

## 5 Protokolle

### 5.1 Protokollaspekte

### 5.2 Ein Beispiel: HDLC-Protokoll

### 5.3 ARQ-Verfahren

### 5.4 Verfahren der Gleitfenster

### 5.5 Virtueller Kanal

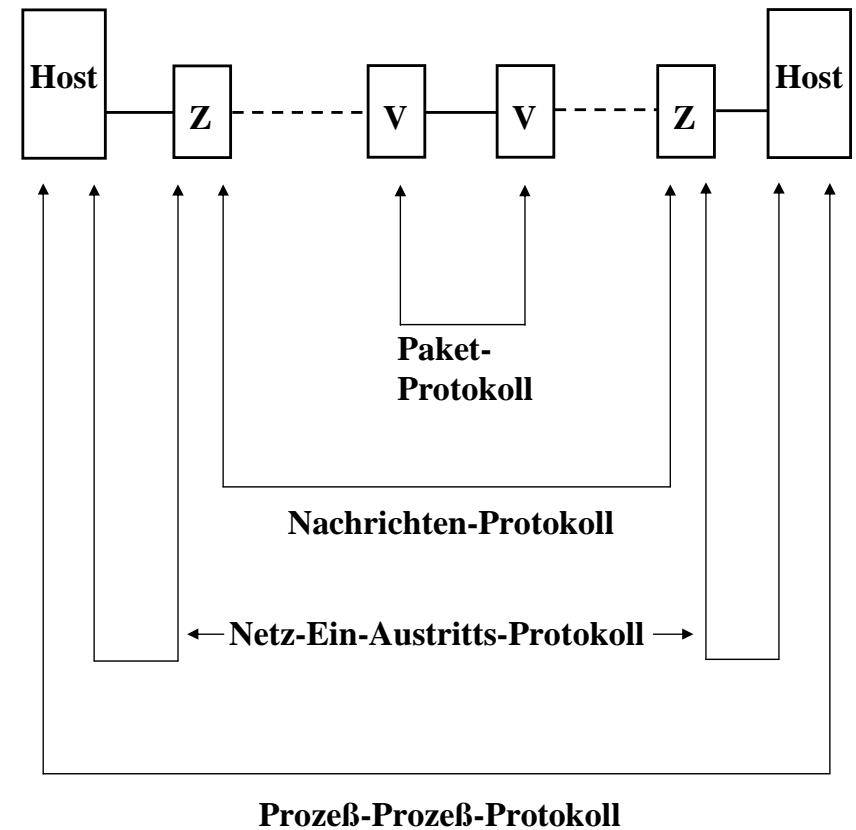
### 5.6 Transparenz

### 5.7 Bemerkungen zu TCP

### 5.8 Auf- und Abbau von Verbindungen

### 5.9 Staukontrolle und Timeout-Intervall

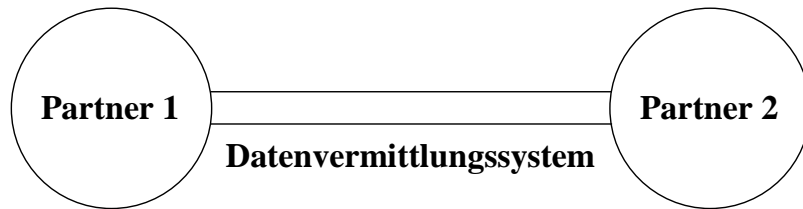
### Beispiel einer Protokoll-Hierarchie:



Bezeichnung: V = Vermittlungsrechner,  
Z = Netz-Zugangs-Rechner.

Bemerkung: Die Einheiten, die an einer genuinein gemeinsamen Aufgabe arbeiten, sind Prozesse in den Host.

## Verständigung zweier Partner über Nachrichtenmedium:



### Hauptproblem:

Wie erkennt ein Kommunikationspartner die Erwartungen des jeweils anderen?

### Die Antwort lautet:

Man vereinbart verbindlich ein Kommunikationsprotokoll.

### Protokoll – Phasen:

Verbindungsaufbau

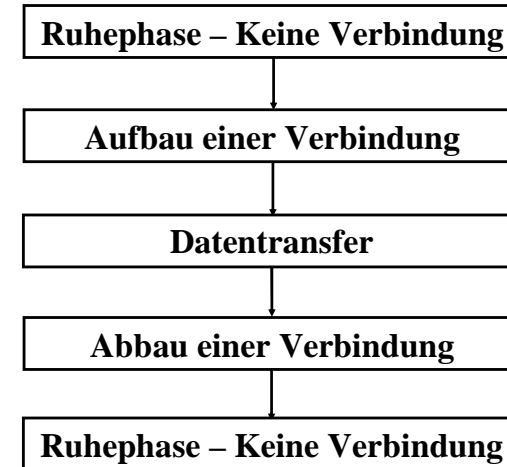
Initialisierung

Nachrichtenaustausch

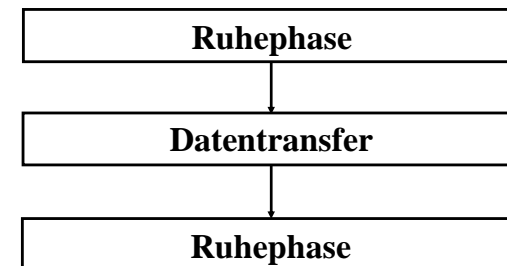
Verbindungsabbau

## Verbindungsorientierte und verbindungsfreie Kommunikation:

### Verbindungsorientierter Datenaustausch:



### Verbindungsfreier Datenaustausch:



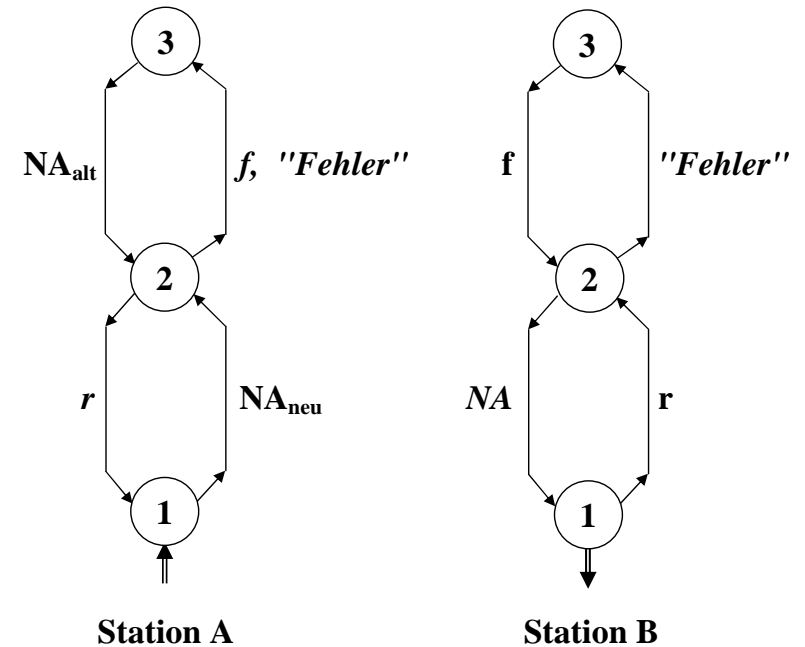
**Bemerkung:** Der verbindungsorientierte Datenaustausch ermöglicht gute Fluß- und Fehlerkontrollen.

## Protokollaspekte:

1. Technische Details  
(Betriebsart, Übertragungsgeschwindigkeit, Codes, ...)
2. Synchronisation  
(Initiative, Steuerzeichen, Nachrichtenrahmen, ...)
3. Datensicherung  
(Prüfzeichen, Zahl der Wiederholungen, Nummerierungsschemata, ...)
4. Flußkontrolle  
(Zeitschranke, Unterbrechung, Netzkonfiguration, ...)
5. Transparenz
6. Verbindungsverwaltung  
(Protokollphasen, Adressen, ...)

Ein einfaches Protokoll P1 zum Transport von Daten von der Station A zur Station B mit Quittierung:

Voraussetzung: Jeder Übertragungsfehler wird erkannt.



Bemerkung:

Eine Folge von Nachrichten wird von Station A nach Station B übertragen;  $r$  steht für den Empfang korrekter Daten,  $f$  für den Empfang fehlerhafter Daten; die gesendeten Nachrichten sind normal ausgezeichnet, die empfangenen kursiv.

Beabsichtigt ist folgendes:

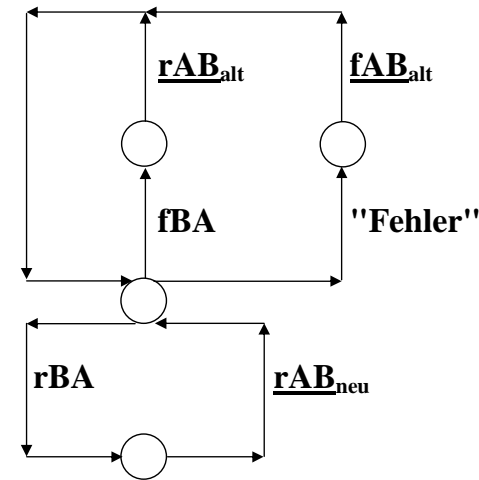
- (i) Beide Stationen werden initialisiert, Station A im Zustand 1 und Station B im Zustand 2.
- (ii) Station A erhält in Zustand 1 eine neue Teilnachricht, sendet sie an Station B und wartet auf eine Quittung.
- (iii) Station B empfängt die Nachricht korrekt oder fehlerhaft, im korrekten Fall übergibt sie die Nachricht an den Endempfänger und sendet eine positive Bestätigung, sonst wird die empfangene Bitfolge vernichtet und das Auftreten eines Fehlers an Station A gemeldet.
- (iv) Bei Empfang einer positiven Quittung fordert Station A eine neue Teilnachricht an, ansonsten wiederholt Station A die letzte gesendete Teilnachricht.

Das Protokoll ist fehlerhaft. Dies zeigt die Ereignissequenz:

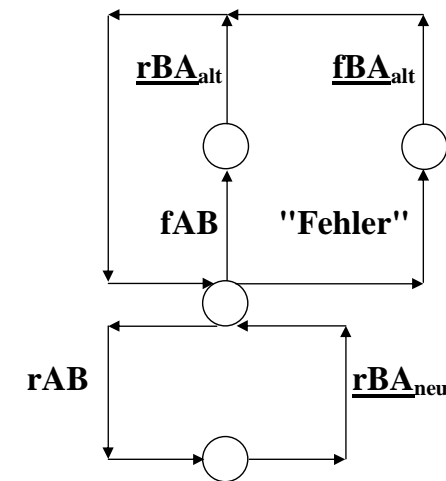
A sendet NA an B,  
 B empfängt NA korrekt,  
 B quittiert korrekten Empfang mit r,  
 A stellt Übertragungsfehler fest,  
 A wiederholt die Nachricht NA !!!

Halbduplex-Prozedur P2 für Datentransport:

P2 entsteht aus P1 durch Hinzufügen von Nachrichten für die Gegenrichtung und Hinzufügen von Gültigkeitsbits für die Gegenrichtung.



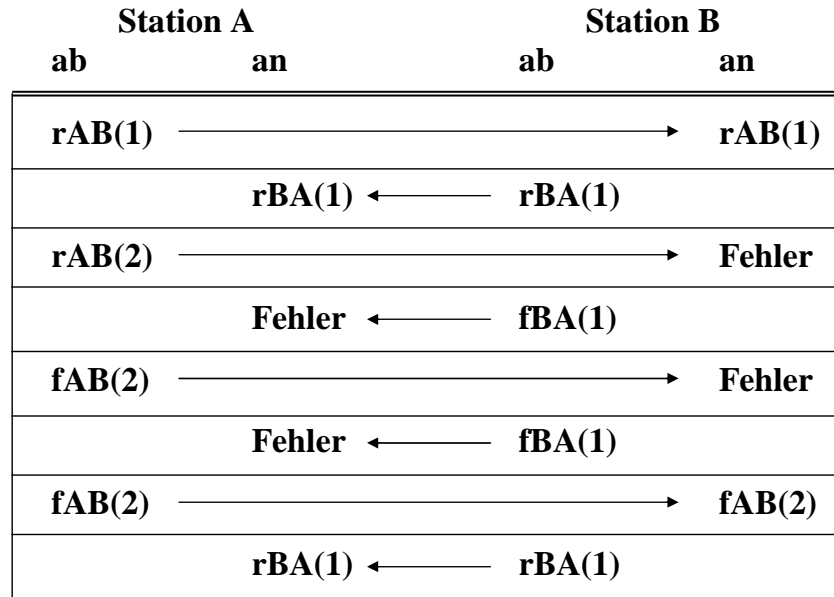
Station A



Station B

Bemerkung:  
 Sendungen sind unterstrichen.

Die Prozedur P2 erbt den P1-Fehler, wie nachfolgend demonstriert.

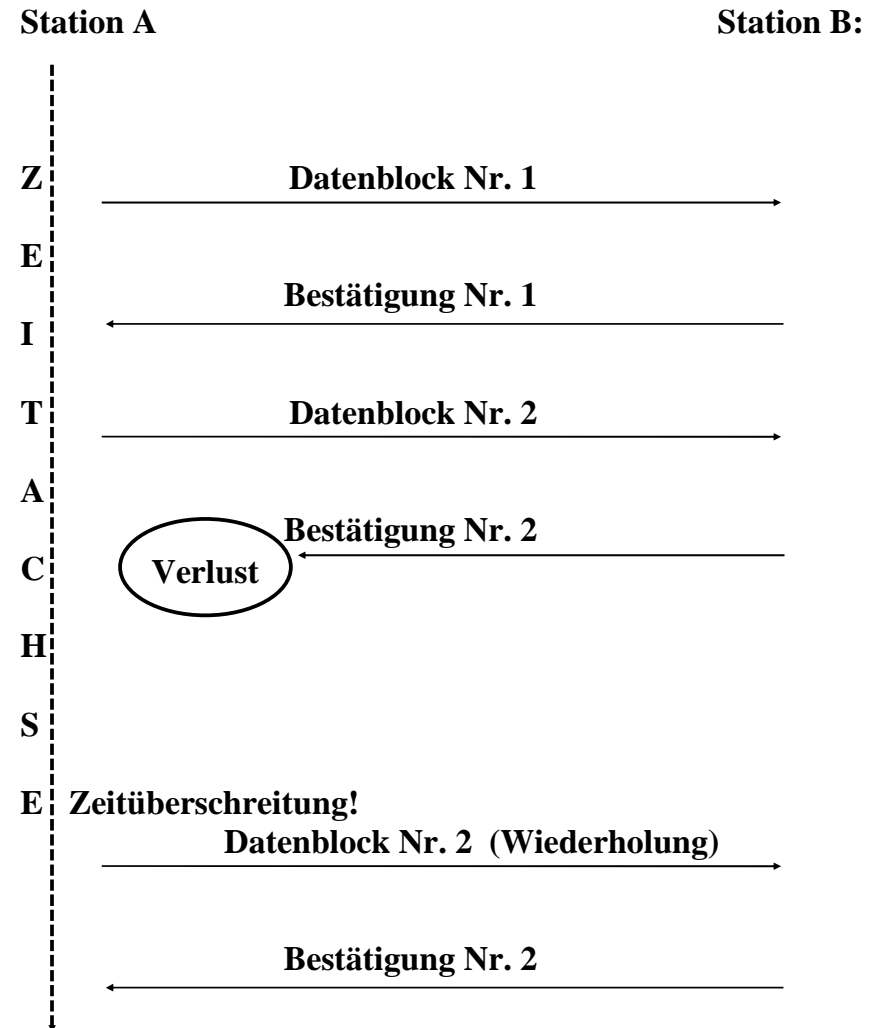


**Fehler:** BA(1) wird zweimal akzeptiert.

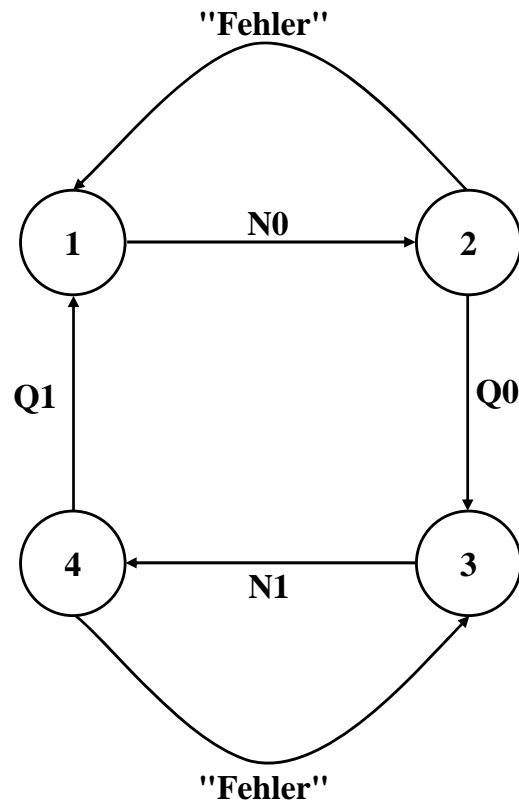
**Bemerkung:** Die Prozedur P2 lässt sich durch Einführung der Nachrichtennummern 0 und 1 zu einem zuverlässigen Protokoll erweitern.

**Sendenummern:**

**Bemerkung:** Jeder Block wird sofort bestätigt. Die Sendenummer verhindert das Entstehen von Duplikaten. In diesem Fall genügen zwei unterschiedliche Nummern.



**Kernautomat bei Verwendung eines Nachrichten-numerierungsschemas modulo 2.**



**Bemerkung:** Die Numerierung der Nachrichten und Quittungen kann mittels des gleichen Bit im Datenrahmen erfolgen.

**Beispiele zeichenorientierter Protokolle:**

- ANSI X3.28** Procedure for use of the communication control characters of ASCII in specified data communication links
- DIN 66 019** Steuerungsverfahren mit dem 7-Bit-Code bei Datenübertragung
- IBM BSC** Binary Synchronous Communication
- DDCMP** Digital Data Communication Message Protocol

**Beispiele bitorientierter Protokolle:**

- ANSI X3.66**  
**ADDCCP** - Advanced Data Communication Control Procedures
- ISO 3309, 4335, 6159, 6259, 13239**  
**HDLC** - High Level Data Link Control Procedure
- ITU-T X.25, ITU-T X.75**  
**LAP-B** - Link Access Procedure Balanced
- ITU-T Q.920, Q.921**  
**LAP-D** - Link Access Procedure D-Channel
- IEEE 802.2 (ISO 8802.2)**  
**LLC** - Logical Link Control
- IBM SDLC**  
**SDLC** - Synchronous Data Link Control
- RFC 1661, RFC 1662**  
**PPP** - Point-to-Point Protocol

## Beispiel eines Protokolls:

HDLC = High level data link control

## Charakteristika:

### Drei Stationsarten:

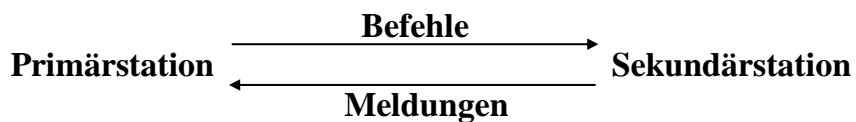
Primärstation,  
Sekundärstation,  
Doppelstation.

### Zwei Verbindungsarten:

balanciert,  
nicht balanciert.

### Drei Transfermodi:

NRM: Normal response mode,  
ABM: Asynchronous balanced mode,  
ARM: Asynchronous response mode.



## HDLC-Rahmenstruktur:

Flag	Adresse	Kontrolle	Info.	Prüfsumme	Flag
------	---------	-----------	-------	-----------	------

**Bemerkung:** Sowohl das Adreßfeld als auch das Kontrollfeld sind in Einheiten von jeweils 8 Bit erweiterbar.

### Aufbau des 8-Bit-Kontrollfeldes:

	0	1	2	3	4	5	6	7
<b>I-Rahmen:</b>	0	N(S)			P/F	N(R)		
<b>S-Rahmen:</b>	1	0	S	P/F	N(R)			
<b>U-Rahmen:</b>	1	1	M	P/F	M			

### Bezeichnungen:

Flag = 01111110  
I-Rahmen = Informationsrahmen  
S-Rahmen = Überwachungsrahmen  
U-Rahmen = Unnumerierter Rahmen  
P/F = "Poll/Final" Bit  
N(S) = Sendenummer  
N(R) = Empfangsnummer  
S = Codierung der Überwachungsbefehle  
M = Codierung der allgemeinen Befehle  
Prüfpolynom = CRC-16 oder CRC-32

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

## Befehle und Meldungen im HDLC-Protokoll:

### Information command: I

Mittels I-Blöcken wird der Austausch von Nutzdaten zwischen Endgeräten durchgeführt, Hilfsaufgaben wie Bestätigungen und Anklopfen werden als Zusätze codiert.

### Supervisory commands:

RR	Receive Ready,
RNR	Receive Not Ready,
REJ	Reject,
SREJ	Selective Reject.

RR- und RNR-Blöcke dienen der Bestätigung von I-Blöcken, zusätzlich signalisiert ein RNR-Block, daß der Empfänger nicht gewillt ist, weitere I-Blöcke abzunehmen. REJ- und SREJ-Blöcke dienen der Fehlererholung, ein REJ-Block fordert den Empfänger auf, alle I-Blöcke ab der mitgeteilten Nummer zu wiederholen, ein SREJ-Block fordert nur die Wiederholung eines Blockes. Da Wiederholungsaufforderungen gleichzeitig Bestätigungen enthalten, kann nur ein SREJ-Block pro Verbindung unbeantwortet sein.

## Normal commands and responses:

(Legende: c = command, r = response)

UI c/r Unnumbered Information

UI-Blöcke ermöglichen den Nachrichtenaustausch, ohne vorangehende Vereinbarung von Rahmenbedingungen. Eine dezidierte Fehlerkontrolle ist daher nicht möglich.

SNRM c Set Normal Response Mode

SNRME c Set Normal Response Mode  
Extended

SARM c Set Asynchronous Response Mode

SARME c Set Asynchronous Response Mode  
Extended

SABM c Set Asynchronous Balanced Mode

SABME c Set Asynchronous Balanced Mode  
Extended

Die Befehle SNRM, SARM und SABM leiten eine Datenaustauschsitzung ein, sie erfordern eine Bestätigung durch UA-Meldungen. Die E-Varianten werden benutzt, um Nachrichtennumerierungen modulo 128 zu vereinbaren.

DISC c Disconnect

DISC beendet eine Sitzung. Eine Bestätigung durch eine UA-Meldung wird erwartet.

**DM**            r    **Disconnected Mode**

Mit einer DM-Meldung zeigt eine Sekundärstation an, daß sie nicht betriebsbereit ist.

**FRMR**        r    **Frame Reject**

Eine FRMR-Meldung berichtet den Grund, warum ein Rahmen nicht akzeptiert werden konnte.

Mögliche Gründe sind:

- ungültiges Kontrollfeld,
- unerlaubtes Informationsfeld,
- Informationsfeld überschreitet zulässige Länge,
- ungültiger Empfangszähler.

Im Normalfall erfolgt die Fehlererholung mittels eines Reset-Befehls.

**UA**            r    **Unnumbered Acknowledgement**

Meldung zu einem SET-Befehl.

**RSET**        c    **Reset**

Der Befehl RSET wird zur Fehlererholung genutzt; die Zähler N(R) und N(S) werden neu gesetzt.

**SIM**            c    **Set Initialization Mode**

Die Kontrollfunktionen werden in der adressierten Station ausgeführt.

**RIM**            r    **Request Initialization Mode**

Bitte um SIM-Befehl.

**RD**            r    **Request Disconnect**

Bitte um DISC-Befehl.

**UP**            c    **Unnumbered Poll**

Verhandlung über Kontrollzustand.

**XID**            c/r   **Exchange Identification**

XID-Befehle und Meldungen dienen der Vereinbarung eines HDLC-Profiles zwischen den beteiligten Stationen. Hierzu gehören unter anderem:

- die Adressierungsart,
- das Numerierungsschema,
- das CRC-Polynom,
- die maximale Sendelänge des Informationsfeldes,
- die maximale Empfangslänge des Informationsfeldes,
- die Sendefenstergröße,
- die Empfangsfenstergröße.

**TEST**        c/r   **Test**

Austausch identischer Felder für Testzwecke.

## HDLC Commands and Responses

Format	Control Field Bit Encoding								Commands	Responses
	1	2	3	4	5	6	7	8		
<b>Information</b>	0	N(S)	P	N(R)					I	I
<b>Supervisory</b>	1	0	0	0	P	N(R)			RR	RR
	1	0	0	1	P	N(R)			REJ	REJ
	1	0	1	0	P	N(R)			RNR	RNR
	1	0	1	1	P	N(R)			SREJ	SREJ
<b>Unnumbered</b>	1	1	0	0	P	0	0	0	UI	UI
	1	1	0	0	P	0	0	1	SNRM	
	1	1	0	0	P	0	1	0	DISC	RD
	1	1	0	0	P	1	0	0	UP	
	1	1	0	0	P	1	1	0		UA
	1	1	0	0	P	1	1	1	TEST	TEST
	1	1	1	0	P	0	0	0	SIM	RIM
	1	1	1	0	P	0	0	1		FRMR
	1	1	1	1	P	0	0	0	SARM	DM
	1	1	1	1	P	0	0	1	RSET	
	1	1	1	1	P	0	1	0	SARME	
	1	1	1	1	P	0	1	1	SNRME	
	1	1	1	1	P	1	0	0	SABM	
	1	1	1	1	P	1	0	1	XID	XID
	1	1	1	1	P	1	1	0	SABME	

### Legende:

I	Information		
RR	Receive Ready	RNR	Receive Not Ready
REJ	Reject	SREJ	Selective Reject
UI	Unnumbered Information	UP	Unnumbered Poll
UA	Unnumbered Acknowledgement		
XID	Exchange Identification	FRMR	Frame Reject
DISC	Disconnect	DM	Disconnected Mode
RIM	Request Initialization Mode		
SIM	Set Initialization Mode	RD	Request Disconnect
SNRM(E)	Set Normal Response (Extended) Mode		
SARM(E)	Set Asynchronous Response (Extended) Mode		
SABM(E)	Set Asynchronous Balanced Response (Extended) Mode		

## "Bit Stuffing":

Nach jeweils fünf aufeinanderfolgenden 1-Bit wird ein 0-Bit eingeblendet.

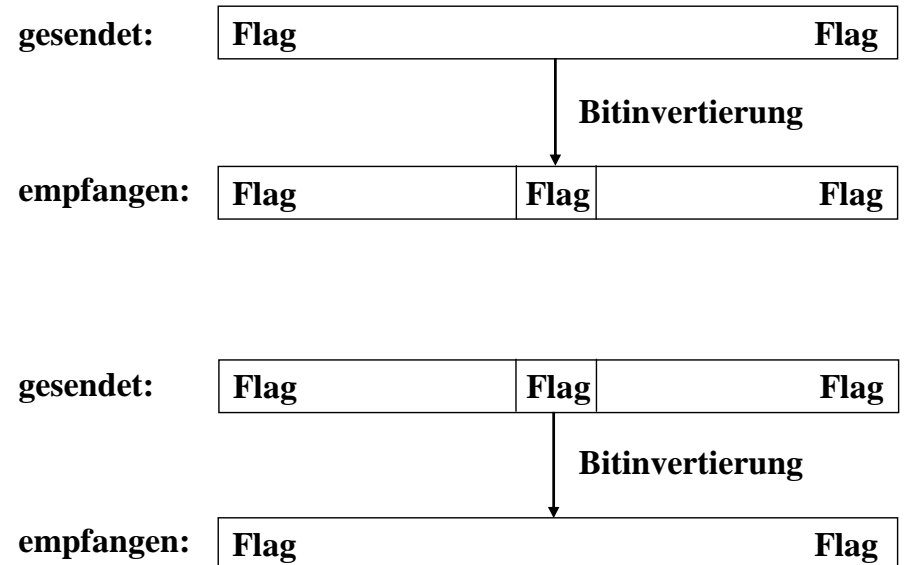
Urfolge:

1111111111101111101111110

nach Einfügen von Trennbit:

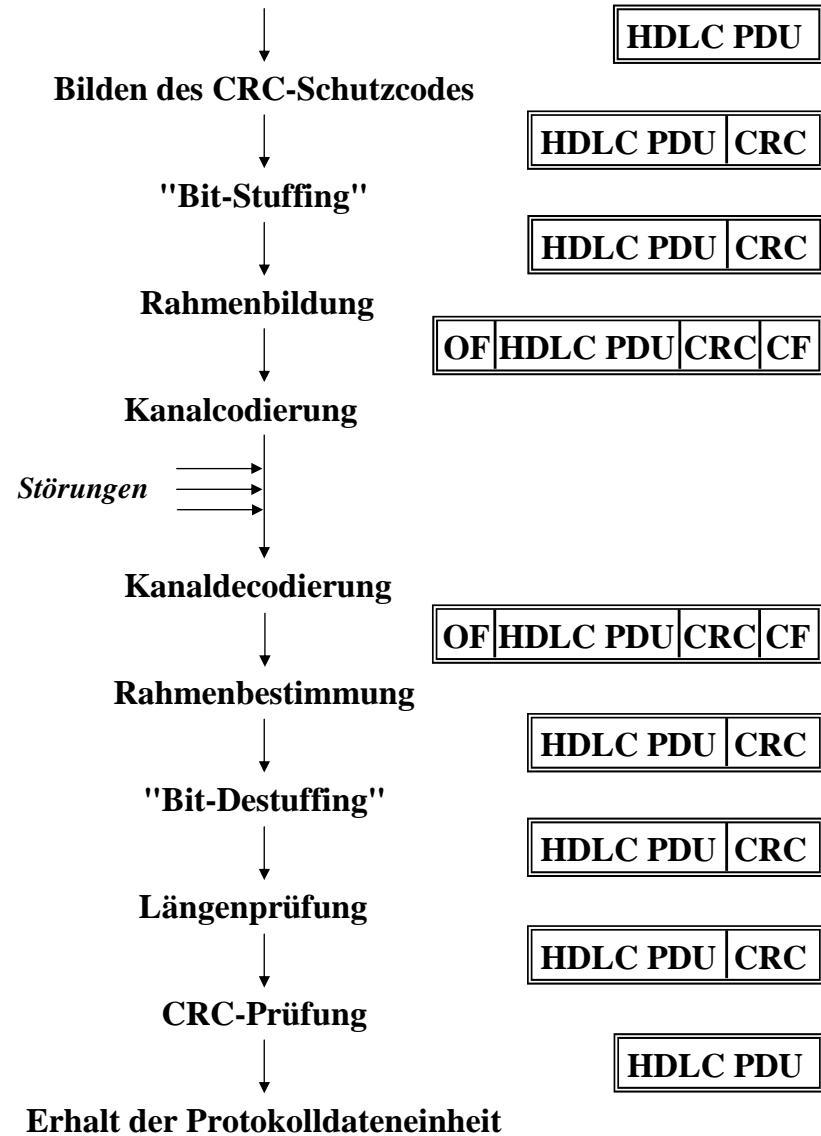
1111101111101101111101011111010

Mögliche Fehler bei Bitinvertierung:

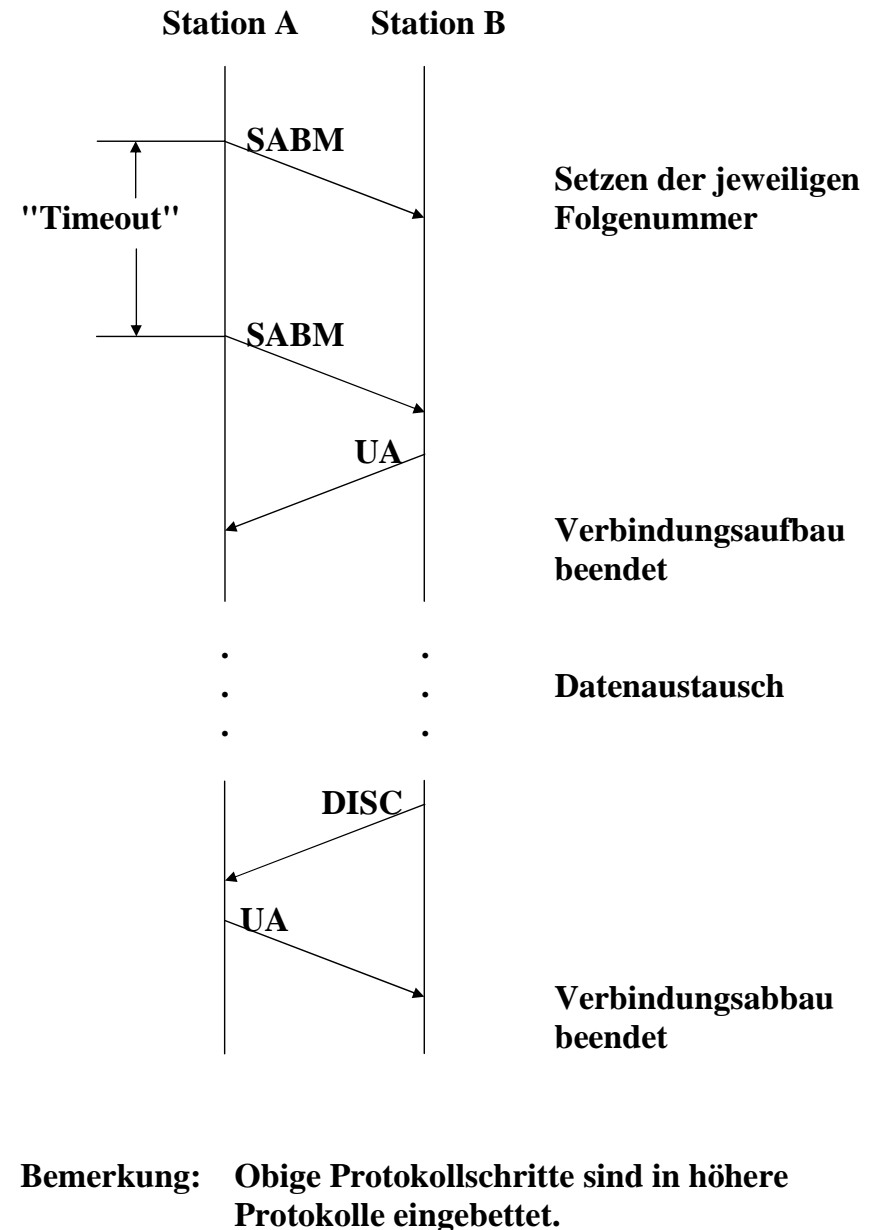


## Ablauf einer Datenübertragung unter HDLC:

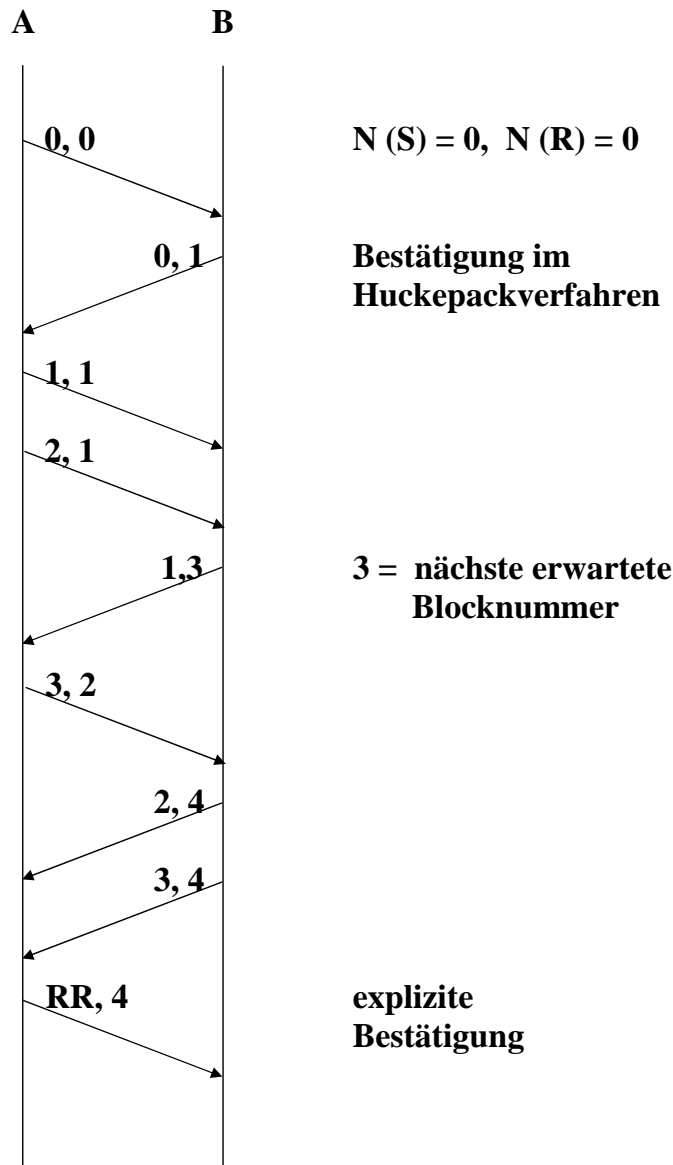
### Bildung der Protokolldateneinheit



## Beispiel 1 zu Protokollablauf:

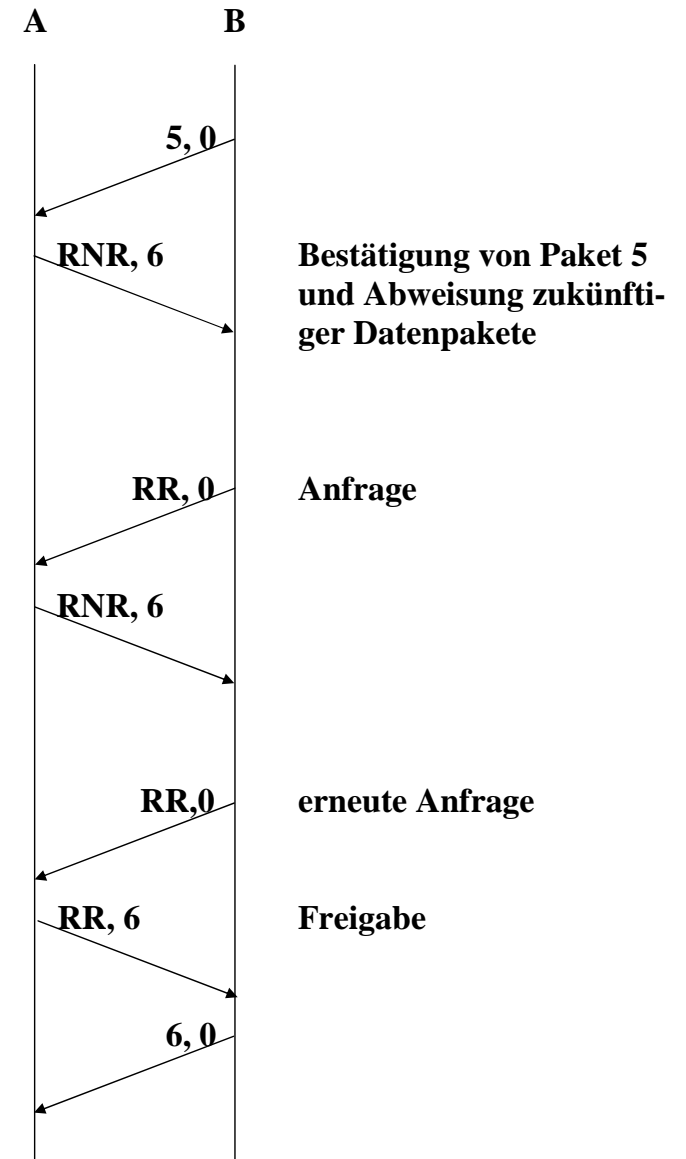


**Beispiel 2 zu Protokollablauf: Datenübermittlung**



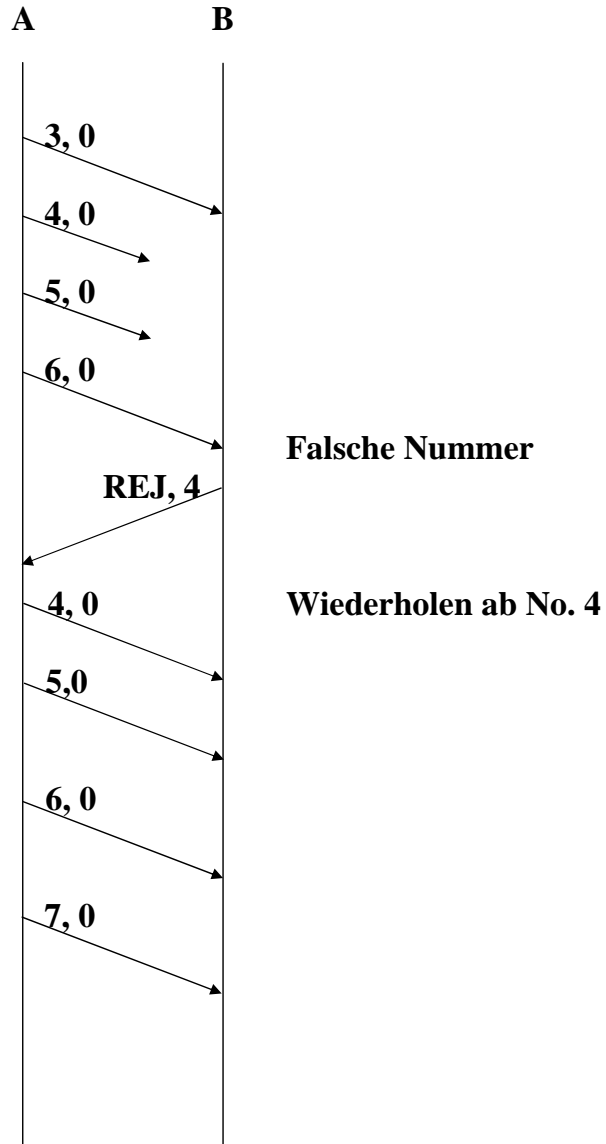
**Beispiel 3 zu Protokollablauf:**

**Vorübergehender Stopp der Datenübermittlung**



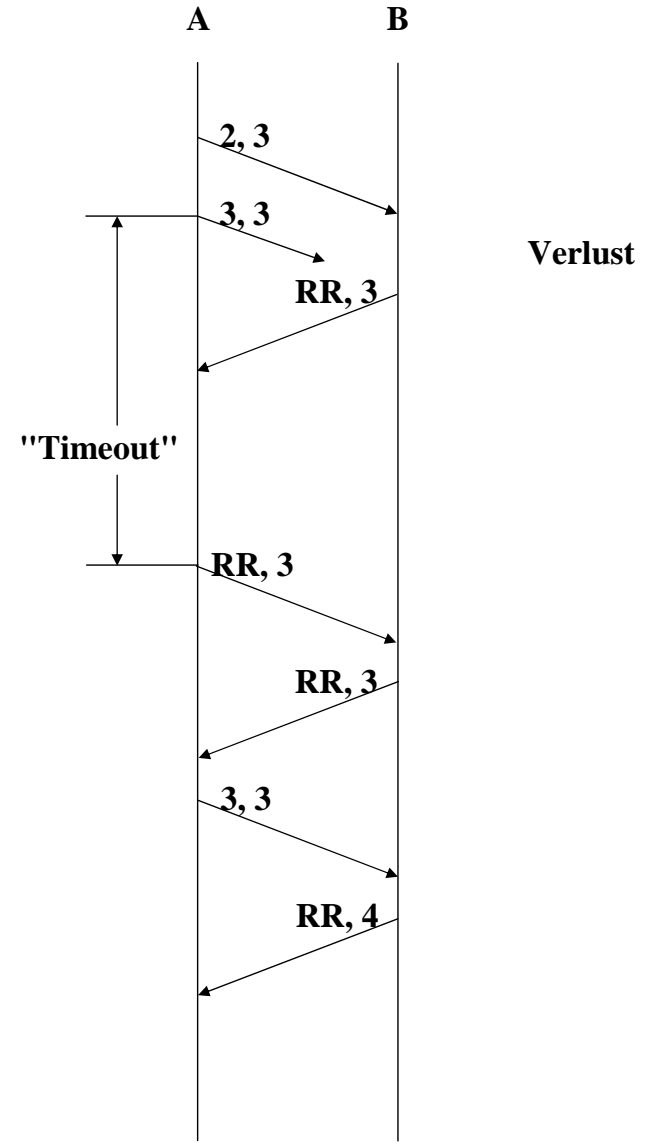
**Beispiel 4 zu Protokollablauf:**

**Verlust eines Datenpakets**

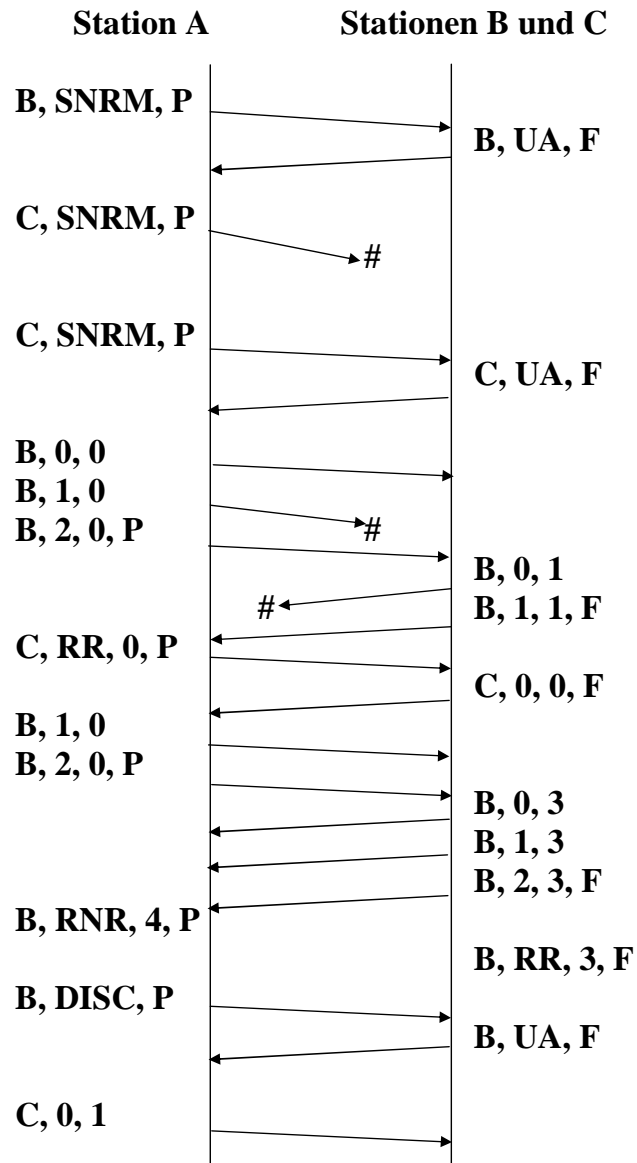


**Beispiel 5 zu Protokollablauf:**

**Verlust eines Datenpakets, "Timeout"-Mechanismus**

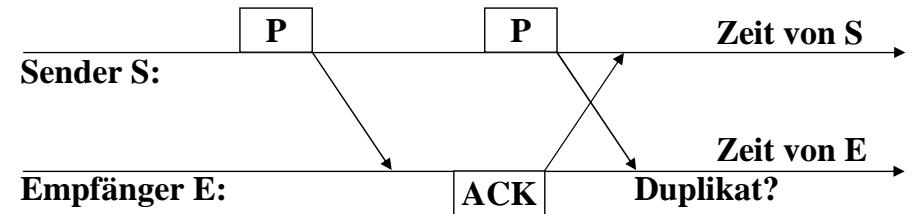


### Beispiel 6 zu Protokollablauf: Sekundärstationen

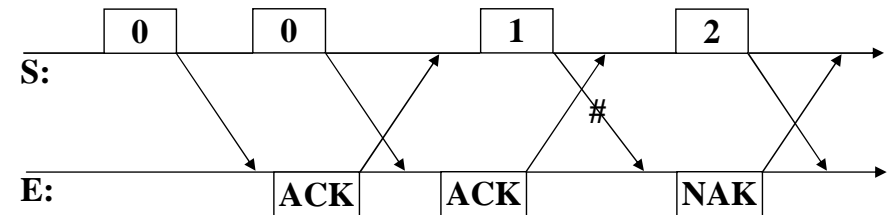


### ARQ-Verfahren:

#### Beispiel: nichtnumerierte Pakete:

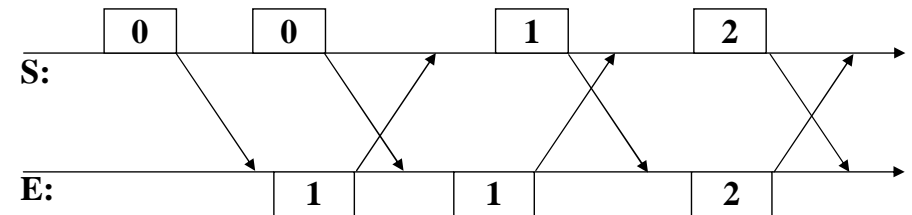


#### Beispiel: nichtnumerierte Bestätigungen:



Der Sender kann nicht unterscheiden, ob die zweite Quittung das Duplikat 0 oder das Paket 1 bestätigt.

#### Beispiel: numerierte Pakete und Bestätigungen:



Bemerkung: ARQ = Automatic Repeat Request

## ARQ-Verfahren:

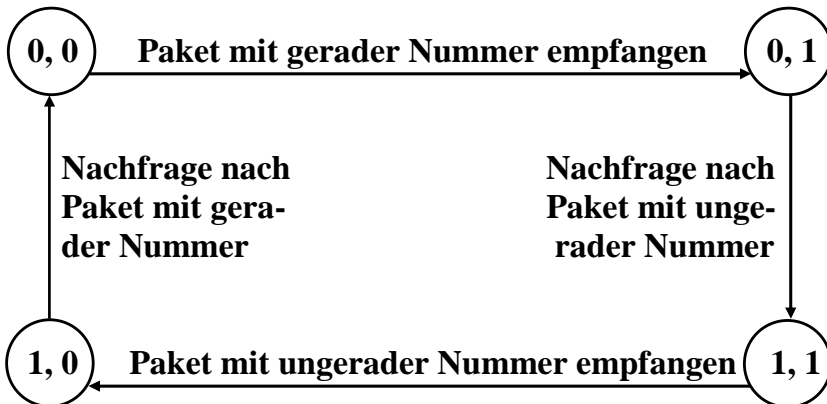
### Algorithmus im Sender:

1. Setze Sendezähler  $SN = 0$ .
2. Gib nächstem Paket die Nummer  $SN$ .
3. Sende Paket mit Nummer  $SN$ .
4. Trifft eine Bestätigung mit Nummer  $RN > SN$  ein, dann setze  $SN = RN$  und gehe zu 2.  
Trifft keine Bestätigung für  $SN$  nach angemessener Wartezeit ein, dann gehe zu 3.

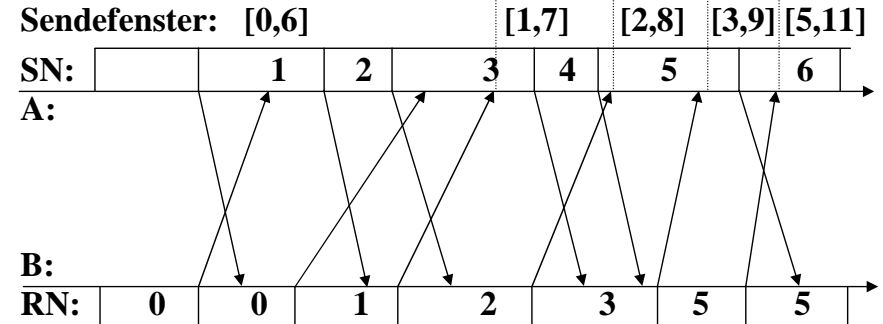
### Algorithmus im Empfänger:

1. Setze Empfangszähler  $RN = 0$ .
2. Wiederhole ad infinitum:  
Trifft ein Paket mit Nummer  $SN = RN$  ein, dann erhöhe  $RN$ , sende Bestätigung mit Nummer  $RN$ .

Als Zählvariable für Sendenummer und Bestätigungsnummer genügt jeweils eine 1-Bit-Variable. Den Zustandsraum bildet das Zahlenpaar  $(SN \bmod 2, RN \bmod 2)$ .



## GO-Back-n-ARQ:



### Algorithmus (Sendeteil):

1. Setze  $SN_{\min} = 0$  und  $SN_{\max} = 0$ .
2. Wiederhole Schritte 3, 4, 5 in beliebiger Reihenfolge:
3. Falls  $SN_{\max} < SN_{\min} + n$ , gib nächstem Paket die Nummer  $SN_{\max}$ , erhöhe  $SN_{\max}$  um 1, sende Paket.
4. Wird ein Paket mit der Quittungsnummer  $RN > SN_{\min}$  fehlerfrei empfangen, dann setze  $SN_{\min} = RN$ .
5. Trifft keine Quittung für  $SN_{\min}$  in angemessener Frist ein, dann wiederhole  $SN_{\min}$  oder ein anderes Paket  $SN$  aus dem Intervall  $SN_{\min} \leq SN < SN_{\max}$ .

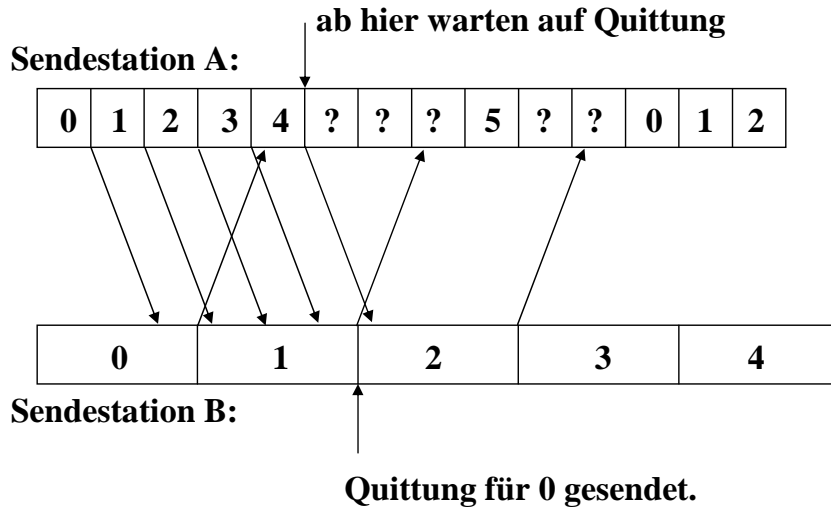
### Algorithmus (Empfangsteil):

1. Setze Zähler  $RN = 0$ .
2. Wiederhole ad infinitum:
  - a) Trifft ein Paket mit Nummer  $SN = RN$  ein, dann erhöhe  $RN$  um 1.
  - b) Sende Bestätigung mit Nummer  $RN$ .

**Ineffizienzen bei Go-Back-n-ARQ:**

**(a) unterschiedliche Rahmengrößen in den Einzelrichtungen:**

**Beispiel: Numerierung modulo 6,  
Rahmengröße von A nach B = 1,  
Rahmengröße von B nach A = 3.**



**Abhilfe: Vergrößerung des Nummernraumes.**

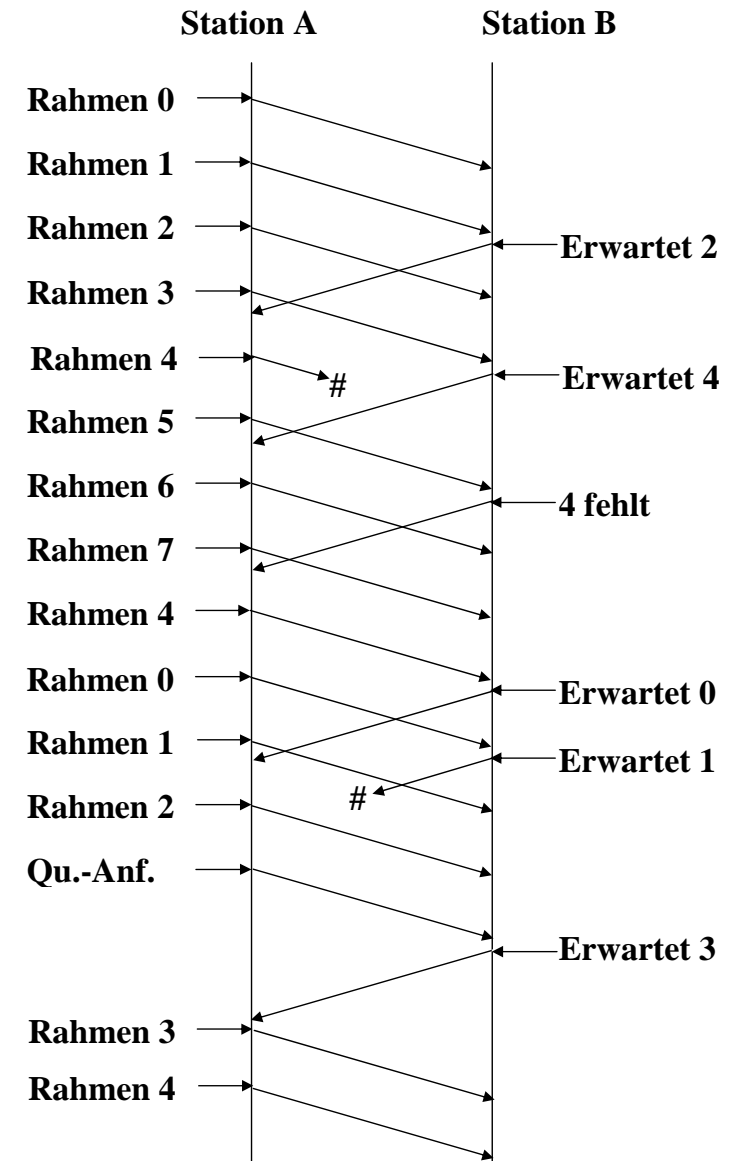
**(b) Fehler in Quittierungsrichtung:**

**Abhilfe: Vergrößerung des Nummernraumes**

**(c) Fehler in der Senderichtung:**

**Abhilfe: Geschickte Wahl des Timeout-Intervalls**

**Beispiel zum selektiven ARQ-Verfahren:**



## Gleitende Fenster:

Der Algorithmus der gleitenden Fenster benutzt zwei Fenster, ein Sendefenster und ein Empfangsfenster. Er dient drei Belangen:

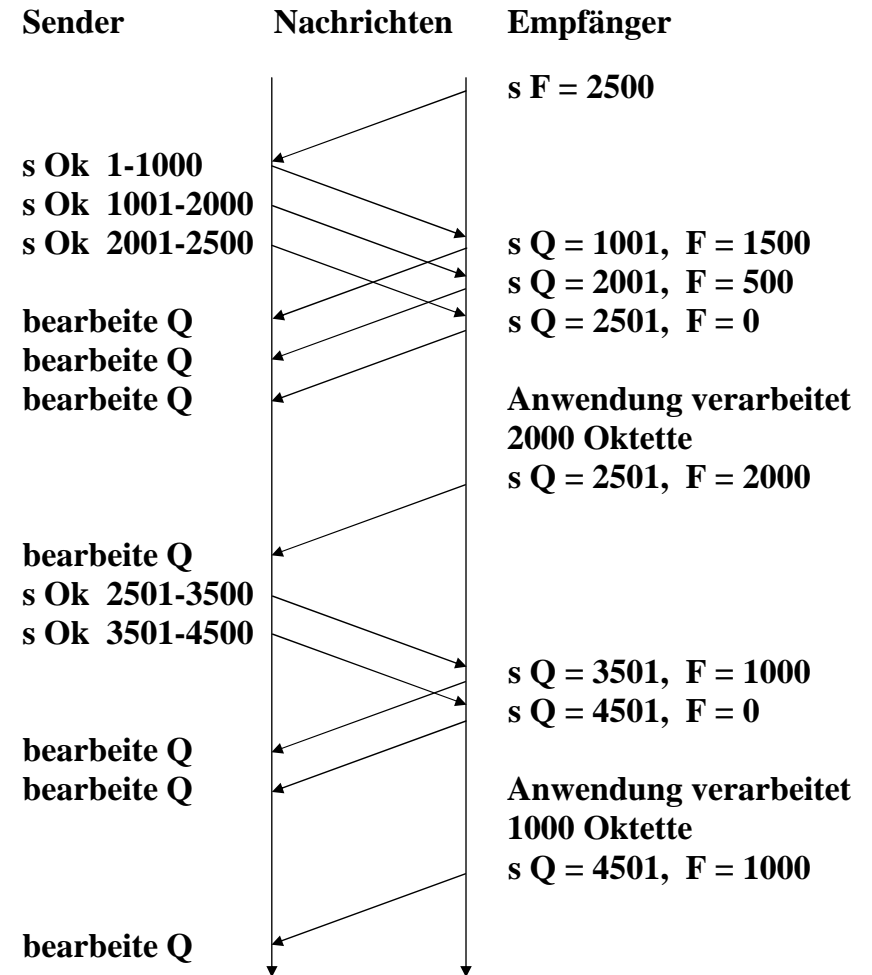
1. der Fehlerkontrolle,
2. der reihenfolgerechten Ablieferung,
3. der Flußkontrolle.

**Ad 3:** Die Empfangsfenstergröße wird vom Sender als Sendeerlaubnisse interpretiert. Durch dynamische Veränderungen der Empfangsfenstergröße steuert der Empfänger den Datenfluß. So führt eine Empfangsfenstergröße von 0 zum Erliegen jeder Sendetätigkeit.

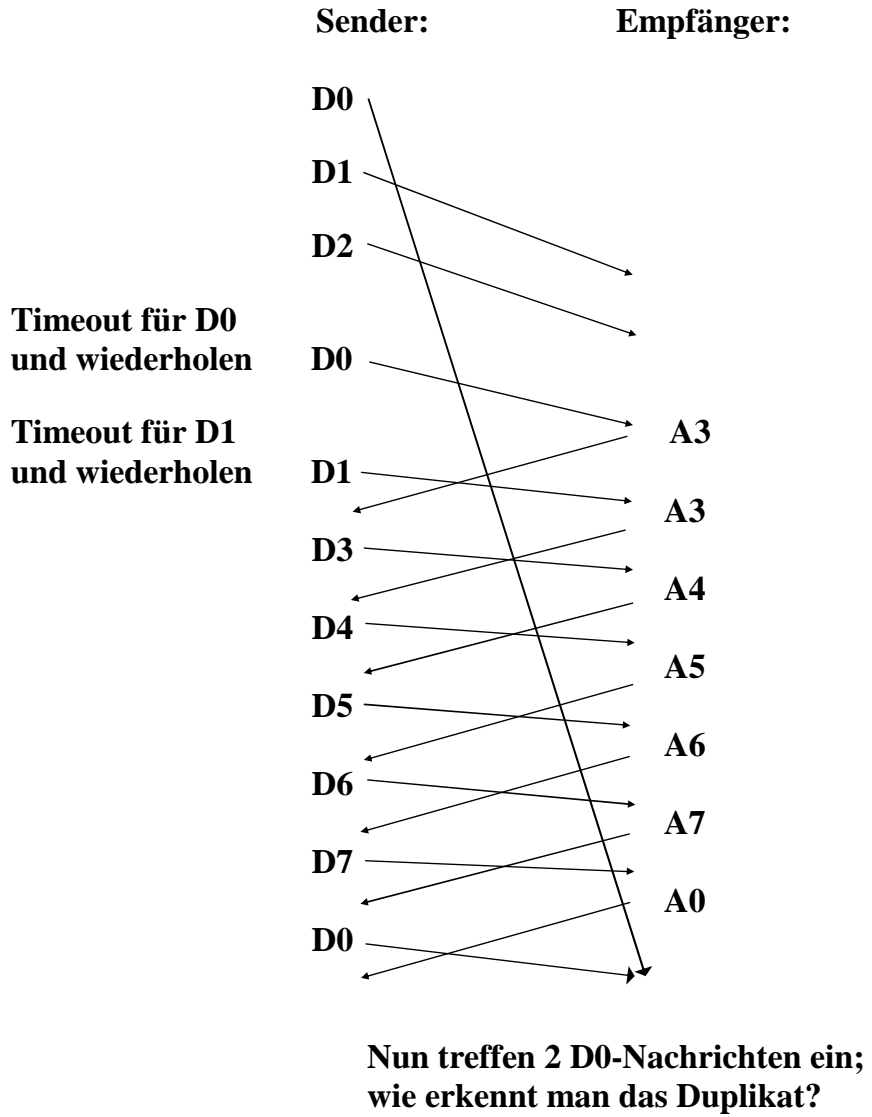
**Bemerkung:** Die Empfangsfenstergröße E, die Sendefenstergröße S und die Größe des Nummernraumes N beeinflussen die Effizienz der Datenübertragung. Alle drei Größen sind auf die mögliche Zahl der Nachrichten in Transit abzustimmen. Diese Zahl wird durch das Produkt aus Datenrate und Übertragungszeit bestimmt.

## Beispiel zur Fensterverwaltung:

**Bemerkung:** s steht für sende,  
F für Empfangsfenstergröße,  
Ok für Oktett,  
Q für Quittung.

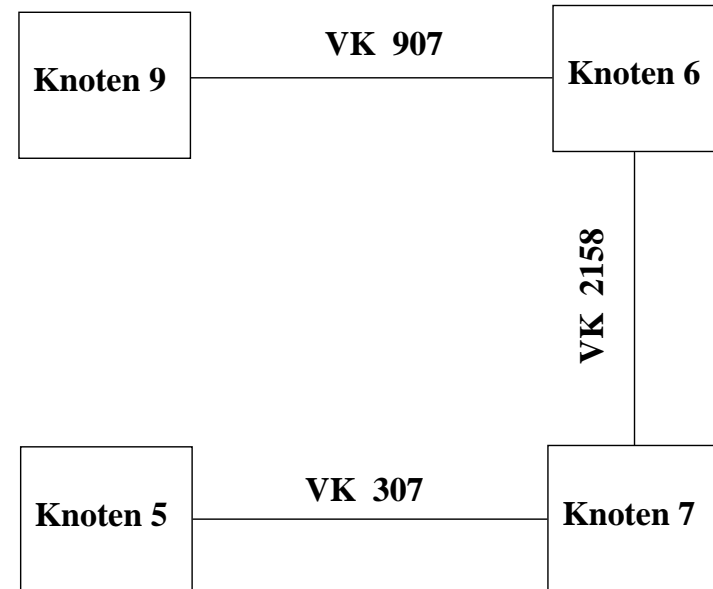


**Problem: Zerstörung nicht rechtzeitig eintreffender Nachrichten**



**Virtueller Kanal:**

**Bemerkung:** In einem Kanal können Nachrichten nicht überholt werden. Dies ist z. B. wichtig bei der Verschlüsselung von Nachrichtenfolgen.



Der virtuelle Kanal zwischen Knoten 5 und Knoten 9 trägt auf jedem Teilabschnitt eine andere Bezeichnung, hier 307, 2158, 907. Die Umbenennung wird in Tabellen durchgeführt. Für beide Richtungen wird jeweils die gleiche Bezeichnung verwendet.

## Transparenz:

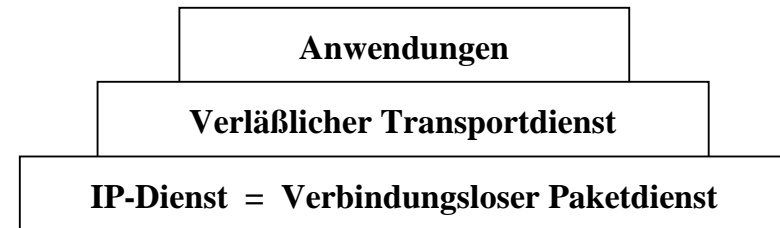
Man nennt ein Übertragungsprotokoll transparent, falls es in der Lage ist, jede Datenfolge zu übertragen.

Bei den HDLC-Protokollen erreicht man Transparenz, indem man durch das Einblenden zusätzlicher Bit verhindert, daß Rahmenbegrenzer innerhalb eines Rahmens auftreten können.

Bei den BSC-Leitungsprotokollen leitet die Umschaltsequenz DLE STX einen reinen Textmodus ein, die Steuerzeichen verlieren innerhalb des Textmodus bis auf das Zeichen DLE ihre Sonderbedeutung. Der Textmodus wird beendet durch das Auftreten von DLE ETX oder DLE ETB, das Zeichen DLE wird durch Verdopplung DLE DLE codiert. Außer den Zeichen DLE, ETX, ETB, SYN darf im Textmodus kein anderes Zeichen auf DLE folgen.

Eine weitere Möglichkeit, Transparenz herzustellen, besteht in der Verwendung von Zählern. Zum Beispiel gibt man an einer festen Position im Rahmenkopf an, wie lang das Feld der Nutzdaten ist.

## Bemerkungen zum TCP-Dienst:



Der TCP-Dienst ist ein Nutzer des IP-Dienstes.

Der IP-Dienst wird charakterisiert als ein "best effort"-Dienst; mit anderen Worten, der IP-Dienst ist unzuverlässig. Im einzelnen bedeutet dies:

- Ein Datagramm kann verloren gehen.
- Ein Datagramm kann verfälschte Daten liefern.
- Datagramme können einander überholen.
- Datagramme können dupliziert werden.

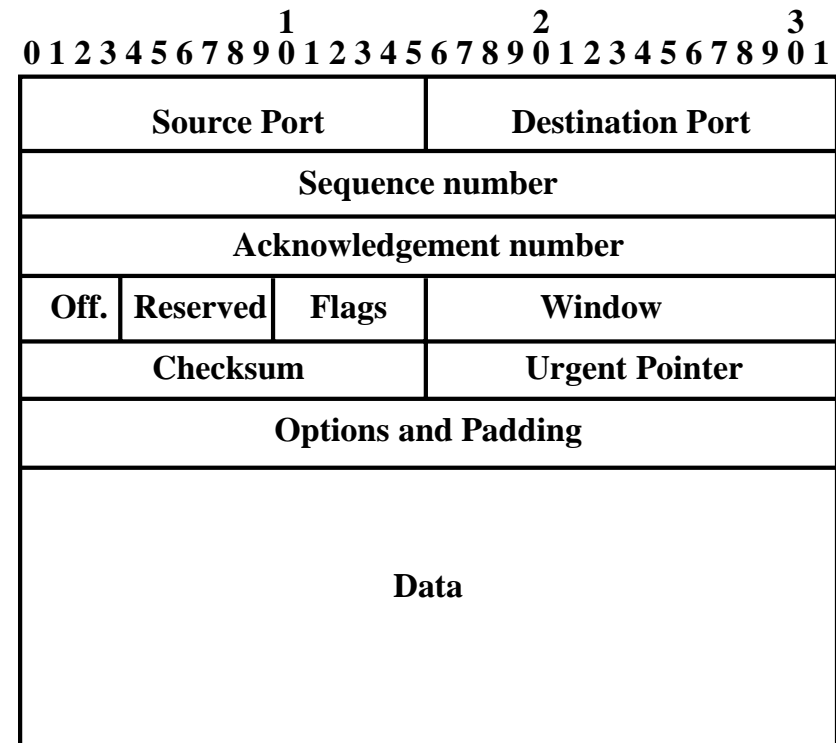
Es ist die Aufgabe der Nutzer des IP-Dienstes, durch geeignete Maßnahmen die Unzulänglichkeiten des IP-Dienstes zu beheben.

Beim Empfang eines Datagramms können Plausibilitätsprüfungen durchgeführt werden, wie Überprüfung der korrekten Kopflänge, der korrekten Datagrammlänge, der IP-Versionsnummer, der Protokollnummer, der Adreßklassen; zum Schutz der Integrität des Kopfes eines Datagramms dient eine Prüfsumme.

### Eigenschaften von TCP:

- (1) TCP ist ein universelles Datenaustauschprotokoll, je zwei Prozesse in beliebigen Rechnern können miteinander kommunizieren.
- (2) TCP erstellt eine virtuelle Verbindung zwischen den Endteilnehmern. Diese Verbindung verwaltet zwei getrennte Zeichenströme, Teile des Zeichenstroms werden zwischengespeichert, ein TCP-Segment wird in der Regel erst dann abgesandt, falls der vorgesehene Puffer gefüllt ist.
- (3) In TCP erfolgt die Fehlerkontrolle Ende zu Ende, daher kann jedes beliebige Transportnetz überbrückt werden.
- (4) Ein Fenstermechanismus ermöglicht die Flußkontrolle, getrennt für jede Richtung.
- (5) Der Zeichenstroms wird reihenfolgetreu beim Endnutzer abgeliefert, die Transporteinheiten sind weitgehend unabhängig von den vom Sender gelieferten Datenabschnitten.
- (6) Die Anpassung des Retransmissionsintervalls an das Verkehrsaufkommen im Netz ermöglicht eine rudimentäre Staukontrolle.
- (7) Ein "Urgent"-Modus fordert die sofortige Benachrichtigung des Endempfängers, dies ist wichtig, um einen zeitgerechten Abbruch unnötiger Aktivitäten des Partnerprozesses zu veranlassen.

### Transmission Control Protocol Segment:



### Erläuterung zu den einzelnen Feldern:

**Source port (16 Bit):** Identifikator des Senders in einem Rechensystem.

**Destination port (16 Bit):** Identifikator des Empfängers in einem Rechensystem.

**Sequence number (32 Bit):** Folgennummer des ersten Oktetts im Segment, außer wenn das SYN-Bit gesetzt ist, in diesem Fall handelt es sich um die Initialisierungsfolgennummer.

**Acknowledgement number (32 Bit):** Folgennummer des nächsten von der Gegenseite erwarteten Oktetts.

**Off. = Data offset (4 Bit):** Zahl der 32-Bit-Einheiten des Kopfes

**Reserved (6 Bit):** Reserviert für zukünftige Erweiterungen.

**Flags (6 Bit) = Anzeigen:**

**URG = Urgent Pointer gültig.**

**ACK = Acknowledgement gültig.**

**PSH = Umgehende Weitergabe aller bisher erhaltenen Daten an die Anwendung**

**RST = Rücksetzen der Verbindung.**

**SYN = Synchronisiere Folge Nummern und eröffne TCP-Verbindung.**

**FIN = Keine weiteren Daten vom Sender.**

**Window (16 Bit):** Durch den Sender eingeräumter Kredit in Oktetten.

**Checksum (16 Bit):** Das Einerkomplement der Summe modulo  $2^{16}-1$  aller 16-Bit-Einheiten des Segments und eines Pseudokopfes

**Urgent Pointer (16 Bit):** Zeigt unmittelbar auf das erste Oktett nach den wichtigen Daten.

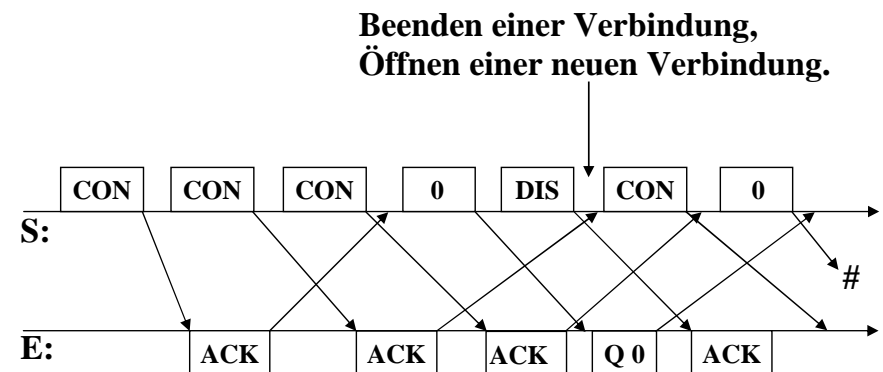
**Options (variable Bitzahl):** Im RFC 793 ist nur eine Option definiert, die maximale Segmentgröße.

**Aufbau einer Verbindung:**

Zunächst erscheint das Problem des Aufbaus einer Verbindung trivial. Der Initiator sendet eine Aufbaunachricht und wartet auf ihre Bestätigung.

Station A sendet Verbindungswunsch an Station B. Station B antwortet mit Bestätigung. Ab jetzt steht eine initialisierte Verbindung zum Datenaustausch zur Verfügung.

Ein erstes Problem stellt die Wahl eines Timeout-Intervalls dar. Folgender Ablauf ist möglich:



In diesem Fall wird wegen großer Zeitverzögerungen bei E die Nachricht 0 der zweiten Verbindung, die auf dem Übertragungsweg verloren geht, durch die Quittung zur Nachricht 0 der ersten Verbindung von S als bestätigt erkannt.

### Dreifaches Händeschütteln zur Authentifikation:

**Aufgabe:** Zwei Partner A und B möchten über ein Netz verschlüsselte Nachrichten austauschen. Es sei angenommen, sie haben vorher schon einen gemeinsamen geheimen Schlüssel GS vereinbart.

### Verfahren:

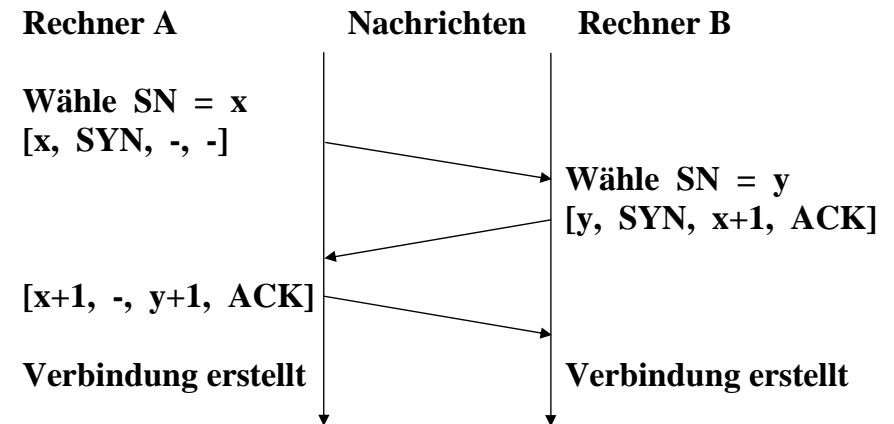
- (1) A wählt Zufallszahl  $x$ , bildet verschlüsselte Nachricht  $V(x, GS)$  und sendet sie zusammen mit seinem Namen  $IdA$  an B.
- (2) B empfängt die Nachricht  $V(x, GS)$  und  $IdA$ . Er wählt den zugehörigen Schlüssel  $GS$ , entschlüsselt  $V$  und bildet  $x+1$ . Anschließend wählt er eine Zufallszahl  $y$  und sendet die verschlüsselte Nachricht  $W(x+1, y, GS)$  an A.
- (3) A empfängt  $W$ , entschlüsselt  $x+1$  und  $y$  und antwortet mit  $U(y+1, GS)$ . Die Teilantwort  $x+1$  bestätigt A, daß er wirklich mit B kommuniziert.
- (4) Der Empfang der verschlüsselten Summe  $y+1$  bestätigt B, daß A der ist, der er vorgibt zu sein. Nun sendet er an A einen für die eigentliche Kommunikation gewählten Sitzungsschlüssel  $SI$ , um die Möglichkeit der Kompromittierung des Schlüssels  $GS$  zu mindern.

### Aufbau einer TCP-Verbindung (1):

TCP benutzt zur Eröffnung einer Verbindung im Normalfall einen Austausch von drei Nachrichten - "3 way handshake".

Fall 1: Rechner A initiativ, Rechner B reaktiv

Benutztes Nachrichtenformat: [Sequenznummer, SYN-Bit, Quittungsnummer, ACK-Bit]

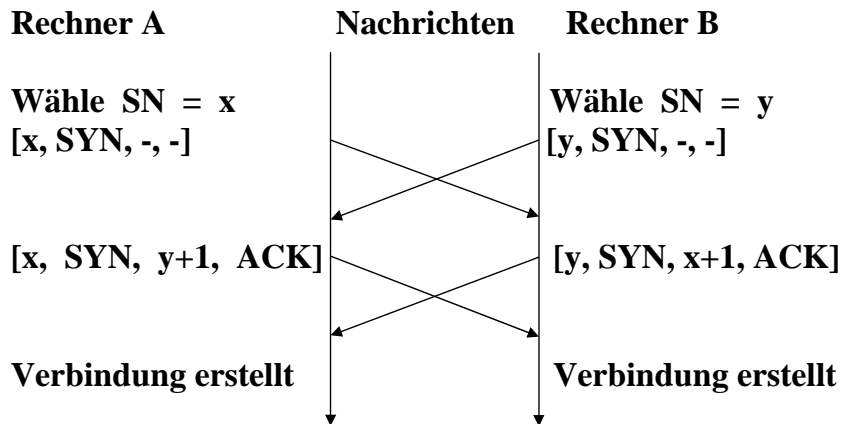


**Bemerkung:** Die Sequenznummern  $x$  und  $y$  sind sorgfältig zu wählen, insbesondere ist zu verhindern, daß Segmente einer geschlossenen Verbindung mit Segmenten einer aktiven Verbindung verwechselt werden können. Dies erreicht man, indem Sequenznummern zufällig gewählt oder aus der Uhrzeit abgeleitet werden. SYN-Segmente verbrauchen eine Sequenznummer.

## Aufbau einer TCP-Verbindung (2):

Fall 2: Rechner A und B eröffnen gleichzeitig eine Verbindung.

Benutztes Nachrichtenformat: [Sequenznummer, SYN-Bit, Quittungsnummer, ACK-Bit]



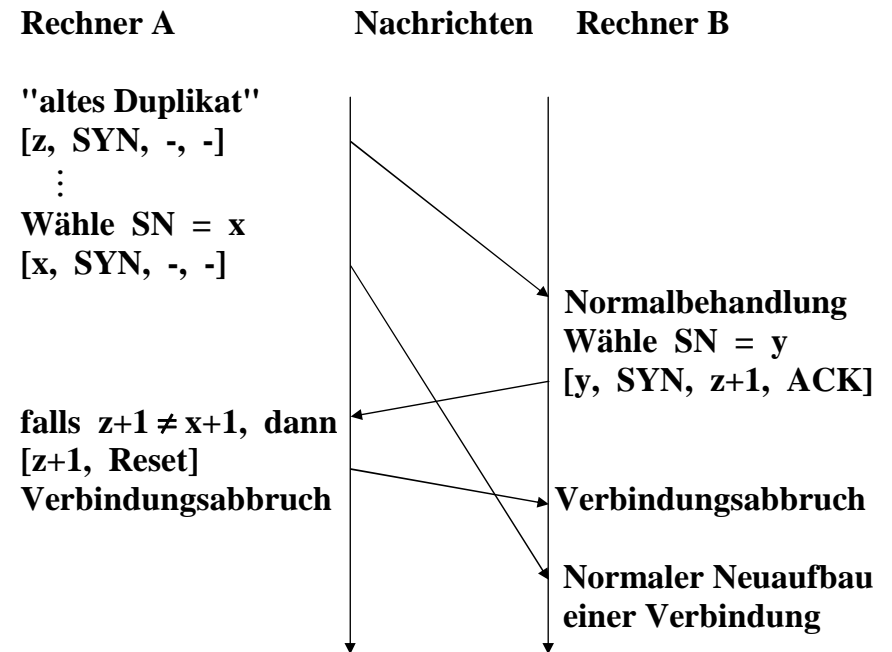
**Bemerkung:** Der Fall, daß beide Partner eines TCP-Dialoges gleichzeitig versuchen, eine Verbindung aufzubauen, tritt sehr selten auf. Benötigt wird dieses Szenario zu Testzwecken für die Erstellung einer Kurzschlußverbindung. Obige Nachrichtenfolge wird im Automaten zur Verwaltung von TCP-Verbindungsautomaten nicht "direkt" berücksichtigt.

## Aufbau einer TCP-Verbindung (3):

Fall 3: Ein altes SYN-Segment stört den Aufbau einer Verbindung.

Benutztes Nachrichtenformat:

- (i) [Sequenznummer, SYN-Bit, Quittungsnummer, ACK-Bit]
- (ii) [Sequenznummer, Reset]



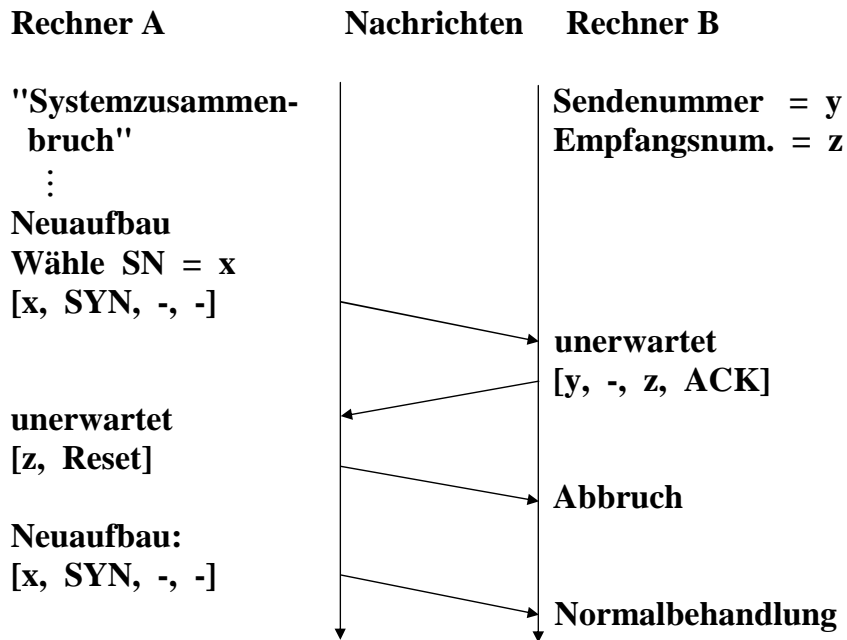
**Bemerkung:** Im Beispiel wurde angenommen, daß die Nachricht [x, SYN, -, -] nach erfolgreichem Verbindungsabbruch im Rechner B eintrifft. Eine falsche SYN-Nachricht führt immer zum Abbruch.

## Aufbau einer TCP-Verbindung (4):

**Fall 4: Versuch des Neuaufbaus einer Verbindung nach einem Systemzusammenbruch.**

**Benutztes Nachrichtenformat:**

- (i) [Sequenznummer, SYN-Bit, Quittungsnummer, ACK-Bit]
- (ii) [Sequenznummer, Reset]



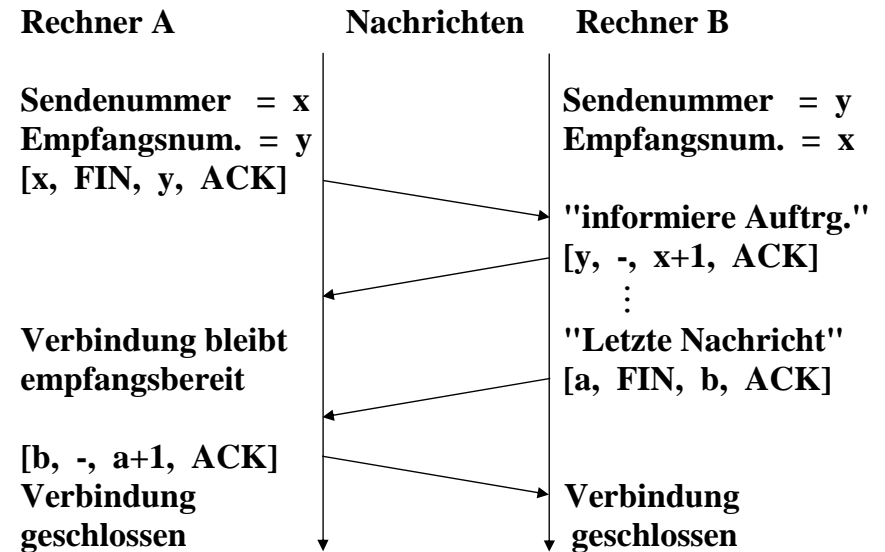
**Bemerkung:** Während eines normalen Datenaustausches wird keine Initialisierungsnachricht erwartet. Rechner B antwortet durch Bekanntgabe der nächsten erwarteten Sequenznummer.

## Abbau einer TCP-Verbindung (1):

**Der Abbau einer TCP-Verbindung folgt einem dem Aufbau ähnlichen Protokoll. Der Normalfall wird im Diagramm dargestellt.**

**Fall 1: Rechner A initiativ, Rechner B reaktiv**

**Benutztes Nachrichtenformat:** [Sequenznummer, SYN-Bit, Quittungsnummer, ACK-Bit]

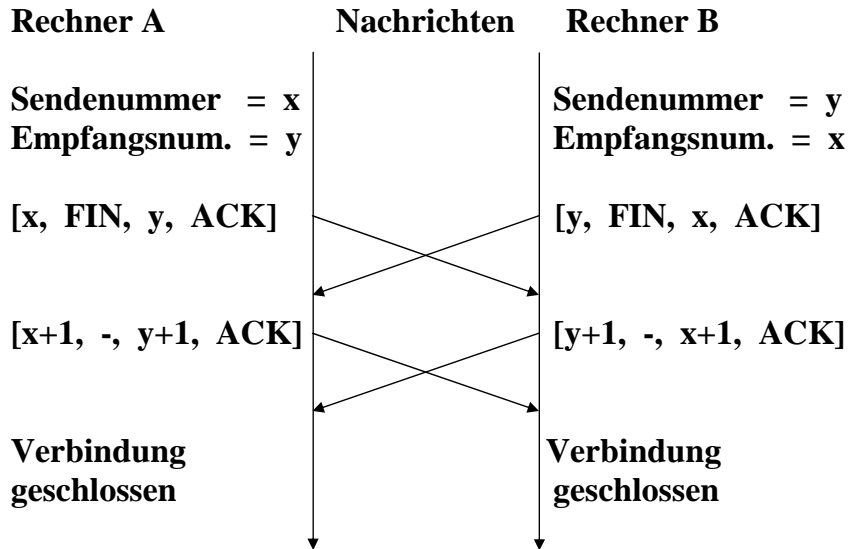


**Bemerkung:** Bevor Rechner A die Verbindung schließt, wartet er nach dem Absenden der letzten Quittung noch die Zeitdauer 2 MSL (Maximum Segment Lifetime).

## Abbau einer TCP-Verbindung (2):

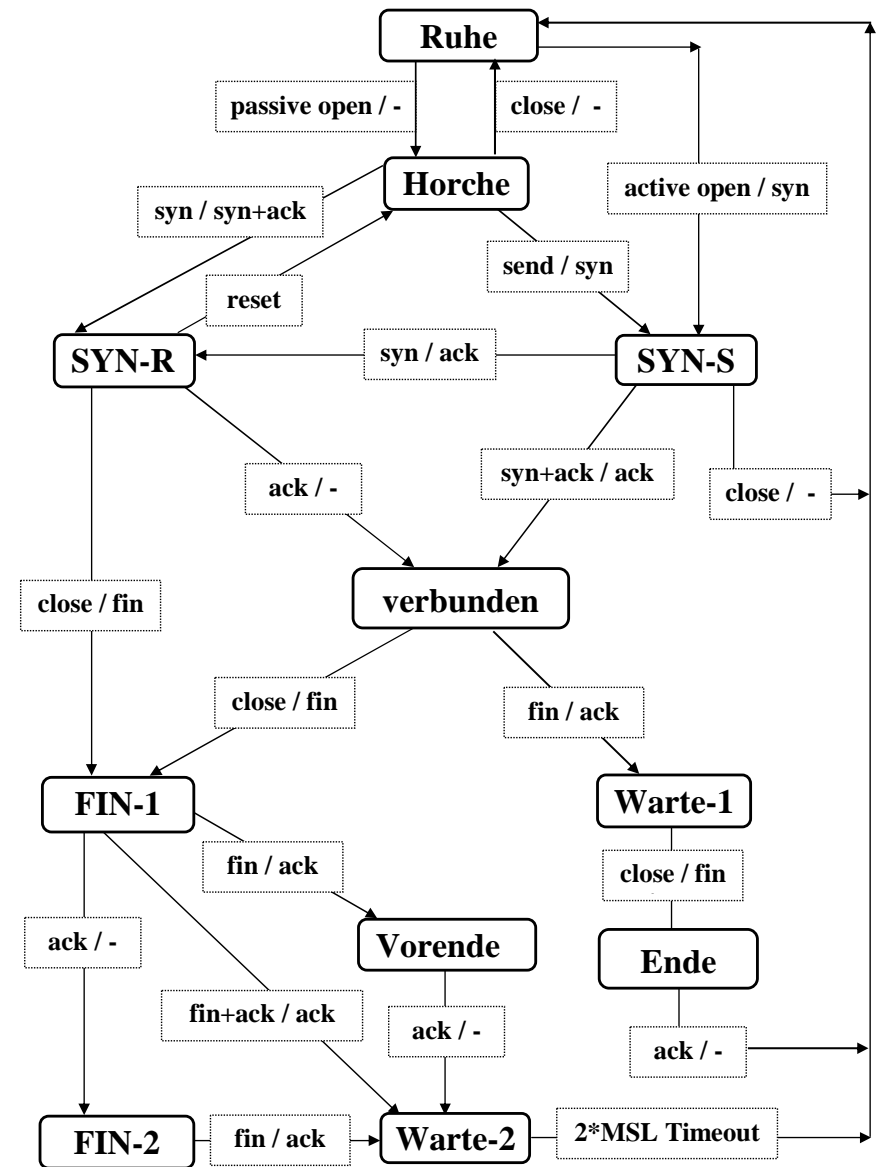
Fall 2: Rechner A und B schließen gleichzeitig eine Verbindung.

Benutztes Nachrichtenformat: [Sequenznummer, SYN-Bit, Quittungsnummer, ACK-Bit]



**Bemerkung:** Auch in diesem Fall warten beide Partner nach dem Eintreffen der Bestätigung für den Verbindungsabbau noch 2 MSL, bevor die Verbindung endgültig geschlossen wird.

## Informeller TCP-Automat (unvollständig):



**Bemerkung:** MSL = Maximum Segment Lifetime

## Staukontrolle:

Tritt ein Stau in den Zwischensystemen auf, dann führt das zum nochmaligen Senden von Paketen. Diese zusätzlichen Pakete verstärken den Stau. Die von TCP zur Staumilderung eingesetzten Mechanismen sind "multiplicative decrease" und "slow start".

Neben dem Fenster für die unbestätigt ausstehenden Oktette verwaltet der Sender auch ein Staufenster. Die Größe des Staufensters wird in Abhängigkeit vom Quittungsfenster initialisiert.

"Multiplicative decrease": Trifft für ein Segment die Quittung nicht im vorgesehenen Zeitintervall ein, dann nimmt TCP an, daß das Segment aufgrund von Netzüberlastung vernichtet wurde. Daraufhin wird die Zahl der Segmente im Staufenster halbiert und die Wartezeit bis zur Retransmission verdoppelt. Gehen mehrere Segmente in Folge verloren, dann befindet sich schließlich nur noch ein Segment im Staufenster.

"Slow start": Nach Empfang einer Quittung wird das Staufenster um 1 vergrößert. Dies führt, anders als man zunächst vermutet, zu einem exponentiellen Wachstum, denn aufeinanderfolgende Fenstergrößen in ähnlichen Zeitintervallen sind: 1, 2, 4, 8, ... Um nun die Wachstumsrate zu verlangsamen, wird, nachdem die halbe Fenstergröße vor Staueintritt erreicht wurde, nur noch dann die Fenstergröße um 1 erhöht, nachdem alle Segmente eines Fensters bestätigt wurden. Dies ist nur noch ein lineares Wachstum.

## Adaptive Timeout-Intervalle:

**Problem:** Sinnvolle Wahl der Wartedauer auf eine Bestätigung: Eine zu kurze Zeitspanne führt zu unnötiger zusätzlicher Last, eine zu lange führt zu ineffizienter Ressourcennutzung. In Weitverkehrsnetzen sollte man die starken Schwankungen der Übertragungszeiten berücksichtigen.

### Ursprünglicher Algorithmus für TCP:

- (i) Adaptive Anpassung der Schätzung SRTT der Zeitspanne RTT zwischen Absenden eines Segmentes und Empfang der Quittung:

**Formel:**  $SRTT = \alpha * SRTT + (1 - \alpha) * Mrtt$ ,  
mit  $Mrtt$  = jüngster Meßwert für RTT,  
 $\alpha$  = Anpassungsfaktor ( $0 < \alpha < 1$ ).

- (ii) Das Timeout-Intervall TOI wählt man etwas größer als SRTT.

**Formel:**  $TOI = \beta * SRTT$ ,  
mit  $\beta > 1$  frei wählbarer Vergrößerungsfaktor.

**Bemerkung:** Es wurde empfohlen, für  $\alpha$  Werte nahe 1 und für  $\beta$  Werte nahe 2 zu wählen. Der ursprüngliche Algorithmus hat eine Reihe von Mängeln, er wurde mehrfach verbessert. Ein Meßproblem stellt im Falle von Retransmissionen die Zuordnung einer Quittung zur quitierten Nachricht dar. Eine Verbesserung stammt von Karn.

**Zur Zuordnung von Bestätigung und Nachricht:**

**Beispiele fehlerhafter Zuordnung von Bestätigung und Nachricht:**

