



Systeme II

**5. Vorlesungswoche
26.05. – 30.05.2008**

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Christian Schindelhauer
Sommer 2008

Systeme II

Kapitel 4

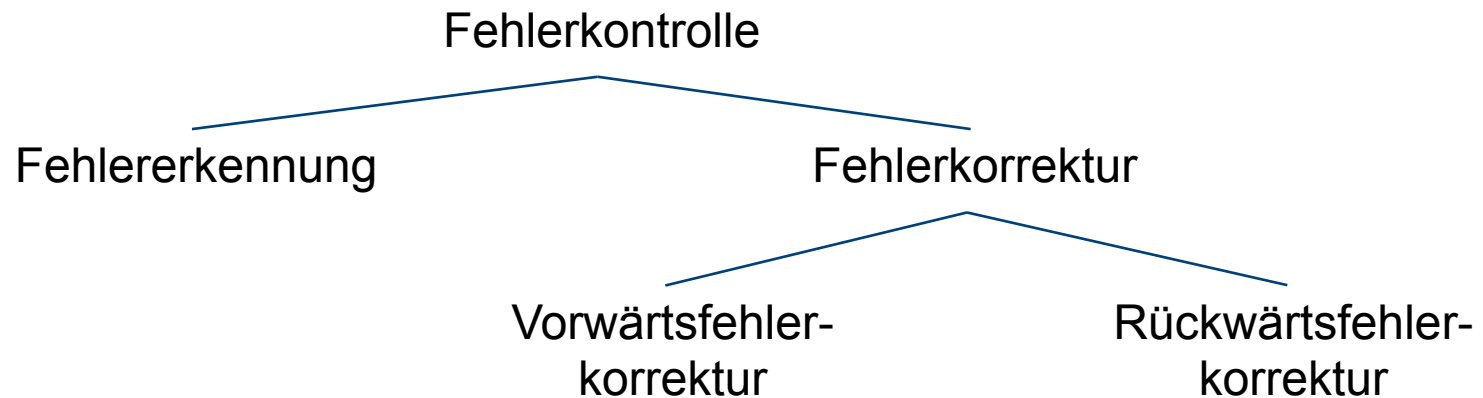
Sicherungsschicht

Einige Folien aus diesem Kapitel
sind aus der Vorlesung „Computer Networks“
von Holger Karl (Universität Paderborn)
übersetzt worden

2

Fehlerkontrolle

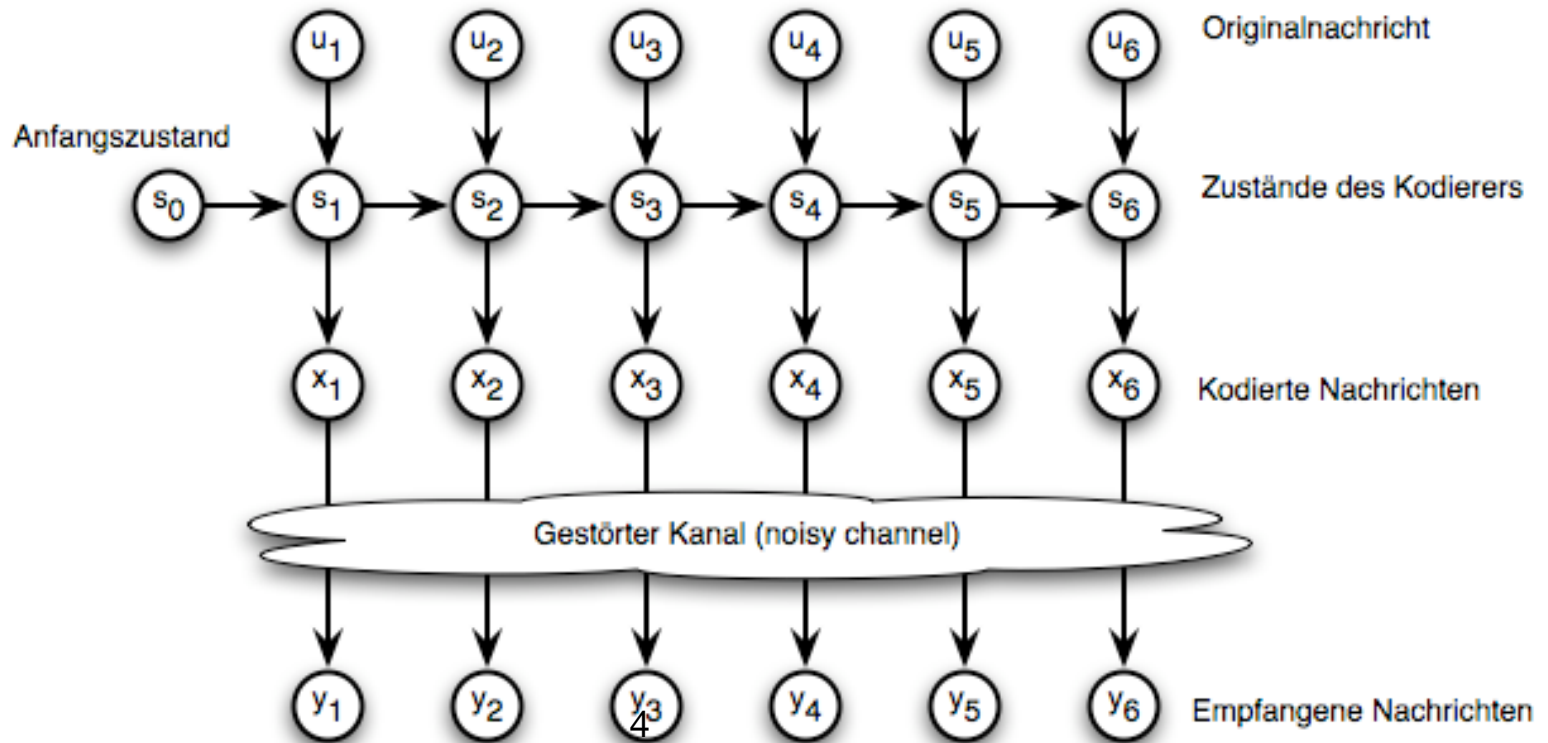
- ▶ **Zumeist gefordert von der Vermittlungsschicht**
 - Mit Hilfe der Frames
- ▶ **Fehlererkennung**
 - Gibt es fehlerhaft übertragene Bits?
- ▶ **Fehlerkorrektur**
 - Behebung von Bitfehlern
 - Vorwärtsfehlerkorrektur (Forward Error Correction)
 - Verwendung von redundanter Kodierung, die es ermöglicht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



Faltungs-Codes

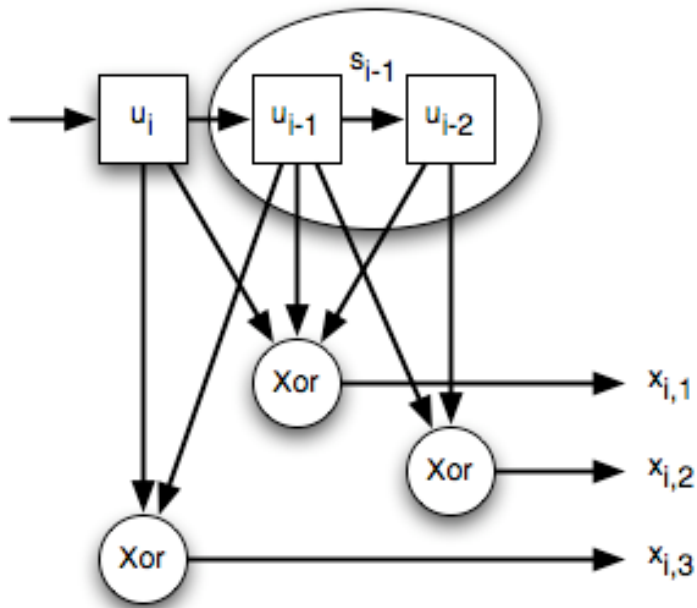
► Faltungs-Codes (Convolutional Codes)

- 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.

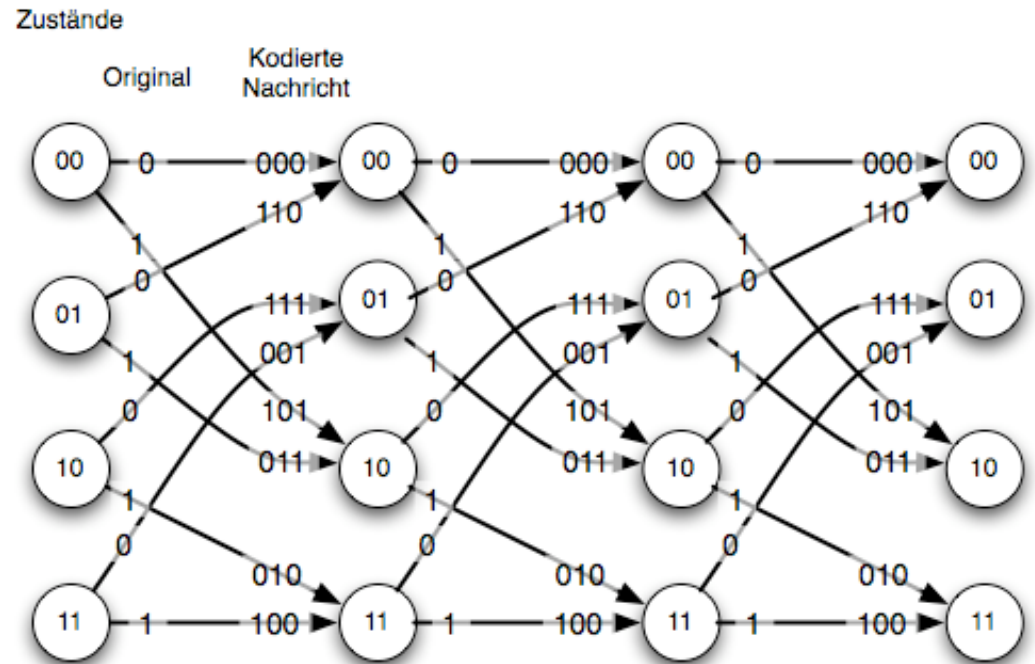


Beispiel

Faltungskodierer



Trellis-Diagramm

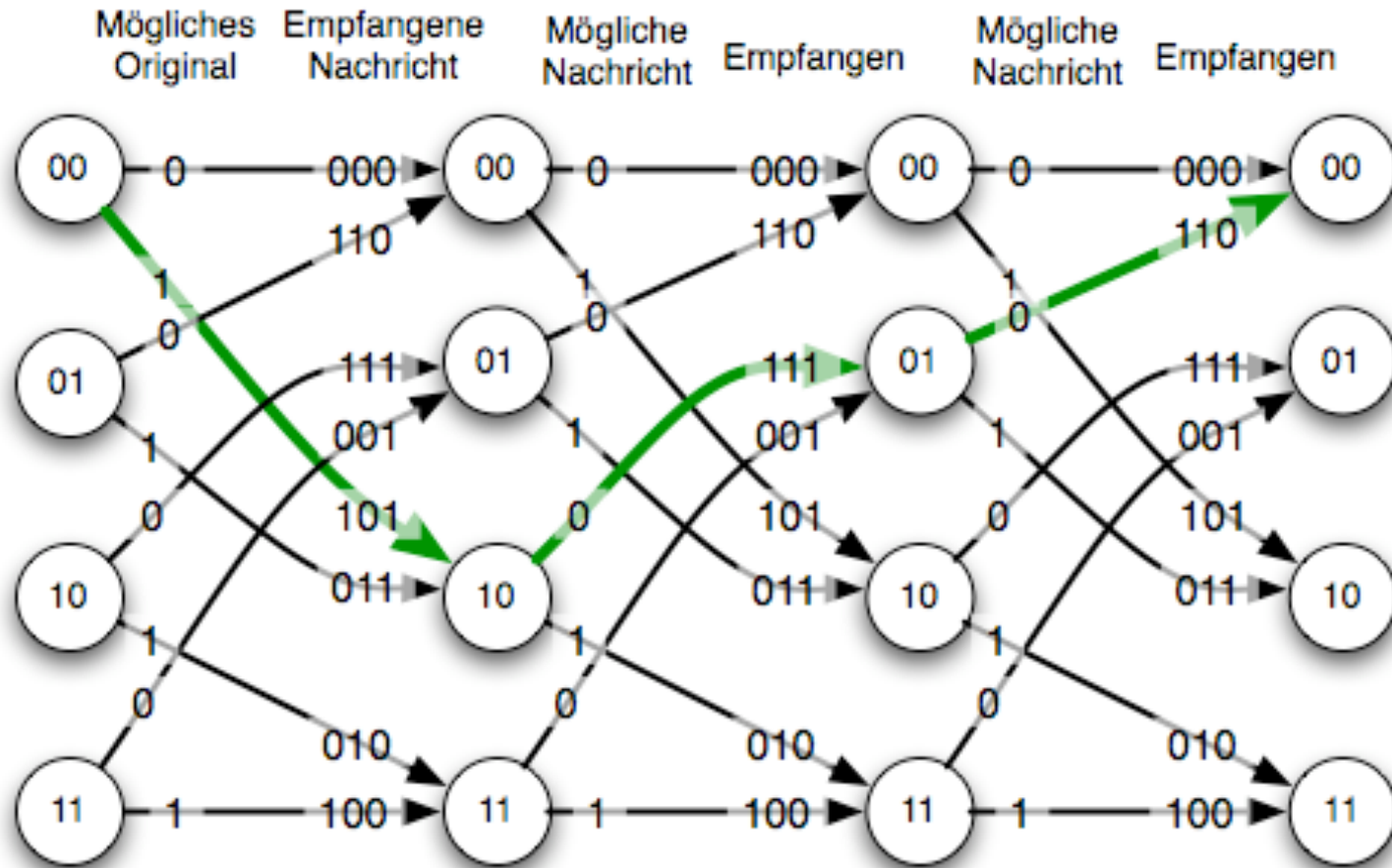


Dekodierung der Faltungs-Codes: Algorithmus von Viterbi

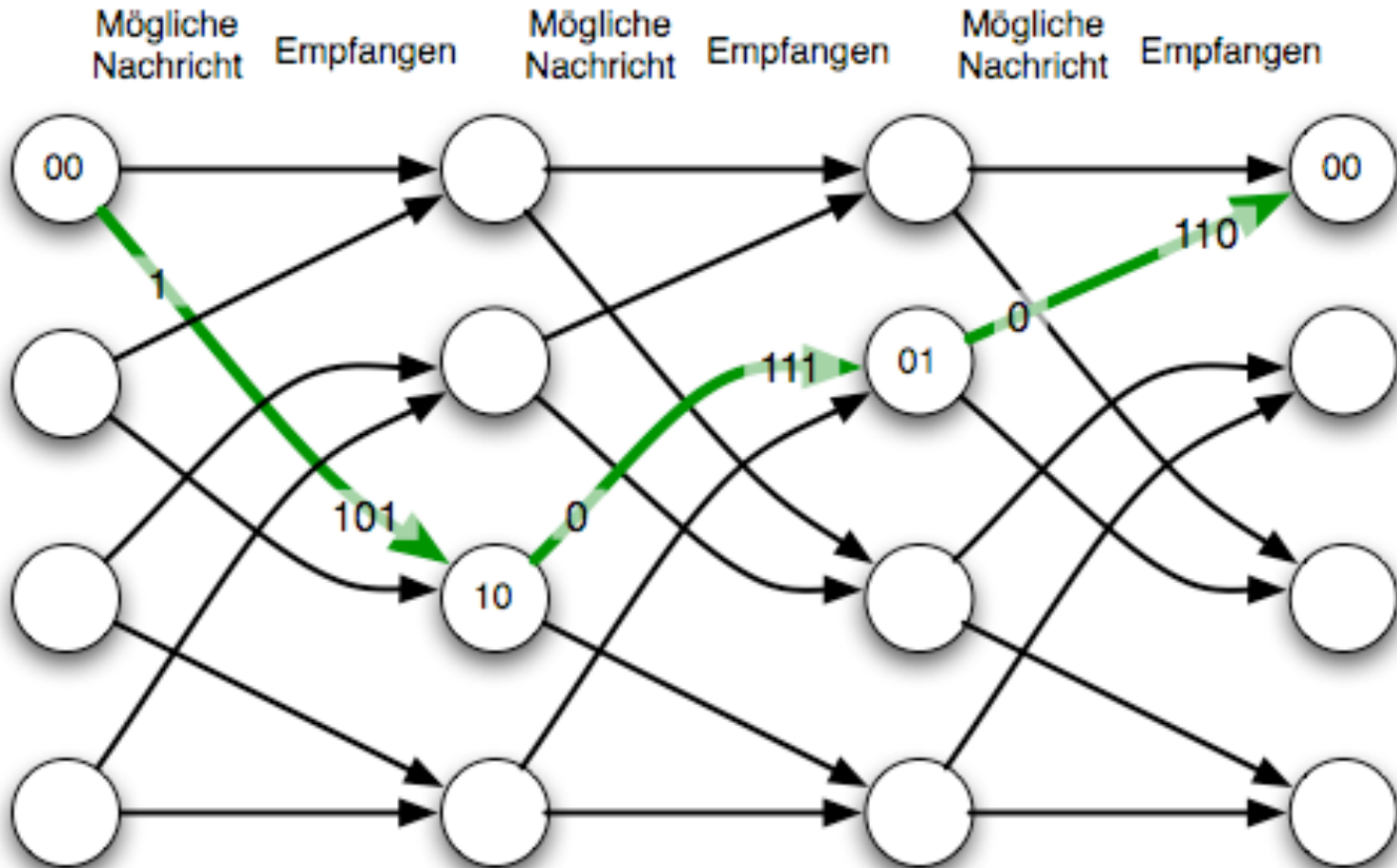
- ▶ **Dynamische Programmierung**
- ▶ **Zwei notwendige Voraussetzungen für Dekodierung**
 - (für den Empfänger) unbekannte Folge von Zuständen
 - beobachtete Folge von empfangenen Bits
(möglicherweise mit Fehler)
- ▶ **Der Algorithmus von Viterbi bestimmt die wahrscheinlichste Folge von Zuständen, welches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

Dekodierung (I)

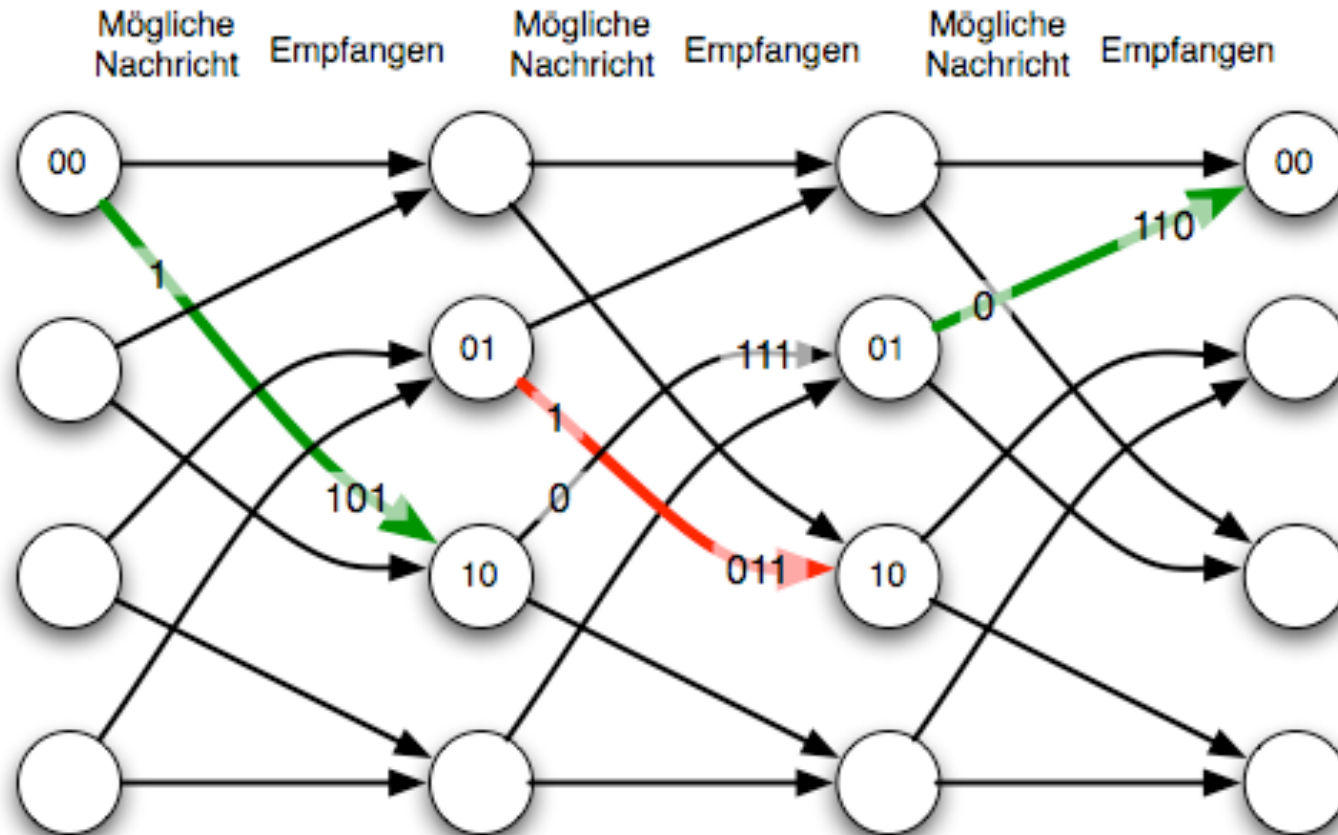
Zustände



Dekodierung (II)

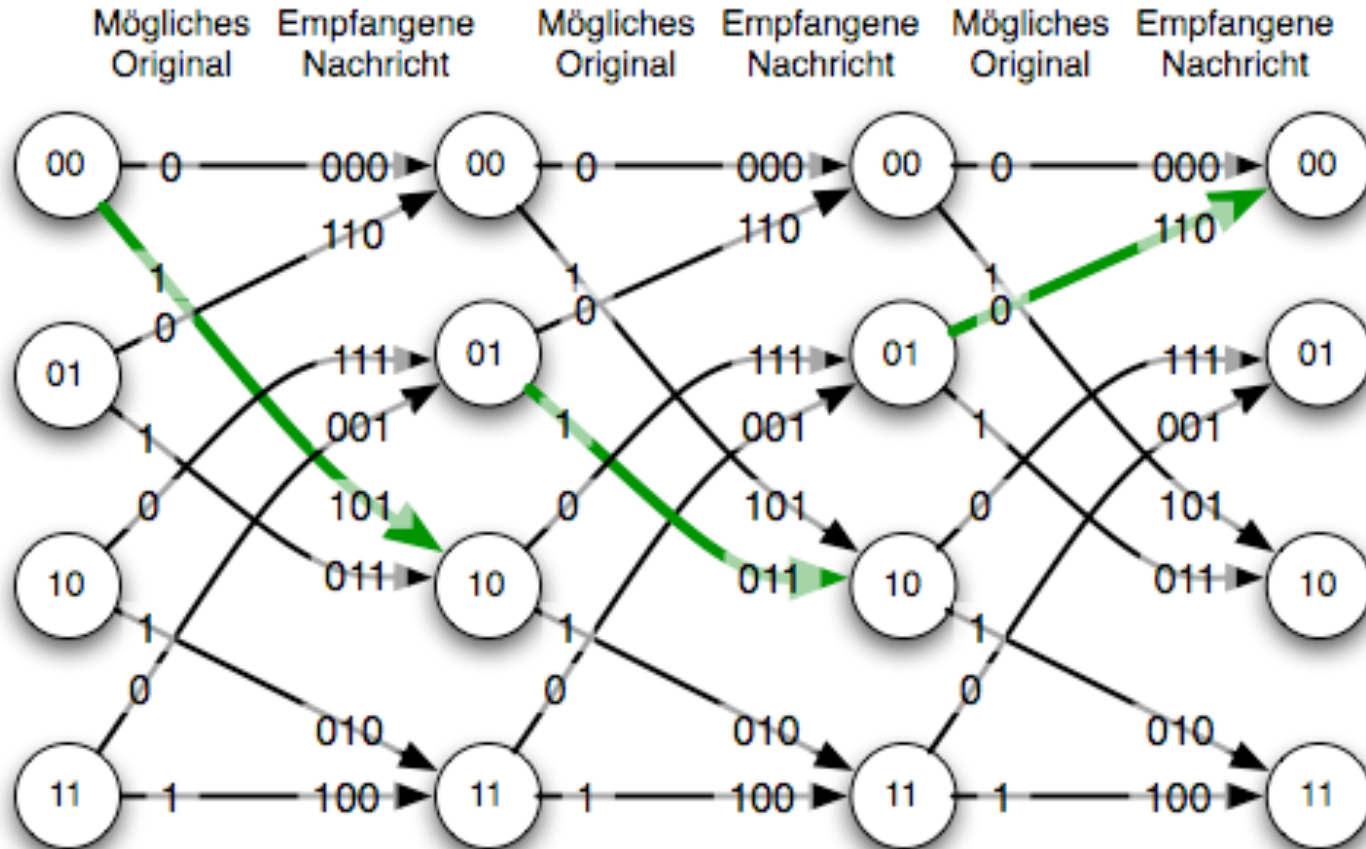


Dekodierung (III)



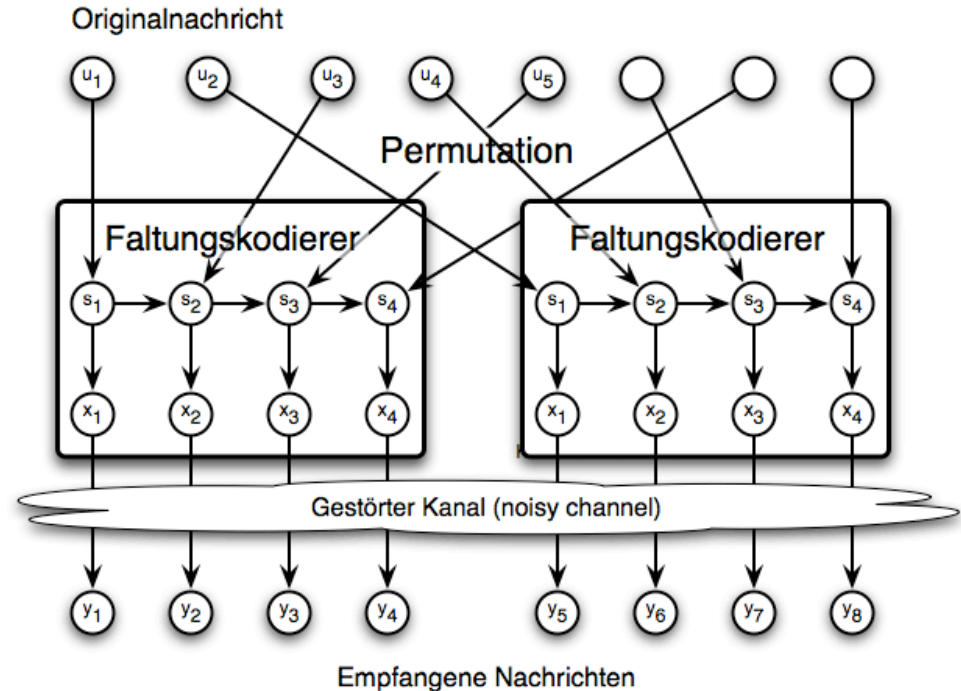
Dekodierung (IV)

Zustände



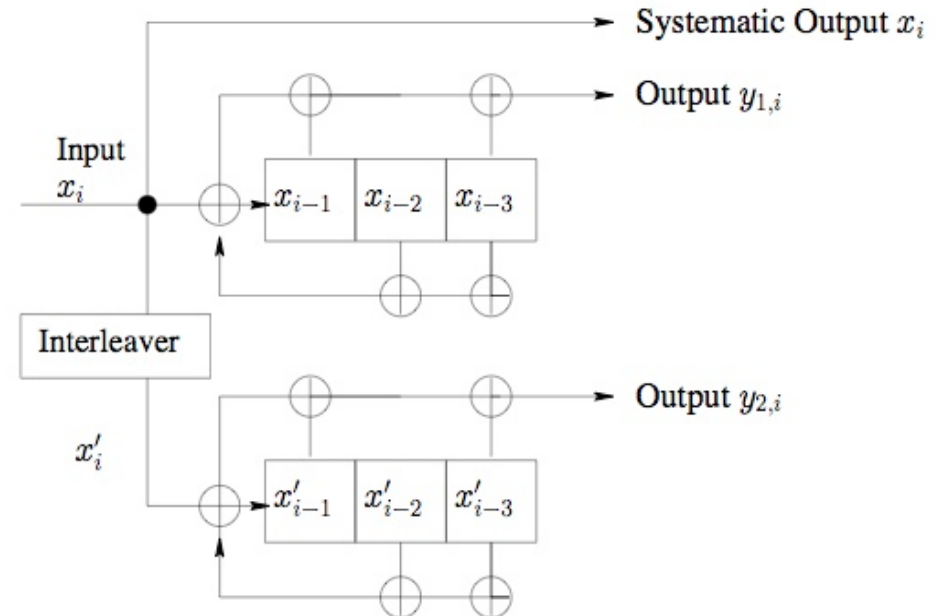
Turbo-Codes

- ▶ **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



Turbo-Codes

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



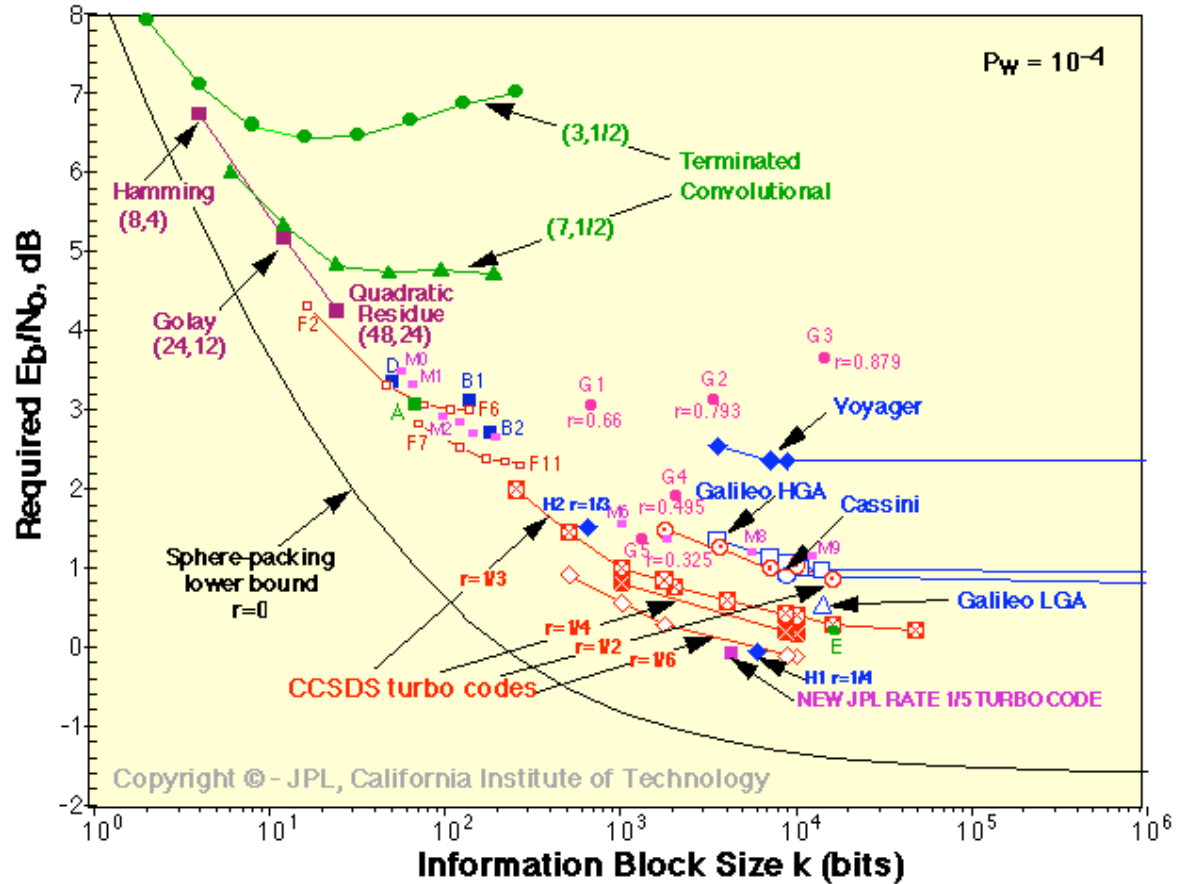
Interleavers

- ▶ **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
 - Rückpermutiert: 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

Codes im Vergleich

► **Code-Rate versus Signal-Ra Verhältnis**

- Stand 1998: (www331.jpl.nasa.gov/public/AllCodesVsSize.Gll)



Fehlererkennung: CRC

- ▶ **Effiziente Fehlererkennung: Cyclic Redundancy Check (CRC)**
- ▶ **Praktisch häufig verwendeter Code**
 - Hoher Fehlererkennungsrate
 - Effizient in Hardware umsetzbar
- ▶ **Beruh auf Polynomarithmetik im Restklassenring Z_2**
 - Zeichenketten sind Polynome
 - Bits sind Koeffizienten des Polynoms

Rechnen in \mathbb{Z}_2

- ▶ **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: $0 + (1 \cdot 0) + 1 + (1 \cdot 1) =$

Polynomarithmetik modulo 2

- ▶ **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**

Zeichenketten und Polynomarithmetik

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

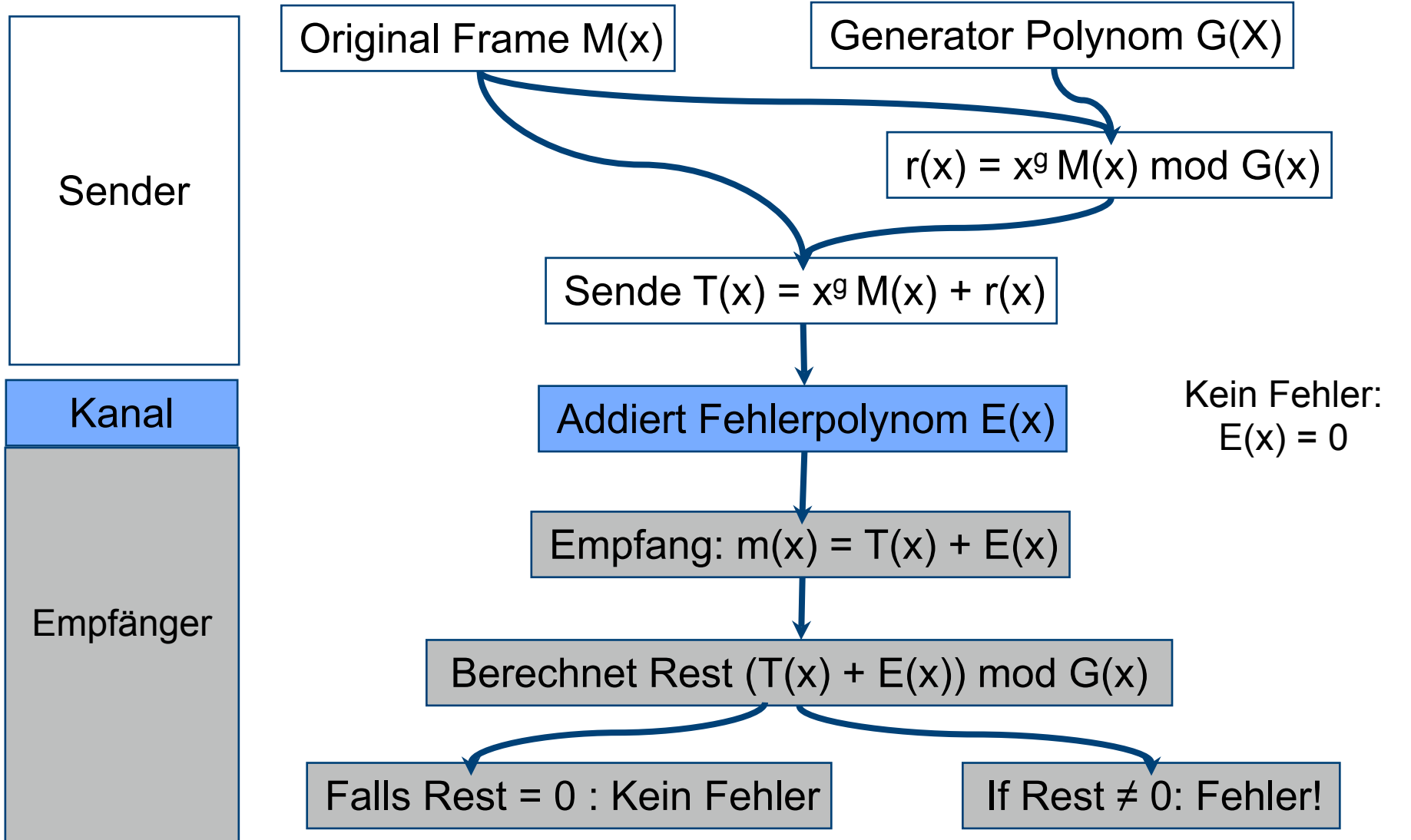
- ▶ **Definiere ein Generatorpolynom $G(x)$ von Grad g**
 - Dem Empfänger und Sender bekannt
 - Wir erzeugen g redundante Bits
- ▶ **Gegeben:**
 - Frame (Nachricht) M , als Polynom $M(x)$
- ▶ **Sender**
 - Berechne den Rest der Division $r(x) = x^g M(x) \bmod G(x)$
 - Übertrage $T(x) = x^g M(x) + r(x)$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

CRC Übertragung und Empfang

- ▶ **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 $m(x) \bmod G(x)$
 - Kein Fehler: $m(x) = T(x)$,
 - dann ist der Rest 0
 - Bit errors: $m(x) \bmod G(x) = (T(x) + E(x)) \bmod G(x)$

$$= \underbrace{T(x) \bmod G(x)}_0 + \underbrace{E(x) \bmod G(x)}_{\text{Fehlerindikator}}$$

CRC – Überblick



Der Generator bestimmt die CRC-Eigenschaften

- ▶ **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
- ▶ **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

CRC in der Praxis

- ▶ **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

Fehlerkontrolle

‣ **Zumeist gefordert von der Vermittlungsschicht**

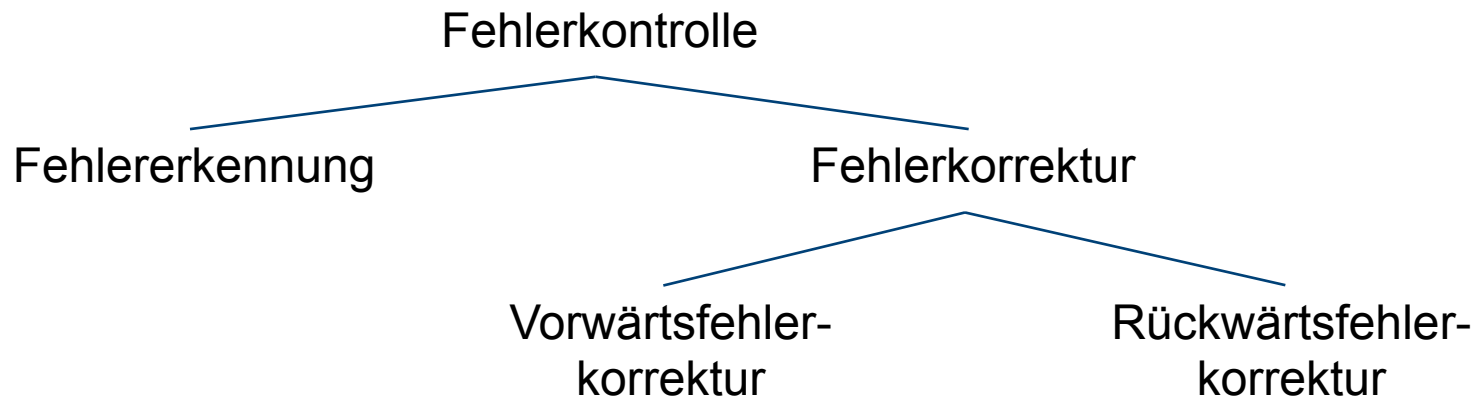
- Mit Hilfe der Frames

‣ **Fehlererkennung**

- Gibt es fehlerhaft übertragene Bits?

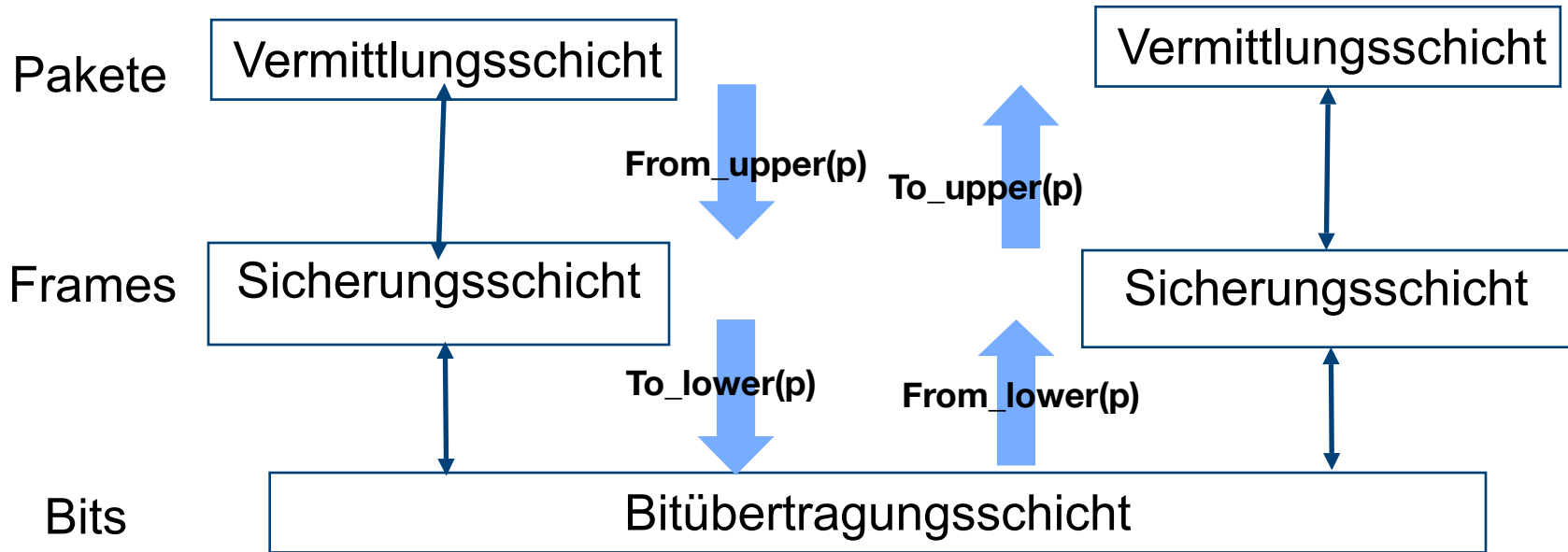
‣ **Fehlerkorrektur**

- 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

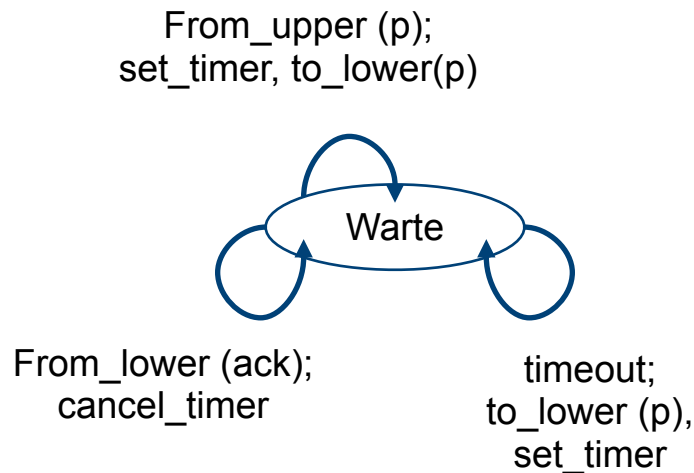
- ▶ Bei Fehlererkennung muss der Frame nochmal geschickt werden
- ▶ Wie ist das Zusammenspiel zwischen Sender und Empfänger?



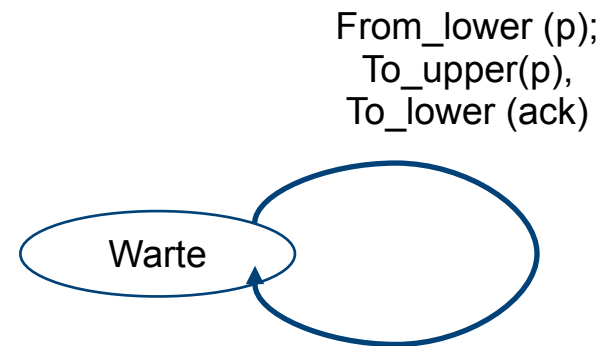
to_lower, from_lower beinhalten CRC
oder (bei Bedarf) Vorwärtsfehlerkorrektur

Einfaches Simplex-Protokoll mit Bestätigungen

- ▶ **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



Diskussion

▶ Probleme

- Sender ist schneller als Empfänger

- Was passiert, wenn Bestätigungen verloren gehen?

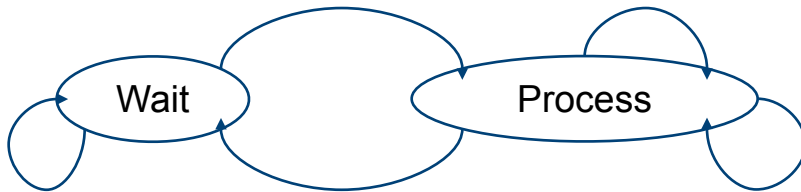
2. Versuch

- ▶ Lösung des ersten Problems
 - Ein Paket nach dem anderen

Sender

From_higher(p);
To_lower(p),
set_timer

From_higher(p);
to_higher (busy)



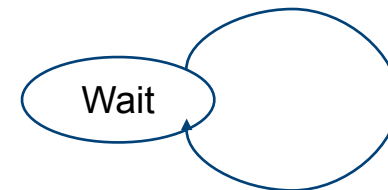
timeout;
error

From_lower(ack);
Cancel_timer

timeout;
to_lower (p),
set_timer

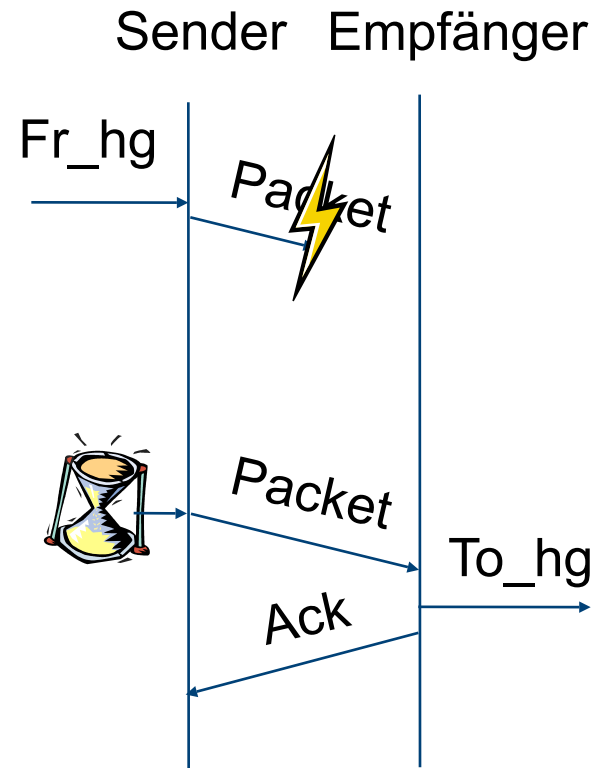
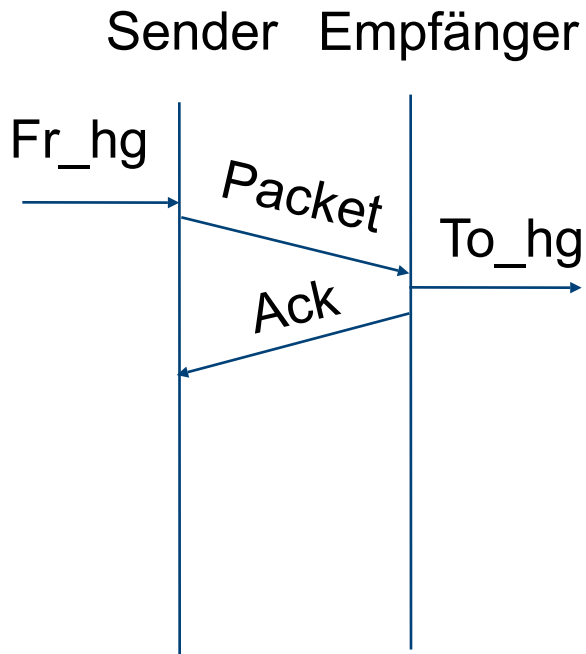
Empfänger

From_lower (p);
To_upper(p),
to_lower (ack)



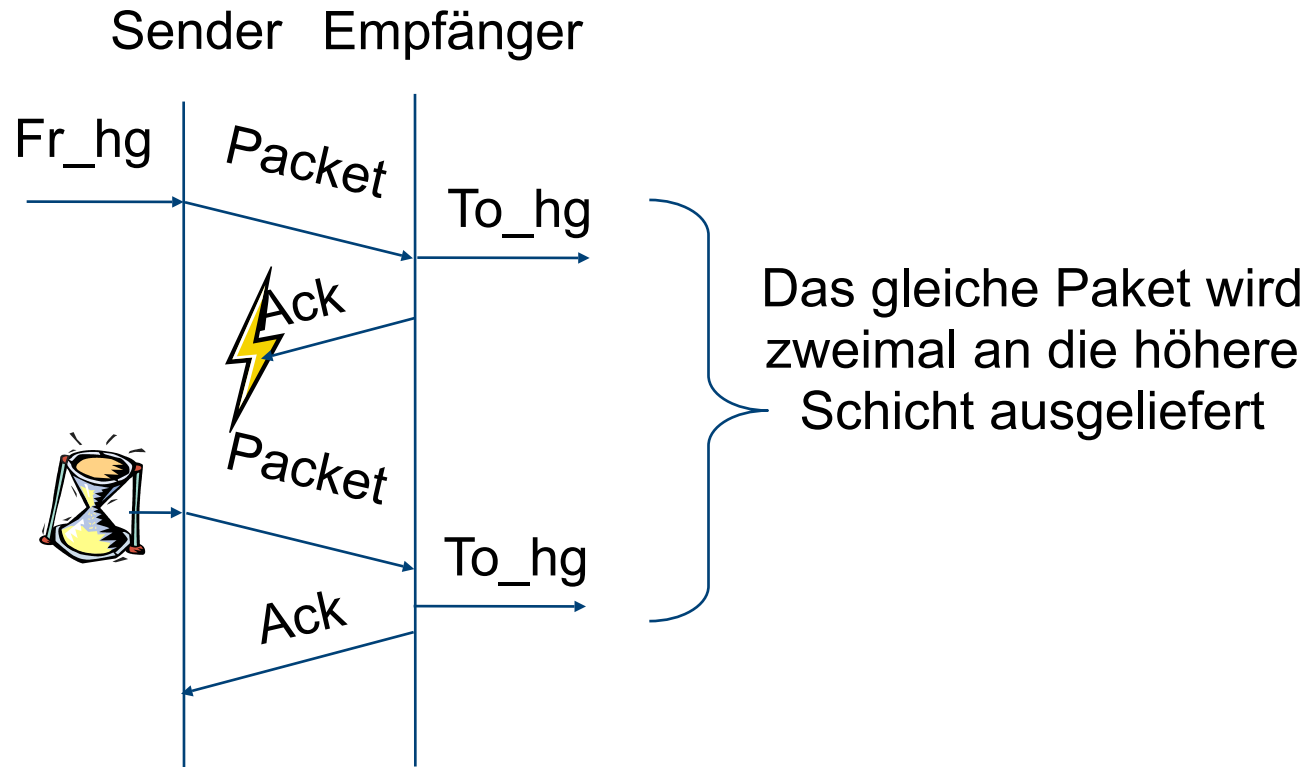
Diskussion

- ▶ **Protokoll etabliert elementare Flusskontrolle**



Diskussion

▶ 2. Fall: Verlust von Bestätigung



Probleme der 2. Version

- ▶ **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)

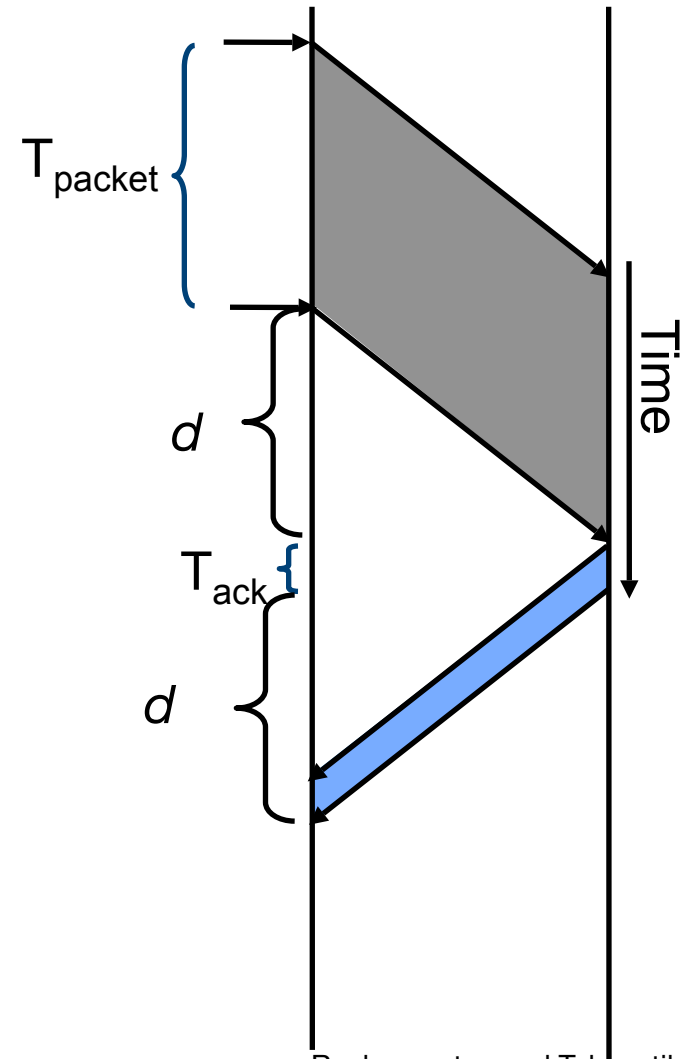
3. Version

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

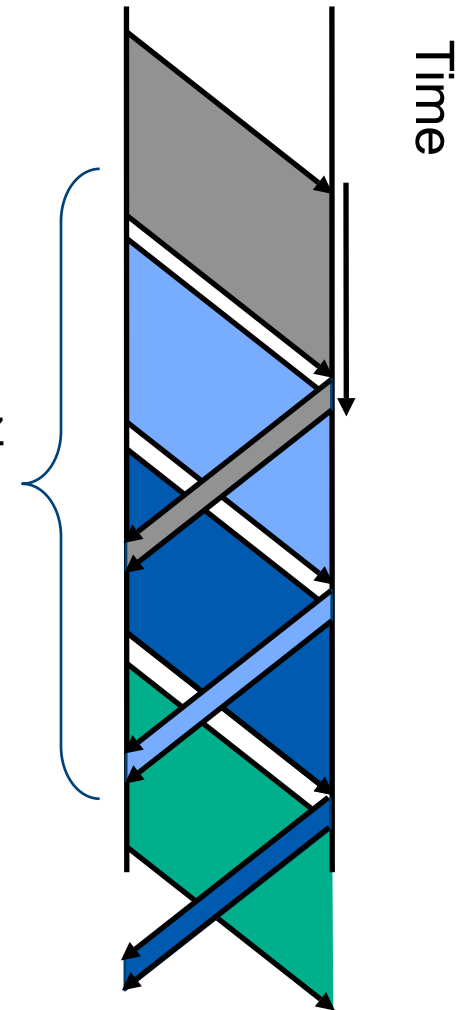
- ▶ **Effizienz η**
 - Definiert als das Verhältnis zwischen
 - der Zeit um zu senden
 - und der Zeit bis neue Information gesendet werden kann
 - (auf fehlerfreien Kanal)
 - $\eta = T_{\text{packet}} / (T_{\text{packet}} + d + T_{\text{ack}} + d)$
- ▶ **Bei großen Delay ist das Alternating Bit Protocol nicht effizient**



Verbesserung der Effizienz

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

Sender ist immer aktiv:
Hohe Effizienz

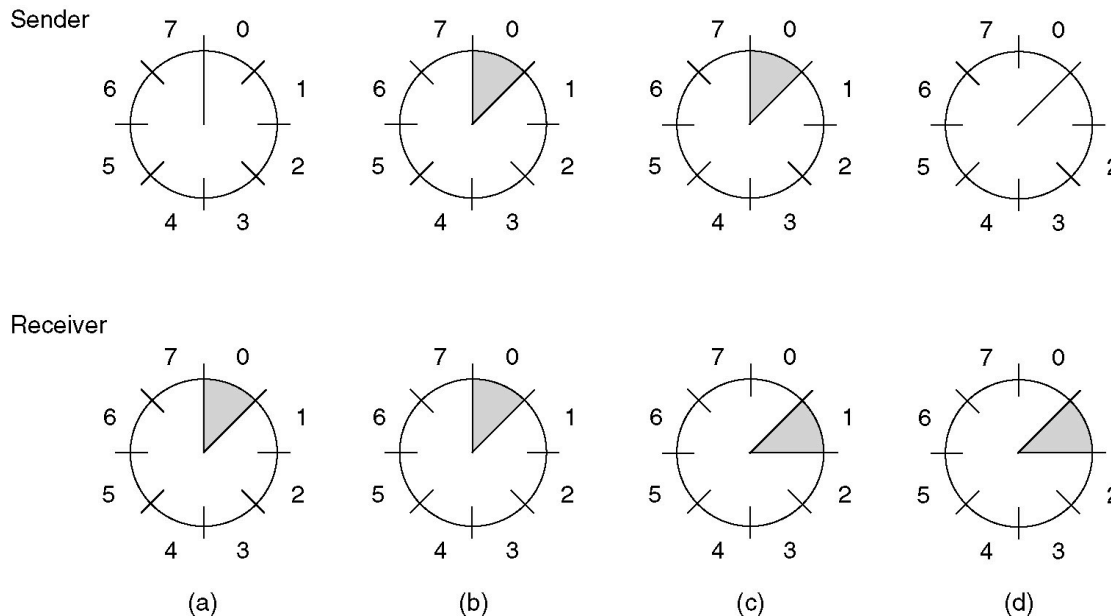


Gleitende Fenster

- ▶ **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



- Initial: Nichts versendet
- Nach Senden des 1. Frames mit Seq.Nr. 0
- Nach dem Empfang des 1. Frame
- Nach dem Empfang der Bestätigung

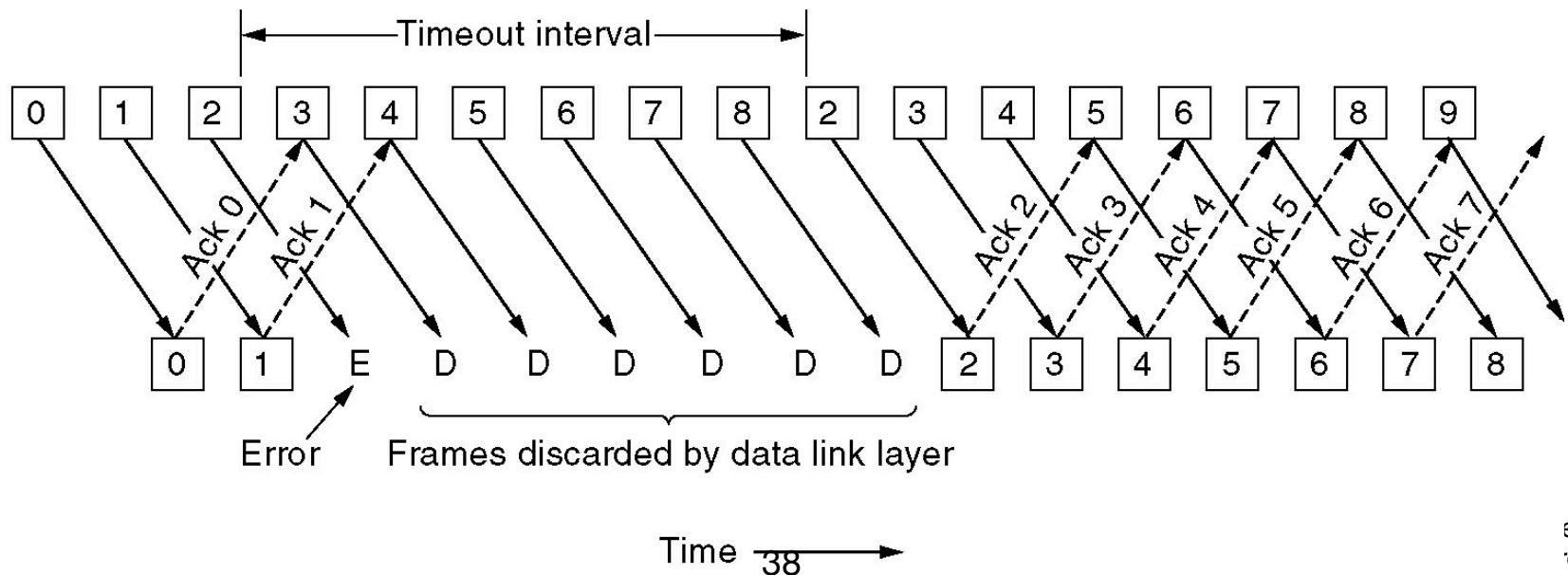
Übertragungsfehler und Empfangsfenster

► 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



Go-back-N

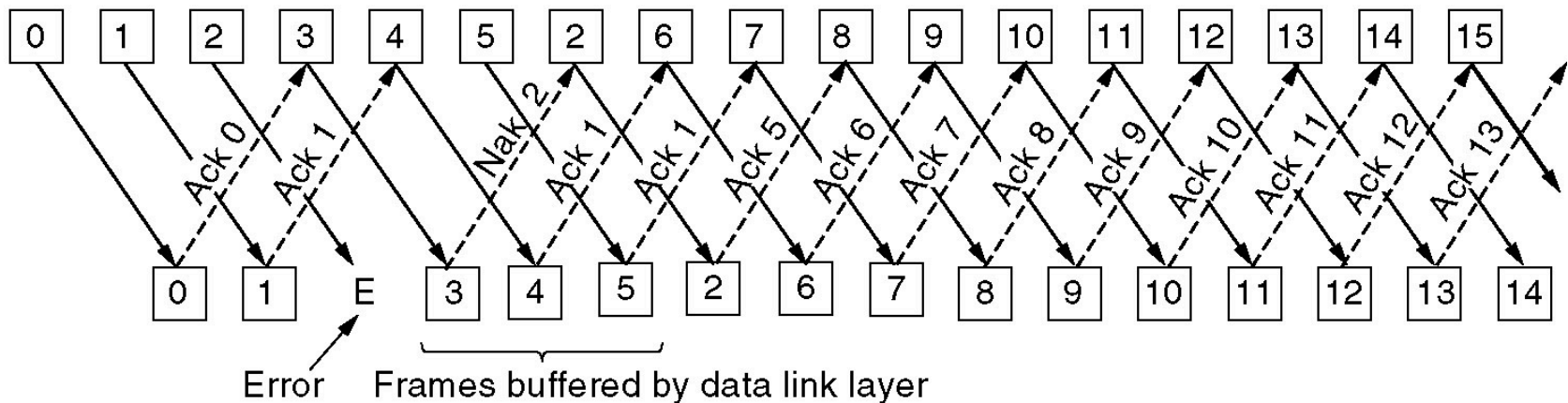
- ▶ **Mit Empfangsfenster der Größe 1 können die Frames, die einem verlorenen Frame folgen, nicht durch den Empfänger bearbeitet werden**
 - Sie können einfach nicht bestätigt werden, da nur eine Bestätigung für das letzte korrekt empfangene Paket verschickt wird
- ▶ **Der Sender wird einen “Time-Out” 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

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 entgegenkommenden Frames gepackt



Systeme II

Kapitel 4

Mediumzugriff in der Sicherheitsschicht

Der Mediumzugriff in der Sicherungsschicht

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

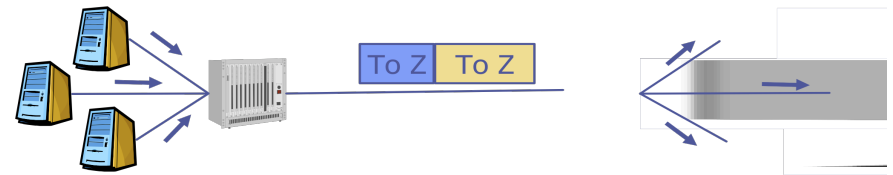
Der Mediumzugriff in der Sicherungsschicht

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

Statisches Multiplexen

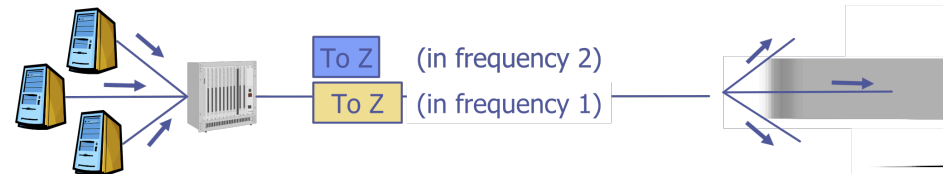
► **Gegeben sei eine einzelne Leitung (Ressource)**

- Mehreren Kommunikationsverbindungen werden feste Zeiträume/Kanäle (slots/channels) zugewiesen
- Oder: Feste Frequenzbänder werden ihnen zugewiesen ... oder ...



► **Gute Lösung falls**

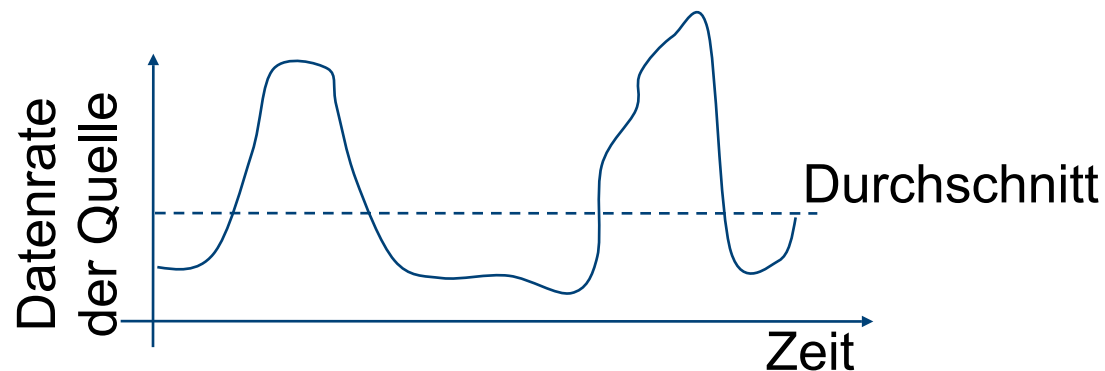
- Feste Datenraten und entsprechenden Anteilen am Kanal
- Quellen lasten die Leitung aus



Verkehrsspitzen (*bursty traffic*)

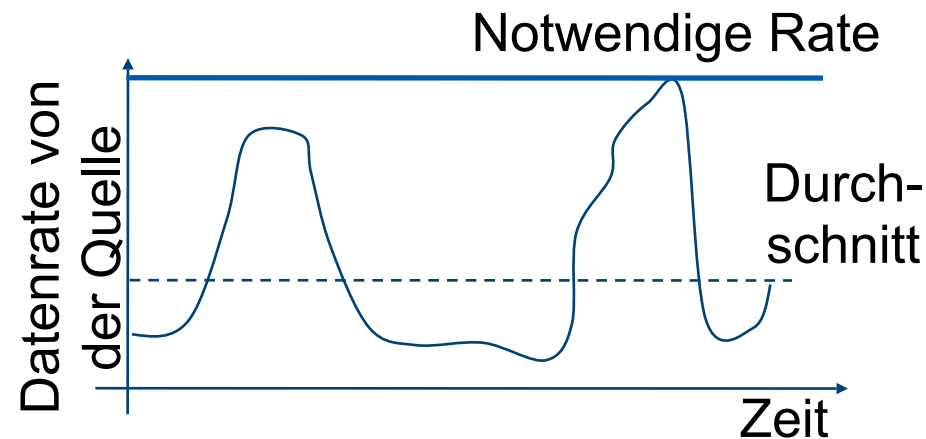
► **Problem: Verkehrsspitzen (*bursty traffic*)**

- Definition: Großer Unterschied zwischen Spitze und Durchschnitt
- In Rechnernetzwerken: Spitze/Durchschnitt = 1000/1 nicht ungewöhnlich



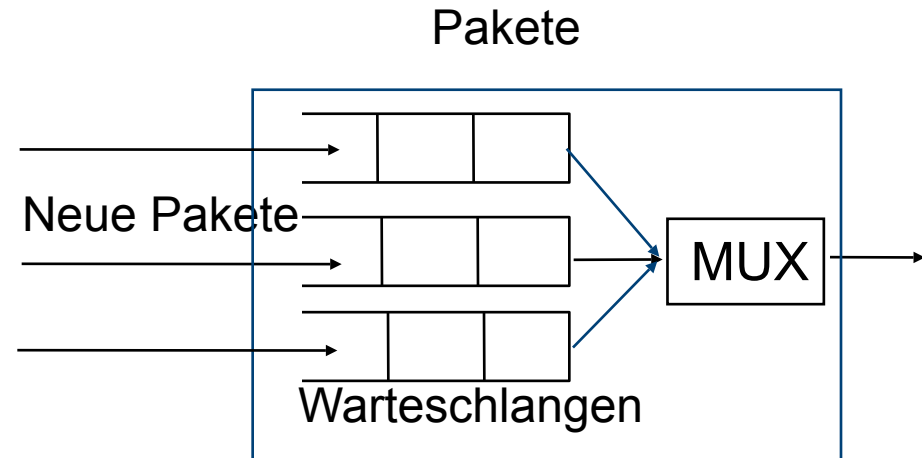
Verkehrsspitzen und statisches Multiplexen

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



Systeme II
Sommer 2008

47



Rechnernetze und Telematik
Albert-Ludwigs-Universität Freiburg
Christian Schindelhauer

Verkehrsspitzen und statisches Multiplexen - Verzögerung

- ▶ **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

Der Mediumzugriff in der Sicherungsschicht

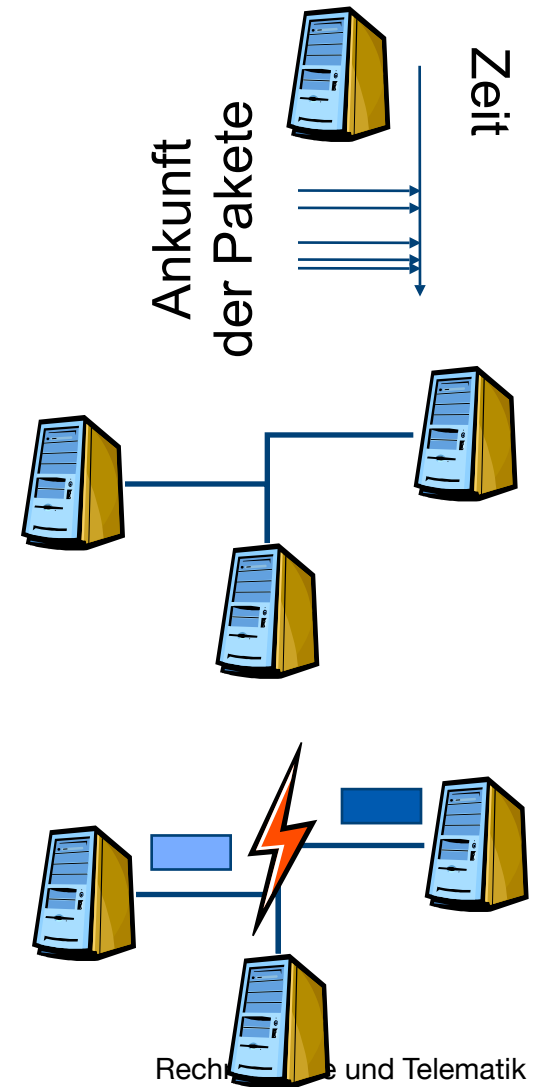
- ▶ **Statisches Multiplexen**
- ▶ ***Dynamische Kanalbelegung***
 - Kollisionsbasierte Protokolle
 - Kollisionsfreie Protokolle (contention-free)
 - Protokolle mit beschränktem Wettbewerb (limited contention)
- ▶ **Fallbeispiel: Ethernet**

Dynamische Kanalzuweisung – MAC

- ▶ **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)***

Annahmen

- ▶ **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



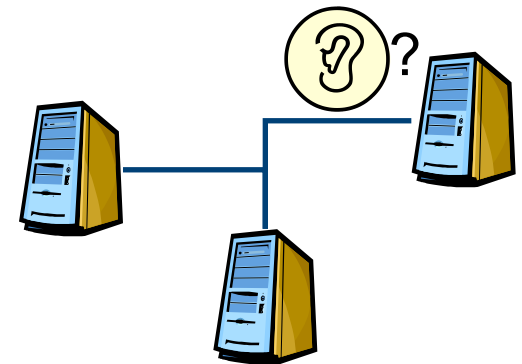
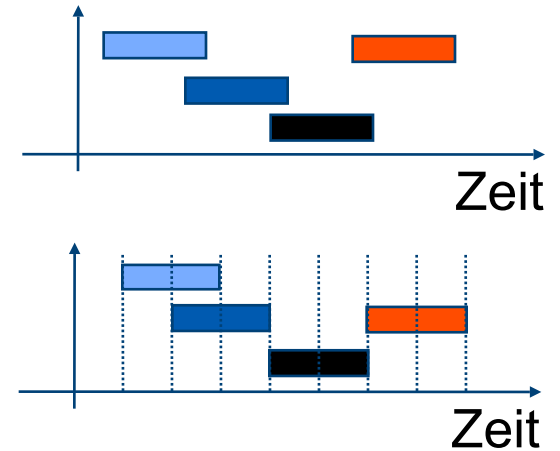
Annahmen

▶ 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



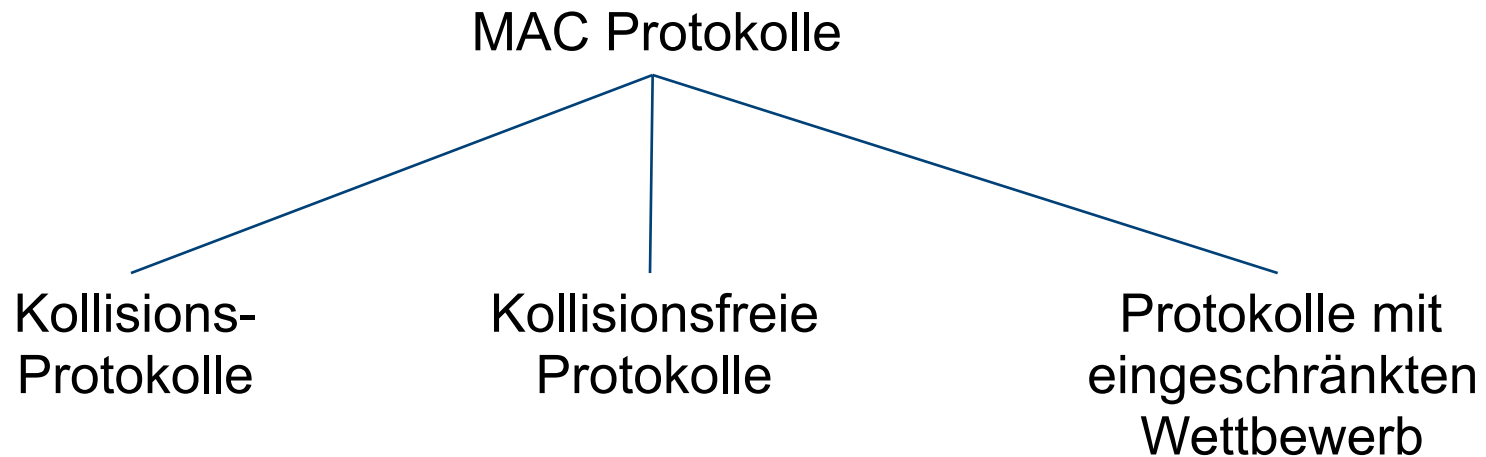
Bewertung des Verhaltens

- ▶ **Methoden zur Bewertung der Effizienz einer Kanalzuweisung**
- ▶ **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

Mögliche MAC-Protokolle

▸ **Unterscheidung: Erlaubt das Protokoll Kollisionen?**

- Als Systemscheidung
- Die unbedingte Kollisionsvermeidung kann zu Effizienzeinbußen führen



System mit Kollisionen: **Contention System**

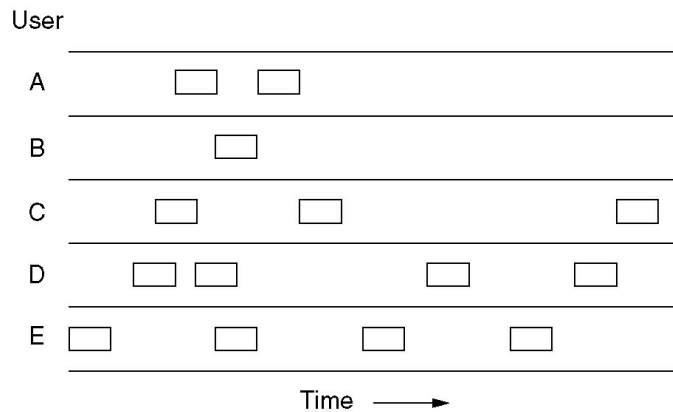
ALOHA

▶ Algorithmus

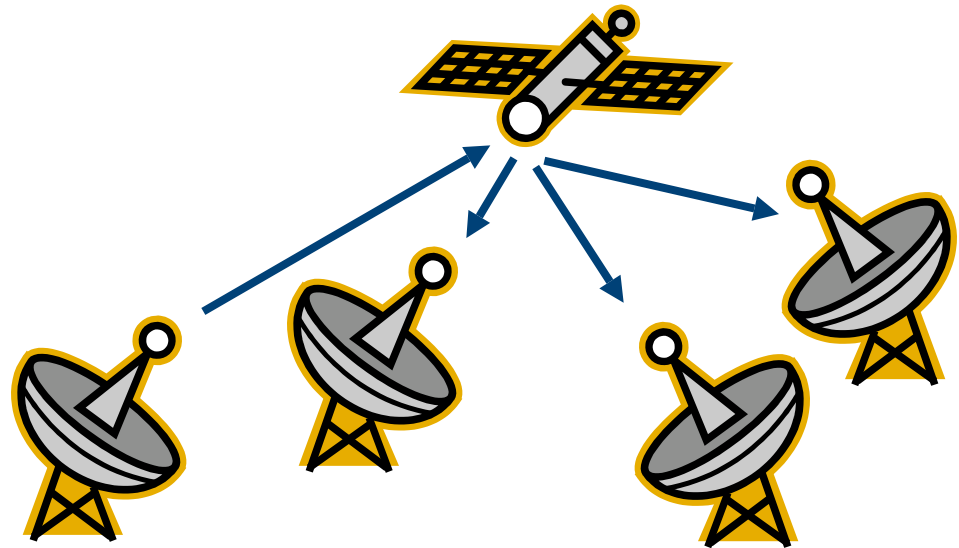
- Sobald ein Paket vorhanden ist, wird es gesendet

▶ Ursprung

- 1985 by Abrahamson et al., University of Hawaii
- Ziel: Verwendung in Satelliten-Verbindung



Pakete werden zu beliebigen Zeiten übertragen



ALOHA – Analyse

▶ 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

ALOHA – Effizienz

▶ **Betrachte Poisson-Prozess zur Erzeugung von Paketen**

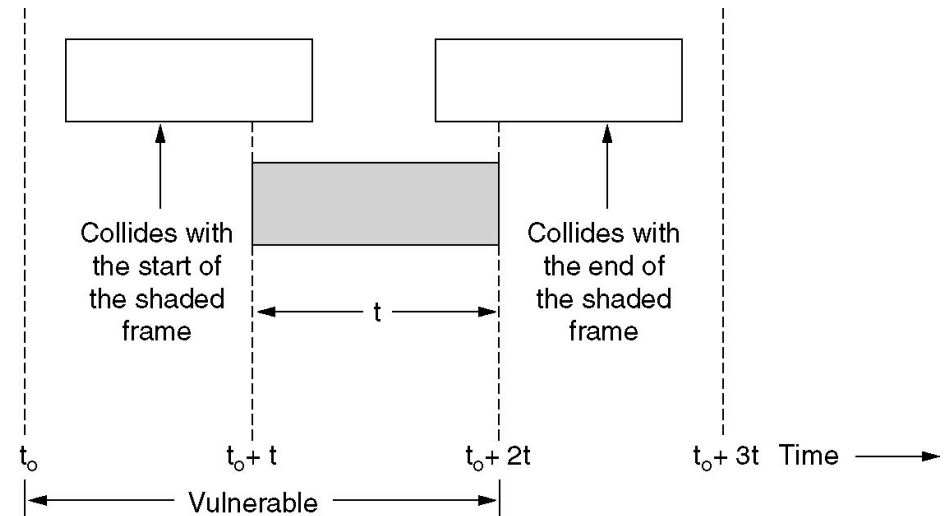
- Entsteht durch “unendlich” viele Stationen, die sich gleich verhalten
- Zeit zwischen zwei Sendeversuchen ist exponentiell verteilt
- Sei G der Erwartungswert der Übertragungsversuche pro Paketlänge
- Alle Pakete haben gleiche Länge
- Dann gilt

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

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

ALOHA – Effizienz

- ▶ **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**

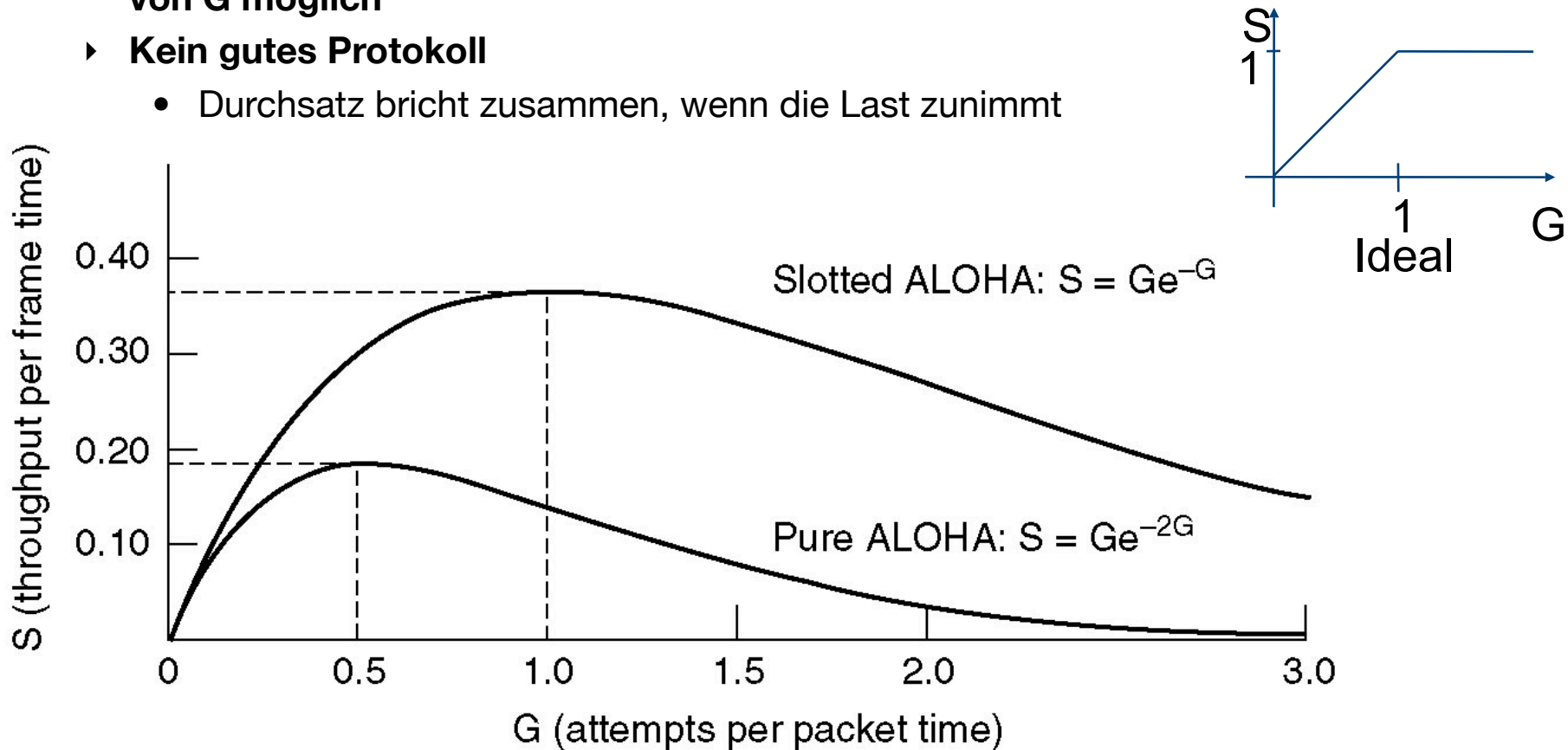


Slotted ALOHA

- ▶ **ALOHA's 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





Systeme II

Ende der 5. Vorlesungswoche

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Christian Schindelhauer
Sommer 2008