

Systeme II



Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Christian Schindelhauer

Sommersemester 2006

9. Vorlesung

24.04.2006

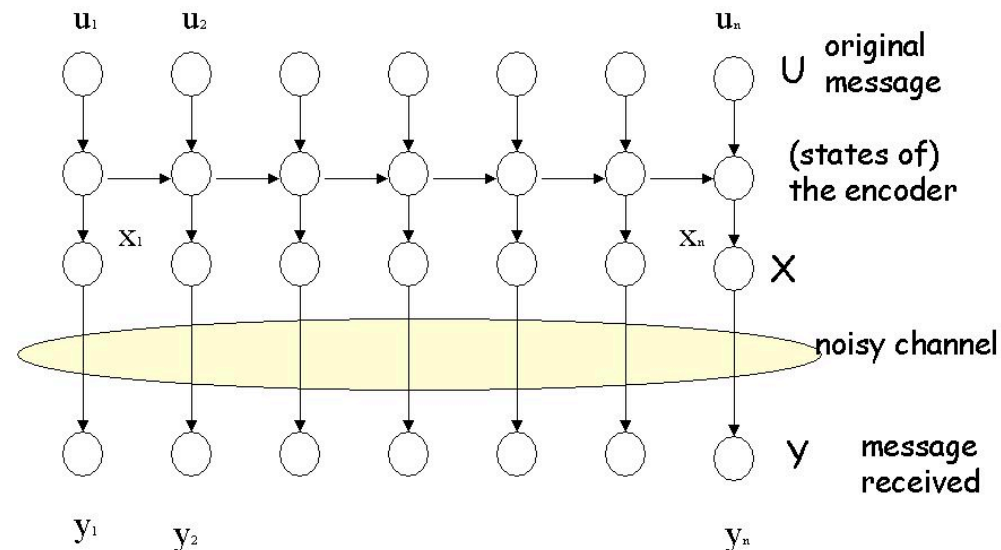
schindel@informatik.uni-freiburg.de



Faltungs-Codes

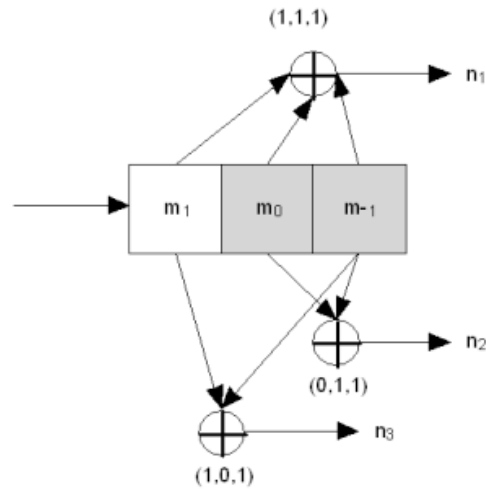
➤ Faltungs-Codes (Convolutional Codes)

- 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.

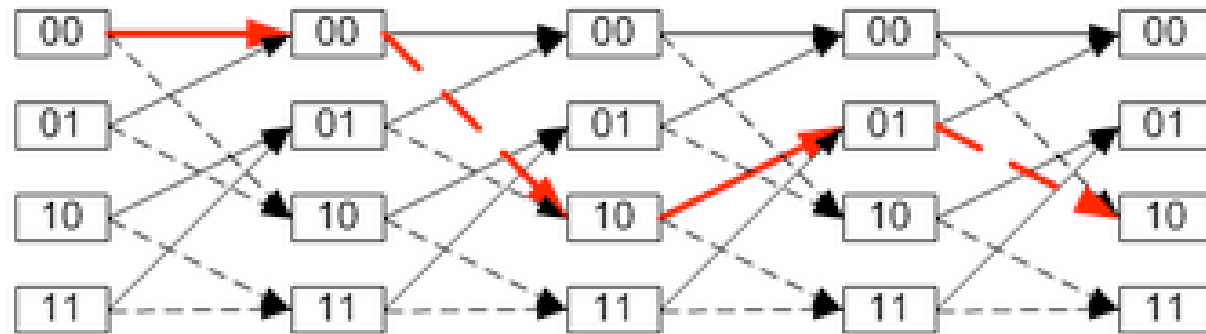




Beispiel



Trellis-Diagramm:





Dekodierung der Faltungs- Codes: Algorithmus von Viterbi

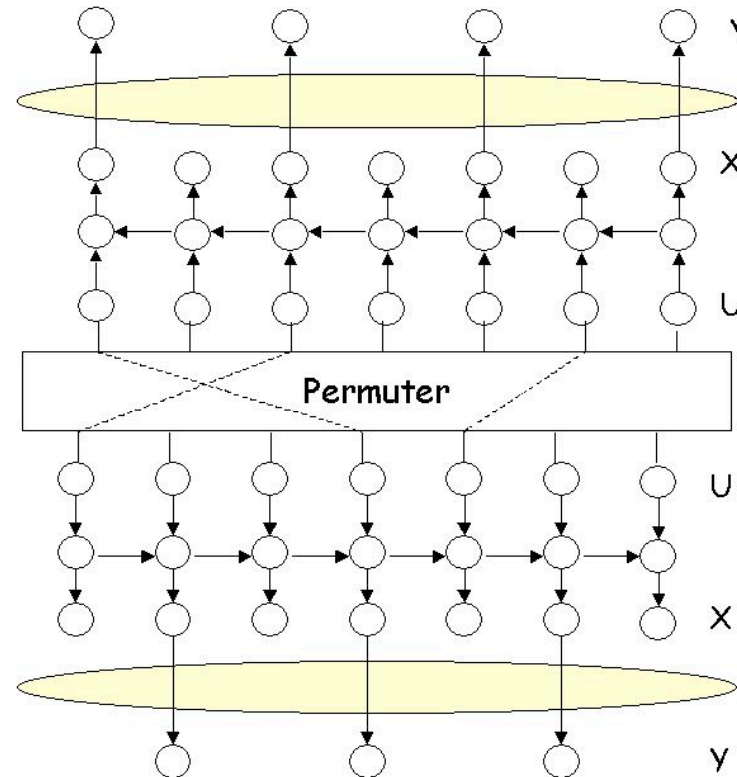
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **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
 - Weniger effizient als die Turbo-Codes



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

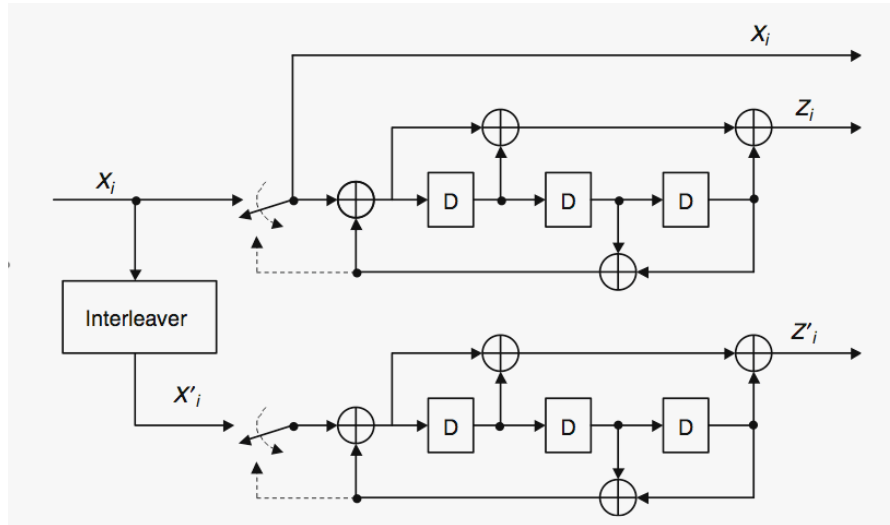




Turbo-Codes

- **Beispiel:**
 - UMTS Turbo-Kodierer

- **Dekodierung von Turbo-Codes ist effizienter möglich als bei Faltungscodes**

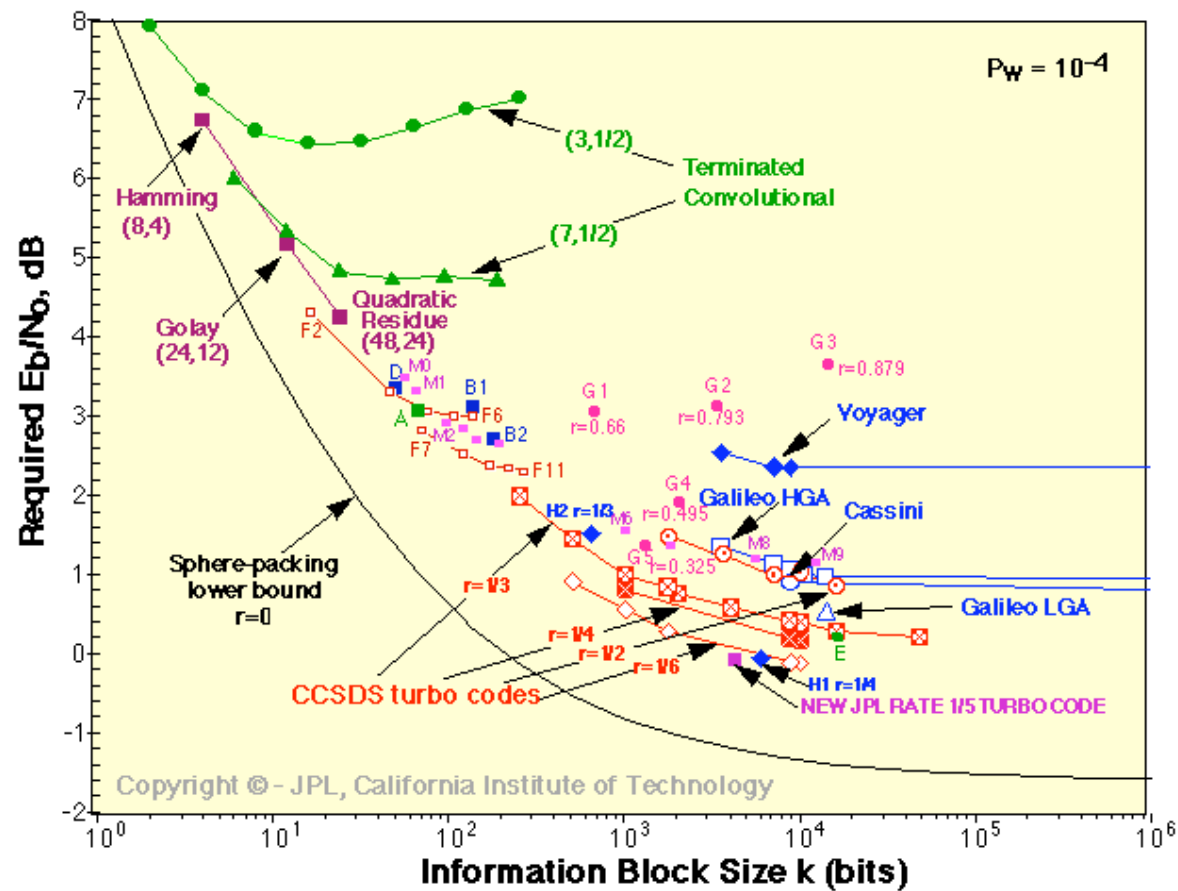




Codes im Vergleich

➤ Code-Rate versus Signal-Rausch-Verhältnis

– Stand 1998: (www331.jpl.nasa.gov/public/AllCodesVsSize.GIF)





Fehlererkennung: CRC

- **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



Rechen in \mathbb{Z}_2

➤ Rechnen modulo 2:

➤ Regeln:

– Addition modulo 2

Subtraktion modulo 2

Multiplikation modulo 2

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

– Example: 0110111011

$$+ 1101010110 = 1011101101$$



Polynomarithmetik modulo 2

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

➤ **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

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

➤ 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

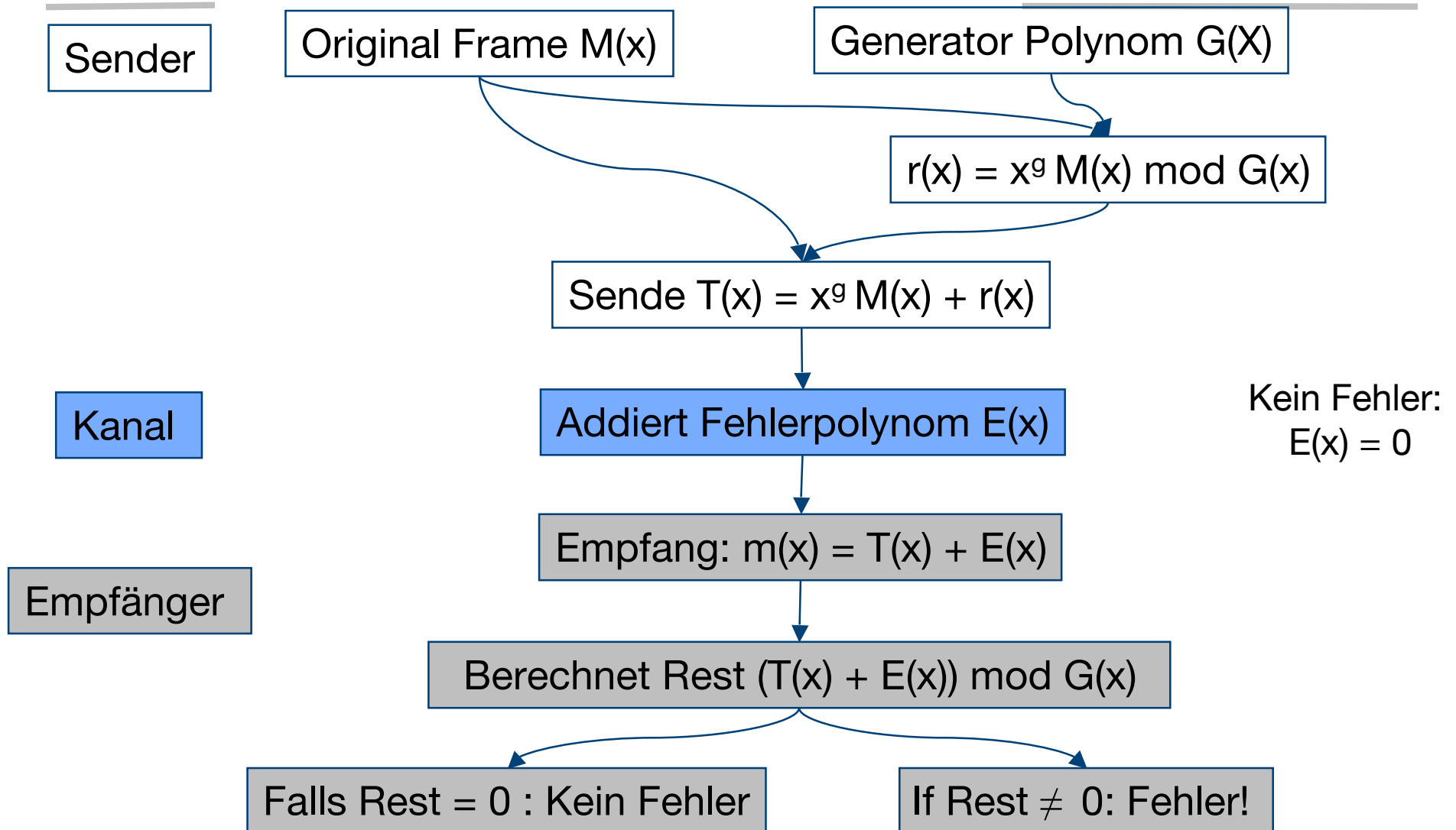
➤ Empfänger

- 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.

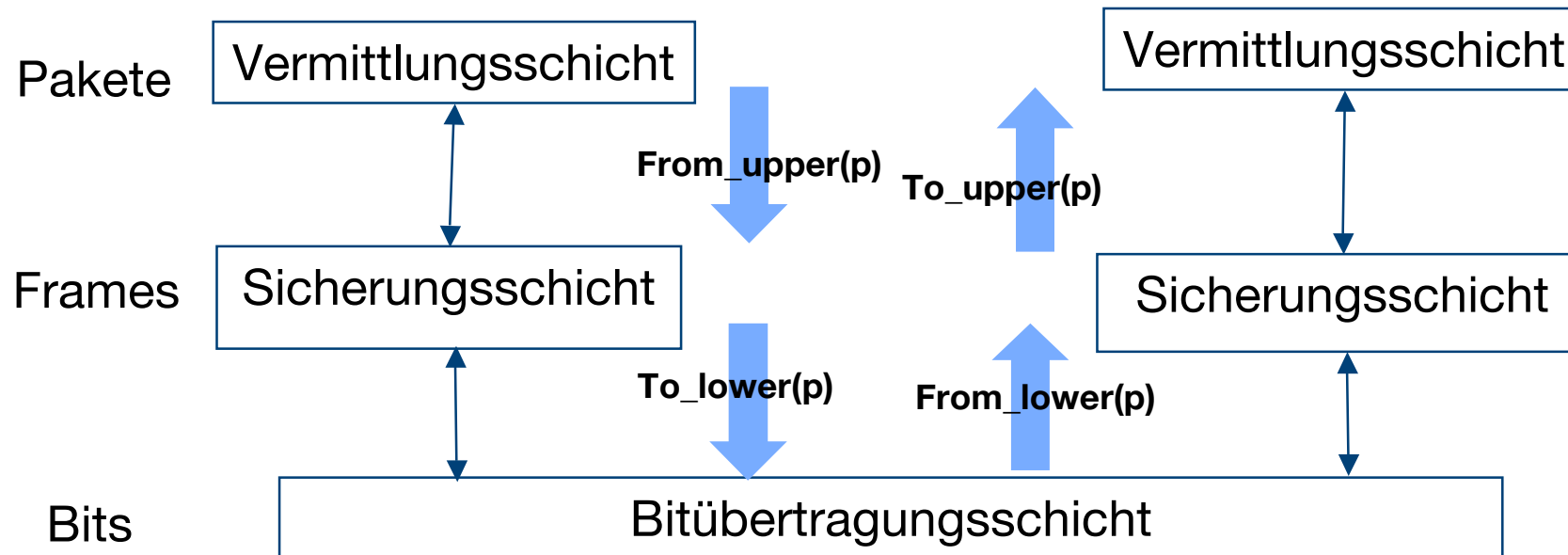
➤ **Implementation:**

- 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



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



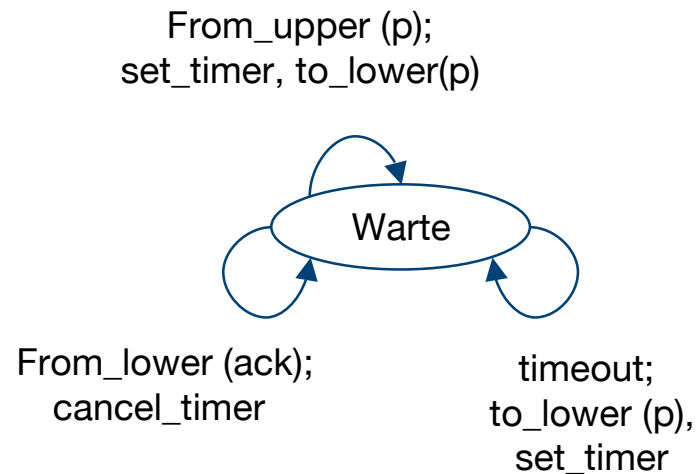
Einfaches Simplex-Protokoll mit Bestätigungen

➤ Empfänger bestätigt Pakete dem Sender

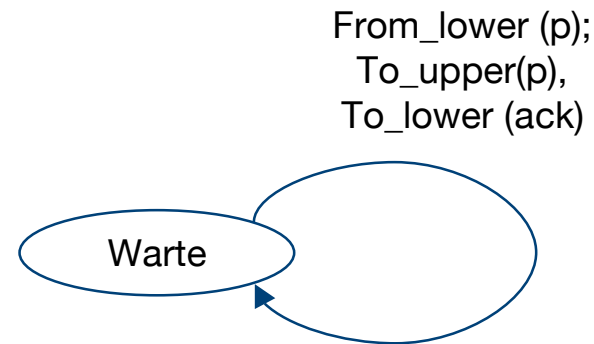
- Der Sender wartet für eine bestimmte Zeit auf die Bestätigung (acknowledgment)
- Falls die Zeit abgelaufen ist, wird das Paket wieder versendet

➤ Erster Lösungsansatz

- **Sender**



Empfänger





Diskussion

➤ Probleme

- Sender ist schneller als Empfänger

- Was passiert, wenn Bestätigungen verloren gehen?

Ende der 9. Vorlesung



Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Systeme II
Christian Schindelhauer
schindel@informatik.uni-freiburg.de