Datensicherungsschicht

L3..L7	höhere Schichten
L2.2	Datensicherungsschicht - Warteschlange - segmentation/reassembly - Adressierung
L2.1	- Fehlererkennen - Fehlerbeheben - Prioritäten - Überlaststeuerung
L2.0	MAC Vielfachzugriff
L1	physikalische Schicht

SLIP: Serial Line Interface Protocol

TCP/IP-Meldungen über serielle Leitungen (Modemverbindungen)

219	IP-Meldung	192

SLIP unterstützt lediglich die Rahmensynchronisation

Transparenz durch Charakter-stuffing

219 oder 192 innerhalb einer Meldung vor, wird

192 wird durch 221 und das Zeichen 219 durch 220 erweitert (gestopft)

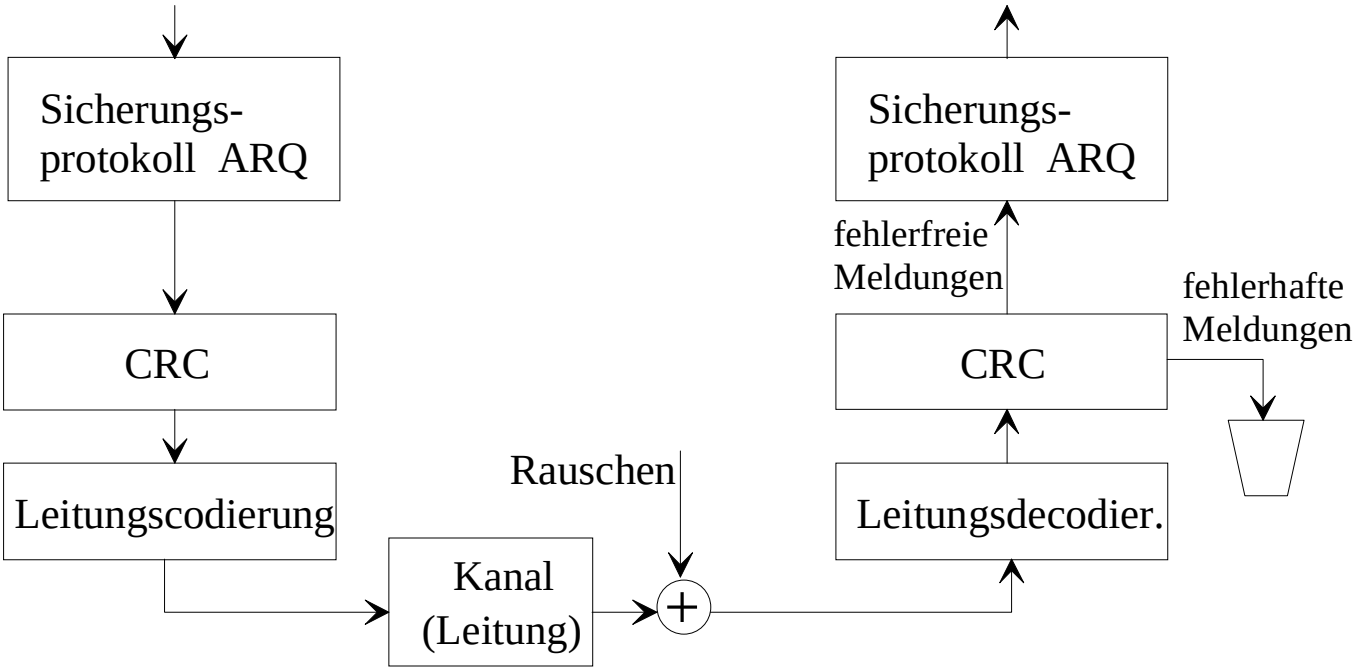
Fehlererkennung

Block Check Character BCC

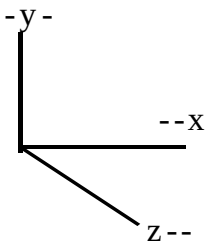
0	1	0	0	1	0	0	1	Daten Byte 1
0	1	1	0	1	1	0	1	Daten Byte 2
1	1	0	0	0	1	1	0	Daten Byte 3
1	1	0	1	1	0	0	1	Daten Byte 4
1	0	0	0	1	0	0	0	Daten Byte 5
1	**0**	**1**	**1**	**0**	**0**	**1**	**1**	**Block Check Character**

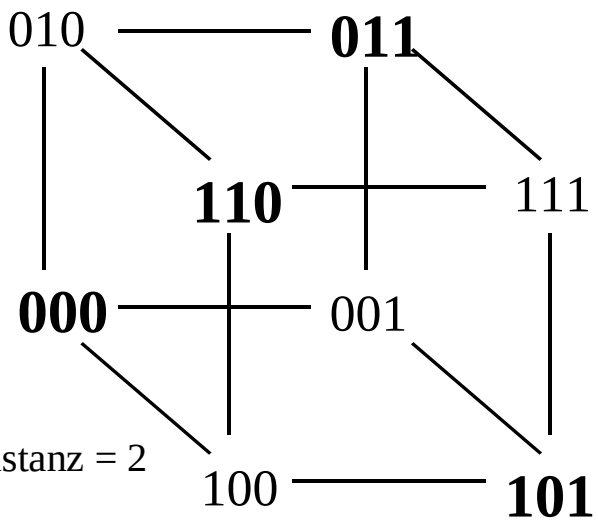
Data	BCC

Cyclic Redundancy Check: CRC

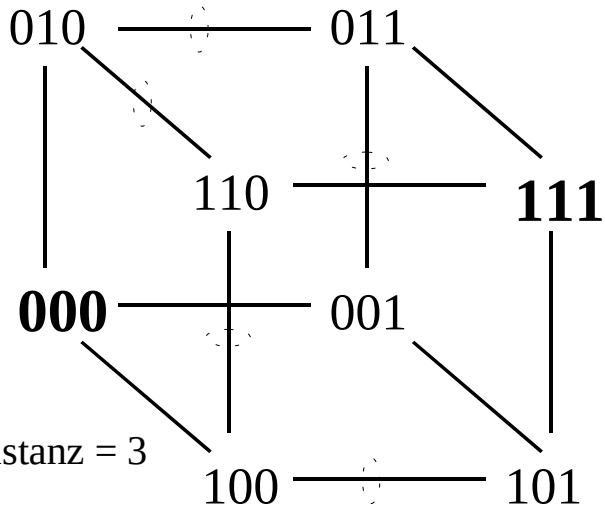


Kanalcodierung





Distanz = 2



Distanz = 3

CRC = Cyclic Redundancy Check

= FCS = Frame Check Sequence

Hamming Code:

Generatorpolynom g(x) ist ein primitives Polynom vom Grad k

⇨ maximale Blocklänge $m = 2^k-1$ (k=16 ➔ 65535 Bits = 8.191 Bytes)

⇨ Distanz d = 3 ➔ 2 Fehler erkennen oder 1 Fehler korrigieren

Ethernet: Hamming Code mit dem Generatorpolynom g(x) =

$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

$m = 2^{32}-1$ ➔ max. 4 294 967 295 Bits werden geschützt.

Fehlerbündel der Länge $l \leq 32$ kann erkannt werden.

Abramson-Code

$g_1(x)$ sei das Generatorpolynom vom Grad k eines Hamming-Codes;

⇨ maximale Blocklänge $m = 2^{k1}-1$

⇨ Distanz d = 3

⇨ $g_2(x) = g_1(x) \cdot (x+1)$ ist Abramson-Code

Eigenschaften des Abramson-Code

- maximale Blocklänge $m = 2^{k1}-1$ wie Hamming
- Distanz d = **4** ➔ 3 Fehler erkennen oder 1 Fehler korrigieren

Beispiele für Abramson-Code

CRC-16: $g(x) = (x^{15} + x + 1) \cdot (x+1) = x^{16} + x^{15} + x^2 + 1$

CCITT-16: $g(x) = (x^{15} + x^{14} + x^{13} + x^{12} + x^4 + x^3 + x^2 + x + 1) \cdot (x+1) = x^{16} + x^{12} + x^5 + 1$

Einsatz von CCITT-16: X.25, ISDN (LAP-B, LAP-D), SS7

CRC Codierung :

Daten sind ein Polynom U(x) (eine binäre Zahl, die aus n Bits besteht)

U(x) soll mit k Prüfbits geschützt werden (z.B. k=16 oder 32).

Das Generatorpolynom ist $G(x) = x^k + ... + 1$.

Codierer:

1. An U(x) werden k Nullen angehängt: $U'(x) := U(x) \cdot 2^k$. $U'(x) \cdot$ besteht aus m = n+k Bits.

2. U'(x) / G(x) := E(x) + R(x)/ G(x) Ergebnis E(x); Restpolynom R(x)

3. U''(x) := U'(x) + R(x)

4. Das Polynom U''(x) wird übertragen.

Decodierer:

5. Input: V(x) falls keine Fehler aufgetreten sind, gilt V(x) = U''(x)

6. V(x) / G(x) = V'(x) + W(x) / G(x)
Geht die Division ohne Rest auf (W(x) = 0), dann war die Übertragung fehlerfrei.

CRC-Beweis:

Übertragen wird U''(x) := U'(x) + R(x) = U'(x) - R(x),
Bei Modulo-2 Operationen ist die Addition identisch mit der Subtraktion.

Decodierer: V(x) / G(x) = U''(x) / G(x) Übertragung fehlerfrei: V(x) = U''(x)

U''(x) / G(x) = [U'(x) + R(x)] / G(x) U''(x) := U'(x) + R(x)

[U'(x) + R(x)] / G(x) = [U'(x) - R(x)] / G(x) Modulo-2: Addition = Subtraktion

U'(x) / G(x) - R(x) / G(x) = E(x) + R(x)/ G(x) - R(x) / G(x) = E(x).

Tritt Übertragungsfehler auf, gilt $V(x) \neq U''(x)$.
Fehler durch ein Fehlerpolynom F(x) darstellen: V(x) = U''(x) + F(x).

V(x) / G(x) = [U''(x) + F(x)] / G(x) = [U'(x) - R(x) + F(x)] / G(x) =

= U'(x) / G(x) - R(x) / G(x) + F(x) / G(x) =

= E(x) + R(x)/ G(x) - R(x) / G(x) + F(x) / G(x) = E(x) + **F(x) / G(x)**

Nur wenn F(x) ein Vielfaches von G(x) ist, wird der Fehler nicht erkannt. Da die Polynome 4 Einsen besitzen, müssen mindestens 4 Fehler auftreten, damit diese unerkannt bleiben.

CRC-Beispiel: Generatorpolynom ist die Primzahl 19 : $\mathbf{x^4 + x + 1 = 1\,0\,0\,1\,1}$.

Wieviele Fehler können mit dem Generatorpolynom $x^4 + x + 1$ erkannt werden ?

Da es sich nicht um einen Abramson-Code handelt können maximal 2 Fehler entdeckt werden.

Wieviele Informationsbits können damit geschützt werden ?

Da es sich nicht um einen Abramson-Code handelt und k=4 ist, gilt
$m = n + k = 2^k - 1 = 2^4 - 1 =$ **15** Bit. $n = m - k = 15 - 4 =$ **11**.

Es können maximal 11 Bit geschützt werden.

Es sollen die Daten U(x) = 1 0 1 0 0 1 1 0 1 0 durch den CRC-Code mit dem Generatorpolynom $x^4 + x + 1$ geschützt werden. Welcher Datenstrom wird übertragen ?

1. U(x) = 1 0 1 0 0 1 1 0 1 0
2. $U'(x) = U(x) * 2^k = U(x) * 2^4$ = 1 0 1 0 0 1 1 0 1 0 * 1 0 0 0 0 = 1 0 1 0 0 1 1 0 1 0 0 0 0 0
3. U'(x) / G(x) = `10100110100000 / 10011 = 1011101001`

```
        10011          |
          11111        |
          10011        |   Modulo-2   0 + 0 = 0
           11000       |   Addition   0 + 1 = 1
           10011       |   1 + 1 = 0
            10111      |
            10011      |
              10000    |
              10011    |
                 11000
                 10011
Rest R(x)         1011
```

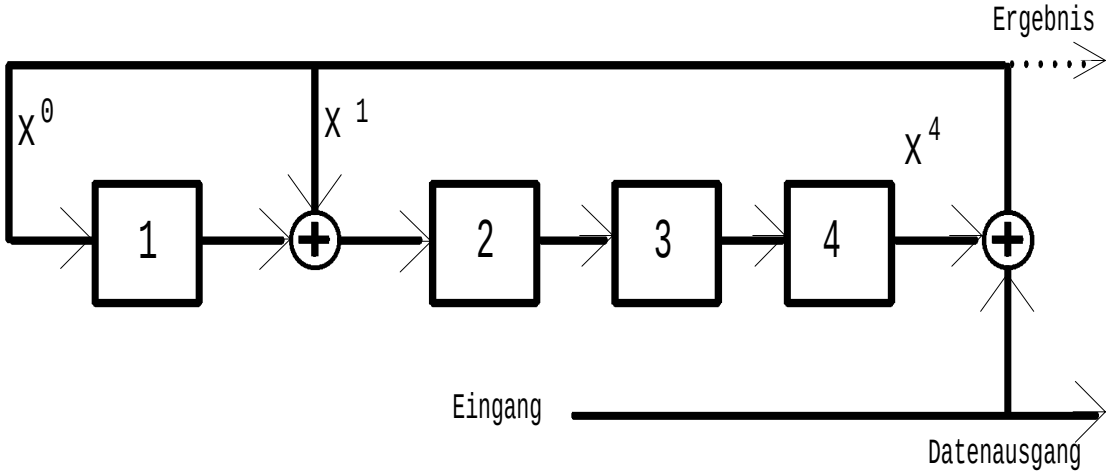
Übertragen wird: 10100110101**011**

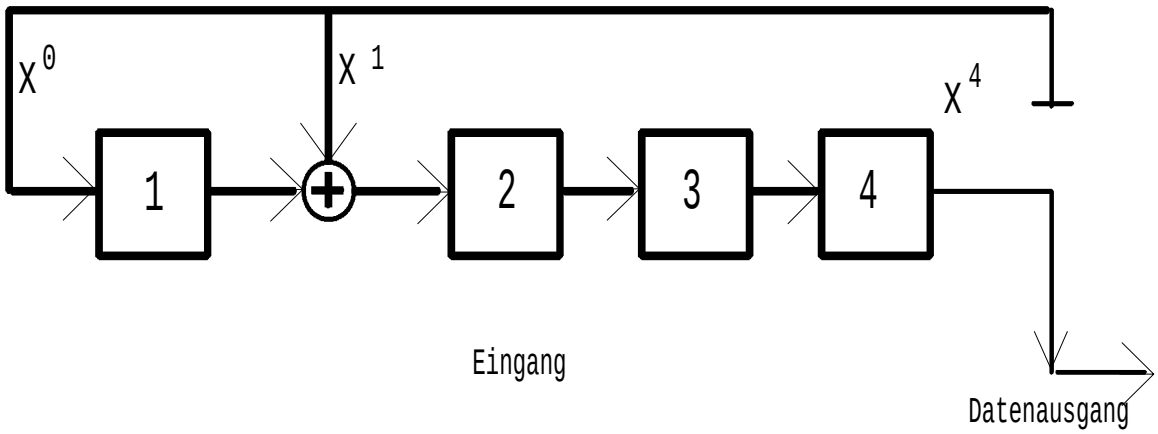
Decodierer: 101001101010 11 / 10011 = 1011101001

```
10011          |
  11111        |
  10011        |
   11000       |
   10011       |
    10111      |
    10011      |
      10001    |
      10011    |
         10011
         10011
             0
```

Rest R(x) 0 Der Rest ist 0, d.h. es liegt kein Übertragungsfehler vor.

Realisierung: durch SW oder durch HW (rückgekoppelte Schieberegister)





BDC-Protokoll

BDC = **B**asic mode control for procedures for **D**ata **C**ommunication systems

Basic Mode

BSC Binary Synchronous Control

Bisync

ISO-Standard und DIN 66019.

V.24-Schnittstelle zeichenorientiert 'In-band'-Signalisierung CCITT Code Nr. 5

Fehlerprüfung: Blockprüfung = Zeilenweise Parity; Spalte Block Check Character BCC

Steuerzeichen aus dem CCITT-Code Nr. 5 :

TC Transmission Control oder Übertragungssteuerzeichen:

SYN synchronisation SOH, start of header STX, start of text
ETB, end of transmission block ETX, end of text
ACK, NAK, DLE, data link escape ENQ, enquiry
EOT, end of transmission

Formatsteuerzeichen:

BS Backspace HT Horicontal Tab
CR Carriage Return LF Line Feed
FF Form Feed VT Vertical Tab

Gerätesteuerzeichen:

DC1, DC2, DC3, DC4 vom Endgerät abhängig

Informationssteuerzeichen: Zur Strukturierung der Information in Datensätzen

FS File Separator
GS Group Separator
RS Record Separator
US Unit Separator

Sonstige Steuerzeichen:

NUL Füllzeichen, wird im Empfänger überlesen
BEL Bell = akustisches Signal
CAN Cancel = Zeichenfolge ist zu ignorieren
EM Ende des Datenträgers
SUB Substitude = ersetze verhergehendes Zeichen durch nachfolgendes
SP Space = Leerzeichen
DEL Delete = oft nur Füllzeichen, bei Lochstreifen zum überstanzen

SYN	SYN	STX	Text	ETX	BCC

Meldung aus einem Block ohne Kopf

SYN	SYN	SOH	Iden	tifier	STX	Text	ETX	BCC

|----------header----------|

Meldung aus einem Block mit Kopf

SYN	SYN	SOH	Identifier	STX	Text	ETB	BCC

erster Block einer Meldung, die aus mehreren Blöcken besteht.

SYN	SYN	STX	Text	ETB	BCC

zweiter bis vorletzter Block einer Meldung aus mehreren Blöcken..

SYN	SYN	STX	Text	ETX	BCC

letzer Block einer Meldung aus mehreren Blöcken

SYN	SYN	ACK

Positives Acknowledge

SYN	SYN	NAK

Negatives Acknowledge

SYN	SYN	P/S	Adresse	ENQ

Aufforderung zum Auswählen oder Pollen

SYN	SYN	EOT

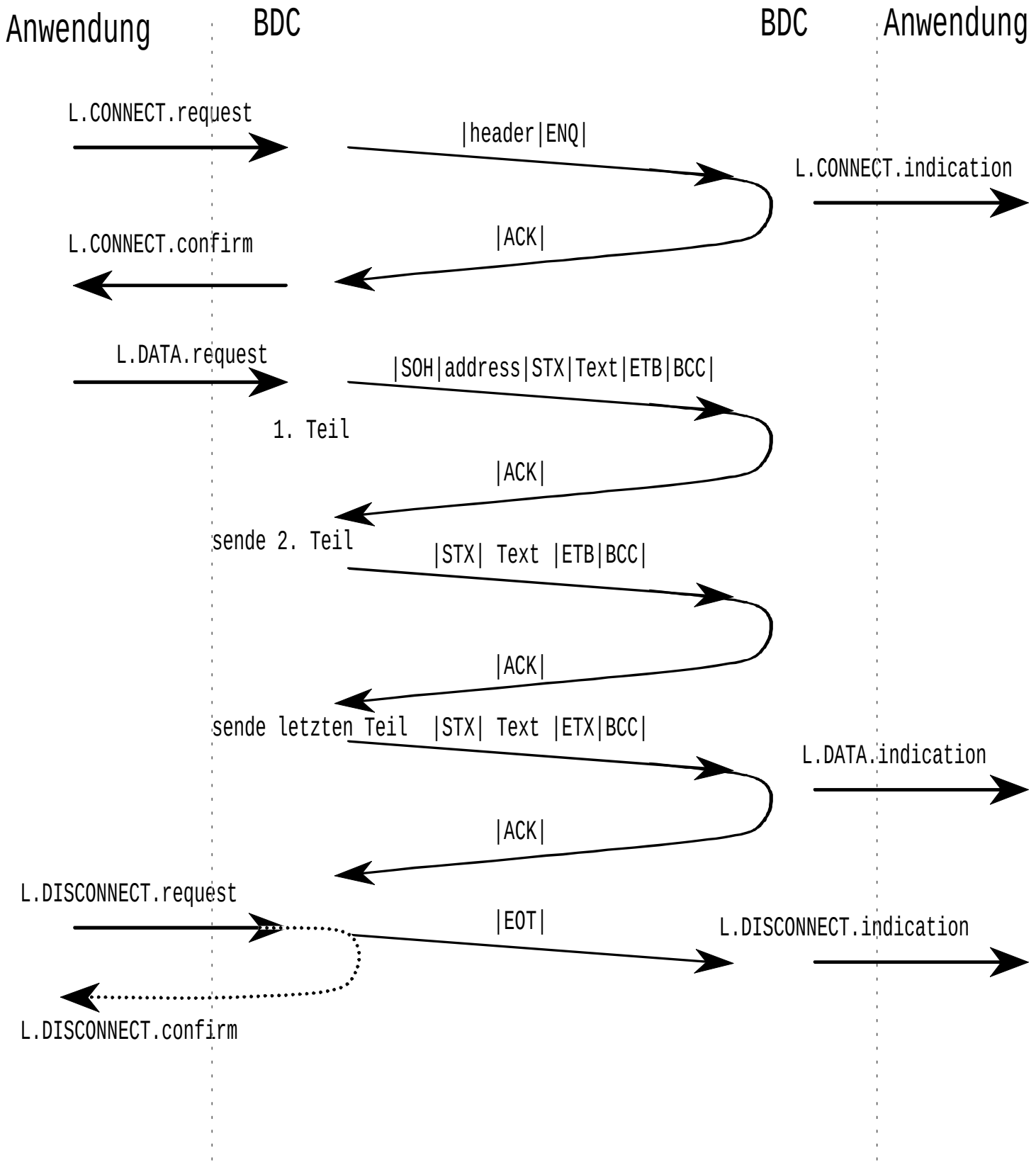
Verbindung beenden

	Station 1 → Station 2	
Prüfen ob Station 2 bereit oder pollen	ENQ → ← ACK	Station 2 ist bereit
Station 2 pollen	ENQ → ← NAK	Station 2 hat keine Daten zu übertragen
Beide Stationen senden gleichzeitig starte Zufallstimer	ENQ → ← ENQ ← ENQ ACK →	 starte Zufallstimer timer der Station 2 war kürzer
Station 2 antwortet nicht nach einigen Wiederholungen wird mit EOT die Verbindung abgebrochen	ENQ → ENQ → EOT →	keine Reaktion
Fehlerfreie Übertragung	STX|Text|ETB → ← ACK0 STX|Text|ETB → ← ACK1 STX|Text|ETX → ← ACK0	Das ACK ist nummeriert und wird modulo 2 gezählt.

Übertragung mit Fehler Der Block wird wiederholt	← ACK0 STX|Text|ETX → error ← NAK STX|Text|ETX → ← ACK1	letztes ACK = 0 Fehler wird erkannt.
Übertragung mit Fehler Timer läuft aus, Enquiry ACK0 zeigt an, daß Meldung nicht empfangen wurde, nochmals senden.	← ACK0 STX|Text|ETX → error ENQ → ← ACK0 STX|Text|ETX → ← ACK1	letztes ACK = 0. Meldung nicht empfangen, kein ACK oder NAK . ENQ wird mit dem letzten ACK beantwortet.
Überlastbehandlung Anfrage wegen verspätetem ACK Anfrage wegen verspätetem ACK Übertragung normal fortsetzen	← ACK0 STX|Text|ETX → ← ACK1 ENQ → ← ACK1 ENQ → ← ACK1	letztes ACK = 0 Ack nach 2s Delay Ack nach 2s Delay ENQ normal beantworten

Überlastbehandlung	STX|Text|ETX → ← NAK	Überlastabwehr durch ablehnen der Meldung
Fehler im Acknowledge Timer läuft aus Zuletzt wurde ACK0 empfangene, also ist die Meldung richtig angekommen	← ACK0 STX|Text|ETX → error ← ACK1 ENQ → ← ACK1	letztes ACK = 0 Der Block wird richtig empfangen Station 2 antwortet mit dem letzten Ack

OSI-Dienstprimitive und BDC-Meldungen



Transparenz (Character stuffing)

Problem : Sonderzeichen aus dem CCITT Alphabet No. 5 kommen im Text vor.
Die Schicht 2 reagiert darauf, obwohl das Zeichen für die Anwendung bestimmt ist

SYN	SYN	SOH	Identifier	STX	Text	ETX	BCC

Lösung: DLE wird neu eingeführt. DLE kündigt an, daß Zeichen für Schicht 2

SYN	SYN	**DLE**	SOH	Ident.	STX	Text	**DLE**	ETX	BCC

Problem : DLE darf nicht im Text vorkommen.

Lösung: DLE im Text wird verdoppelt

Beispiel: A B DLE C ETX D sollen übertragen werden

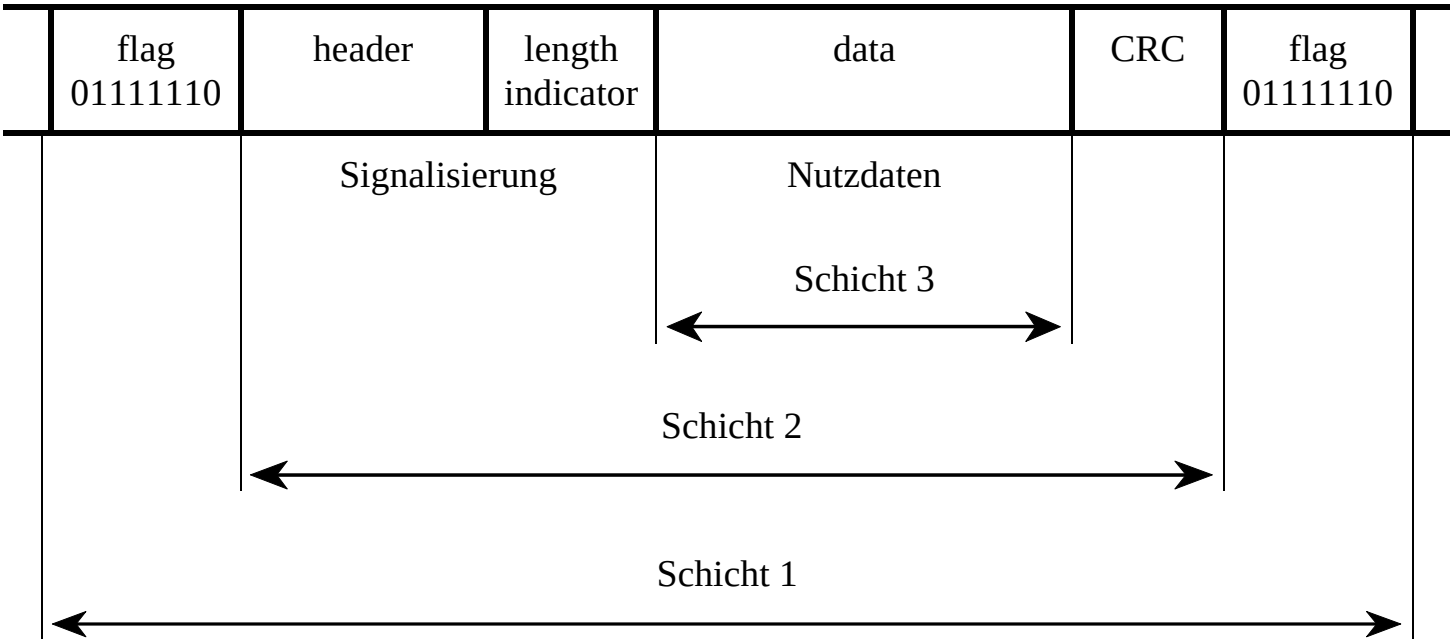
DLE	STX	A	B	DLE	DLE	C	ETX	D	DLE	ETX	BCC

Synchrone Übertragung

asynchron nur für niedrige Übertragungsraten geeignet synchron schnell und effektiv

1. Bit-Synchronisation: Leitungscodierung mit ausreichendem Taktgehalt
2. Byte-Synchronisation: Synchronisationsbyte, meist 01111110
3. Rahmen-Synchronisation: Flag am Anfang und am Ende, meist 01111110

Frame formatting (L2)



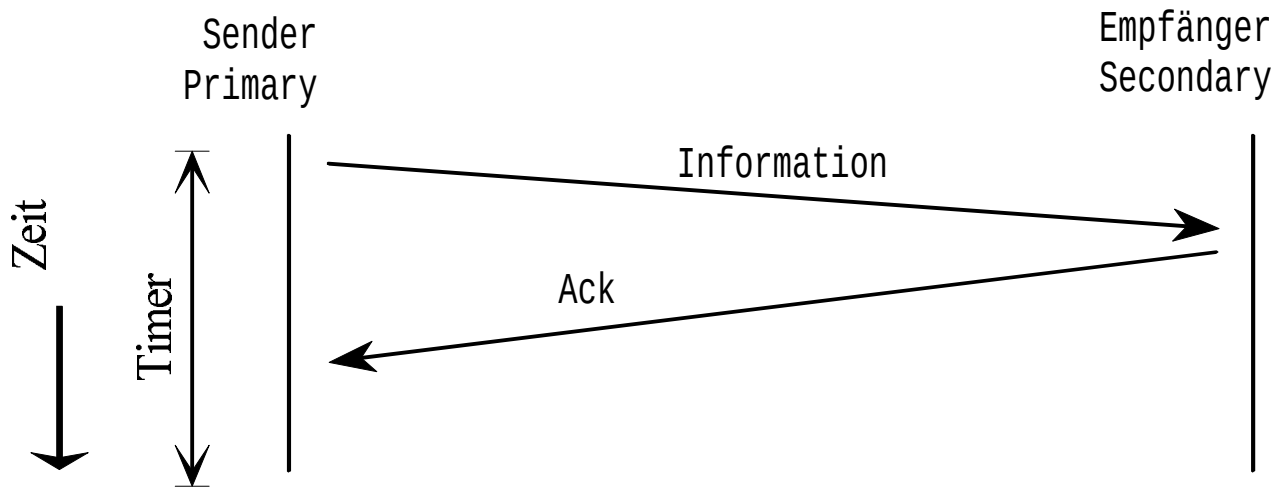
Bitstuffing (L1)

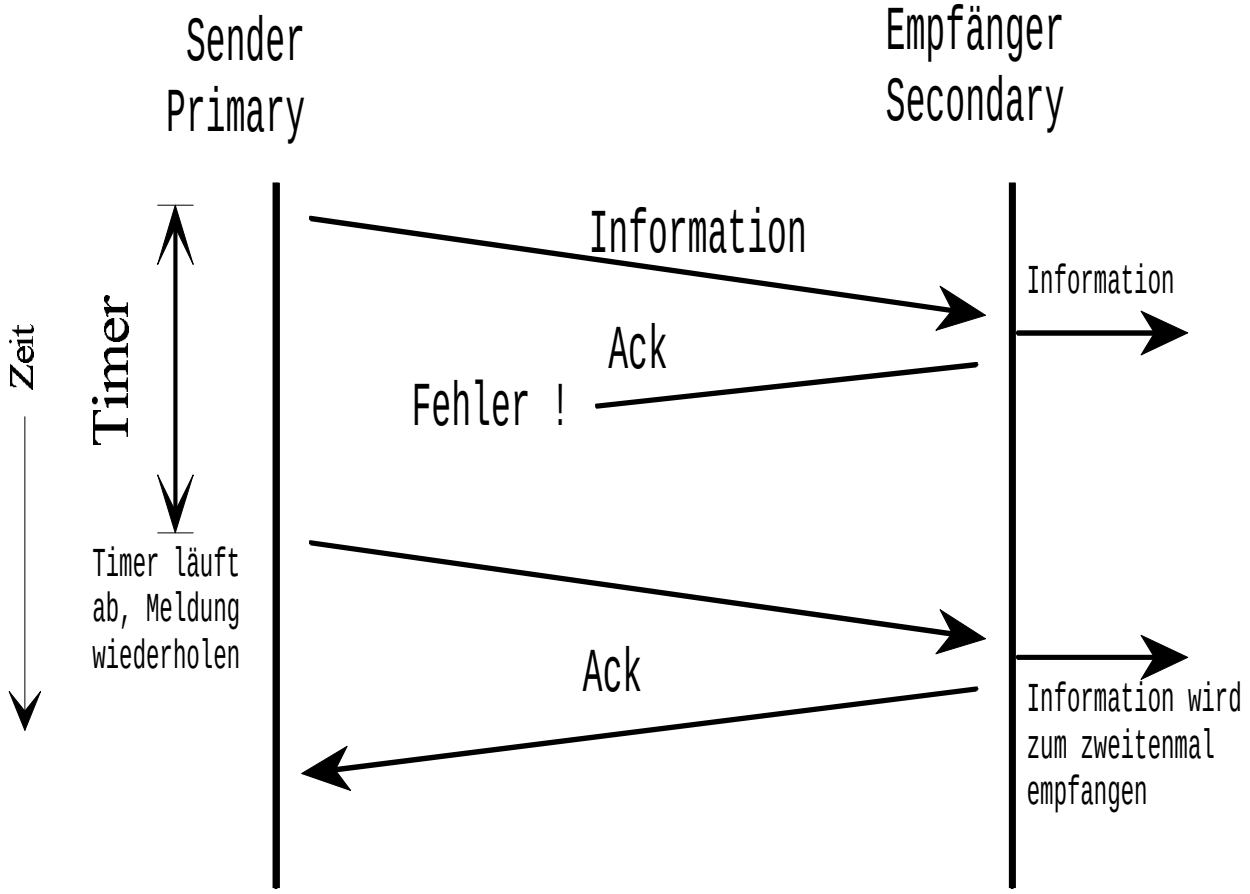
	Beispiel 1	Beispiel 2	Beispiel 3
Nutzdaten	0 1 1 1 1 0 1	0 1 1 1 1 1 0 1	0 1 1 1 1 1 1 0 1
übertragene Bitfolge	0 1 1 1 1 0 1	0 1 1 1 1 1 **0** 0 1	0 1 1 1 1 1 **0** 1 0 1

Send-and-wait Protokoll

'Automatic Repeat Request' ARQ:

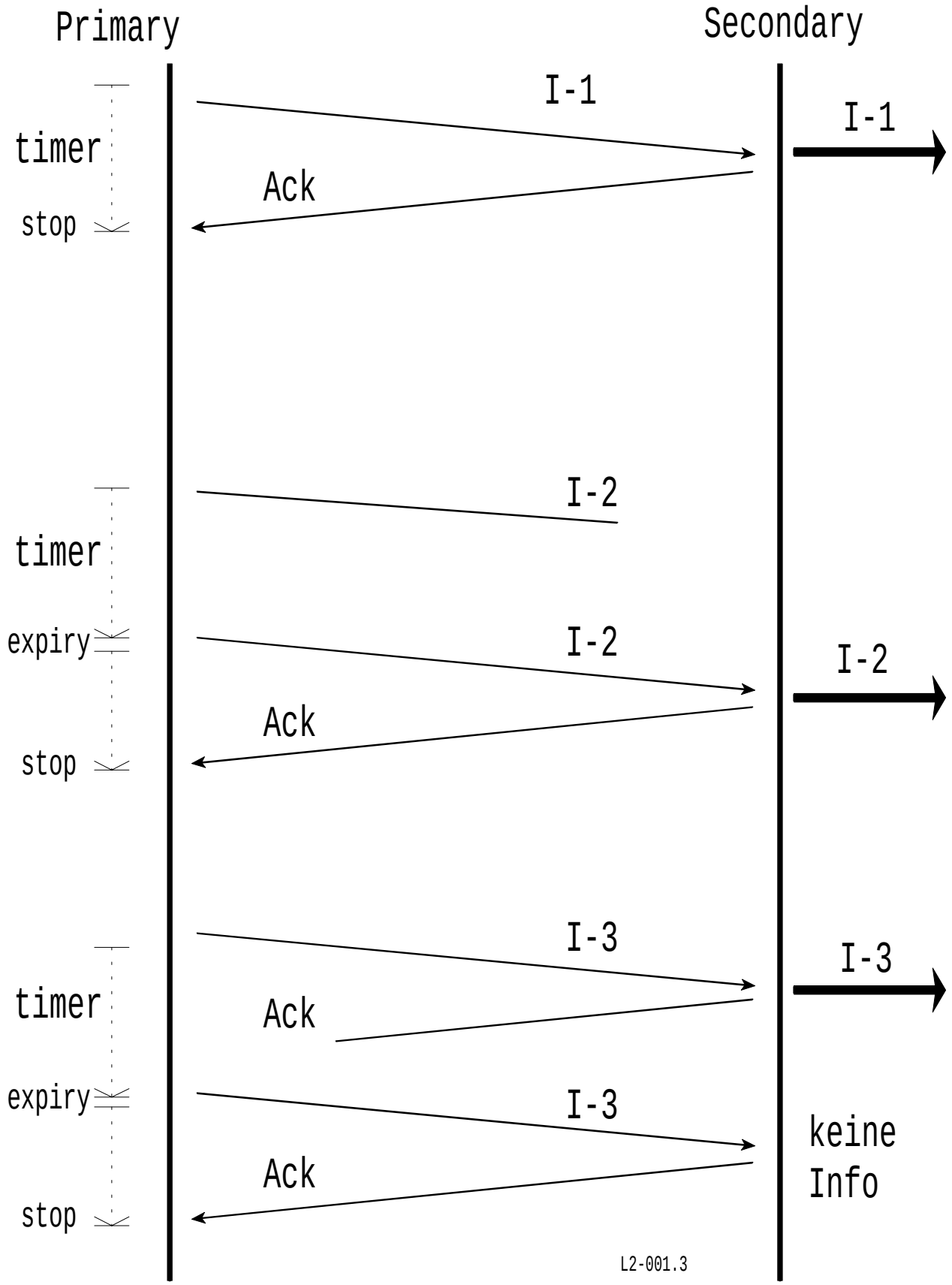
- der Sender (Primary) ist verantwortlich für das sichere Abliefern der Meldung
- der Empfänger hat nicht die Aaufgabe sich eine Meldung abzuholen.
- ungültige Meldungen werden vom Empfänger ignoriert;
- der Empfänger muß den Empfang gültiger Meldungen quittieren





Send and Wait

- kein Übertragungsfehler (oben)
- Meldung geht verloren (mitte)
- das Ack geht verloren (unten)

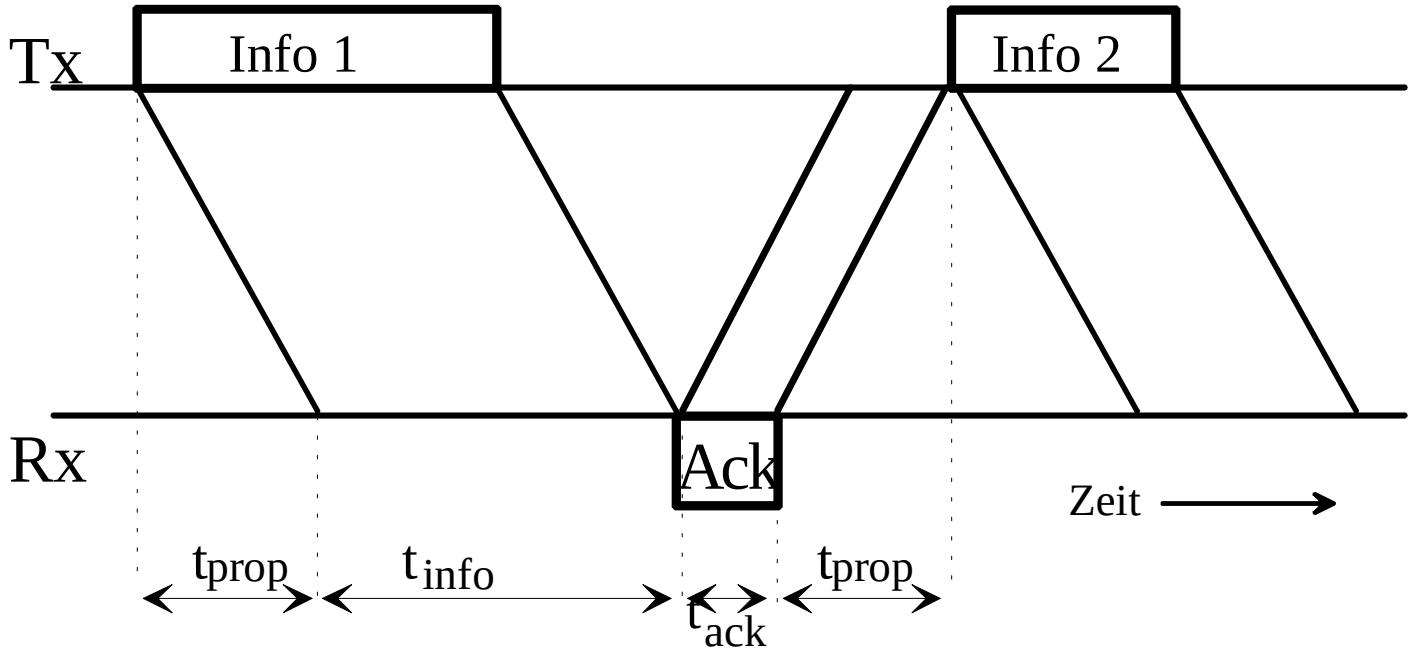


Leistungsfähigkeit

Kanalnutzung (Utilisation, Effizienz)

$$U = \frac{Zeit\ \ für\ \ Nutzdaten}{Gesamtzeit}$$

Verarbeitungszeit in Sender und Empfänger ist nicht berücksichtigt.



$$U = \frac{t_{info}}{t_{prop} + t_{info} + t_{ack} + t_{prop}} = \frac{t_{info}}{t_{info} + t_{ack} + 2\,t_{prop}}$$

t_{info} = Meldungslänge I [bit] / Übertragungsgeschwindigkeit v_D [b/s]
t_{ack} = Länge des Ack [bit] / Übertragungsgeschwindigkeit v_D [b/s]
t_{prop} = Entfernung / Ausbreitungsgeschwindigkeit + Verzögerungszeiten durch Multiplexer usw.

Zeiten in Bit ausgedrückt:

$$U = \frac{t_{info} \cdot v_D}{(t_{prop} + t_{info} + t_{ack} + t_{prop})\, v_D} = \frac{I\ (in\ Bits)}{I + Ack + 2\,t_{prop}\ v_D}$$

I Information in Bits
A Acknowledge in Bits
$t_{prop} * v_D$ = Bits auf dem Übertragungsweg

Beispiel 5.3-1:

Gegeben: Übertragungsstrecke: L = 100 km = 10^5 m;

Ausbreitungsgeschwindigkeit auf der Leitung $v_{prop} = 2*10^8$ m/s;

Verzögerung durch Multiplexer = 0,5 ms;

Meldungslänge l_{info} = 100 Bit;

Länge des Ack l_{Ack} = 10 Bit;

Übertragungsgeschwindigkeit v_D = 1 kb/s

Laufzeit $\mathbf{t_{prop}} = L/v_{prop}$ + delay = 10^5 m / $2*10^8$ m/s + 0,5 ms = **1 ms**

$$U = \frac{l_{info}}{l_{info} + l_{ack} + 2\, t_{prop}\, v_D} = \frac{100\,b}{100\,b + 10\,b + 2 \cdot 1\,ms \cdot 1\,kb/s} = 89\ \%$$

Beispiel 5.3-2: Übertragungsstrecke wie in Beispiel 5.3-1

Gegeben: Laufzeit t_{prop} = 1 ms;

Übertragungsgeschwindigkeit v_D = 1 Mb/s

$$U = \frac{l_{info}}{l_{info} + l_{ack} + 2\, t_{prop}\, v_D} = \frac{100\,b}{100\,b + 10\,b + 2 \cdot 1\,ms \cdot 1\,Mb/s} = 4{,}7\ \%$$

Die Kanalnutzung hängt von der Anzahl von Bits auf der Leitung ab.

erstes Beispiel: 1 kb/s * 1 ms = 1 Bit auf der Leitung

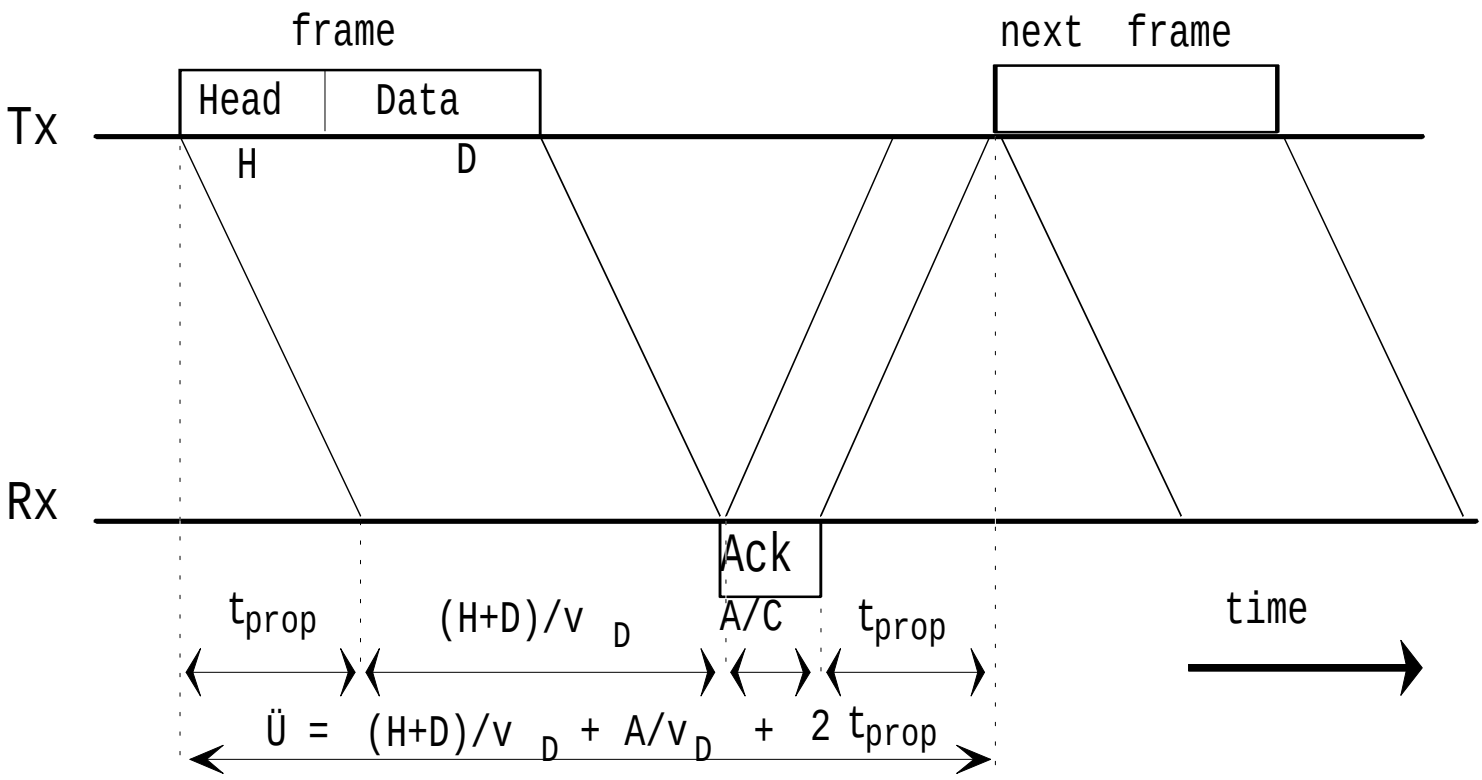
zweites Beispiel 1 Mb/s * 1 ms = 1000 Bit sind.

Leistungsfähigkeit mit Berücksichtung des Signalisierungsanteils

Zeit in Bits ausdrücken: Bitdauer $T_B = 1 / v_D$

Beispiel: Datenübertragungsgeschwindigkeit v_D = 1000 b/s ==> Bitdauer T_B = 1/1000 b/s = 1 ms/b

Die Zeit t_{prop} entspricht $t_{prop} / T_B = t_{prop} * v_D$ Bits



Kanalnutzung des send-and-wait-Protokolls ohne Bitfehler Gl. 5.3-4

Die Zeiten sind in Bit ausgedrückt

$$U = \frac{D}{H+D+A+2t_{prop}v_D}$$

$$= \frac{D}{H+D+A} \bullet \frac{H+D+A}{H+D+A+2t_{prop}v_D}$$

D = Anzahl der Datenbits, H = Anzahl der Bits im Header und im CRC-Feld, A = Anzahl der Bits der Ack-Meldung

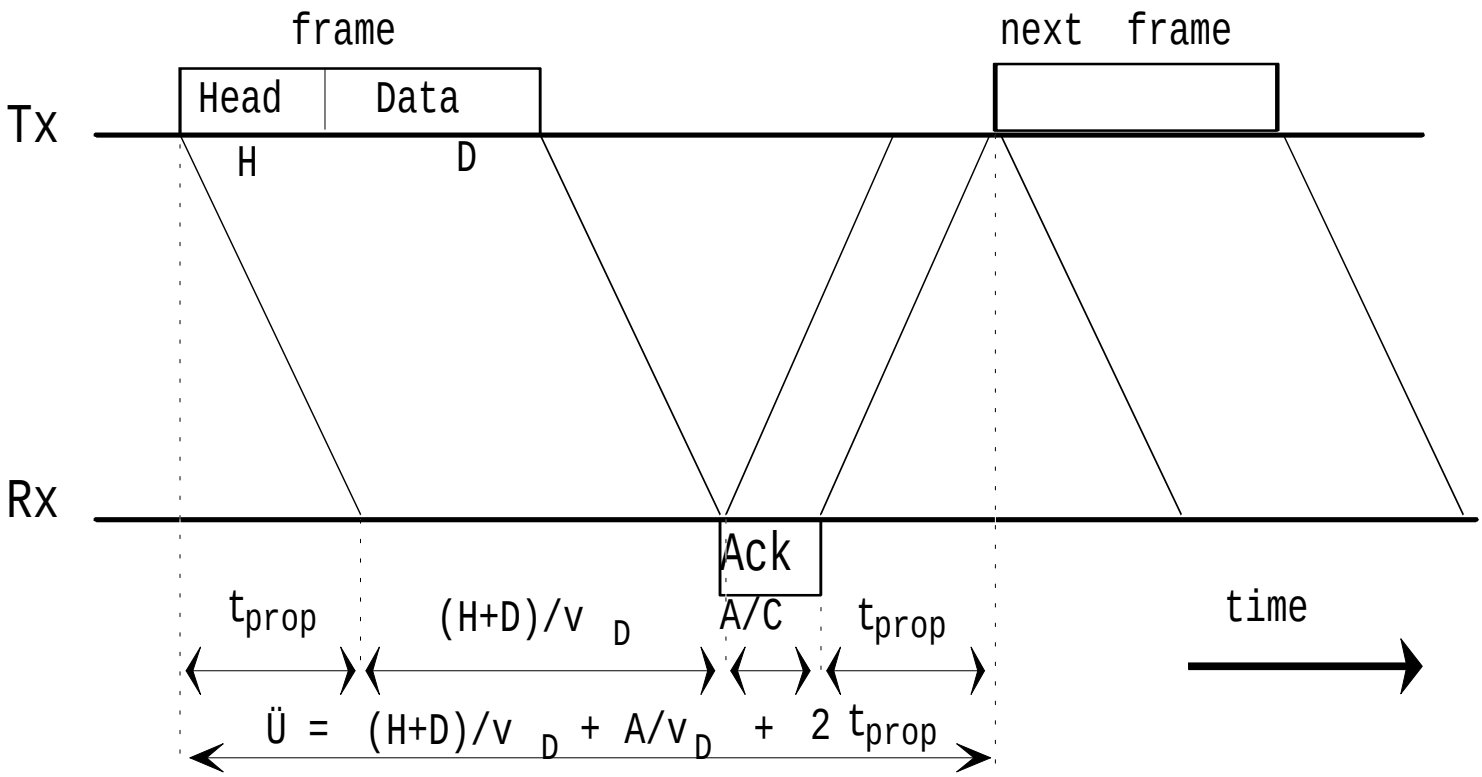
Leistungsfähigkeit des Send-and-Wait-Protokolls mit Übertragungsfehlern

Ü = Wahrscheinlichkeit, daß Meldung oder Ack fehlerhaft sind

Die Meldung wird 1 + Ü/(1+Ü) mal übertragen

1	erfolgreiche Übertragung,	Dauer = $H+D+A+2\, t_{prop}\, v_D$
Ü/(1+Ü)	Wiederholungen durch Fehler	Dauer = $H+D+A+T\, v_D$

t_{prop} = propagatio time v_D = Datenübertragungsgeschwindigkeit T = Wiederhol-Timer



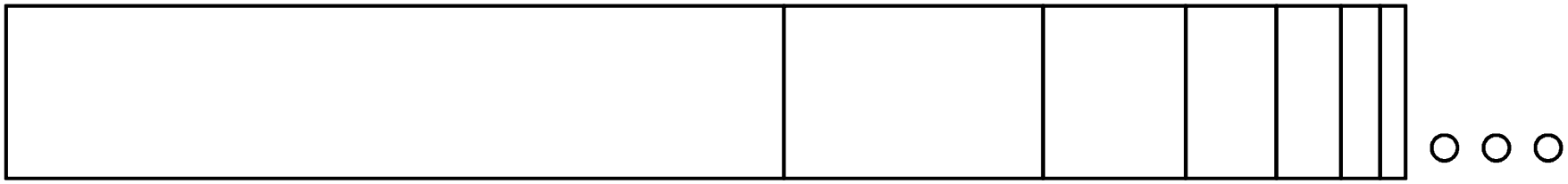
Kanalnutzung bei send-and-wait mit Fehler

$$U = \frac{D}{H+D+A+2t_{prop}v_D + (H+D+Tv_D)*\ddot{U}/(1-\ddot{U})}$$

Kanalbelegung durch Meldungsübertragung und Meldungswiederholungen beim Stop-and-wait - Protokoll

Ü = Übertragungsfehlerrate (Fehler in Meldung oder Acknowledge)

Auftrittswahrscheinlichkeit: 1 Ü $Ü^2$ $Ü^3$ $Ü^4$ $Ü^5$



mittlere Kanalbelegungszeit durch :

erste Übertragung der Meldung

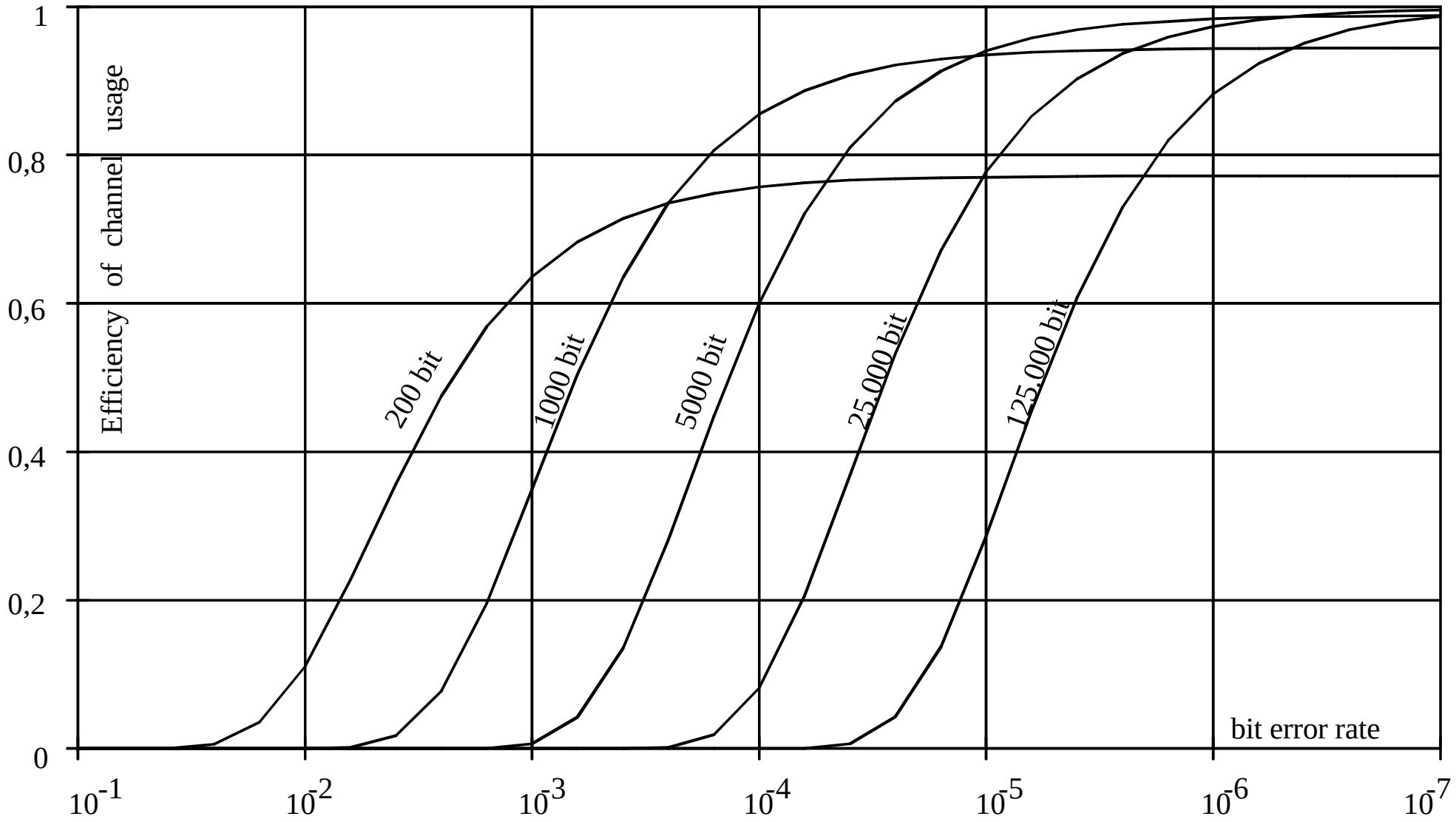
1. Wiederholung wegen Übertragungsfehler in Meldungsübertragung

2. Wiederholung wegen Übertragungsfehler in 1. Wiederholung

3. Wiederholung wegen Übertragungsfehler in 2. Wiederholung

4. Wiederholung wegen Übertragungsfehler in 3. Wiederholung

Es gibt $1 + Ü + Ü^2 + Ü^3 + Ü^4 \ldots = \frac{1}{1 - Ü}$ Übertragungen

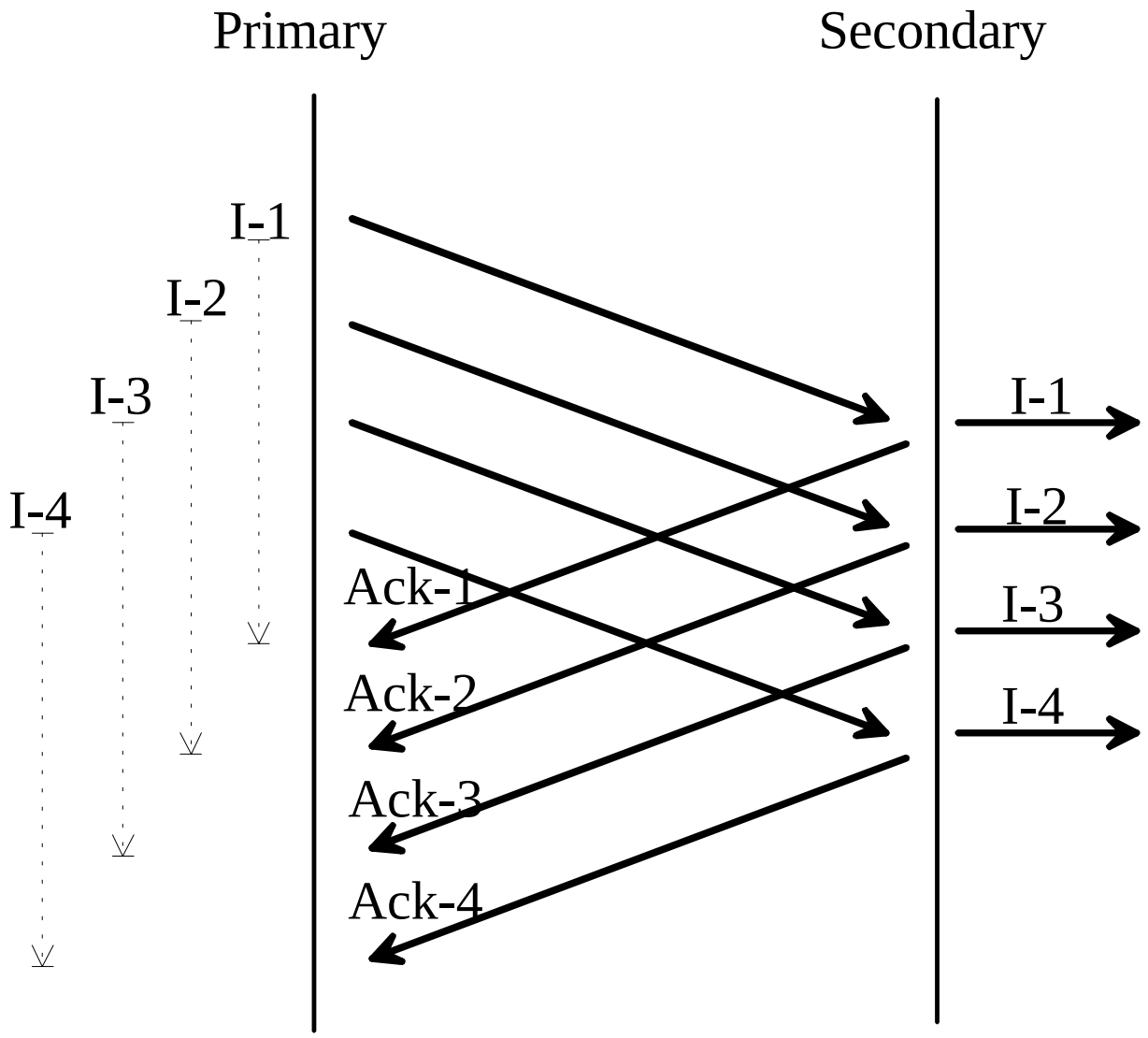


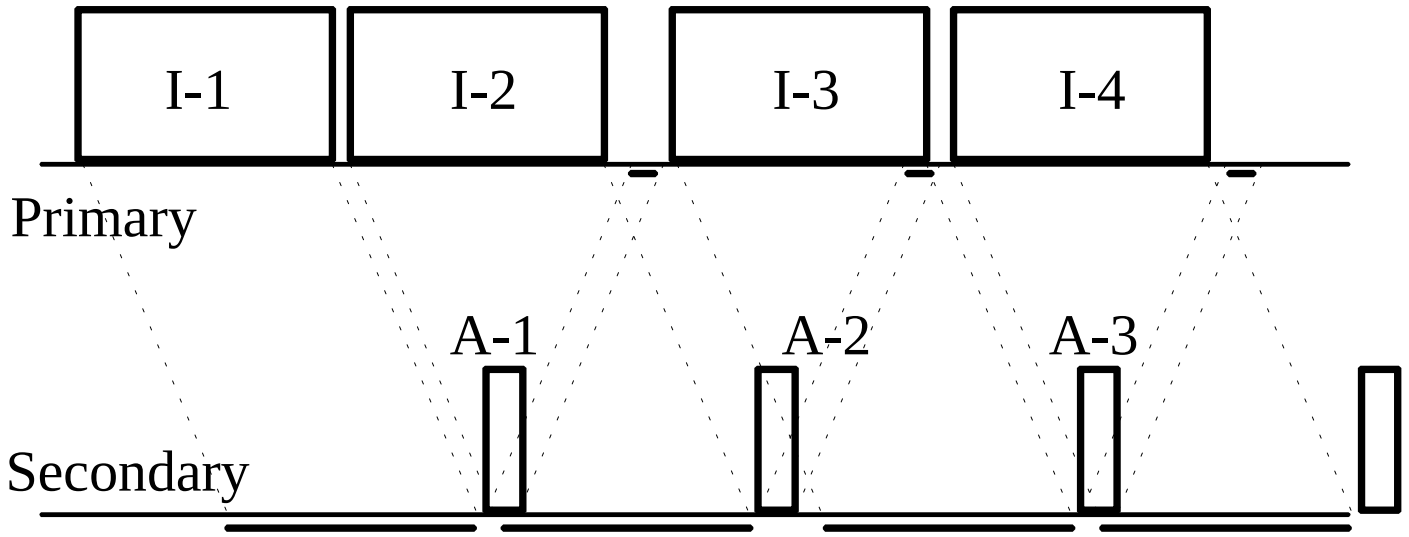
Send-and-wait-Protokoll H=24, A=16, delay = 1ms, Bitrate=9600 b/s

Effizienz der Protokolle

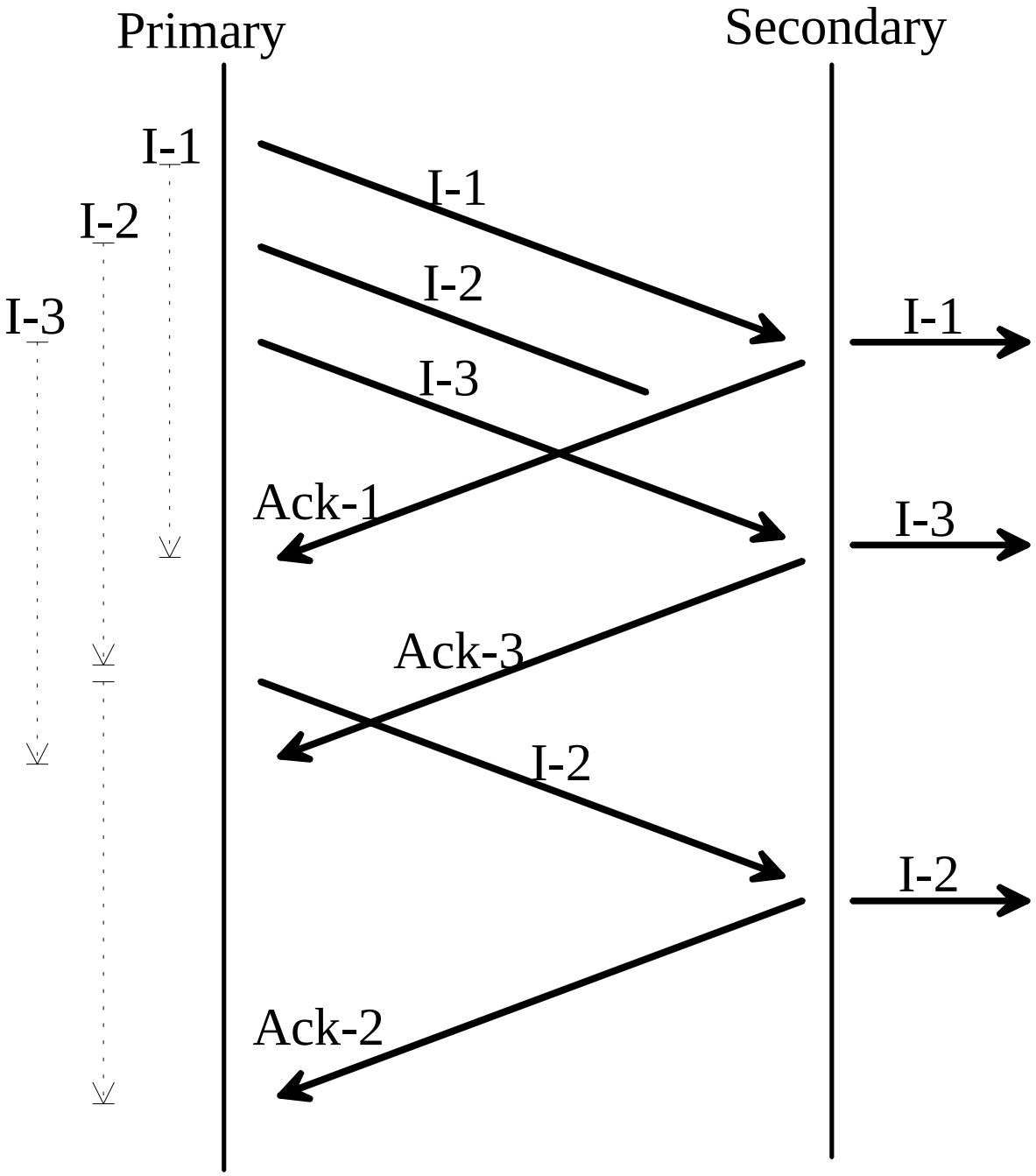
		Header-Faktor	Protokoll-Faktor	Fehler-Faktor
send-and-wait		$\frac{D}{H+D}$	$\frac{H+D}{H+D+A+2t_{prop}v_D}$	$1-\ddot{U}$
go-back-n	großes Fenster	$\frac{D}{H+D}$	1	$\frac{1-\ddot{U}}{1+\frac{(A+2t_{prop}v_D)}{H+D}\ddot{U}}$
go-back-n	kleines Fenster	$\frac{D}{H+D}$	$w\cdot\frac{H+D}{H+D+A+2t_{prop}v_D}$	$\frac{1-\ddot{U}}{1-\ddot{U}+w\ddot{U}}$
selective repeat	großes Fenster	$\frac{D}{H+D}$	1	$1-\ddot{U}$
selective repeat	kleines Fenster	$\frac{D}{H+D}$	$w\cdot\frac{H+D}{H+D+A+2t_{prop}v_D}$	$1-\ddot{U}$

Erhöhung der Effizienz



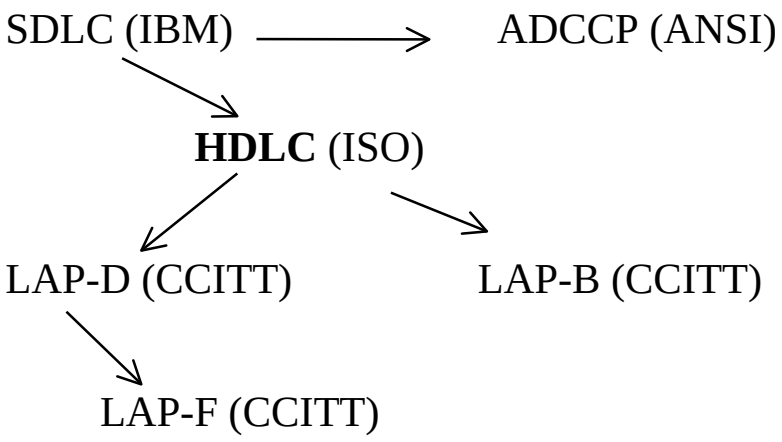


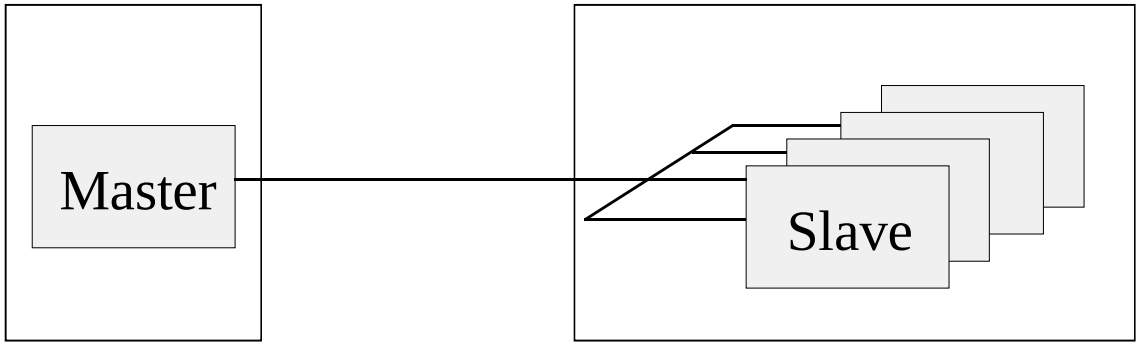
Einhaltung der Sequenz



HDLC

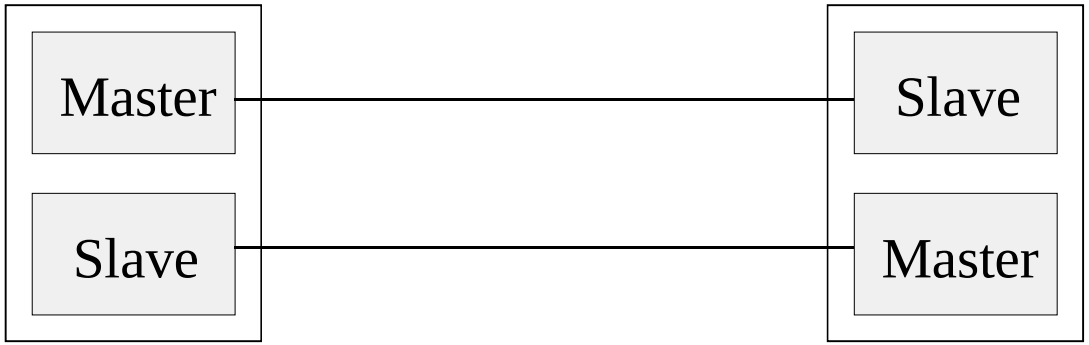
Stammbau von Schicht-2-Protokollen:





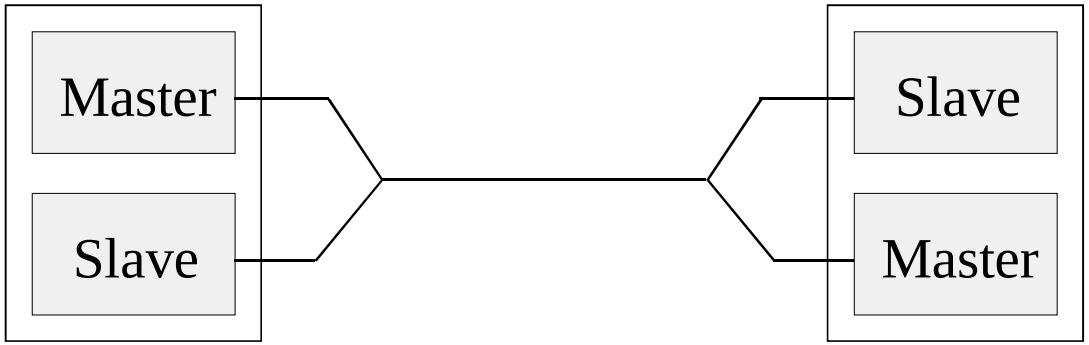
NRM

Normal Response Mode



ARM

Asynchronous Response Mode



ABRM

Asynchronous Balanced Response Mode

LAP-D Protokoll

8	7	6	5	4	3	2	1	
0	1	1	1	1	1	1	0	Flag Octet 1
SAPI						C/R	0	Adresse Octet 2
TEI							1	Adresse Octet 3
Control								Steuerfeld Octet 4 (und 5)
Data								
FCS								Rahmenprüfzeichen Octet N-1
FCS								Rahmenprüfzeichen Octet N-2
0	1	1	1	1	1	1	0	Flag Octet N

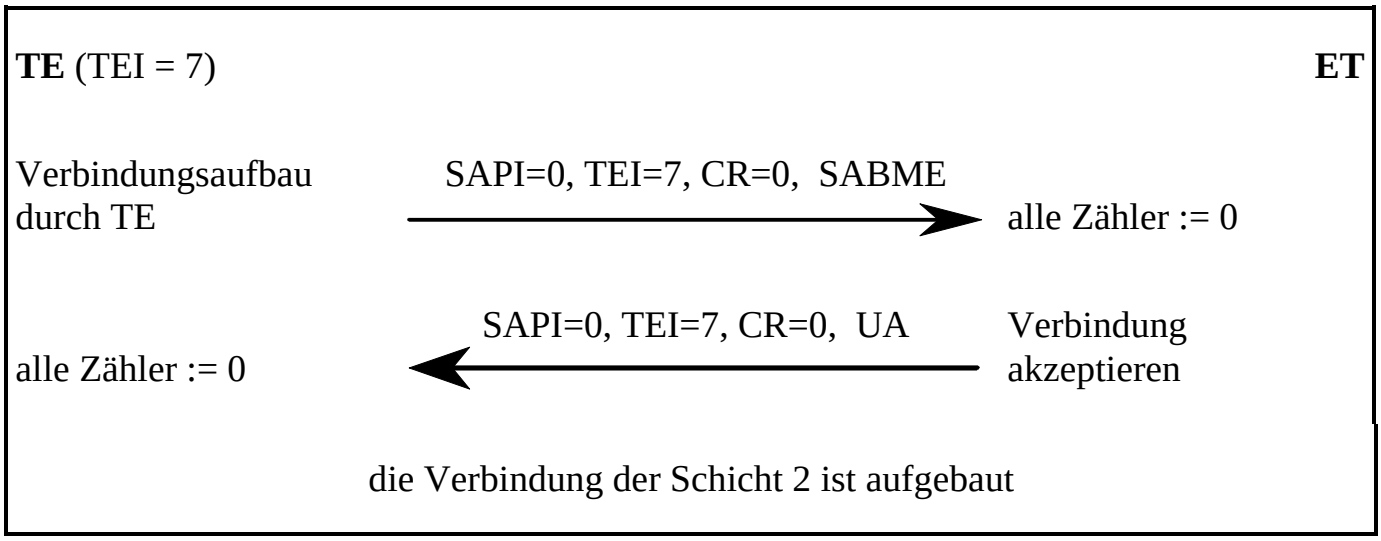
Rahmen der Schicht 2

Control Field Coding

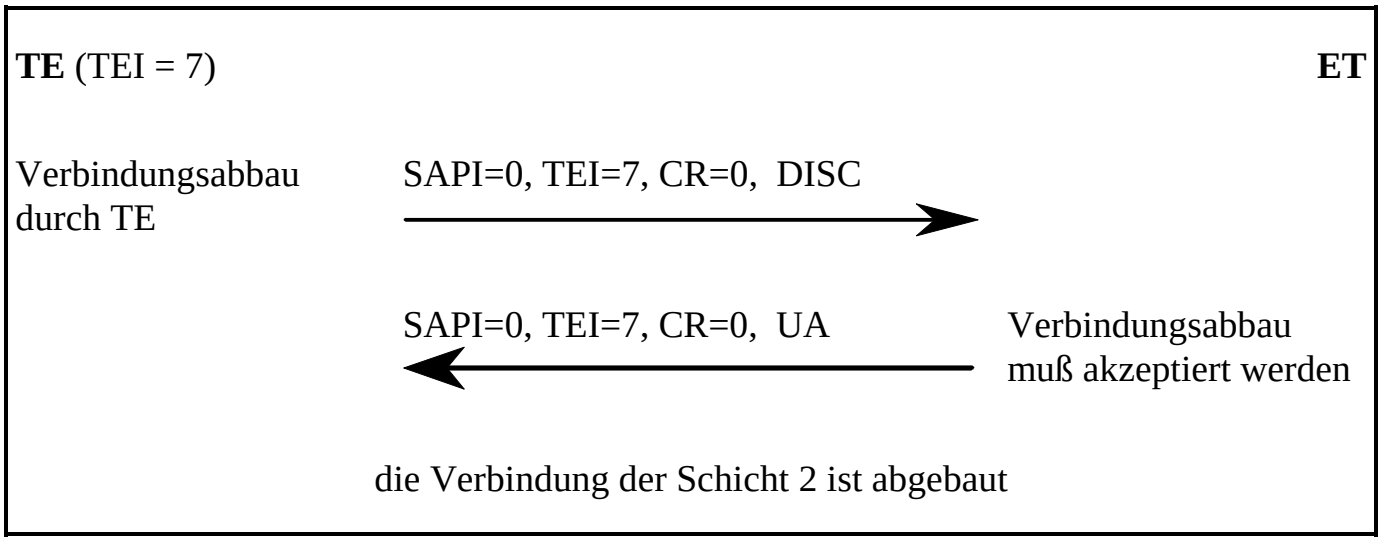
Format	Commands	Responses	Encoding	
I	I Information		N(S)	0
			N(R)	P
S	RR Receive Ready	RR Receive Ready	0 0 0 0 0 0 0 1	
			N(R)	P/F
	RNR Receive Not Ready	RNR Receive Not Ready	0 0 0 0 0 1 0 1	
			N(R)	P/F
	REJ Reject	REJ Reject	0 0 0 0 1 0 0 1	
			N(R)	P/F
U	SABME		0 1 1 P 1 1 1 1	
		DM Disconnec. Mode	0 0 0 F 1 1 1 1	
	UI Unnumbered Info		0 0 0 P 0 0 1 1	
	DISC Disconnect		0 1 0 P 0 0 1 1	
		UA Unnumbered Ack	0 1 1 F 0 0 1 1	
		FRMR Frame Reject	1 0 0 F 0 1 1 1	

	XID Exchange Identific.	XID Exchange Identific.	1 0 1 P/F 1 1 1 1

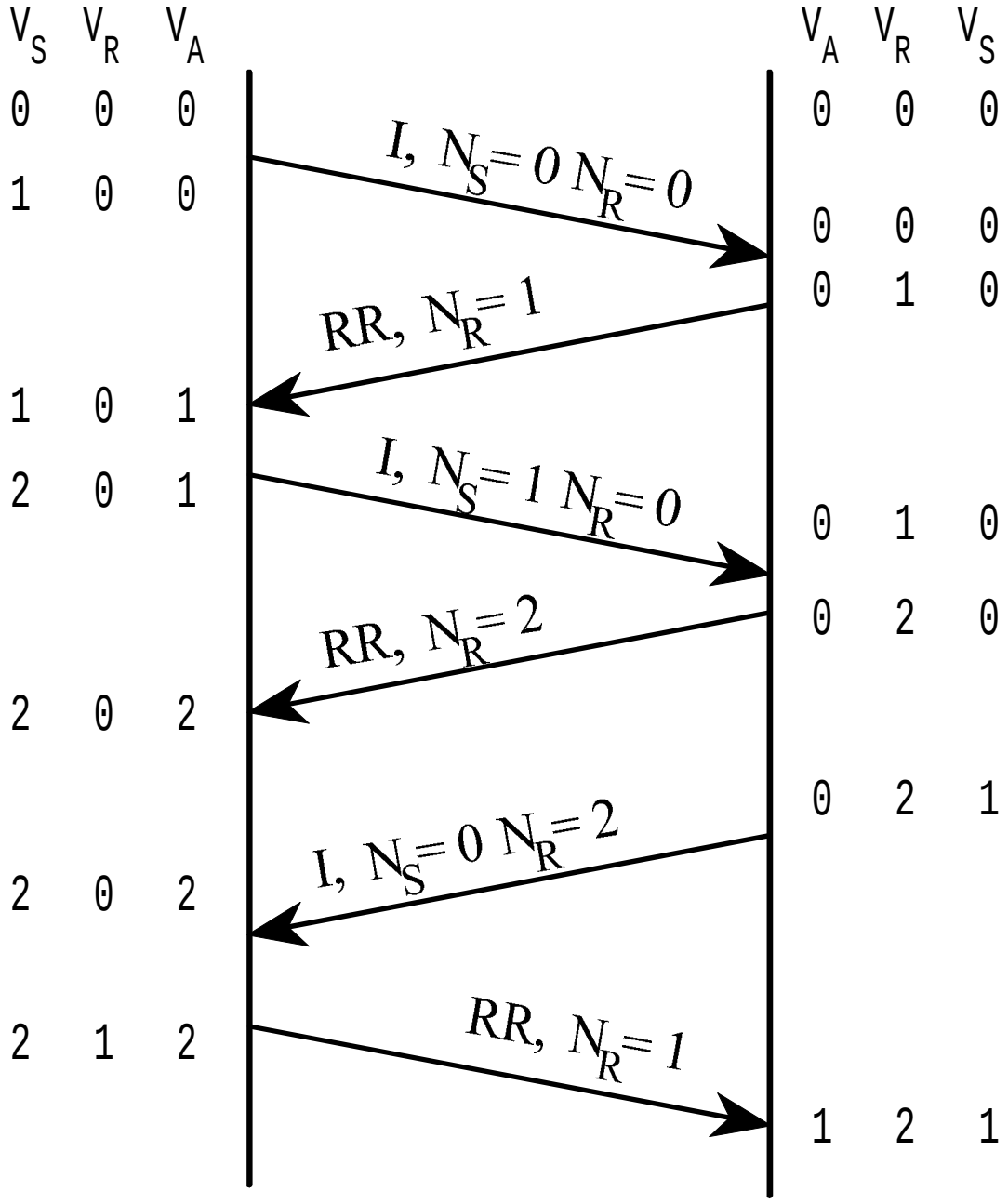
Schicht 2 Verbindungsaufbau



Schicht 2 Verbindungsabbau

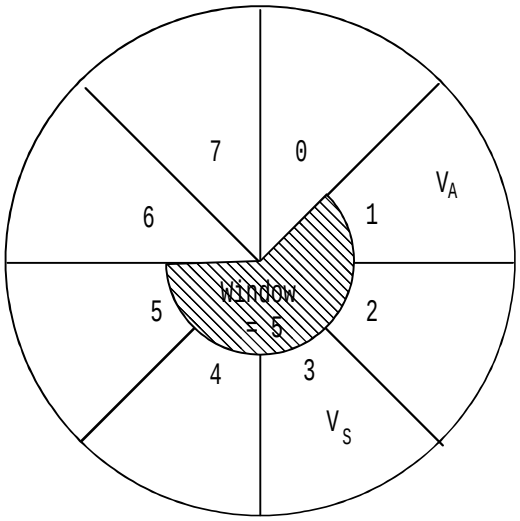


Fehlerfreie Datenübertragung



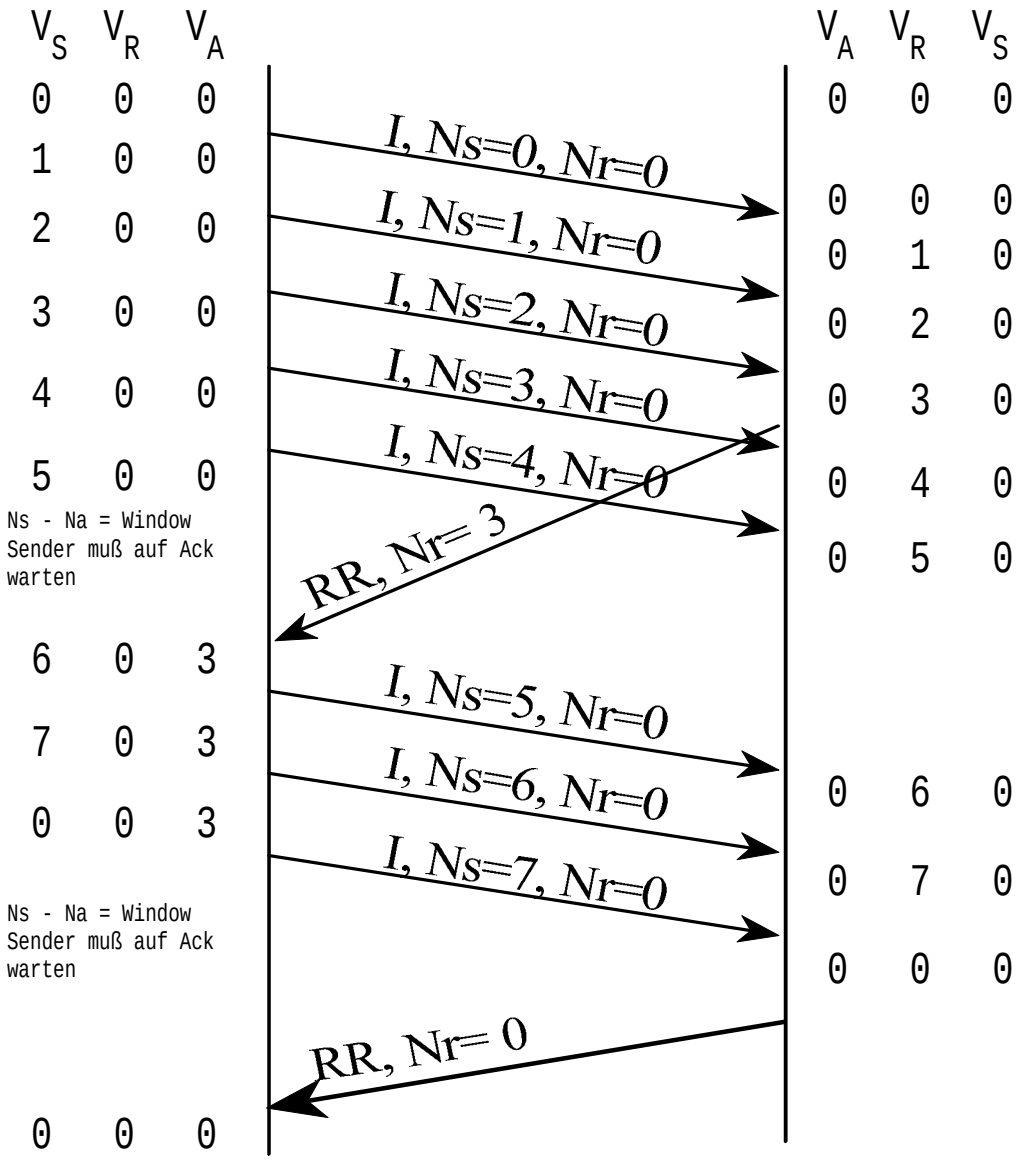
V_R	Receive Sequence Variable (Variable in TE und ET)
V_S	Send Sequence Variable (Variable in TE und ET)
V_A	Acknowledeged Frame Variable (Variable in TE und ET)
N_S	Send Sequence Counter (wird in der Meldung übertragen)
N_R	Receive Sequence Counter (in der Meldung übertragen)

Window - Mechanismus

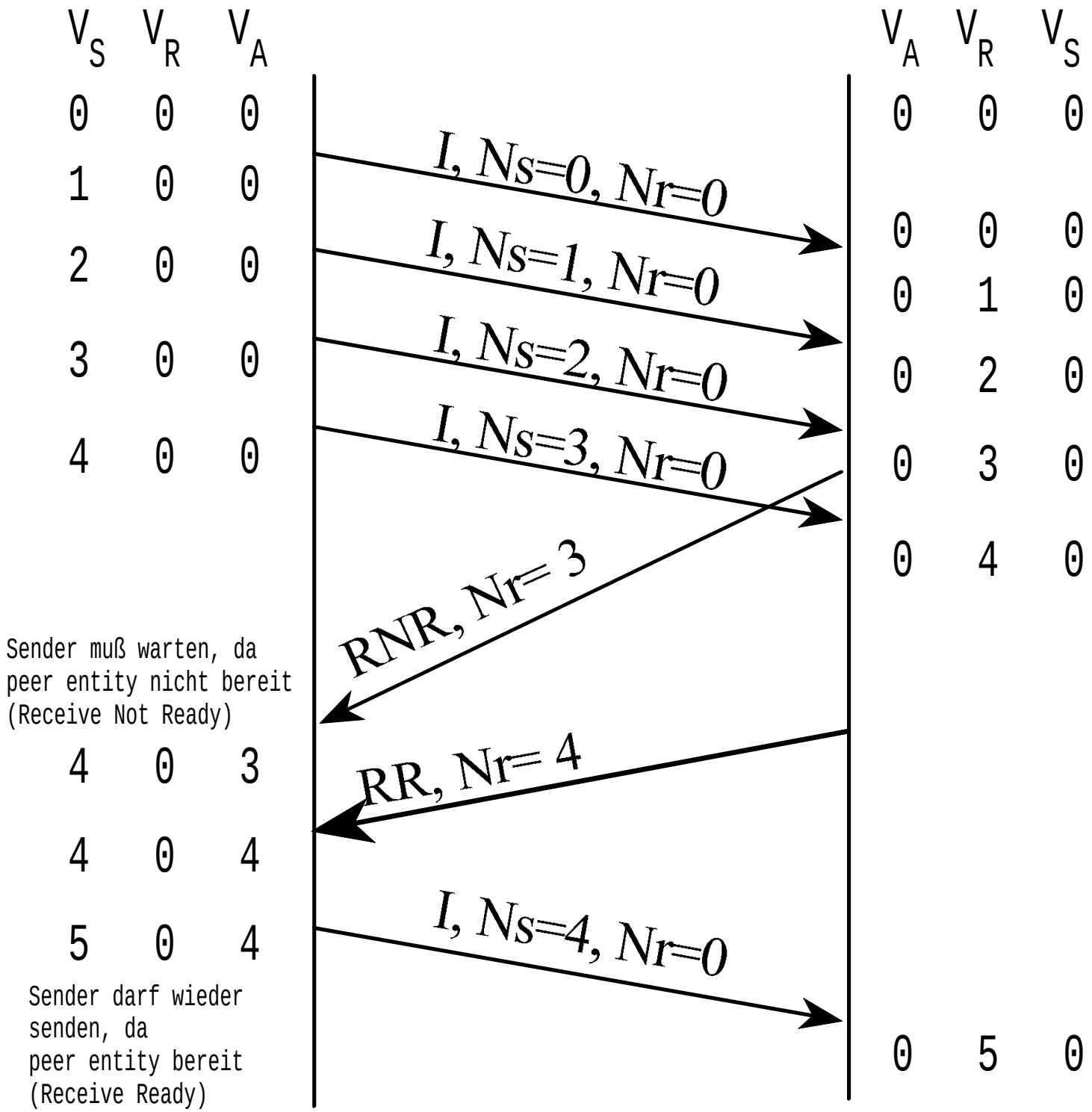


			V_A		V_S						
0	**1**	**2**	**3**	**4**	5	6	7	8	9	10	11

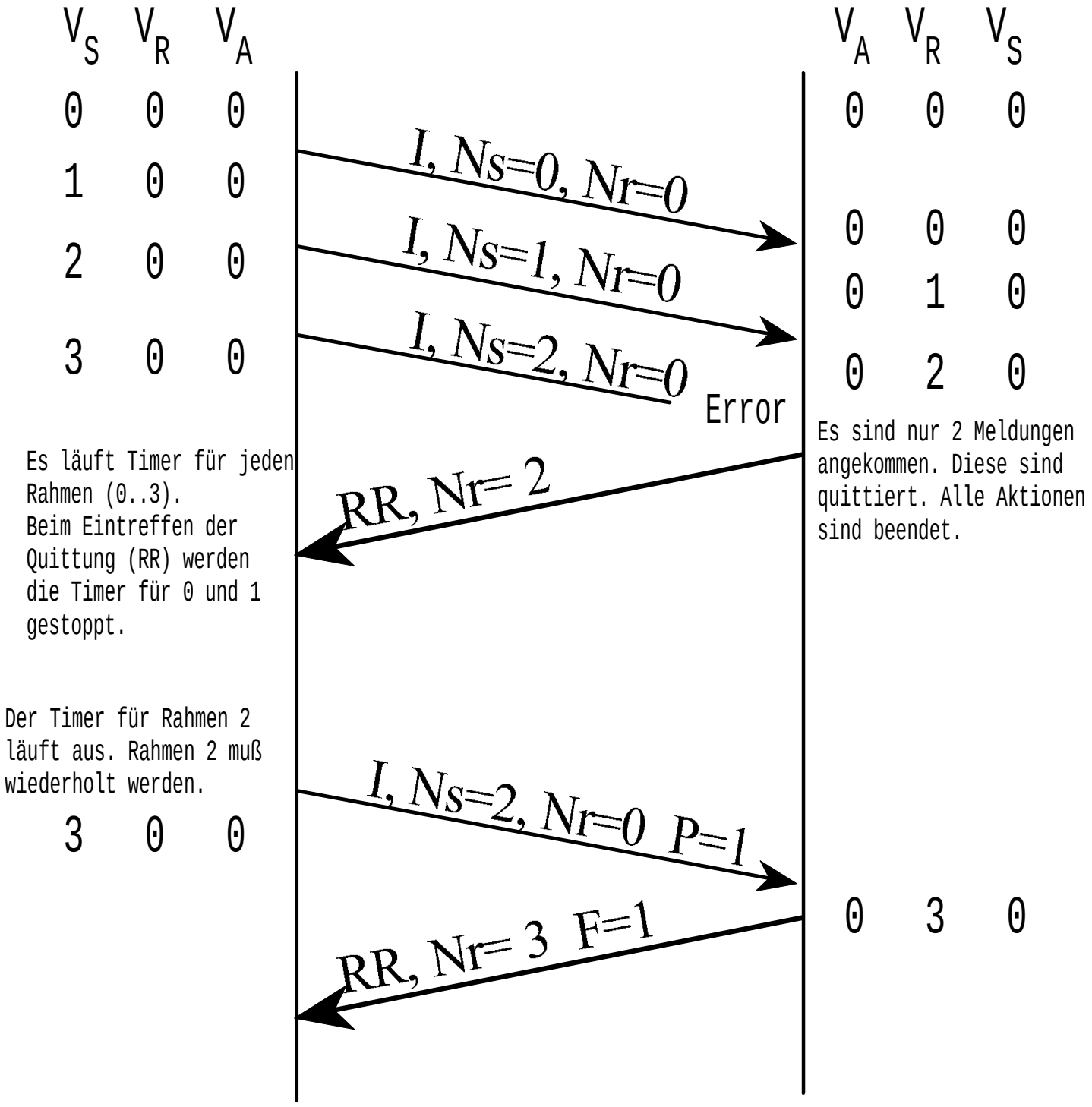
Window: $V_S = 5$ ➔ 0,1,2,3 und 4 sind gesendet
$V_A = 3$ ➔ alles bis 3 (0,1,2) ist quittiert
$V_S - V_A \leq$ Window-Größe

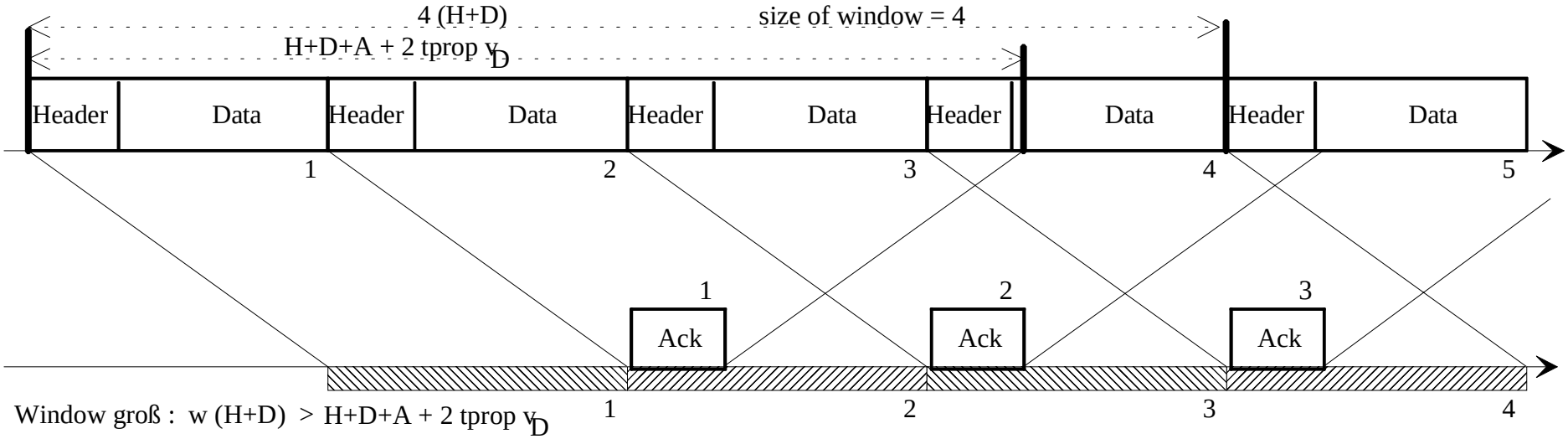


Flow Control

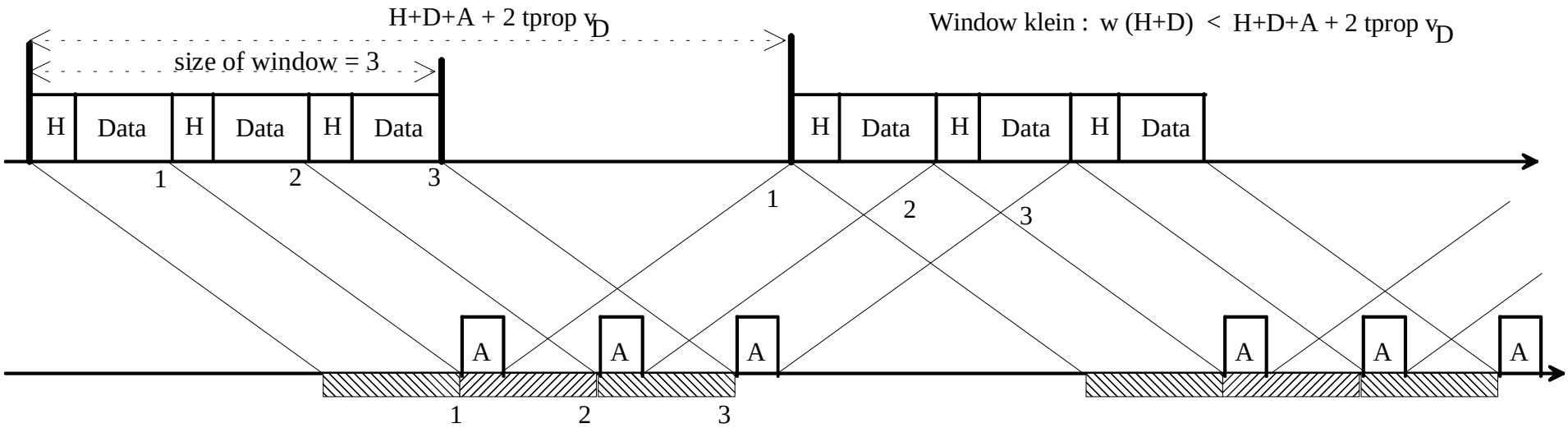


Error Control





go back n Protocol



Streaming Mode Protokoll

ist dem Selectiv Repeat Protokoll sehr ähnlich, es verzichtet jedoch auf ein Acknowledge, solange die Übertragung fehlerfrei verläuft.

Wird ein Fehler entdeckt, sendet der Empfänger ein NAK und die fehlerhafte Meldung wird wiederholt.

Die Übertragung wird fortgesetzt, sobald der Sender ein Ack für die Wiederholung der fehlerhaften Meldung erhält.

Bekannte Modem-Protokolle

Protocol	ARQ	Error Detection	Data Block Size	
IModem	no	no	128	
X-Modem	no or send-and-wait	16 bit CRC	adjustable (128)	
1-K-X-Modem	no or send-and-wait	16 bit CRC	1024	Blöcke werden auf 1K
Y-Modem	no or send-and-wait	16 bit CRC	1024	aufgefüllt
Modem 7	send-and-wait	16 bit CRC	128	Blöcke werden nicht aufgefüllt
Telink	send-and-wait	16 bit CRC	128	
Kermit	send-and-wait	16 bit CRC	90	
X-Modem W	Go-Back-N	16 bit CRC	128	
Sealink	Go-Back-N	16 bit CRC	128	
Kermit W	Selective Repeat	16 bit CRC	90	
Z-Modem	Streaming Mode	32 bit CRC	1024	
				weit verbreitet

Y-Modem: Im Header werden Dateiattribute übertragen (Datum, Uhrzeit)
Y-Modem-G:Speziell für Fehlerkorrekturverfahren (MNP, V.42), keine Steuerinformation, Überprüfung am Schluß
Z-Modem: sehr effizient, Blockgröße paßt sich der Leitungsqualität an, bei Unterbrechung Fortfahren an der gleichen Stelle