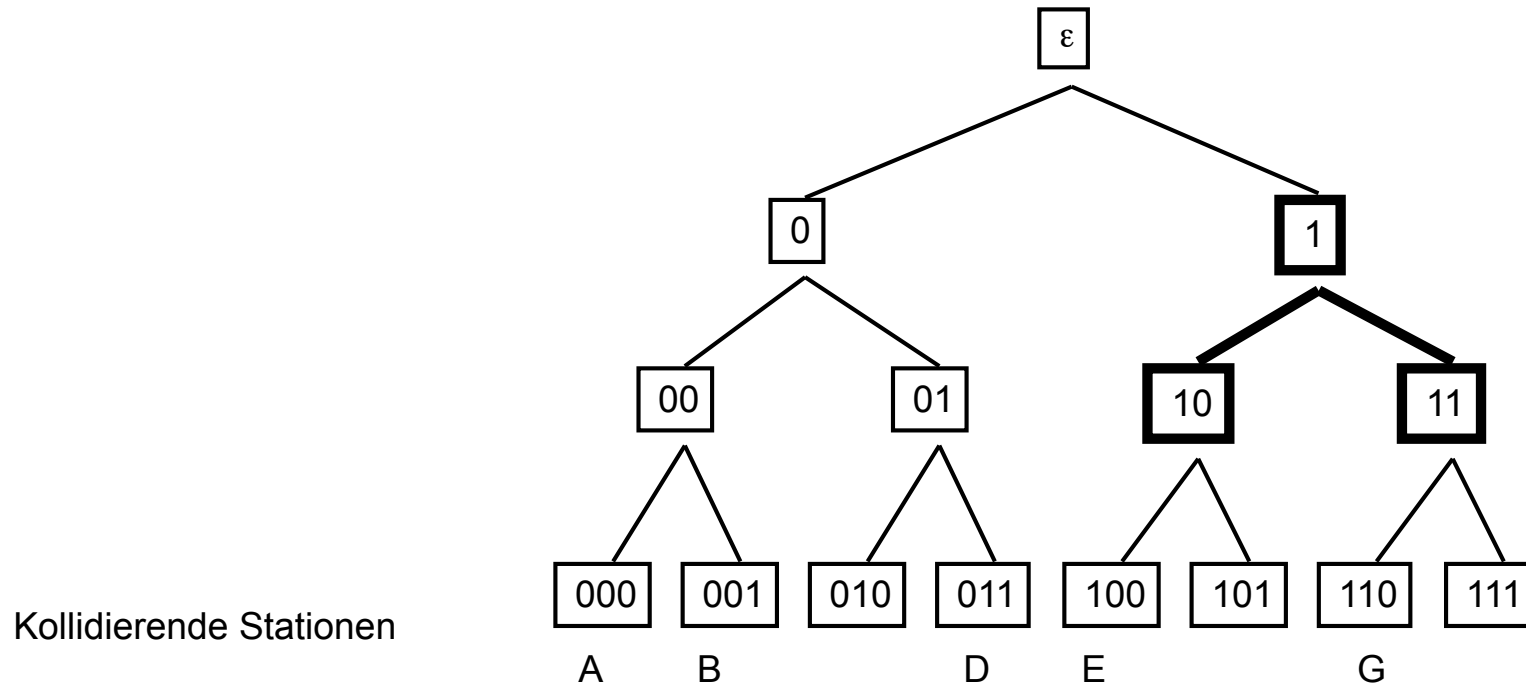


Adaptive-Tree-Walk

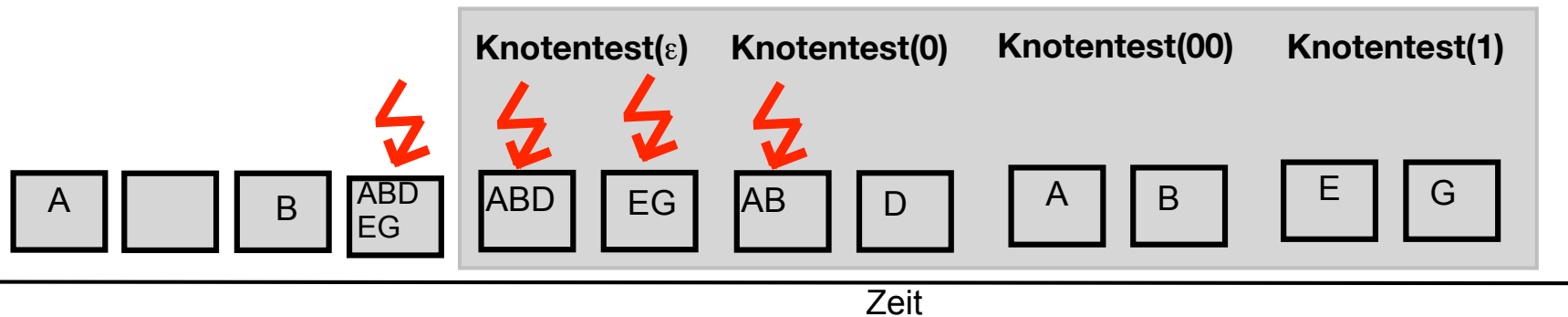


Adaptives Baumprotokoll

Beispiel (5)



Adaptive-Tree-Walk



Systeme II

3. Die Datensicherungsschicht

Christian Schindelhauer

Technische Fakultät

Rechnernetze und Telematik

Albert-Ludwigs-Universität Freiburg

Version 06.05.2013

$$111011 : 101 = 1101 \quad \text{Rest } 10$$

101	1
100	1
101	1
0111	1
101	1
10	

$$\underline{10111010 : 10011 = \cancel{1011}}$$

$$\begin{array}{r}
 101110100000 \text{ mod } 10011 = \\
 \underline{10011} \\
 10001 \\
 \cdot 10011 \\
 \hline
 100000 \\
 10011 \\
 \hline
 1100
 \end{array}$$

$R_q \neq 1100$
 $R =$