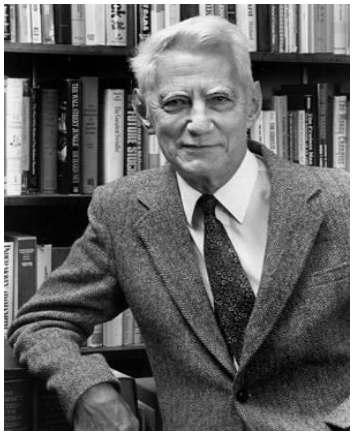
Teil A :

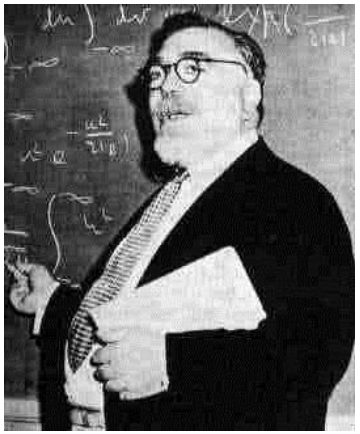
Systeme, Algorithmen, Technologien und Dienste zur Kommunikation

Kapitel 2 Grundlagen der Codierung; Codierungs- und Kompressionsalgorithmen

2.1 Einige Grundbegriffe der Informationstheorie

Informationstheorie (Begründer: Claude SHANNON, Norbert WIENER, 1948):





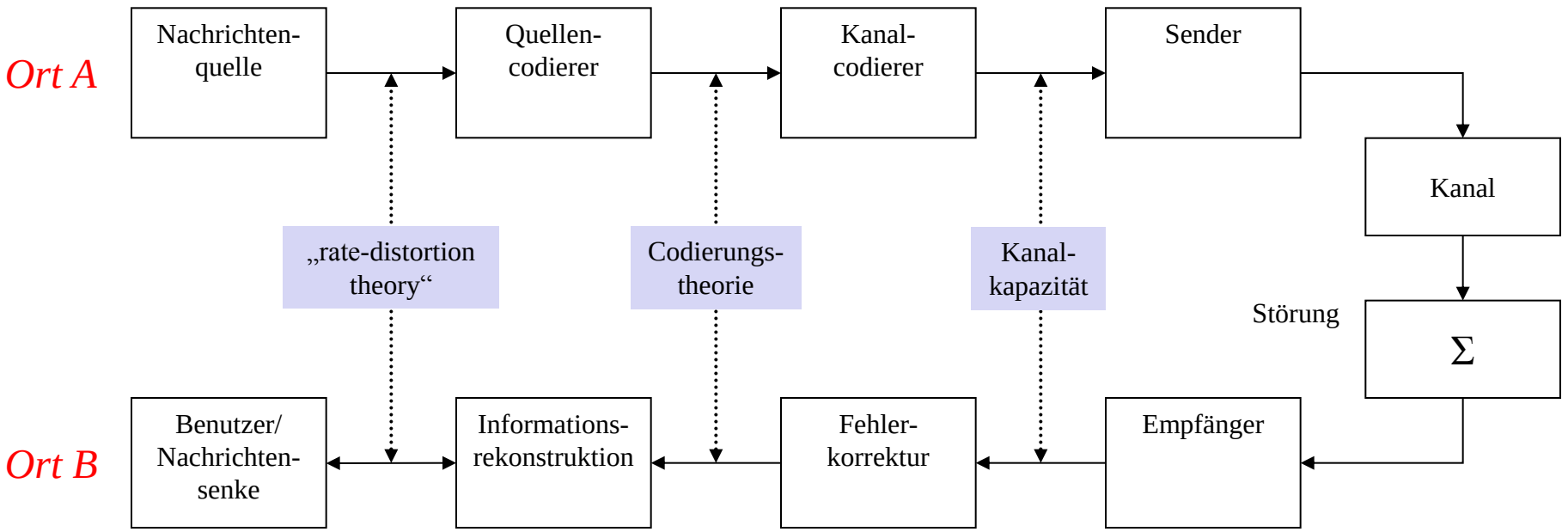
- Zweig der statistischen Kommunikationstheorie (damit auch der Wahrscheinlichkeitstheorie)

- untersucht Informationsgehalt von Nachrichten / physikalischen Beobachtungen sowie Zusammenhang zwischen Informationsgehalt und Übertragung dieser Information von Ort A zu B

nota bene:

Information hier definiert über Wahrscheinlichkeit für Auftreten von Symbolen
→ semantischer und pragmatischer Aspekt von Nachrichten unberücksichtigt!

Zugrundeliegendes Modell eines Nachrichtenübertragungssystems:



- *„rate-distortion theory“* untersucht u.a.
  - Informationsgehalt der Nachricht/Beobachtungen
  - Senderate einer Informationsquelle
  - Beziehung zwischen Senderate und Genauigkeit, mit der Nachricht bei Empfänger rekonstruiert werden kann.
- *Codierungstheorie* :
  - Konstruktion und Analyse von Codierungsverfahren zur Fehlererkennung/-korrektur bei auftretenden Übertragungsfehlern.
- *Kanalkapazität* : Maß für die Geschwindigkeit, mit der ein Kanal Nachrichten ohne Fehler oder mit vorgegebener Fehlerrate übermitteln kann.

Begriffe der Informationstheorie

Hier : Nur kurzes Resümee von einigen wenigen Inhalten des B.Sc.-Pflichtmoduls „Rechnerstrukturen“

Gegeben:
Zeichen Z_i, i = 1, 2, …, n mit Auftrittswahrscheinlichkeit p_i für Zeichen Z_i

Def.: ***Informationsgehalt*** h_i (in [bit]) von Zeichen Z_i :

$h_i = – ld (p_i) = ld (1/p_i)$

Def.: ***Mittlerer Informationsgehalt*** **h** (in [bit]) , auch ***Entropie*** genannt :

$\mathbf{h} = \sum_{i=1,\ldots,n} p_i \bullet h_i = \sum_{i=1,\ldots,n} p_i \bullet (– ld(p_i))$

Maximum von **h**, sofern p_i = const. für alle i und ergo $p_i = 1/n$, da $\sum_{i=1,\ldots,n} p_i = 1$

Somit : $H_0 := \max (\mathbf{h}) = 1/n \sum_{i=1,\ldots,n} (– ld(1/n)) = – ld(1/n) = ld(n)$ [bit]

Def.: ***Absolute Redundanz*** **R:** $\mathbf{R} = H_0 – \mathbf{h}$, wobei $H_0 \equiv$ ***Entscheidungsgehalt***, s.u.
Relative Redundanz **r:** $\mathbf{r} = R / H_0 = (H_0 – \mathbf{h})/ H_0$.

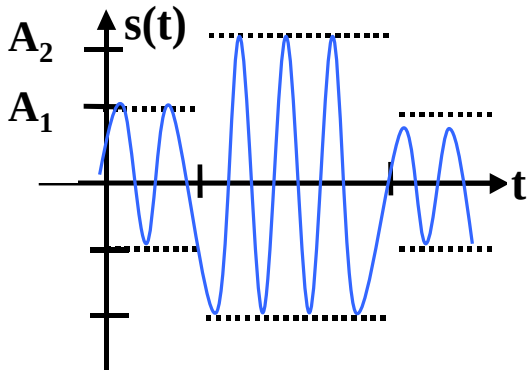
Def. ***Entscheidungsgehalt H_0*** :

Entscheidungsgehalt H_0 ist die Informationsmenge, die nötig ist, um ein Zeichen aus einem Zeichenvorrat ZV von n gleichwahrscheinlichen Zeichen mit Ja/Nein - Entscheidungen auszuwählen :

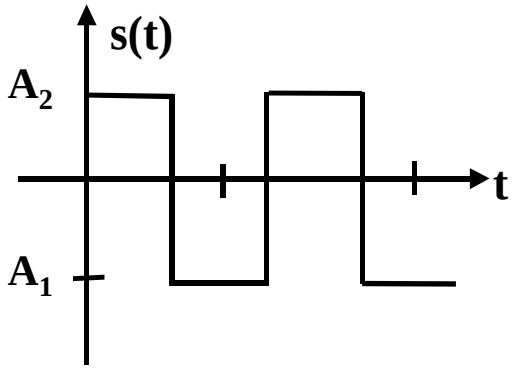
Es gilt : $H_0 = \mathrm{ld}\, n$ [bit] .

Beispiele :

(a) Sei Zeichenvorrat

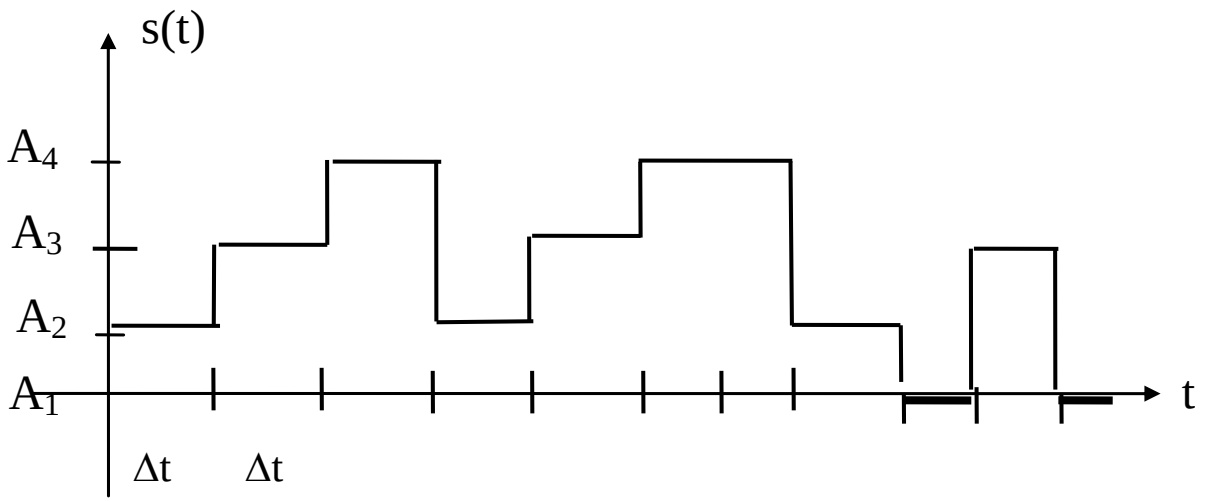


ZV={0,1}
ZV={♠, ♣}
ZV ={X, Y}
ZV = {A1, A2}
•
•
•
o.ä.



Ergo : n = 2 und $H_0 = \mathrm{ld}\,(2) = \mathrm{ld}\,(2^{1}) = 1$

(b) Sei Zeichenvorrat $ZV = \{A_1, A_2, A_3, A_4\}$



Ergo :

$n = 4$ und $H_0 = ld(4) = ld(2^2) = 2$

(c) Sei Zeichenvorrat $ZV = \{A_1, A_2, ..., A_8\}$

Ergo :

$n = 8$ und $H_0 = ld(8) = ld(2^3) = 3$

2.2 Codes und Codierung

Ziele einer Codierung, z.B.

- Absicherung gegen Übertragungsfehler
- Verschlüsselung von Daten
- rechnerinterne Repräsentation einer Menge von Symbolen, Sachverhalten, etc.
- digitale Repräsentation eines analogen Signals
- Komprimierung großer Datenmengen (u.a. zur Reduktion von Speicher- oder Übertragungsressourcen).

Def.: ***Code***

Gegeben: Zwei Zeichenvorräte Z_1, Z_2.
Sei $f : Z_1 \rightarrow Z_2$ eine Abbildung von Z_1 nach Z_2.
Dann nennen wir f eine *Codierungsvorschrift* oder kurz: *Code*.
Teilweise wird auch $f(Z_1)$ als Code bezeichnet.

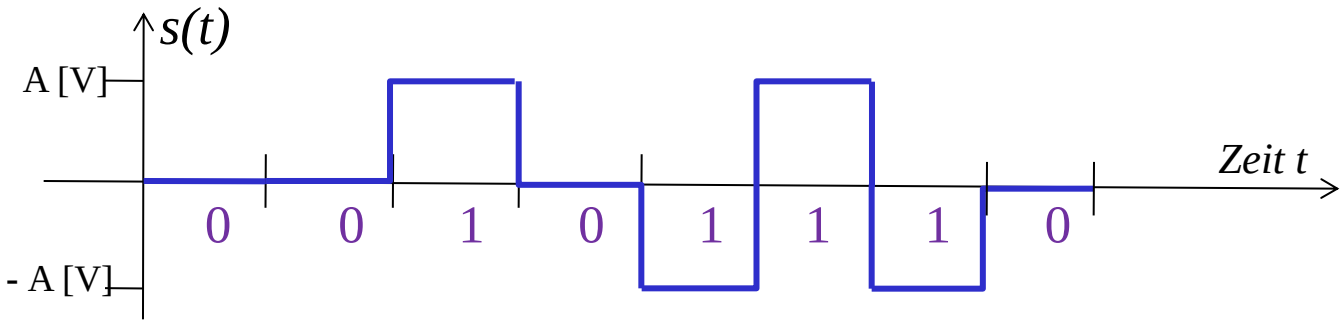
Beispiel: $Z_1 = \{\text{Hund, Katze, Goldfisch}\}$, $Z_2 = \{00, 01, 10, 11\}$

$$f : \begin{cases} \text{Hund} \rightarrow 01 \\ \text{Katze} \rightarrow 10 \\ \text{Goldfisch} \rightarrow 11 \end{cases}$$

Eigenschaften von Codes

- f : ist i.a. *nicht surjektiv* und evtl. auch *nicht injektiv* (z.B. bei verlustbehafteter Codierung).
- Beispiel für eine Codierung f einer Bitfolge, die auf Bitebene keine Abb. (im mathemat. Sinne) darstellt, da "1" auf versch. Werte abgebildet wird.

$$f: \begin{cases} 0 \rightarrow 0\ [\mathrm{V}] \\ 1 \rightarrow \mathrm{A}\ [\mathrm{V}]\ ,\ -\mathrm{A}\ [\mathrm{V}] \quad \text{(alternierend; [V] steht für [Volt])}, \mathrm{A} \in \Re^{+} \end{cases}$$



s(t) bezeichne hier das übertragene Signal (bei Datenübertragung)

Nota bene :

- Umkehrabbildung f^{-1} ist wohldefiniert
- Codierung erfolgt sozusagen zustandsabhängig, aber : Codierung einer gesamten Bitsequenz ist gleichwohl eine Abbildung.

Einige Beispiele zur Codierung in Rechnernetzen

1. Verschlüsselung

Beispiel: Codierung von Daten *d* gemäß geheimer Abbildung f durch Sender S, Übertragung der Daten $f(d)$

→ Empfänger kann entschlüsseln, sofern er

- über f informiert ist und aus f leicht f^{-1} ermitteln kann
- direkt f^{-1} kennt.

2. Codierung von Daten zur Signalübertragung

[zu Details vgl. DÜ-Verfahren in Kap. 3]

Beispiel: Verwendung von Intervallen konstanter Länge Δt pro übertragenem Bit und Codierungsvorschrift, z.B.

- „0“: kein Strom
- „1“: Strom fließt → Entscheidungsschwelle für Empfänger notwendig.

3. Codierung zur Erkennung von Bitverfälschungen

Beispiel: Ergänzung abzusichernder Daten um Redundanz-Bit(s)

→ elementar: (un)gerade Parität, d.h. 1 Zusatz-Bit

Einige Beispiele zur Codierung in Rechnernetzen (Forts.)

4. Redundanzsparende Codierung von Nutzdaten

   Beispiel: deutschsprachiger Text mit Codierungsprinzip

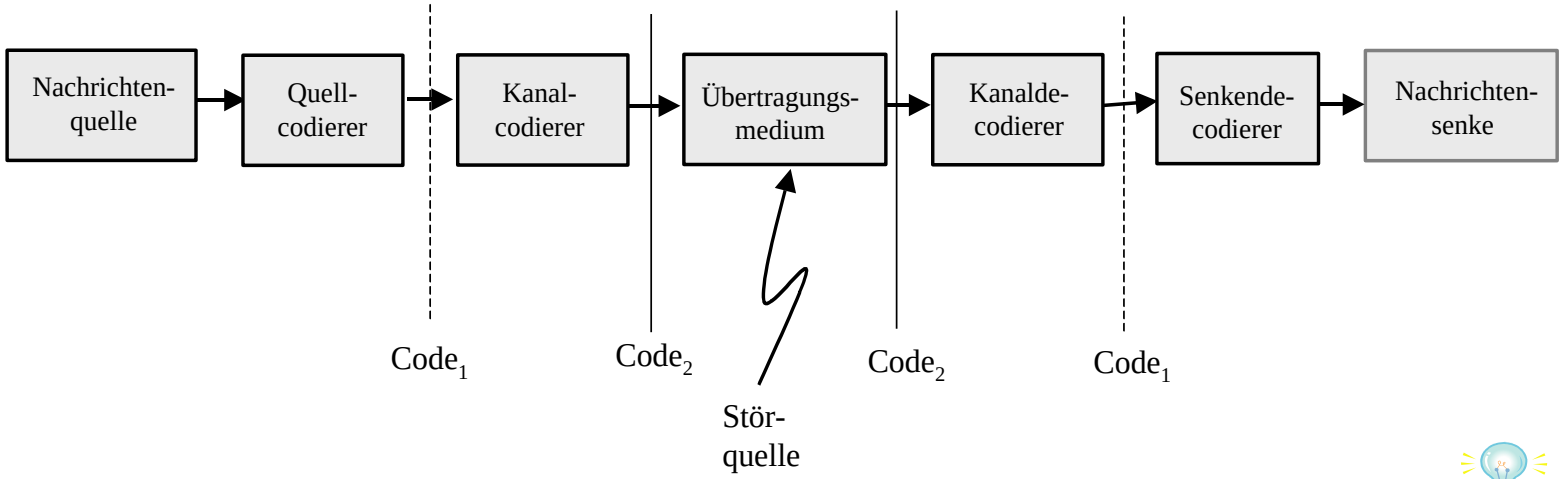
   - wenige Bits zur Codierung häufiger Buchstaben wie „e“, „a“, „n“, …
   - viele Bits zur Codierung seltener Buchstaben wie „x“, „y“, „q“, …

   → “**Präfixbedingung**“ (d.h. “kein Codewort ist Beginn eines anderen“) ermöglicht Decodierung.

5. Codierung zu übertragender Kontrollinformation

   *Beispiel: „*Typ-Feld“ im Header übertragener Protokolldateneinheiten zur Angabe des Kontrollinformationstyps, z.B. Frametypen bei HDLC

Codierung bei der Datenübertragung



Mögliche Ziele für Quellcodierer und Senkendecodierer :

- Geheimhaltung zu übertragender Nachrichten,
  z.B. durch Ver-/Entschlüsselung (Bsp.: DES, vgl. Kap. 12)
- Ermöglichung der Übertragung von Nachrichten als Folge von Binärsignalen,
  z.B. bei Sprachübertragung (Bsp.: PCM, vgl. Abschn. 2.5)
- Reduktion der (zufälligen) Redundanz in den zu übertragenden Nutzdaten,
  wie z.B. Bewegtbildsequenzen → zeitliche und räumliche Redundanz
  (Bsp.: MPEG- sowie H.26x-Normen, vgl. Abschn. 2.4) oder Redundanz-
  reduktion bei natürlichsprachlichen Texten, z.B. *Shannon-Fano-Code*
  (Wahrscheinlichkeit für „x“ << Wahrscheinlichkeit für „e“ !)

→ betroffene Schicht (ISO/OSI-Modell): insbesondere *Darstellungsschicht*

Mögliche Ziele **für Kanalcodierer bzw. -decodierer :**

- Senderseitige Codierung zur Übertragung von Bitfolgen unter Nutzung von DÜ-Verfahren mit Unterstützung einer empfangsseitigen Taktrückgewinnung im Decodierer
  (Bsp.: Manchester-Codierung, vgl. Abschn. 3.7)

- systematische Hinzufügung von Redundanz im Codierer zur Unterstützung einer empfangsseitigen Fehlererkennung durch Codeprüfung und/oder Fehlerkorrektur
  (Bsp.: CRC-Codierung, vgl. Abschn. 2.3.3)

→ betroffene Schichten (ISO/OSI-Modell): insbesondere *Datensicherungs-* und *physikalische Schicht*

Redundanz eines Codes

Sei gegeben:

- eine zu codierende Zeichenmenge $Z = \{Z_1, Z_2, \ldots, Z_n\}$ mit Auftrittswahrscheinlichkeit p_i für Zeichen Z_i ,
- für $Z_i \in Z$ bezeichne $C(Z_i)$ das zu Z_i gehörende Codewort, und
- $l(Z_i)$ bezeichne die Länge von $C(Z_i)$.

Def.: ***Mittlere Codewortlänge*** **L(C)** eines Quell-Codes C :

$$L(C) = \sum_{i=1,\ldots,n} l(Z_i) \bullet p_i(Z_i),$$

wobei

$$Z = \{Z_1, Z_2, \ldots, Z_n\}$$

und $p_i(Z_i)$ die Auftrittswahrscheinlichkeit von Z_i bezeichne .

sowie

Def.: ***Redundanz des Codes*** $\mathbf{R_{Code}}$ **:**

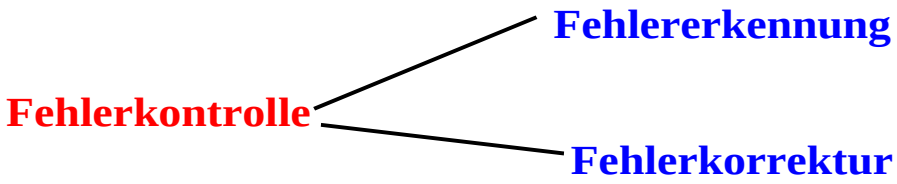
$$\mathbf{R_{Code}} = L(C) - h ,$$

wobei h den mittleren Informationsgehalt (die Entropie) bezeichne,
d.h. $h = \sum_{i=1,\ldots,n} p_i \bullet (- ld(p_i))$, siehe Def. in Abschn. 2.1

2.3 Codierung zur Fehlerkontrolle

Fehlerkontrolle bei Datenübertragung

Aufgaben der Fehlerkontrolle :



⇒ hier betrachtete Fehler :

Verfälschung, **Verlust**, **Duplizierung**, **Reihenfolgeverfälschung**
für übertragene Dateneinheiten, insbesondere auf

- Datensicherungsschicht
- Vermittlungs ~
- Transport ~

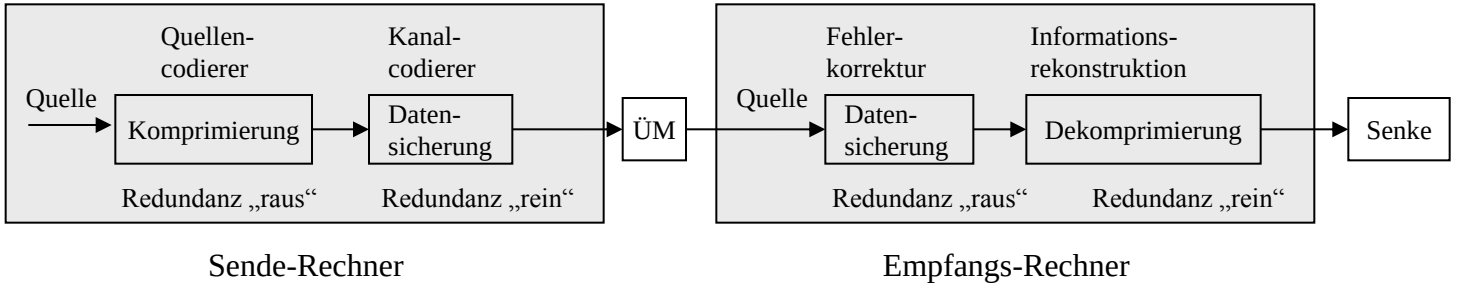
"Denkanstoß" : +/- für Fehlerkontrolle auf verschiedenen Schichten einer Protokollhierarchie ?

- **Erkennung von**
  - *Dateneinheiten (DE) -Verfälschungen*
    → Verwendung fehlererkennender/-korrigierender Codes (vgl. u.a. CRC-Verfahren in 2.3.3)
  - *DE-Verlust*
    → Ausbleiben von Bestätigung (ACK/NAK)
  - *DE-Duplizierung*
    → Nummerierung der DEen
  - *Reihenfolgeverfälschungen*
    → Nummerierung der DEen

- **Korrektur von**
  - *DE-Verfälschungen*
    → Wiederholung der Übertragung (Kopienhaltung notwendig)
    bzw. FEC (forward error correction/control)
  - *DE-Verlust*
    → Wiederholung der verlorenen DE (und ihrer "Nachfolger")
  - *DE-Duplizierung*
    → Vernichtung der Duplikate
  - *Reihenfolgeverfälschungen*
    → Umordnung beim Empfänger
    oder : Wiederholung der ersten DE außerhalb der ursprünglichen Sequenz und ihrer "Nachfolger"

Redundanz bei Datenübertragung

- allgemein:



- Datensicherung, i.a. in Schicht 2
- Komprimierung, i.a. in Schicht 6 (Darstellungs~)

spezielle Beispiele:

a) Codierungsverfahren zur *Komprimierung* :
   - ➢ bei Textübertragung:
     Shannon-Fano-Code (wenige Bits für häufige, indes längere Codewörter für seltene Zeichen)
   - ➢ bei Videoübertragung:
     MPEG, H.261, DVI, o.ä. Codes (Eliminierung von räumlicher und zeitlicher Redundanz bei Bewegtbildsequenzen)

b) Codierungsverfahren zur *Datensicherung* :
   - ➢ nur Fehlererkennung: Paritätssicherung, zykl. Redundanzprüfung (CRC)
   - ➢ Vorwärtsfehlerkontrolle: bei hinr. großem Hamming-Abstand (s.u.)

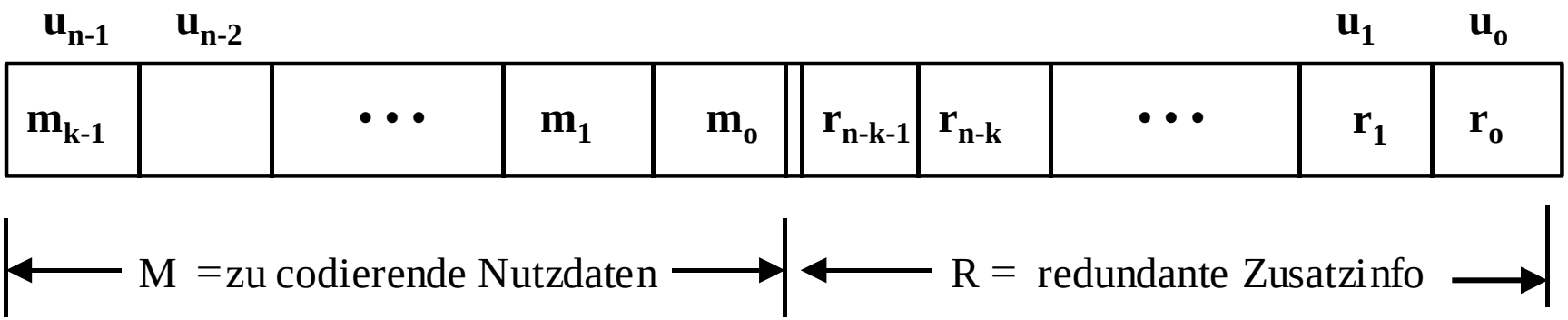
2.3.1 Fehlererkennende Codes

Störungen einer Signalübertragung → Bitverfälschungen → Übertragungsblock-/ Paketverfälschungen

(Übertragungsweg) *(Physikal. Schicht)* *(≥ Datensicherungsschicht)*

- Vorgehensweise bei der Codierung zur Fehlerkontrolle :
  Hinzufügung redundanter Zusatzinformation zu Nutzdaten

- Beispiel für den Aufbau eines (n-Bit) Codewortes U = (u_{n-1}, ..., u_1, u_o) :



Die Codierung definiert eine Abbildung γ_c dergestalt, dass :

$$\gamma_c : \quad \mathbb{M} \longrightarrow \mathbb{U}_c \qquad M \longrightarrow U$$

mit $\mathbb{M}$ = Menge der Bitmuster, die zur Übertragung anstehen können (zu codierende Nutzdaten)

$\mathbb{U}_c$ = Menge der zu verwendenden Codewörter eines Code C.

- **Hamming-Abstand h (U_1, U_2) zweier** (gleich langer) **Codewörter** U_1 und U_2 :
  h (U_1, U_2) = Anzahl der zu ändernden Bitwerte, um U_1 in U_2 überzuführen

- **Hamming-Abstand h_c eines Code C** (mit Codewörtern gleicher Länge) :

$$h_c = \min \{h(U, U') \mid U, U' \in \mathbb{U}_c \ \wedge \ U \neq U'\}$$

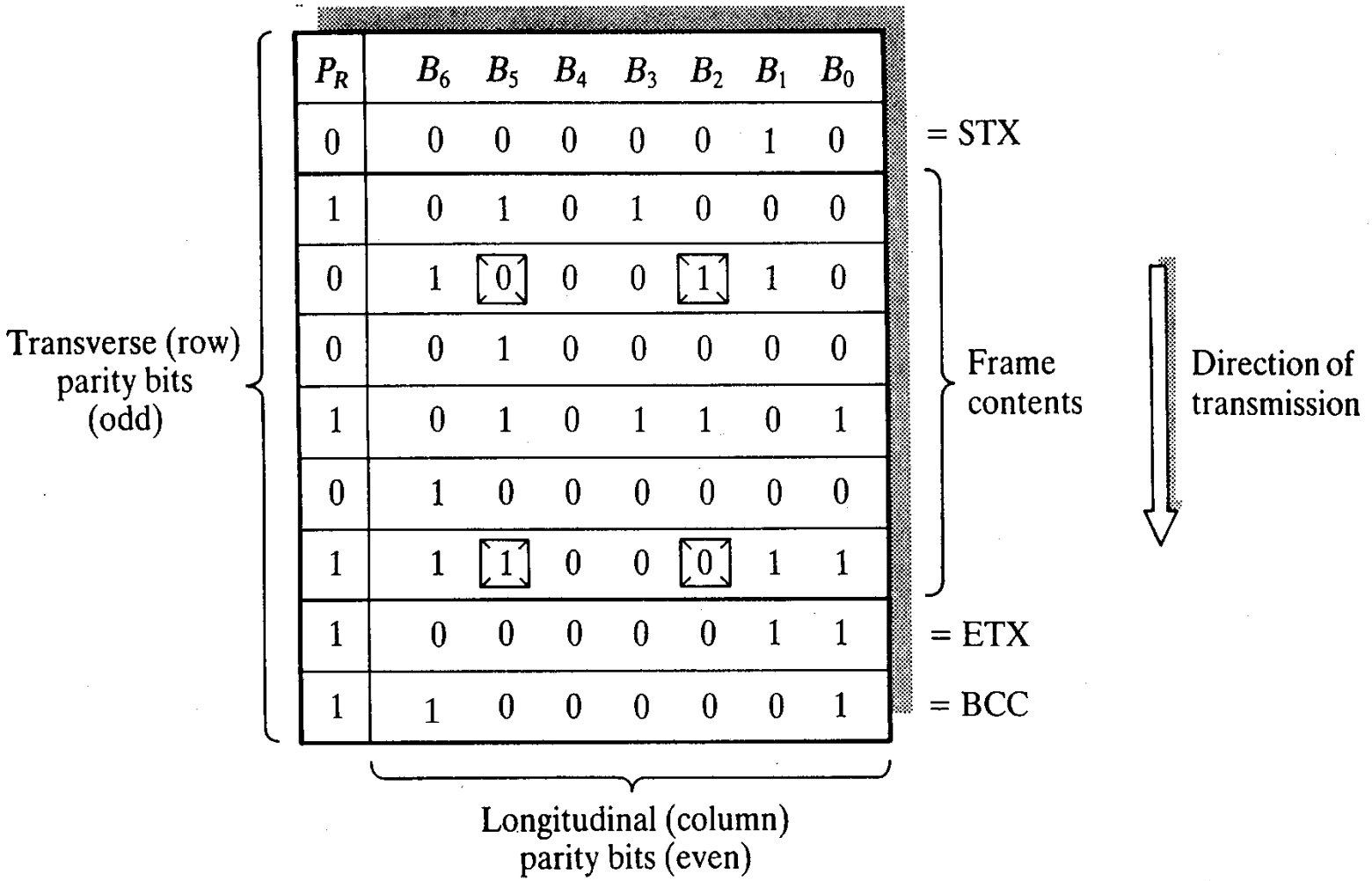
Auswirkung des Hamming-Abstandes auf die Möglichkeit der Fehlerentdeckung/-korrektur :

- Hamming-Abstand **d** → **Entdeckung** aller μ-Bit-Fehler, **$\mu \leq d-1$**, $\mu \in N$
- Hamming-Abstand **2d +1** → **Korrektur** aller ν -Bit-Fehler, **$\nu \leq d$**, $\nu \in N$
  ABER: Risiko fehlerhafter "Korrektur " !

Beispiel : $h_c = 5$ → *Entdeckung* aller 1-, 2-, 3- und 4-Bit Fehler
sowie *Korrektur* aller 1-, 2- Bit Fehler möglich

2.3.2 Paritätsprüfungen

Paritätsprüfung : Zeilen-/Spaltenparität



P_R	B_6	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	1	0	= STX
1	0	1	0	1	0	0	0	
0	1	0	0	0	1	1	0	
0	0	1	0	0	0	0	0	
1	0	1	0	1	1	0	1	
0	1	0	0	0	0	0	0	
1	1	1	0	0	0	1	1	
1	0	0	0	0	0	1	1	= ETX
1	1	0	0	0	0	0	1	= BCC

2.3.3 Zyklische Codierung (CRC)

(n, k) –zyklische Codierung

- ***Bedingungen*** für (n, k)-zyklische Codierung
  (**CRC** : ***c****yclic* ***r****edundancy* ***c****heck*):

  (B1) Länge der zu codierenden Nutzdaten : k [Bit]
  (B2) Länge des Codewortes : n [Bit]
  (d.h. n-k [Bit] redundante Zusatzinformation)
  (B3) mit einem Codewort $U = (u_{n-1}, u_{n-2}, ..., u_1, u_0)$
  ist auch $U' = (u_0, u_{n-1}, u_{n-2}, ..., u_2, u_1)$
  ein Codewort.

- ***Satz :*** Für gegebene n, k, $k < n$, existiert (mindestens) ein Polynom P(X) vom Grad n-k, das einen (n,k)-zyklischen Code erzeugt.

  Bem.: P(X) werde als ***Generatorpolynom*** des (n,k)-zyklischen Code bezeichnet.

- ***Vorgehensweise*** bei (n,k)-zyklischer Codierung :
  Sei M(X) das aus den zu codierenden Nutzdaten
  $M = (m_{k-1}, ..., m_1, m_0)$ resultierende Polynom, d.h.:
  $M(X) = m_{k-1} \bullet X^{k-1} + ... + m_1 \bullet X + m_0$

  Sei ferner

  P(X) ein Generatorpolynom, das (n,k)-zykl. Code erzeugt.

1. Schritt : Multiplikation von M(X) mit X^{n-k} ;

Sei $Y \equiv X^{n-k} \bullet M(X)$.

2. Schritt: Division $\frac{Y}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$

wobei R(X) den bei der Division $\frac{Y}{P(X)}$ entstandenen Rest

bezeichnet, d.h. es gilt :

$Y = Q(X) \bullet P(X) + R(X)$ mit $R(X) = r_{n-k-1} \bullet X^{n-k-1} + ... + r_1 X + r_0$

Bem : Division und Addition erfolgen modulo 2.

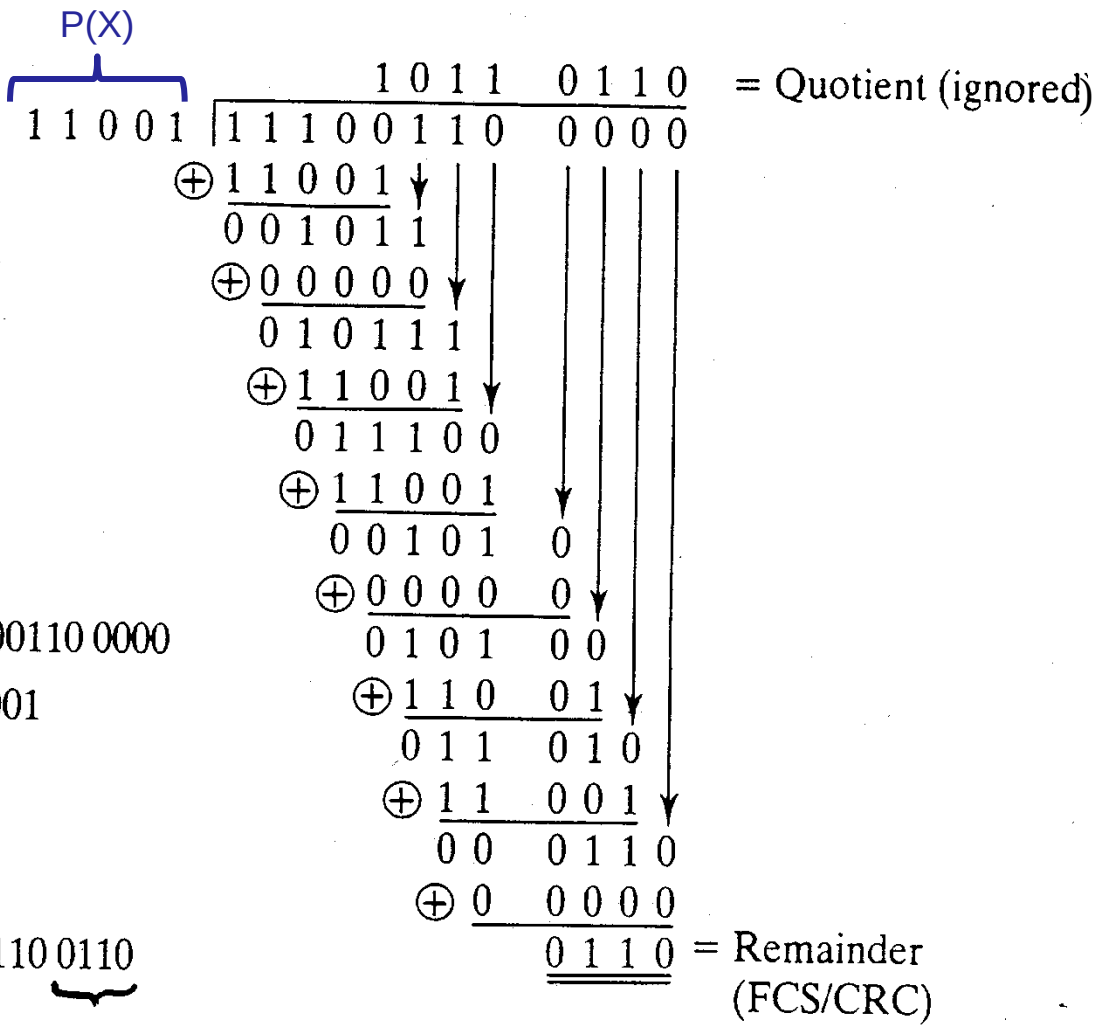
3. Schritt : Abzusendende Dateneinheit

$$U(X) = Y + R(X) = X^{n-k} \bullet M(X) + R(X)$$

Bem : R wird auch als ***Blockprüfzeichenfolge*** bezeichnet.

CRC (Beispiel) :

Sender



Frame contents: 11100110

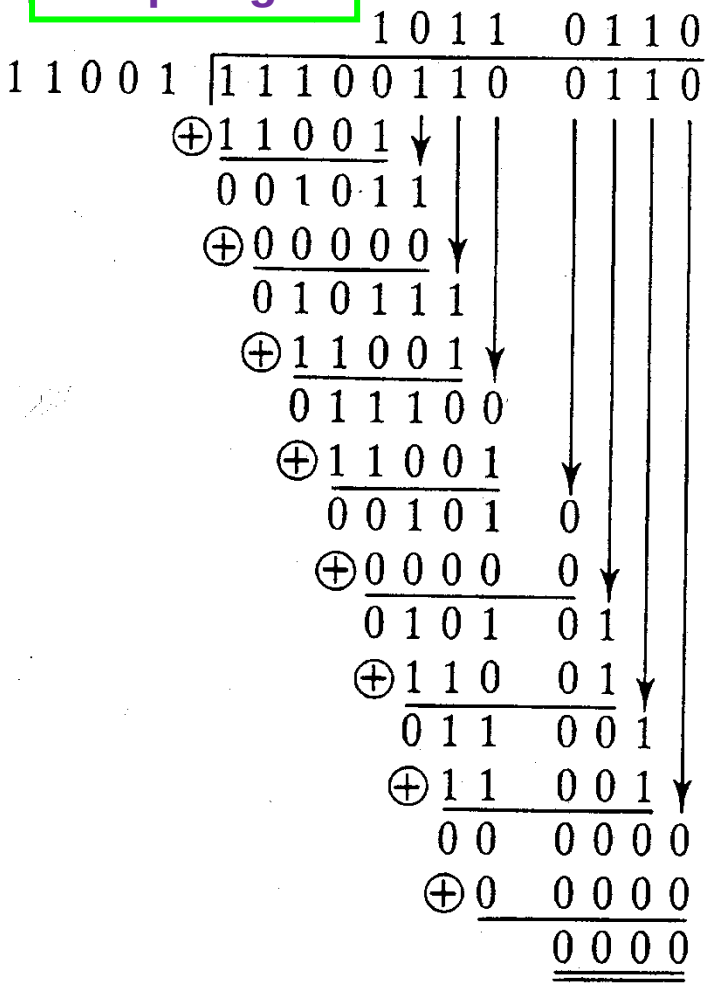
With appended zeros: 11100110 0000

Generator polynomial: 11001

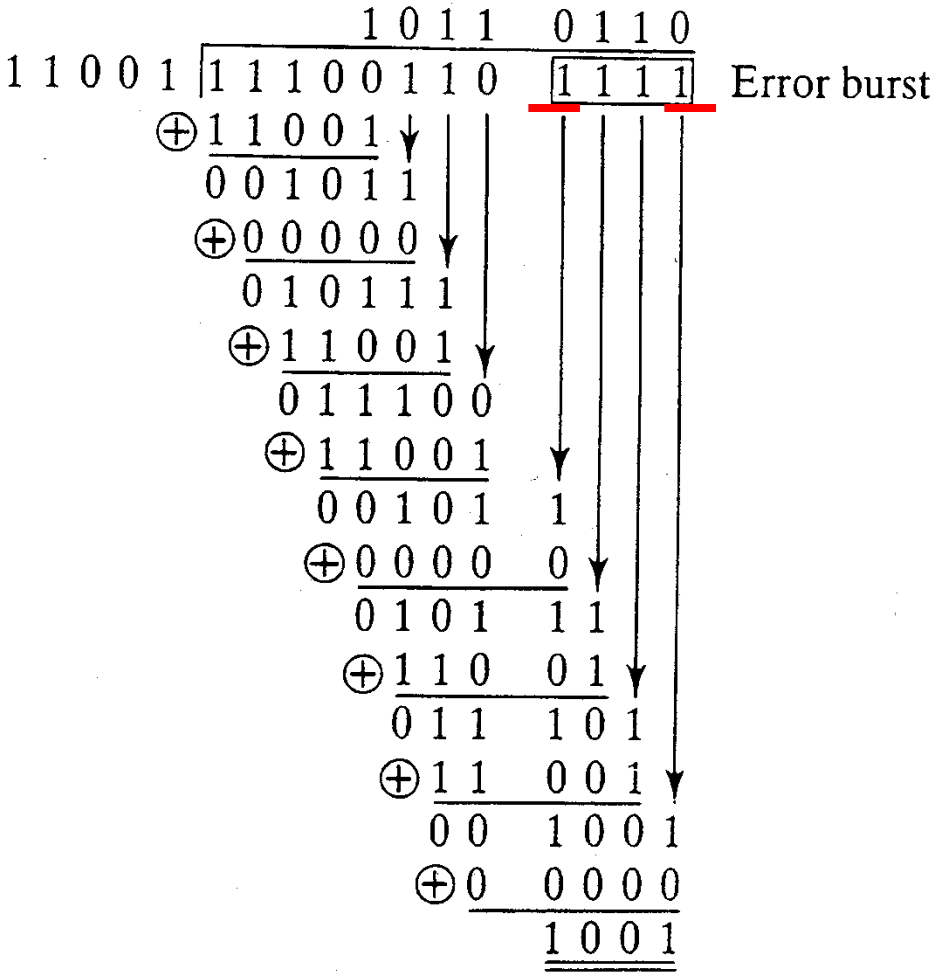
Transmitted frame: 11100110 0110

redundante Zusatzinfo.

Empfänger



Remainder = 0: no errors



Remainder ≠ 0: error detected

Qualität der Fehlererkennung bei zyklischer Redundanzprüfung

$$E(X) = U(X) + F(X) \qquad (*)$$

empfangene Dateneinheiten — abgesandte Dateneinheiten — Fehlerpolynom

Darstellung von E(X) in der Form :

$E(X) = T(X) \bullet P(X) + S(X)$; S ≡ **"Syndrom"**

Damit gilt :

$F(X) = U(X) + E(X)$, wegen Gl. (*) und $F(X) = U(X) + T(X) \bullet P(X) + S(X)$

Da $U(X) = Q(X) \bullet P(X)$ als Codewort → $F(X) = [Q(X) + T(X)] \bullet P(X) + S(X)$.

➢ ***Satz*** : E ist Codewort ⇔ Fehlerpolynom F durch P teilbar.

Beweis : " ⇒ " E ist Codewort → S = 0 → F = (Q+T) • P → F durch P teilbar.
" ⇐ " F durch P teilbar → S = 0 → E = T • P → E ist Codewort. ■

➢ Zur Wahl von Generatorpolynomen

→ *Anforderungen :* Wunsch nach

- Erkennung von 1-Bit-Fehlern :
  1-Bit-Fehler darstellbar in der Form $F(X) = X^i$;
  falls Generatorpolynom P(X) ≥ 2 Terme enthält, folgt :
  $F(X) = X^i$ nicht durch P(X) teilbar; denn ansonsten existierte C(X) mit : F(X) = C(X) P(X)
  ⇒ alle 1-Bit-Fehler erkannt, falls P(X) aus ≥ 2 Termen bestehend.

- Erkennung von 2-Bit-Fehlern :
  2-Bit-Fehler darstellbar in der Form $F(X) = X^i + X^j$, $i \neq j$ und o.B.d.A. $i < j$
  $\rightarrow$ $F(X) = X^i (1+X^{j-i})$;
  $\Rightarrow$ alle 2-Bit-Fehler u. a. erkannt, falls P(X) nicht (ohne Rest!) teilbar durch X
  und
  $X^k + 1$ nicht teilbar durch P(X) $\forall$ k=1, ..., n .

- Erkennung einer ungeraden Anzahl von Fehlern :
  bei ungerader Anzahl von Fehlern $\rightarrow$ F(X) kann Faktor X+1 nicht enthalten
  $\Rightarrow$ jede ungerade Anzahl von Fehlern erkannt, falls P(X) den Faktor X+1 enthält.

- Erkennung von Fehlerbüscheln, die $\leq$ r aufeinanderfolgende Bits innerhalb der Dateneinheit betreffen :
  $\Rightarrow$ jeder (n,k)-zykl. Code mit n-k = r entdeckt o.g. Fehlerbüschel, unter Voraussetzung eines Generatorpolynoms $P(X) = X^r + ... + 1$.

Beispiele verbreiteter, standardisierter Generatorpolynome :

- **CRC-12** : $P(X) = X^{12} + X^{11} + X^3 + X^2 + X + 1$ für 6-Bit-Zeichen
- **CRC-16** : $P(X) = X^{16} + X^{15} + X^2 + 1$
- CRC des CCITT / ITU, z.B. für **HDLC** : $P(X) = X^{16} + X^{12} + X^5 + 1$
- **CRC-32**, z.B. **Ethernet** : $P(X) = X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

CRC-32 erkennt 99,999 999 95 % aller aufgetretenen Bitfehler (!), vgl. [Ste 08].

ABER: CRC leistet nur Fehlererkennung und KEINE (!) Fehlerkorrektur.

2.3.4 Vorwärtsfehlerkontrolle

- Wozu Vorwärtsfehlerkontrolle (FEC) ?
  - Simplex-Übertragung (kein “feedback“),
  - Echtzeitkommunikation (Echtzeitbedingungen),
  - Gruppenkommunikation (Aufwandsreduktion für Sender),
  - Mobilkommunikation (zu häufig Fehler)

- Wie realisierbar auf Datensicherungsschicht ?
  → allgemein : hinreichend großer Hamming-Abstand

- Wie realisierbar bei verlustbehaftetem Vermittlungsnetz ?
  → Redundanz auf der Ebene übertragener Pakete

➢ *“Exkurs“ :*

FEC auf Paketebene

→ Idee : Sende Nutzdaten M in Form einer Folge von Paketen P_1, P_2, ..., P_n (mit dem Ziel, *n* so wählen zu können, dass *x* % Paketverluste noch akzeptabel sind, d.h. durch FEC “maskiert“ werden können),
Details zum Procedere vgl. Folgefolie

FEC auf Paketebene

→ **Idee** (s.o.): Sende Nutzdaten M in Form einer Folge von Paketen P_1, P_2, ..., P_n (mit dem Ziel, *n* so wählen zu können, dass *x* % Paketverluste noch akzeptabel sind, d.h. durch FEC "maskiert" werden können).

Procedere :

- *Schritt 1* : Wir betrachten, wie bei CRC, die zu übertragenden Nutzdaten

m_{k-1}	m_{k-2}	• • •	m_1	m_0

als Polynom $M(X) = m_{k-1} \bullet X^{k-1} + ... + m_1 \bullet X + m_0$ vom Grad k-1.

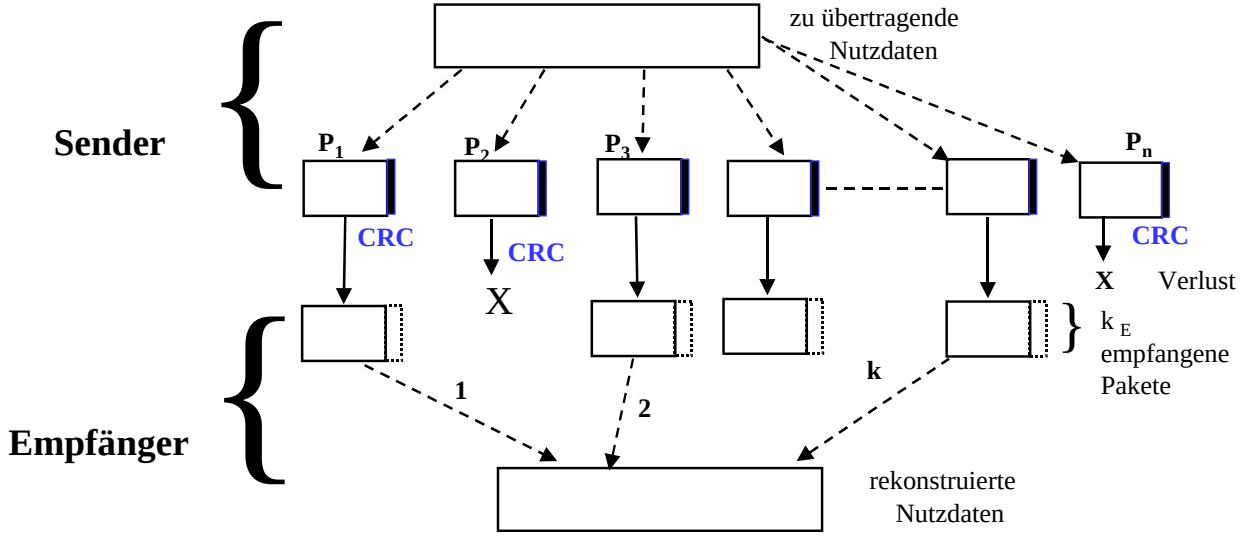
- *Schritt 2* : Sender sendet n ≥ k <u>verschiedene</u> Stützpunkte, z.B. jeden Stützpunkt in einem separaten Paket

- *Schritt 3* : Empfänger erhält k_E Stützpunkte und kann die m_0, m_1, ..., m_{k-1} berechnen (solange $k_E \geq k$) auf Basis von k beliebig, aus den k_E korrekt empfangenen Paketen [*)], ausgewählten Stützpunkten. ■

[*)] ERGO: Werden ***mindestens k Stützpunkte*** korrekt empfangen, so können sämtliche übertragenen Nutzdaten komplett (fehlerfrei) rekonstruiert werden;
werden ***weniger als k Stützpunkte*** korrekt empfangen, so können KEINE Nutzdaten mehr rekonstruiert werden. „ALL OR NOTHING" !!!

FEC auf Paketebene

- Beurteilung :
  - Vorteil : Redundanz kann an beobachtete Paketverlusthäufigkeit (ist indes nur für Vergangenheit bekannt!) des Netzes angepasst werden
  - Nachteile : redundante Pakete zu versenden; Berechnungsaufwand (besonders für Empfänger)
- Erweiterung des vorgestellten FEC-Ansatzes um Maßnahmen zur Fehlerkontrolle für die übertragenen Pakete P_1, P_2, P_3, …, P_n.



➢ Exemplarische *Illustration des Procederes* :

- (m_1, m_0), $m_i \in \mathbb{R}$ zu übertragen

- Wir betrachten : $Y = m_1 \bullet X + m_0$
  ... und versenden *n* Stützpunkte (a_1 , b_1), (a_2 , b_2), ..., (a_n , b_n)
  für die Gerade in jeweils einem Paket

- Solange Empfänger ≥ 2 Pakete – z.B. (a_r , b_r), (a_s , b_s) – erhält, löst er einfach das Gleichungssystem

$$b_r = m_1 \bullet a_r + m_0$$
$$b_s = m_1 \bullet a_s + m_0$$

und gewinnt so (m_1, m_0), d.h. die übertragenen Nutzdaten [*)]. ■

---

*) *Nota bene* : Da in diesem Bsp. $m_i \in \mathbb{R}$ → Approx. wäre notwendig, da Menge der REAL-Zahlen in Rechner immer endlich [ABER : Dieses Problem entfällt, sofern $m_i \in \{0,1\}$ → dann nur 4 Geraden G1, …, G4 möglich, insbesondere
G1: y = 0, G2: y = 1, G3: y = x und G4: y = x+1

... und mögl. Stützpunkte für G4 z. B.: (-1,0), (0,1), (1,2), (2,3), ...]

2.4 Videokomprimierungsalgorithmen

2.4.1 Räumliche und zeitliche Redundanz

Welche Arten von Redundanz in Videostrom ?

- **zeitliche Redundanz**

Bsp.: Nachrichtensprecherin vor konstantem Hintergrund;
langsamer Kameraschwenk (→ Bewegungsvektor relevant!)

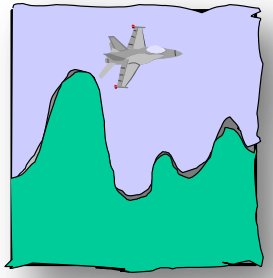


Bild zu $t = t_i$

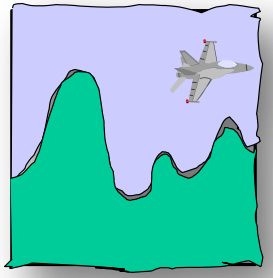


Bild zu $t = t_i + \Delta t$

- **räumliche Redundanz**

Bsp.: größere "homogene" Bildbereiche in einem Bild

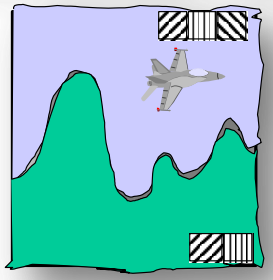


Bild zu $t = t_i$

n.b.: Redundanzreduktion → Verluste besonders problematisch (Fehlerfortpflanzung)

- Bsp. zu Normen für Videocodierung: MJPEG, MPEG-x, H.26x, ...

2.4.2 Das JPEG-Verfahren

➢ **JPEG** (= Joint Photographic Experts Group)

→ ISO/ITU-Standard [ISO-10918 sowie CCITT-Recommendation T.81]

➢ ***ZIEL von JPEG*** :
Standard für Speichern von (Fest-)Bildern in komprimiertem Format

- Erreichtes Komprimierungsverhältnis :
  - 100:1 mit erheblichem Verlust, und
  - 20:1 mit unerheblichem Verlust.

- Wesentlicher benutzter Basisalgorithmus :
  DCT (= *Diskrete Cosinus-Transformation*)

  DCT transformiert Helligkeitswerte in den Frequenzraum, vgl. Fourier-Transformation von Zeit- in Frequenzbereich in Kapitel 3.

Neuerer JPEG-Standard :

JPEG 2000 (seit Jan. 2001) [festgelegt in : ISO/IEC 15444-1 & 2 sowie in ITU-T Recom. T.800 & T.801]

Wesentliche Schritte des JPEG-Algorithmus (nur Grobbeschreibung)

- **Schritt 1 :** (Optionaler) *Wechsel des Farbmodells* von (meist) RGB-Farbraum [Rot-Grün-Blau] in das YCbCr-Farbmodell.

- **Schritt 2 :** *Tiefpassfilterung und Unterabtastung* der Farbabweichungssignale Cb und Cr (verlustbehaftet)

  → dadurch Verringerung der Auflösung der Farbkanäle.

- **Schritt 3 :** Für jedes Teilbild :

  Einteilung in Makroblöcke zu je 8x8 Pixel.

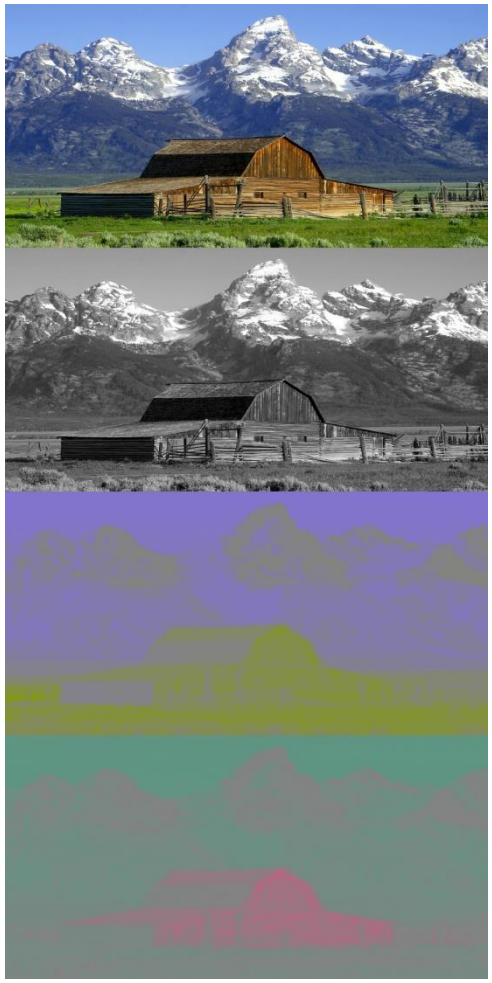
- **Schritt 4 :** Für jeden Makroblock :
  DCT des Helligkeitswertes f(i,j) in den Frequenzraum,
  insbes.

$$F(u,v) = \tfrac{1}{4} \sum_{i,j=0,\ldots,7} f(i,j) \cdot h_u \cdot h_v \cdot \cos((2i+1) \cdot u \cdot \pi/16) \cdot \cos((2j+1) \cdot v \cdot \pi/16)$$

$h_u = 1/\sqrt{2}$ wenn $u=0$, $h_u = 1$ sonst.

Wesentliche Schritte des JPEG-Algorithmus

- Details zu **Schritt 1 :** (Optionaler) *Wechsel des Farbmodells* von (meist) RGB-Farbraum [Rot-Grün-Blau] in das ***YCbCr-Farbmodell***.



Originalbild

Y-Komponente (Helligkeit, Luminanz)

Cb-Komponente (Farbigkeit, Chrominanz):
Abweichung „grau“ in Richtung „blau“/ „gelb“

Cr-Komponente (Farbigkeit, Chrominanz):
Abweichung „grau“ in Richtung „rot“/ „türkis“

Wesentliche Schritte des JPEG-Algorithmus (Fortsetzung)

- **Schritt 5 :** *Quantisierung*,
  d.h. Division von F(u,v) durch eine Zahl q(u,v) und Rundung der Resultate.
  Ergo :

$$F'(u,v) = \text{int}\left(F(u,v) / q(u,v) \right)$$

  → evtl. stark verlustbehafteter Schritt.

- **Schritt 6 :** *Umsortierung der Werte F'(u,v)*
  → u.a. Übertragung gemäß sog. *Zick-Zack-Serialisierung*.

- **Schritt 7 :** *Lauflängen-Codierung* und anschließend *Entropie-Codierung*
  (z.B. Huffman- oder arithmet. Codierung)

nota bene :
Details vgl. u.a. P.A. Henning „Taschenbuch Multimedia“
(Fachbuchverlag Leipzig, 2. Auflage, 2001).

2.4.3 Die Familie der MPEG-Verfahren

- **MPEG** (= ISO/IEC Moving Pictures Experts Group)
  → Serie von Standards
  - ***1993***: **MPEG-1** → - Verwendung u.a. für Video-CDs;
    - MPEG-1 Layer 3 (kurz: MP3) gehört zu Audio-Teil von MPEG-1.
  - ***1994/95***: **MPEG-2** → - Video- und Tonformate in TV-Qualität;
    - Verwendung u.a. für DVD-Videos.
  - *nur geplant, aber nie erschienen*: **MPEG-3**
  - ***1998-2001***: **MPEG-4** → - u.a. Beschreibung von Container-Format (an QuickTime angelehnt);
    - 3D-Sprache ähnlich Virtual Reality Modeling Language (VRML) und insbes. verbesserte Videokompression (im Vergleich zu MPEG-Vorgänger-Standards).
  - ***2002***: **MPEG-7** → - System zur Beschreibung multimedialer Inhalte.

- Überdies: **H.26x-Standards** der ITU-T Video Coding Experts Group (VCEG)
  (insbes. $x \in \{1,2, \ldots 5\}$)
  - ***Jan. 2013***: **H.265** auch „High Efficiency Video Coding“ (HEVC) → Verwendung u.a. für hocheffiziente Komprimierung von TV-Kanälen in HD-Qualität

Einige Grundprinzipien der MPEG-Standards

- **Prinzip :** ***Zerlegung der einzelnen (Video-)Frames in kleinere Einheiten,***
  wie :
  - **Bildscheiben** (“*Slices*“)
  - **Makroblöcke** (“*Macro Blocks*“)
  - **Blöcke** (“*Blocks*“).

→ VORTEIL:

Kleinere Einheiten, z.B. Makroblöcke, können (näherungsweise) in einem Frame oder in Folge-Frames nochmals auftauchen.

Ergo: bei kleinen Veränderungen wenig Information benötigt, da nur der Referenz-Makroblock und die Veränderungen benötigt werden, um den neuen Makroblock vollständig zu berechnen.

MPEG-Grundprinzipien (Fortsetzung)

- **Prinzip :** ***Unterschiedliche Grade von Abhängigkeiten zwischen zeitlich aufeinanderfolgenden Frames***

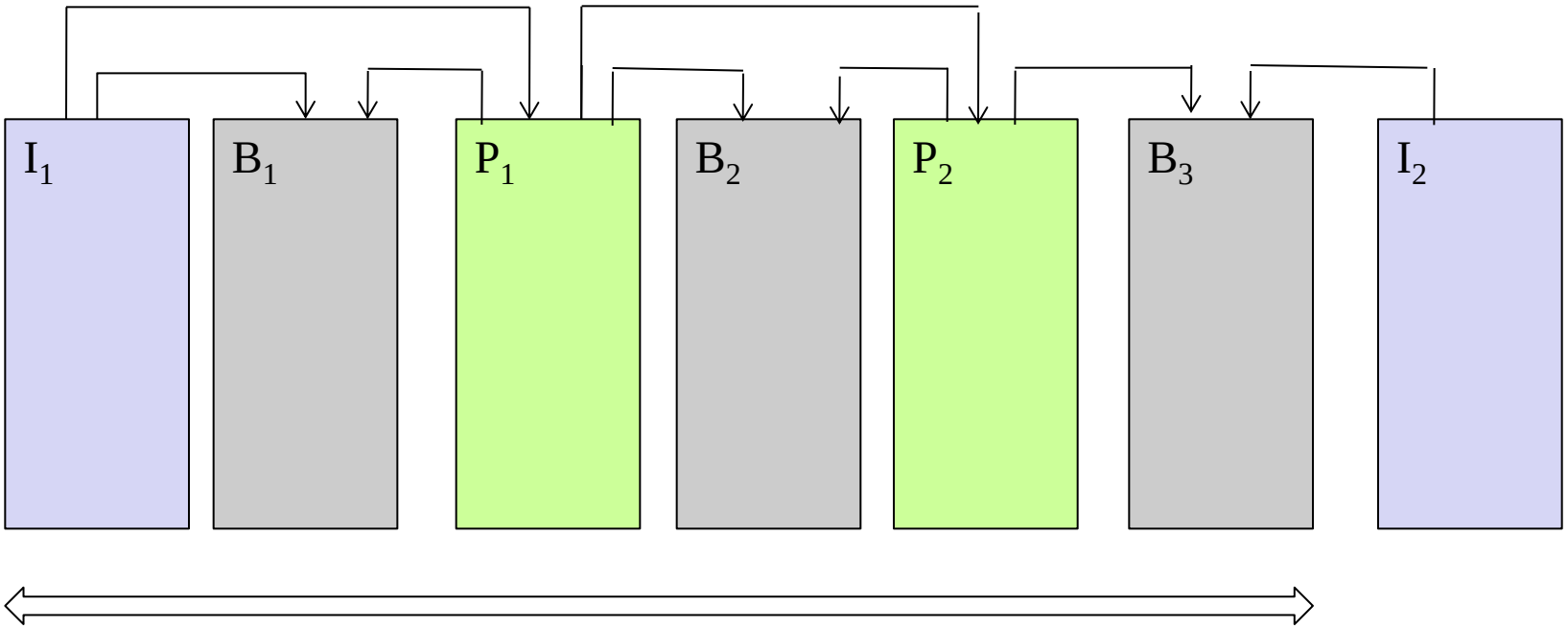
→ u.a. 3 Frame-Typen :

- **I-Frames** ("***I****ntraframes*") : vollkommen unabhängig von anderen Frames codiert.
- **P-Frames** ("***P****redicted Frames*") : durch Bewegungsvorhersage und Differenzbildung aus den vorhergehenden Frames erzeugt.
- **B-Frames** ("***B****idirectionally Predicted Frames* ") : durch Bewegungs-vorhersage aus dem vorhergehenden und dem nachfolgenden I- oder P-Frame erzeugt.

Nota bene :

- I-Frames i.d.R. am größten und B-Frames am kleinsten.
- Starke Fehlerfortpflanzung bei verfälschten/verlorenen I-Frames, geringerer „Schaden" bei P-Frames, und B-Frame „schädigt" nur eigene Decodierung.
- Prädiktive und interpolierende Codierung auch anwendbar für kleinere Einheiten wie Makroblöcke (in Standards auch genutzt).

Gegenseitige Abhängigkeiten und Fehlerfortpflanzung bei MPEG



Sinnvolle Übertragungsreihenfolge:

– $I_1, P_1, B_1, P_2, B_2, I_2, B_3$.

Zulässige GOP-Muster: GOP(n,m) mit n=GOP-Size und m=Abstand zwischen benachbarten I- und P-Frames, $m \leq n$, $n \bmod m = 0$: z.B. *GOP(9,3)* ≡ I, B, B, P, B, B, P, B, B, I, …

Lernmodul „Videokommunikation“

Wesentliche Merkmale der Videokomprimierung gemäß MPEG können dem bei TKRN im Rahmen des eLearning-Projektes *TeleMuM* entwickelten **Lernmodul „Videokommunikation“** entnommen werden (insbes. Kap. 5 „Videocodierung/ -komprimierung : MPEG als ausgewählter Standard“)

→ evtl. Zugriff über die TKRN-Webseiten und Nutzung dieses Lernmoduls (insbes. des in das Lernmodul integrierten Tools zur Qualitätsanalyse bei Audio-/Videokommunikation in Echtzeit) im praktischen Teil der DKR-Übungen.

Weitere Literatur :

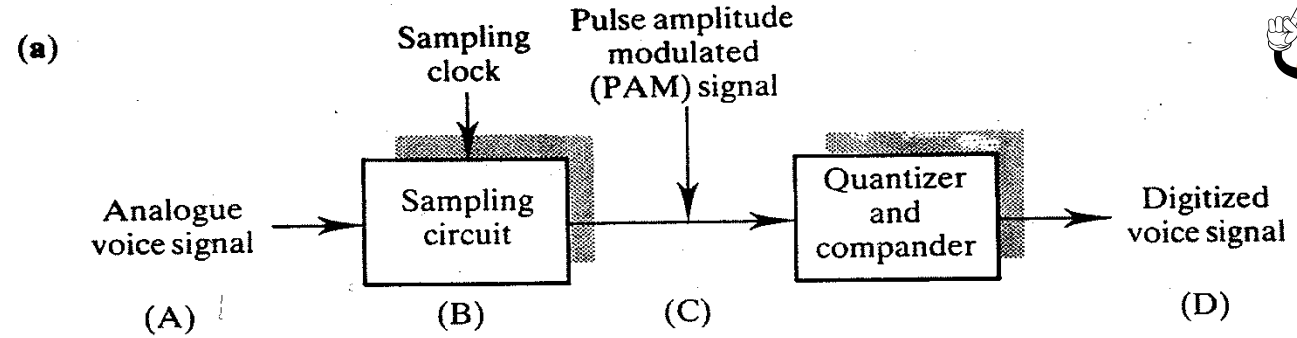
P.A. Henning „Taschenbuch Multimedia“
(Fachbuchverlag Leipzig, 2. Auflage, 2001).

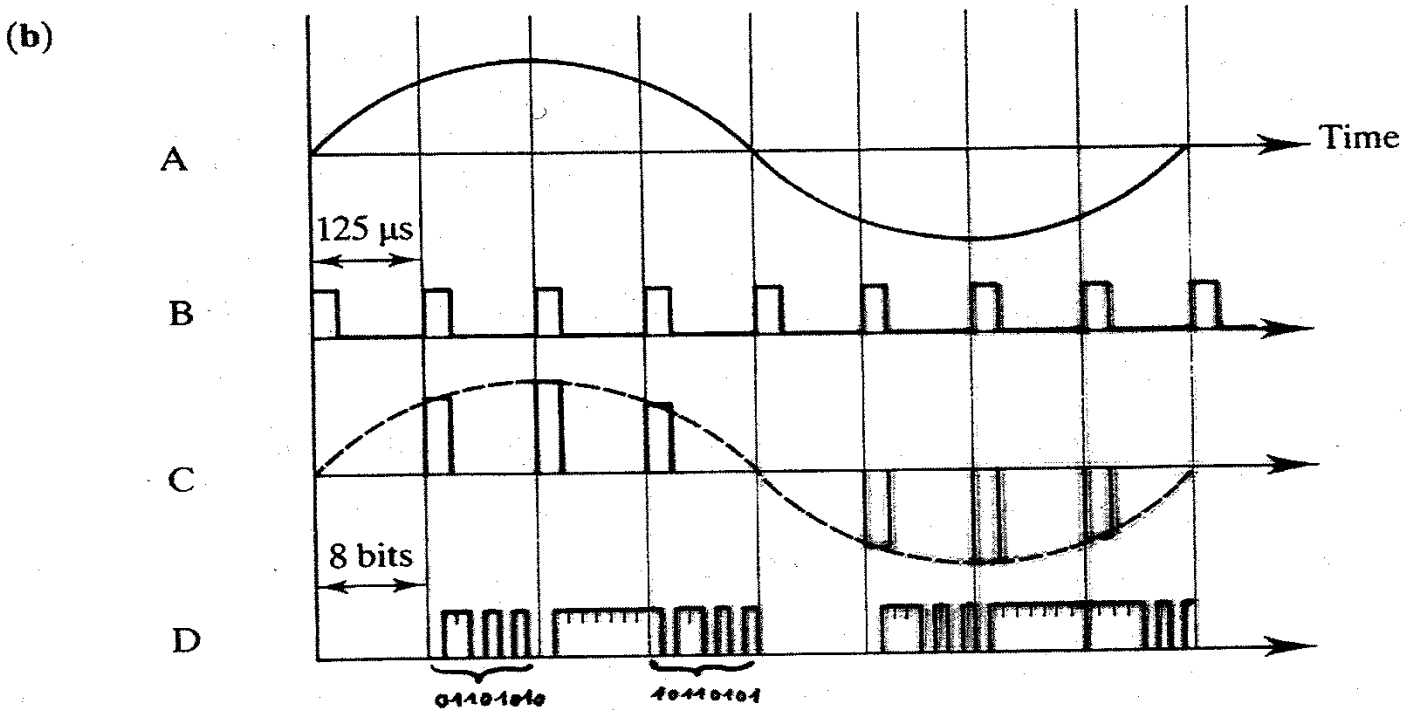
2.5 Sprach- und Audiocodierung

Anforderungen an Sprach- und Audioübertragung

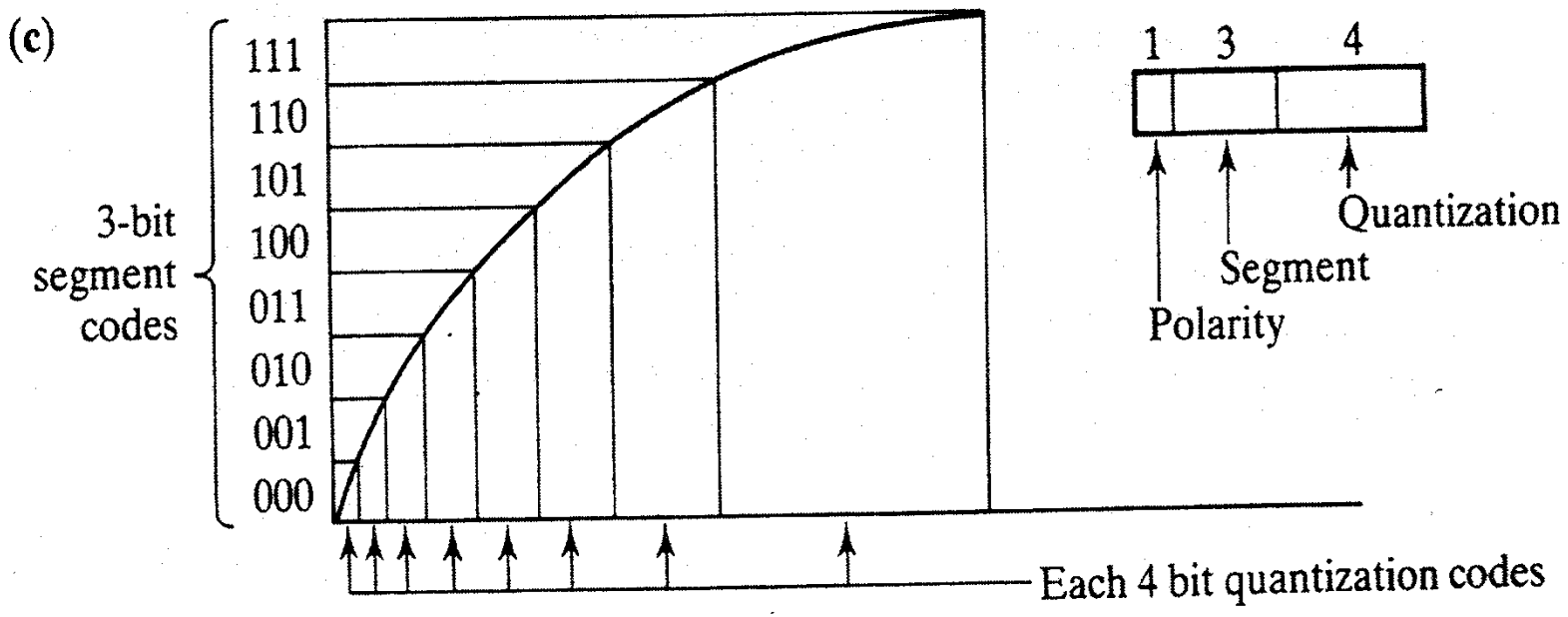
Kriterium	**Sprache**	**Audio**
Qualitätsanforderung	Fernsprechqualität	CD-Qualität
Höchste berücksichtigte Frequenzen	ca. 4.000 Hz	ca. 20.000 Hz
Primär benötigtes Rechner-/Netzbetriebsmittel	Übertragungsressourcen	(Peripher-) Speicherressourcen
Typisches Ausgangs-Signal (vgl. primäres Signal in Kap. 3)	menschl. Sprache	Musik (zumeist Aufnahmen in hoher Qualität)
Typischerweise erreichte Wiedergabe-/Übertragungsqualität	mittel (bei leitungsgebundenen Netzen); eher schwach bei Mobilnetzen	im allg. sehr hoch

PCM für die digitale Sprachübertragung





PCM: Fortsetzung (Codierung)



Resultierende Datenrate bei digitalisierter Sprachübertragung

- **Abtastfrequenz** in [Hz], d.h. Anzahl der Abtastungen pro sec:
  8 kHz, wegen *Abtasttheorem von Shannon*, vgl. Kap. 3
  → 2 • Maximalfrequenz (d.h. 2 • 4.000 Hz)

- **Anzahl benutzter bit pro Abtastwert** :
  7 Bit (USA) bzw. 8 Bit (Europa)

Ergo:
Resultierende Datenrate:

- USA: 7 Bit • 8.000 Hz = 56.000 [Bit • Hz] = 56.000 [Bit / sec] = 56.000 [bps] = **56 [kbps]**
- Europa: 8 Bit • 8.000 Hz = 64.000 [Bit • Hz] = 64.000 [Bit / sec] = 64.000 [bps] = **64 [kbps]** , vgl. Datenrate des sog. „B-Kanals“ im (Schmalband-) ISDN

Da die Datenrate bei Sprachübertragung während der Sprechphasen nicht schwankt, spricht man hier auch von „***Constant Bit Rate***“(CBR)-Verkehr (vgl. spätere Kap. über Echtzeitkommunikation).

ABER bitte vorsichtig (!) mit CBR-Begriff, denn CBR-Verkehr mit z.B. 10 [kBit/sec] kann erreicht werden durch :
• Pakete à 100 [Bit], erzeugt nach jeweils konstant 10 [msec], *oder durch*
• Pakete à 1 [kBit], erzeugt nach jeweils konstant 100 [msec] , u.v.a.m. !!!