

Willkommen bei Kommunikations- und Netztechnik!

Von Kupferkabel, Glasfaser und Mikrowelle über Telefon, Ethernet und TCP zu E-Mail, Webserver und REST.



Heute: **Verlässliche Übertragungen über fehlerbehafteten Kanal, Rechner über „Kabel“ verbunden.**

Wiederholung Bitübertragung II

- Passband: FSK (Frequenz), ASK (Amplitude), PSK (Phase)
- Bandbreitenbedarf, Taktrückgewinnung, Gleichstromfreiheit
- Multiplexing
 - FDM (Frequenz), TDM (Zeit), CDMA (Code)
- CDMA
 - Walshcodes zur Encodierung mehrerer Signale in ein Signal

Weitere Fragen?

Übungsaufgaben

- In Gruppen bis 3 Personen
- Gruppenaufgaben einfach mit Name, da die Matrikelnummer nicht als öffentlich gilt

Ablauf heute

- Grundlagen
 - Dienste der Sicherungsschicht
 - Rahmen bilden
- Konkret
 - Fehlerkorrektur
 - Fehlererkennung
 - Grundlegende Protokolle
 - Schiebefensterprotokolle

Pause um etwa 14:15

Wiederholung Bitübertragung I

- Nyquist Formel:

$$\text{maximum data rate} = 2B \log_2 V$$

- Shannon Formel:
 - maximum number of bits/sec = $B \log_2(1 + S/N)$
 - SNR nach $\frac{S}{N}$: $\text{SNR} = 10 \log_{10}(\frac{S}{N})$
- duplex, simplex
- Unterteilung Übertragungsmedien, Beispiele für Kategorien
 - Kabelgebunden
 - Kupfer
 - LWL
 - Kabellos
- Modulation

Warum Sicherungsschicht?

- Bits zwischen verbundene Geräten übertragen
- Fehlerrate **genug reduzieren** für die Vermittlungsschicht
- Fehler frühzeitig abfangen: Optimierung für geringere Latenz

Ziele heute I

- Sie wissen, dass die Sicherungsschicht als Dienst Pakete aus Bits überträgt
- Sie wissen, dass die Sicherungsschicht als Protokoll Pakete in Rahmen verpackt und übermittelt.
- Sie verstehen, dass je nach Medium unterschiedliche Komplexität sinnvoll ist und können die dabei notwendigen Abwägungen erklären.
- Sie können mit einer Kurzbeschreibung aus (11.7) Hamming-Codierten Daten die korrigierten Nachrichtenbits extrahieren.
- Sie wissen, dass der Hamming-Abstand angibt, ab wievielen Bitfehlern Fehler unentdeckt bleiben können.

Ziele heute II

- Sie verstehen ein 1-Bit Schiebefensterprotokoll und erkennen verschiedene Optimierungen
- Sie haben einen Fehlerkorrekturcode programmiert (und wissen, dass sie es wieder tun könnten).

Versuch 1 (von 3): Auswirkungen von Fehlern

Erste Reihe

Sie haben ein Blatt mit 1-en und 0-en. Nehmen Sie jeweils die entsprechenden kleinen Zettel (Karten), drehen Sie sich nach hinten, schließen Sie die Augen und werfen Sie die Zettel einen nach dem anderen auf den Tisch der Person hinter Ihnen.

Mittlere Reihen

Nehmen Sie die Zettel. Wenn einer runterfällt, ersetzen Sie ihn durch einen zufälligen. Dann drehen Sie sich nach hinten, schließen Sie die Augen und werfen Sie die Zettel.

Letzte Reihe

Nehmen Sie die Zettel und decodieren Sie die Bitfolge mit der ASCII-Tabelle auf der nächsten Seite.

Etwas kalibrieren: 1-2 Fehler von erster zu letzter Reihe. Die Anzahl Fehler einer Satelliten-Übertragung in einem 100kiB Bild.

Corona-Pandemie-Zeit

- Sie wissen, dass die Sicherungsschicht als Dienst Pakete aus Bits überträgt
- Sie wissen, dass die Sicherungsschicht als Protokoll Pakete in Rahmen verpackt und übermittelt.
- Sie verstehen, dass je nach Medium unterschiedliche Komplexität sinnvoll ist und können die dabei notwendigen Abwägungen erklären.
- Sie können mit einer Kurzbeschreibung aus (11.7) Hamming-Codierten Daten die korrigierten Nachrichtenbits extrahieren.
- Sie wissen, dass der Hamming-Abstand angibt, ab wievielen Bitfehlern Fehler unentdeckt bleiben können.

ASCII-Tabelle

	000	001	010	011	100	101	110	111
0000	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL
0001	BS	HT	LF	VT	FF	CR	SO	SI
0010	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB
0011	CAN	EM	SUB	ESC	FS	RS	US	
0100	SP	!	"	#	\$	%	&	'
0101	()	*	+	,	-	.	/
0110	0	1	2	3	4	5	6	7
0111	8	9	:	;	<	=	>	?
1000	@	A	B	C	D	E	F	G
1001	H	I	J	K	L	M	N	O
1010	P	Q	R	S	T	U	V	W
1011	X	Y	Z	[\]	^	_
1100	`	a	b	c	d	e	f	g
1101	h	i	j	k	l	m	n	o
1110	p	q	r	s	t	u	v	w
1111	x	y	z	{		}	~	DEL

ASCII-Steuerzeichen (zum Nachschlagen)

NUL	Null	DLE	Data Link Escape
SOH	Start of Heading	DC1	Device Control 1 (XON)
STX	Start of Text	DC2	Device Control 2
ETX	End of Text	DC3	Device Control 3 (KOFF)
EOT	End Of Transmission (EOF)	DC4	Device Control 4
ENQ	Enquiry (who are you?)	NAK	Negative ACK
ACK	Acknowledgement	SYN	Synchronous Idle
BEL	Bell: echo -as "la":	ETB	End of Transmission Block
BS	Backspace	CAN	Cancel
HT	Horizontal Tab	EM	End of Medium
LF	Linefeed aka -as "la":	SUB	Substitute
VT	Vertical Tab	ESC	Escape
FF	Form Feed	FS	File Separator
CR	Carriage Return	GS	Group Separator
SO	Shift Out	RS	Record Separator
SI	Shift In	US	Unit Separator

Hochhalten und zeigen statt werfen!

Dienste der Sicherungsschicht

Dienst-Arten

- Welche Arten von Diensten sind sinnvoll?
- Woran erinnern sie sich noch aus der Übersicht?

Je 6 Leute zusammen, 5 Minuten, dann sammeln wir.

Beispiel: Verbindungsorientiert, bytestrom, unbestätigt.

Stärken und Schwächen

- Welche Charakteristiken haben diese Arten von Diensten?

Rahmen bilden

- Längenbyte
- Flagbyte
- Flagbits
- Codierungsverletzung
- Kombiniert

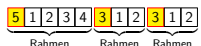
Längenbyte



(1)

Brauchen **canonical S-expressions** einen fehlerfreien Kanal?
(4: this22: Canonical S-expression3: has1: 55: atoms)

Längenbyte



(1)

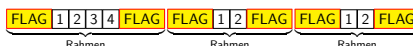
Brauchen **canonical S-expressions** einen fehlerfreien Kanal?
(4: this22: Canonical S-expression3: has1: 55: atoms)

Byteverlust und Desynchronisierung



(2)

Flagbyte mit Bytestopfen



(3)

Flagbyte mit Bytestopfen



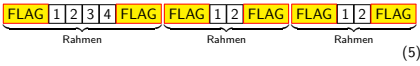
(3)

Byteverlust



(4)

Flagbyte mit Bytestopfen



(5)

Flagbyteverlust



(6)

Codierungsverletzung (Abstraktionsbruch)

Codierung auf Bitübertragungsebene verwendet nur ein Subset der Zeichen, z.B. für Taktrückgewinnung.

Beispiel: 4B/5B

Signalisierung über verbotene Zeichen; unproblematisch, da selten.

Zusammenfassung der abstrakten Grundlagen

Dienste

- Verbindungslos, unbestätigt
- Verbindungslos, bestätigt
- Verbindung, bestätigt, geordnet

Rahmen

- Längenfeld
- Flagbyte/-bit
- Codierungsverletzung

Bestätigung als Optimierung:
Früher korrigieren.

Arten von Fehlern

Welche Arten von Fehlern gibt es? Welche Struktur haben die Veränderungen?

- Verlorene Einzelbits
- Einzelne Bitflips
- Burst-Fehler (mehrere)

Hamming-Abstand

Codewörter

- 000000000
- 000001111
- 111110000
- 111111111

Korrigierbar:

- 000000000
- 000000011
- 000001111

Erkennbar:

- 000000000
- 000001111
- 000001111

Hamming-Abstand:

- 000000000
- 000001111
- 000011111

p=95% korrekte Bits:

- HA 1: 40% beschädigt (1 / 3)
- HA 2: 8.6% beschädigt (1 / 12)
- HA 3: 1.2% beschädigt (1 / 87)
- Ha 5: 0.0096% beschädigt (1 / 10k)

Summe binom(10, p, 11-HA)

→ <https://de.wikipedia.org/wiki/Binomialverteilung>

Prüfbits Minimum, Einzelbitfehler

n gesamtbits, m Nachrichtenbits und r Prüfbits. $n = m + r$

m bits → 2^m mögliche Nachrichten → Pro Nachricht zusätzlich n verbotene Codewörter mit Abstand 1 nötig.

Verbotene Codewörter: Für jede erlaubte Nachricht jedes Bit flippen (⇒ n flips). ⇒ n + 1 Bitmuster für jede erlaubte Nachricht.

$$(n + 1) \cdot 2^m \leq 2^n = 2^{m+r} \quad (10)$$

$$(m + r + 1) \leq 2^r \quad (11)$$

$$m \leq 2^r - r - 1 \quad (12)$$

Flagbytes: Escaping nötig



(7)



(8)

Wie in Java-Strings:

String withQuote = "ab\"c";

Codierungsverletzung (Abstraktionsbruch)

Codierung auf Bitübertragungsebene verwendet nur ein Subset der Zeichen, z.B. für Taktrückgewinnung.

Beispiel: 4B/5B

Signalisierung über verbotene Zeichen; unproblematisch, da selten.

Aber Abstraktionsbruch:

Wechsel der Codierung erfordert Wechsel der Rahmenbildung.

Fehlerkorrektur oder -erkennung

Arten von Fehlern

- Erkennung vs. Korrektur
- Hamming-Abstand
- Fehlerkorrektur
- Fehlererkennung

Erkennung vs. Korrektur

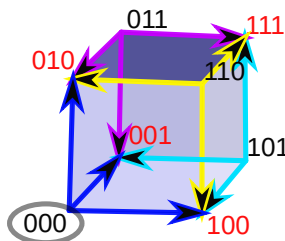
Effizienzfrage:

- Immer Zusätzliche Bandbreite
- Teure Neu-Übertragungen (Round-Trips) im Fehlerfall

Fehlerkorrektur, Methoden

- Vollständige Suche
- Hamming-Code
- Binäre Faltungscodes
- Reed-Solomon-Codes
- LDPC-Codes (Low-Density Parity Check)

Verbotene Wörter, 1 bit erkennen



Flagbits mit Bitstopfen

FLAG 01111110
Escape x011111x → x0111110x
Unescape x0111110x → x011111x



(9)

Kombiniert

- Beispiel: Präambel + Längenfeld in WiFi (72 Bit Präambel!)

Frage: Wofür ist die Länge noch nützlich?

Arten von Fehlern

Welche Arten von Fehlern gibt es? Welche Struktur haben die Veränderungen?

Hamming-Abstand

Codewörter

- 000000000
- 000001111
- 111110000
- 111111111

p=95% korrekte Bits:

- HA 1: 40% beschädigt (1 / 3)
- HA 2: 8.6% beschädigt (1 / 12)
- HA 3: 1.2% beschädigt (1 / 87)
- Ha 5: 0.0096% beschädigt (1 / 10k)

Summe binom(10, p, 11-HA)

→ <https://de.wikipedia.org/wiki/Binomialverteilung>

Vollständige Suche

- Komplexität: O(N) (alle möglichen Codewörter vergleichen)

⇒ zu teuer.

⇒ Hinweise, wo der Fehler korrigiert werden kann.

Hamming-Code

Prüfbits gemappt auf Bits

```
Reine Daten: 10000001 (14)
```

```
Mit Prüfbits: 0010000010001 (15)
```

Prüfbit in Position 4, genutzt für Bit 5, 6, 7.

```
0010000010001 (16)
```

```
0010101001001 (17)
```

(11,7) Hamming: Hilfsfunktionen

```
define : mod2sum . numbers
. "Modulo-2 sum, i.e. for even parity"
## : tests : test-equal 1 : mod2sum 1 0 1 1 0
modulo (apply + numbers) 2

define H mod2sum ;; for brevity

define : flip numbers index
. "flip the bit-number (0--1 or 1--0) at the index."
## : tests : test-equal '(1 0 1) : flip '(0 0 1) 0
append
take numbers index
list : mod2sum 1 : list-ref numbers index
drop numbers (index + 1)
```

Hacks, um auf Zahlen zu arbeiten. Sauberer: Bitvector.

(11,7) Hamming: Encode

```
Header
Body
define : 11/7-encode numbers
match numbers
list
: 13 15 16 17 19 110 111
H 13 15 17 19 111 ;; bit 1
H 13 16 17 110 111 ;; bit 2
. 13 ;; bit 3
. '(0 0 1 0 0 0 0 1 0 0 1)
H 15 16 17 ;; bit 4
. 15 16 17 ;; bit 5, 6, 7
H 19 110 111 ;; bit 8
. 19 110 111 ;; bit 9, 10, 11
```

(11,7) Hamming: Decode

```
define : 11/7-decode numbers
define broken-bit
match numbers
: h1 h2 h3 h4 h5 h6 h7 h8 h9 i10 i11
+
* 1 : h h1 h2 h3 h4 h5 h6 h7 h8 h9
* 2 : h h2 h3 h4 h5 h6 h7 h8 h9 i10
* 4 : h h4 h5 h6 h7 h8 h9 i10 i11
* 8 : h h8 h9 i10 i11
define fixed
if : zero? broken-bit
. numbers
flip numbers {broken-bit - 1}
match fixed
: h1 h2 h3 h4 h5 h6 h7 h8 h9 i10 i11
list 13 15 16 17 19 110 111
```

Zum Nachhören

Eine schöne Beschreibung von Hamming-Codes mit einem 15/11 Beispiel finden Sie in den 3Blue1Brown Videos

- Hamming codes, h.w to ov. rco. e.n.ise (https://www.youtube.com/watch?v=X8jsijh11IA)
Hamming codes part 2, the elegance of it all (https://www.youtube.com/watch?v=b3NxrZ0u_CE)

Weitere Codes

- Binäre Faltungscodes: Ursprünglich für die Voyager-Missionen der NASA, heute in WLAN. Erzeugt aus jedem Bit und internem Zustand zwei Bits, kann Unsicherheiten einbeziehen.
Reed-Solomon: Polynome mit zusätzlichen Stützpunkten.
LDPC-Codes (Low-Density Parity Check): Dünn besetzte Matrizen, iterativ

Fehlererkennung, Methoden

- Parität
Prüfsummen
Cyclic Redundancy Check (CRC)

Corona-Pandemie-Zeit

Hochhalten und zeigen statt werfen!

Parität

Zusatzbits: Anzahl der 1-Bits gerade (oder ungerade, je nach Methode).
00101011(gerade)
10101010(gerade)

Versuch 2 (von 3): Fehlererkennung mit Hamming

Erste Reihe: Sie haben ein Blatt mit 1-en und 0-en. Berechnen Sie die Hamming-Encodierte Bitfolge...
Mittlere Reihen: Nehmen Sie die Zettel. Wenn einer runterfällt, ersetzen Sie ihn durch einen zufälligen...
Letzte Reihe: Nehmen Sie die Zettel, Hamming-Dekodieren Sie die Bitfolge und berechnen Sie das entsprechende Zeichen mit der ASCII-Tabelle.

Parity mit Versatz

Gegen Burstfehler: Versatz (Interleaving) => Parität Byteübergreifend.
Original vs. Mit Burstfehlern (interleaved).
Burst: auch kurze Störung paralleler Übertragung.

Cyclic Redundancy Check (CRC)

Die Bitfolge geteilt durch Generatorpolynom (das CRC-Polynom) mod(2) (Polynomdivision); behält Rest. Rest = CRC-Wert -> anhängen.
Bsp: 110101 -> 1x^5 + 1x^4 + 0x^3 + 1x^2 + 0x^1 + 1

Verifizieren: Daten + CRC durch CRC-Polynom teilen. Rest -> Fehler!

Zusammenfassung

- Hamming Abstand: Ab wie vielen Fehlern nicht mehr erkannt
Erkennbare Bits: Abstand - 1
Korrigierbare Bits: 0.5 * Erkennung
m <= 2^r - r - 1

Fehlerkorrektur allgemeiner: Fountain codes

- Fountain Codes: https://en.wikipedia.org/wiki/Fountain_code
Patente auf Raptor-Codes sind freigegeben.
Wiederherstellung von Daten mit ausreichendem Subset.
A fountain code is optimal if the original k source symbols can be recovered from any k encoding symbols

In Freenet / Hyphanet und gunnet wird FEC (Forward Error Correction) genutzt, bei der beliebige 50% der Daten gebraucht werden.

Prüfsummen

Verallgemeinerte Parität, Internetprüfsumme in IP im Einerkomplement: Überlauf wird auf niedrigstes Bit addiert => 100 + 100 = 001

Fehlerkorrektur allgemeiner: PAR2

- PAR2: http://archive.sourceforge.net/
Easily verify and regenerate single missing parts out of a set of data-blocks

Aus den Usenet-Zeiten. Heute meist nicht mehr nötig, weil Fehler schon bei Übertragung abgefangen werden.
Könnte ähnliches bei Latenz-Jitter helfen? — bei den langsamsten Antworten oder p90, die alles blockieren?

Grundlagen der Protokolle

- Trennung von höheren Schichten: Sicherungsschicht läuft teils auf der Netzwerkkarte
Rahmen liefert Zusatzinformationen (z.B. Prüfsumme)

Simplexprotokoll

```
void Sender() {
Frame toSend;
Packet buffer;
while (true) {
...
}
}
void Receiver() {
Frame received;
while (true) {
wait_for_event();
received = from_physical_layer();
to_send_info = buffer;
to_physical_layer(toSend);
to_network_layer(received.info);
}
}
```

