6. Grundlegende Protokollmechanismen

6.1. Prüfsummen/CRC Fehlersicherung

- Ursachen von Übertragungsfehlern:
  - physikalische Störungen im Übertragungskanal,
  - Ausser Tritt geraten der Taktsynchronisierung,
  - HW- & SW-Fehler in den Partner-Stationen.
  - Zugriffskollisionen im LAN,
  - Fehler in Drittstationen,
  - Überlastung des Empfängers ("Overrun"),
  - Überlastung des Senders ("Underrun"),
  - Absichtlicher Abbruch der Meldung d.Sender.
- Fehlercharakteristiken:
  - Fehlerwahrscheinlichkeit (10^{-2} .. 10^{-15}),
  - zufällige und periodische Fehler,
  - Bitfehler oder Burstfehler.
- Diese Störungen sollen erkannt und falls möglich korrigiert werden:
  - Byteparität,
  - Langs- und Querparität,
  - Zyklische Redundanzprüfung "CRC",
  - fehlerkorrigierende Codes.

6.1.1. Paritätsprüfungen

- z.B. 8-Bit Code wird um ein Paritätsbit erweitert. -> Start-Stop Datenübertragung
- Paritätsoptionen:
  - insbesondere bei Start-Stop Betrieb,
  - Even / Odd / None / Zero / One.
- Vorwärts-Fehlerkorrektur, "Forward Error-Correction":
  - Fehler kann ohne **Rückfrage** korrigiert werden.
- Mit Längs- und Querparität kann ein einzelner Fehler auch korrigiert werden:

STX	**M**	**E**	**L**	**D**	**U**	**N**	**G**	**.**	**7**	**ETX**	**CHK**
0	1	1	0	0	1	0	1	0	1	1	**0**
1	0	0	0	0	0	**1**	1	1	1	1	**0!**
0	1	1	1	1	1	1	1	1	1	0	**1**
0	1	0	1	0	0	1	0	1	0	0	**1**
0	0	0	0	0	1	0	0	0	1	0	**0**
0	0	0	0	0	0	0	0	1	1	0	**0**
0	1	1	1	1	1	1	1	0	0	0	**1**
1	**0**	**1**	**1**	**0**	**0**	**0!**	**0**	**0**	**1**	**0**	**0**

6.1.2. Zyklische Prüfsummen ("CRC")

- Wesentlich verbesserte Fehlererkennung.
- Wird ein Fehler erkannt, verwirft der Empfänger die Nachricht.
- Der Sender wiederholt dann die Nachricht.
- Länge L der Prüfsumme 12, 16 oder 32 Bit.
- Übertragungsfehler wird erkannt:
  - wenn Fehlersequenz kürzer als 16 bzw. 32 Bit,
  - wenn Anzahl der Fehlerbits 1,2 oder ungerade,
  - und 99,99% aller längeren Burstfehler.
- Modulo 2 Arithmetik:

  0+1 = 1 1+0 = 1 0+0 = 0 1+1 = 0

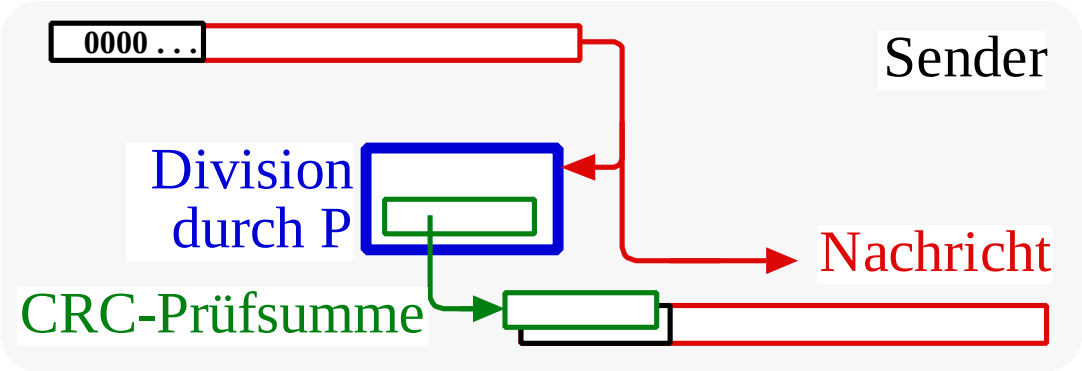
  0- 1 = 1 1- 0 = 1 0- 0 = 0 1- 1 = 0

  Multiplikation als sukzessive Addition,
  Division als sukzessive Subtraktion,

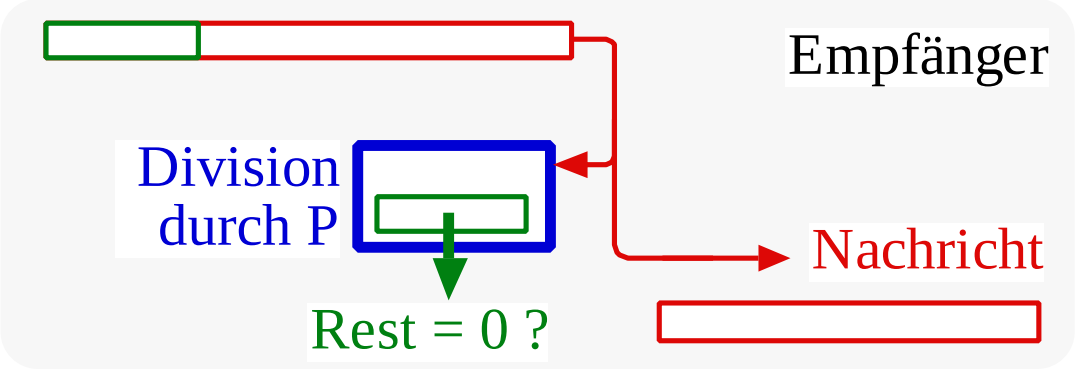
  => XOR-Funktion: a Å b

- Die XOR-Funktion lässt sich leicht in Hardware implementieren (XOR-Gatter).

- Erzeugen der CRC-Prüfsumme:
  - Nachricht mit 2^L multiplizieren,
  - Nachricht durch eine feste Prüfzahl P dividieren,
  - Division geschieht ohne Überträge (Modulo 2),
  - Divisionsrest von Nachricht subtrahieren,
  - Paket zum Empfänger übertragen:



- Validieren der Prüfsumme beim Empfanger:
  - Paket durch feste Prüfzahl P teilen,
  - falls Divisionsrest = 0, dann Nachricht OK,
  - Paket aufteilen in Nachricht und Prüfsumme,
  - Nachricht abliefern (falls OK) .



- Alternative Interpretation:
  - ein b Bit langer Bitstrom wird als Polynom des Grades b-1 aufgefaßt,
  - z.B. \$89 = 1000 1001 interpretiert als x^7+x^3+1
- Geeignete Prüfpolynome sind etwa:
  a) CRC-12: $x^{12} + x^{11} + x^3 + x^2 + x + 1$
  b) CRC-16: $x^{16} + x^{12} + x + 1$
  c) CRC-V.41: $x^{16} + x^{12} + x^5 + 1$
  d) CRC-32: $x^{32} + x^{26} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^4 + x^2 + x + 1$

bzw.:

1 1000 0000 1111,
1 0001 0000 0000 0011,
1 0001 0000 0010 0001,
1 0000 0100 0100 0001 0001 1101 1001 0111 .

- Eine verfälschte Meldung entsteht durch Addition eines Fehlerpolynoms E(x). Der Fehler wird nicht entdeckt, wenn auch das Fehlerpolynom durch das Prüfpolynom teilbar ist:
  - 1 Bit Fehler: 2^i ist nicht teilbar durch P.
  - Doppelfehler: E(x) darstellen als $x^i(x^{j-i}+1)$ ...
  - odd Bitfehler: E(x) < > (x+1) Q(x),
  - kurzer Burst: E(x) ist nicht teilbar durch P(x), falls E(x) kürzer als P(x).

=> Peterson, W. & Brown, D. "Cyclic Codes for Error Detection", Proc. IRE, January 1961.

6.1.3. Rechenbeispiel einer CRC-Division:

- Die Nachricht 1101011010*2^4 wird Modulo 2 dividiert durch das Prüfpolynom 10011:

```
 11010110100000:10011= 1100001011
-10011                 (Rest 1101)
 10011
 -10011
  00001
  -00000
   00010
   -00000
    00101
    -00000
     01010
     -00000
      10100
      -10011
       01110
       -00000
        11100
        -10011
         11110
         -10011
          1101 (Rest)
```

- Die Nachricht ist nun:

  (1101011010 , Rest) = 1101011010 1101

6.1.4. Realisierung durch Software:

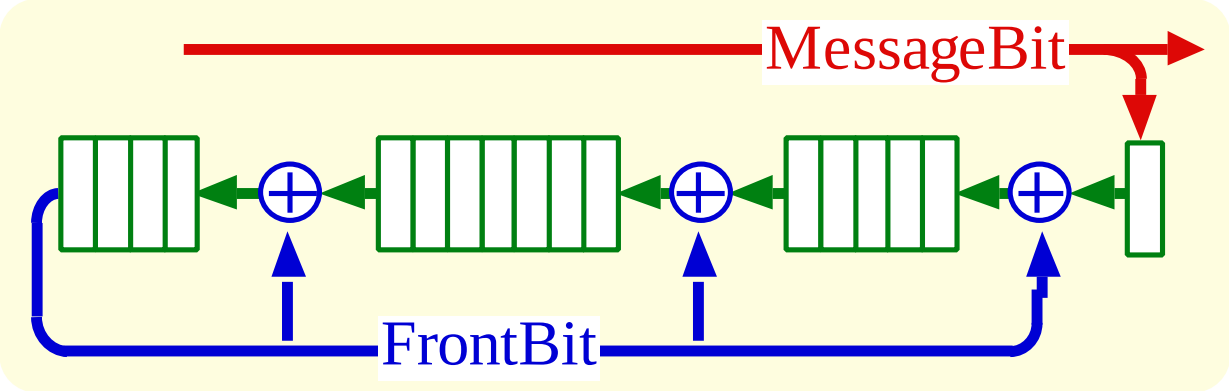
- Nachbildung des Divisionsalgorithmus.

```
const    crcPoly = $13; frontbit = $10;
var      register, xorInput : longint;
procedure AddBit( MessageBit : integer);
begin    if (register >= frontbit)
               then xorInput := crcPoly
               else xorInput := 0;
         register := register XOR xorInput;
         register := register SHL 1;
         register := register + MessageBit;
         (* Now print register value *)
end;
begin .... (*  add all MessageBits *) end.
```

register:	MessageBit
00001	1
00011	1
00110	0
01101	1
11010	0
10011	1
00001	1
00010	0
00101	1
01010	0
10100	0
01110	0
11100	0
11110	0
1101	

6.1.5. CRC-V.41 mit Schieberegister:

- Entsprechend Division und Softwarelösung.



- Registerinhalt pro Takt 1 Stelle verschoben.
- Für nicht verschwindende Polynomkoeffizienten eine Einspeisung ins Schieberegister.
- Rückkopplung über 16 Stufen (CRC-V.41).
- Der im Schieberegister verbliebene Rest wird nach dem letzten Bit der Meldung übertragen.
- Bitstuffing geschieht anschließend.

6.1.6. Vorwärts-Fehlerkorrektur

- Verlorengegangene Pakete rekonstruieren.
- Redundanzpakete hinzufügen.
- z.B. 3 Pakete zu übertragen:

  P1: 10101010
  P2: 00110011
  P3: 00001111

- Redundantes Paket P4:

  (p1 xor p2: 10011001)
  P4: 10010110

- Rekonstruktion des verlorenen P2, z.B.:

  P2 = P4 xor (P1 xor P3)
  P2 = P4 xor (10100101)
  P2 = 00110011 (sic)

- Entsprechend P1 oder P3 rekonstruieren.
- Es muss bekannt sein, welches Paket fehlt.

Einschub: Hamming-Abstand (1)

- Hamming-Abstand d einzelner Codewörter
  - Anzahl der Bitpositionen in denen sich Codewörter c_1, c_2 unterscheiden.
  - Beispiel: d(1000101001, 1101101001) = 2
  - Entspricht Anzahl der Einsen von c_1 XOR c_2 ,
  - Hamming-Abstand vom vollständigen Code C:

    $D(C) := \min\{ d(c_1, c_2) \mid c_1, c_2 \in C, c_1 \neq c_2 \}$.

- Codewort besteht aus m Bits
  - 2^m legale Codewörter aus m Bits möglich,
  - r Prüfbits werden an ein Codewort angehängt,
  - Codewort besteht jetzt aus n = m + r Bits,
  - Pro Codewort existieren n illegale Codewörter der gleichen Länge mit Hamming-Abstand 1.
  - Pro Codewort ein legales Codewort mit Hamming-Abstand 0. (Das korrekte Codewort!)

  => n+1 Codewörter mit Hamming-Abstand ≤ 1.

  => Es gibt 2^m Codewörter wobei jedes Codewort n+1 = m+r+1 Codewörter belegt.

- Frage: Wie muss r gewählt werden, damit **einzelne** Bitfehler behoben werden können.
  - $(n+1)*2^m$ Codewörter mit n Bits darstellbar,
  - $(n+1)*2^m \leq 2^n \implies (m+r+1) \leq 2^r$.

Einschub: Hamming-Abstand (2)

Beispiel: Fehlerbehebender Code

n = 10 Bit; m = 2 Bit; r = 8 Bit;

00 -> 00000 00000	01 -> 00000 11111
10 -> 11111 00000	11 -> 11111 11111

Beispiel für einen Fehler bei dem 2 Bits nicht übereinstimmen:

00000 00111 ==> 00000 11111

2 Bitfehler wird sicher erkannt und behoben.

→ Hamming-Abstand: 5
→ Korrektur von 2-Bitfehlern möglich.

Die Codewörter müssen innerhalb eines Codes derart gewählt werden, dass:

- Der Hamming-Abstand D(C) = e+1 beträgt, damit e-Bitfehler erkannt werden.
- Der Hamming-Abstand D(C) = 2e+1 beträgt, damit e-Bitfehler behoben werden können.

Beachte: $D(C):=\min\{ d(c_1, c_2) \mid c_1, c_2 \in C, c_1 \neq c_2 \}$.

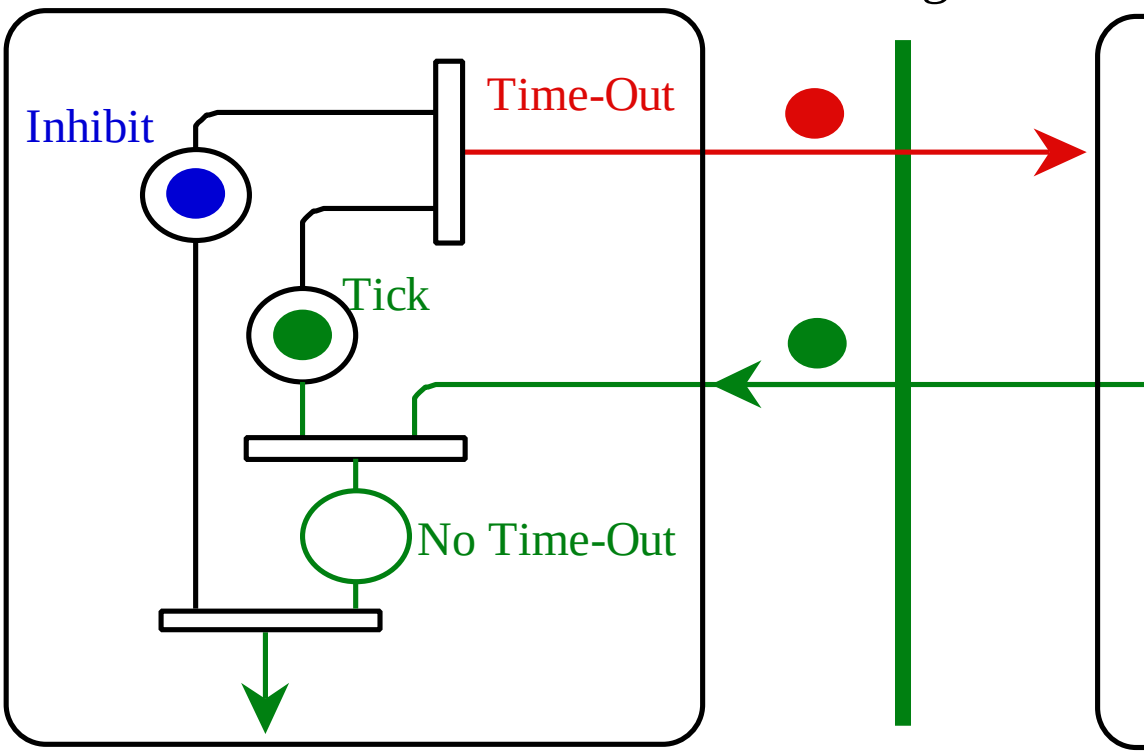
D(C) gibt minimalen Abstand zwischen zwei gültigen, aber verschiedenen Codewörtern an.

D(C) gibt an wie viele Fehler ein fehlerkorrigierender Code erkennen bzw. beheben kann.

6.2. Bestätigungen

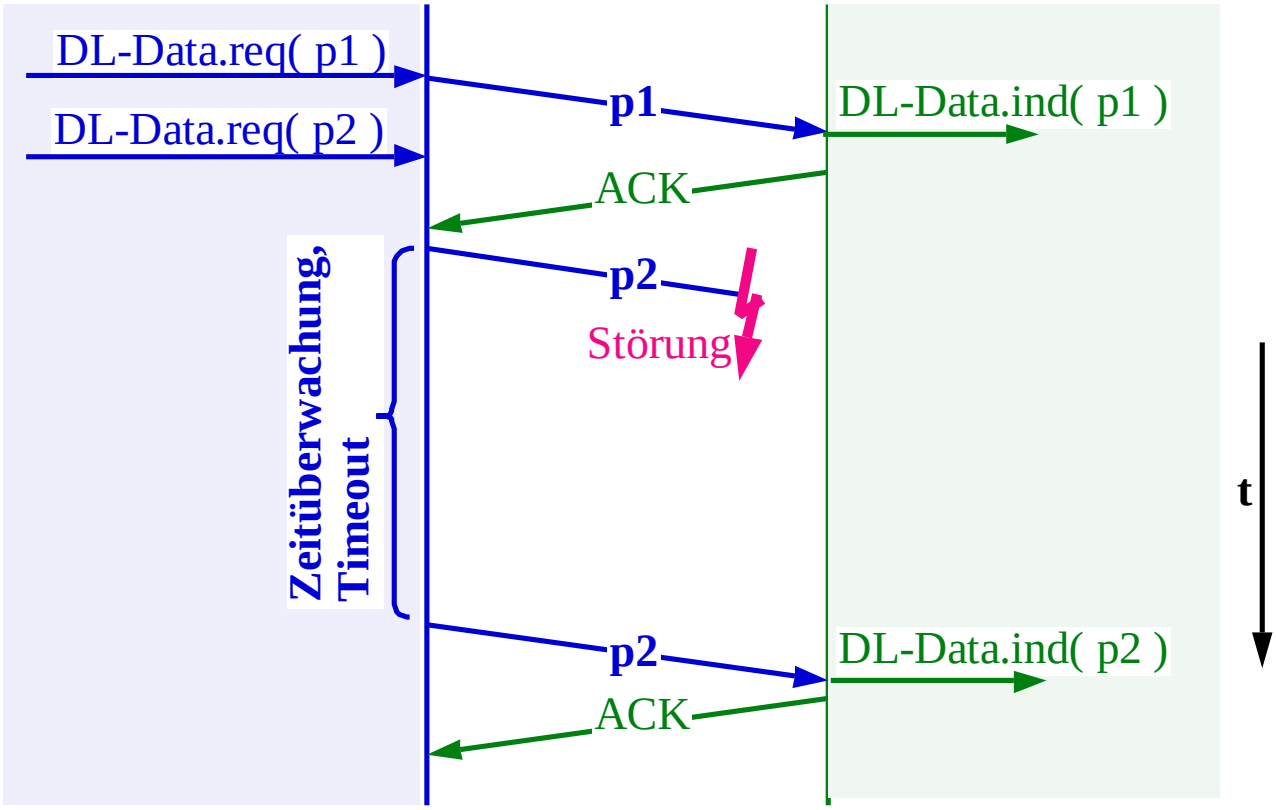
6.2.1. Zeitüberwachung ("Time-Out")

- Die Aktionen eines Senders oder Empfängers werden vom Ablauf eines lokalen Zeitgebers abhängig gemacht:
  - zeitlichen Abstand zw. Nachrichten einhalten,
  - N. wiederholen, wenn die Bestätigung ausbleibt,
  - Feststellen eines Leitungsunterbruchs,
  - "Lebenszeichen" zum Partner,
  - Abbau der Verbindung.
- Petrinetz-Darstellung:
  - **Tick** erzeugt neue Marken im Zeitabstand T nach einer Entladung,
  - **Inhibit** erzeugt Marken im Abstand T+ε ,
  - verhindert nicht deterministische Zündung.



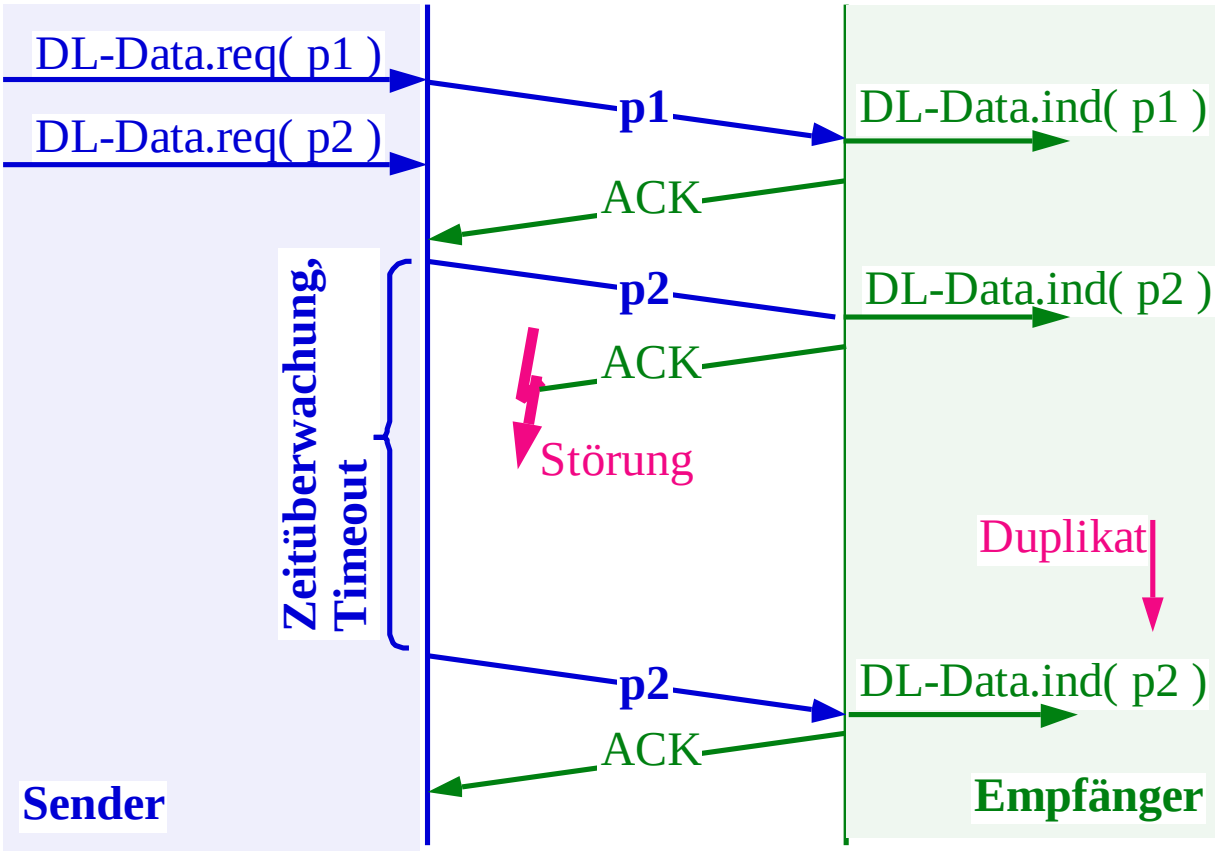
6.2.2. "Stop & Wait" Bestätigung

- Eine Nachricht übertragen und dann auf die Quittung warten (ACKnowkledge).
- Wenn die Quittung ausbleibt, so wird die Übertragung wiederholt.
- Die Wiederholung erfolgt erst nach Timeout.



Aber:

- Ein Risiko besteht, dass eine Meldung doppelt übertragen wird.
- Nämlich, wenn die Quittung verloren geht:

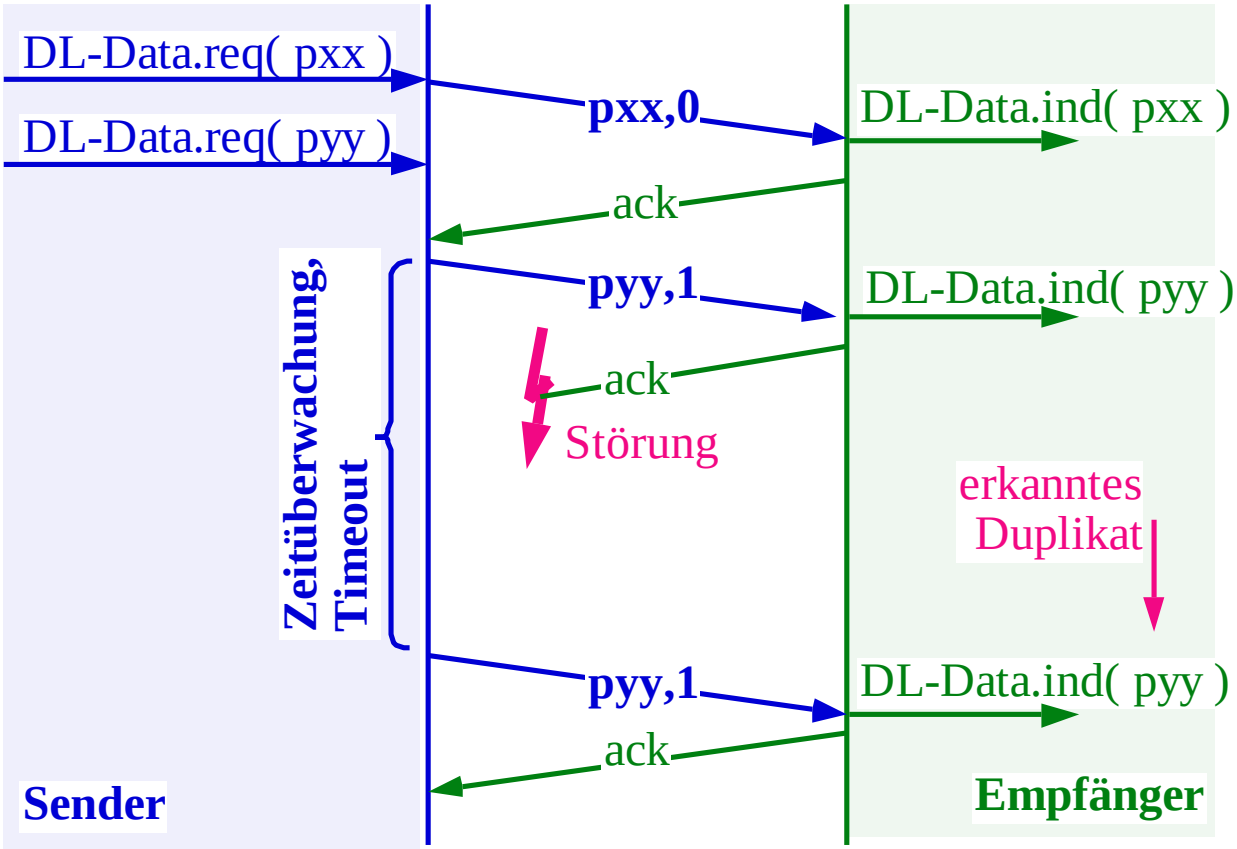


=> Lösung hierzu:

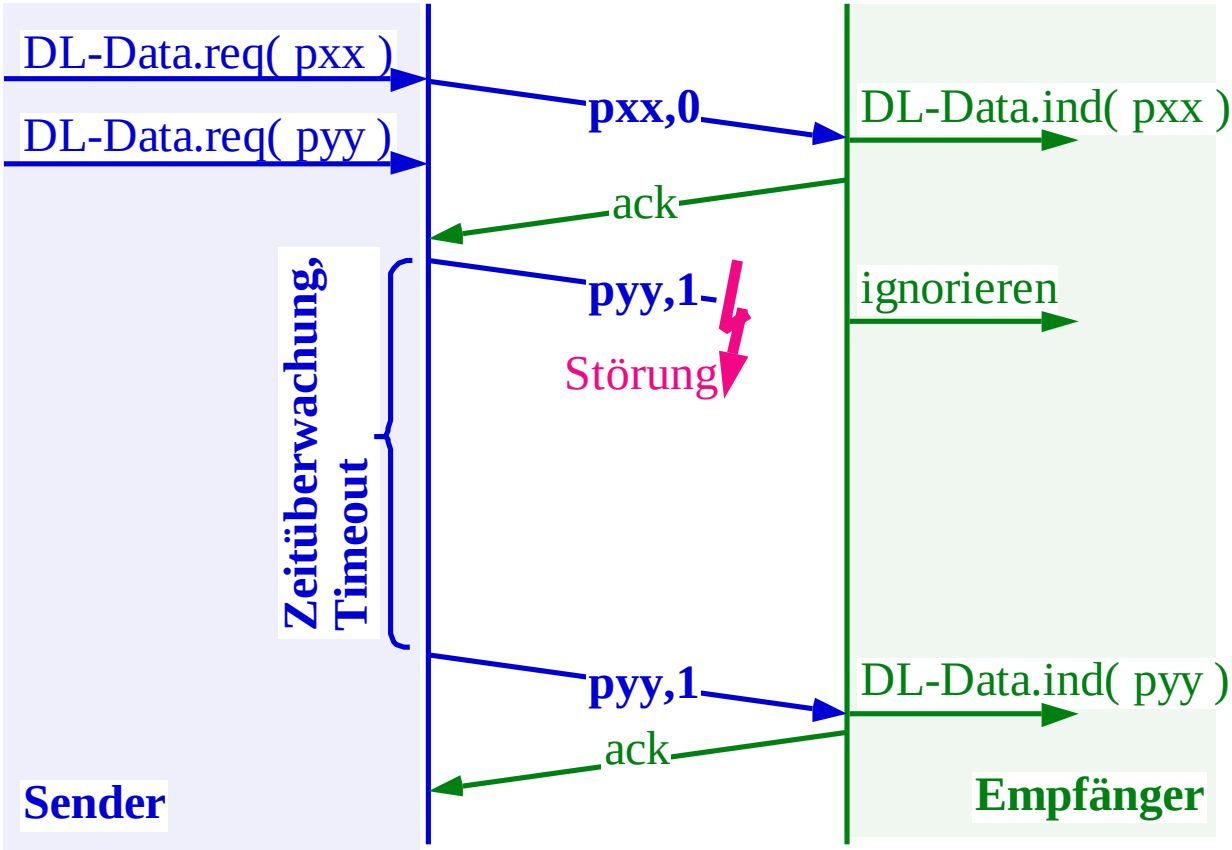
Die Pakete werden nummeriert.

6.2.3. Nummerierung der Pakete

- Der Empfänger erwartet abwechselnd Pakete mit Nummer #0 und #1.
- Das Duplikat wird vom Empfänger als solches identifiziert.
- Wiederholung bei zerstörter Quittung:



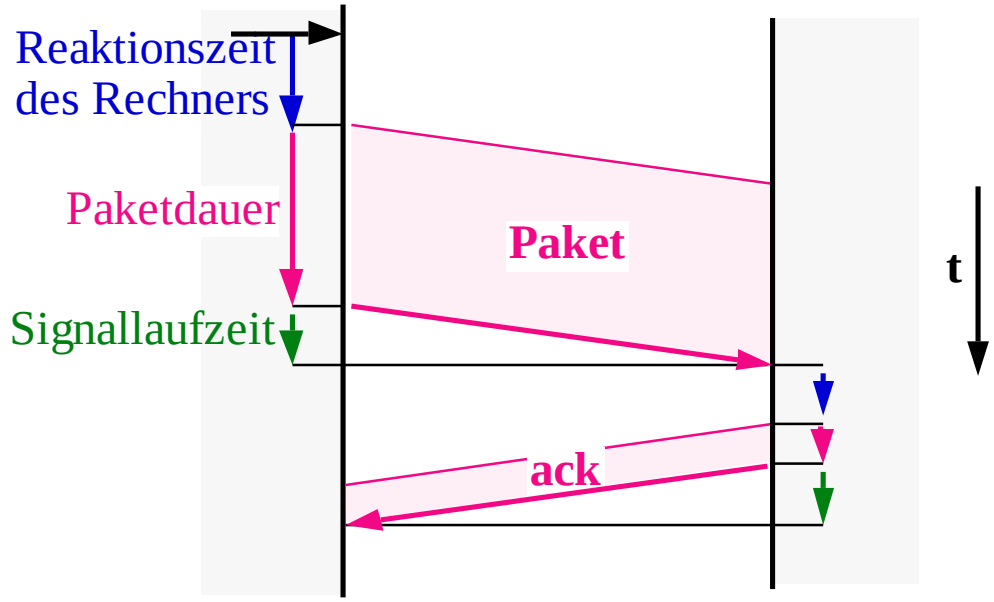
- Wiederholung bei zerstörter Nachricht.
- Teilpakete mit falschem CRC wegwerfen.
- Bestätigungsnummer nicht erforderlich, da immer nur eine unbestätigte Nachricht.



- Nachteil:
  - in der Quittungphase läuft die Leitung leer,
  - Übertragungskapazität geht verloren, wenn die Quittungslaufzeit grösser wird,
  - insbesondere bei vielen Zwischenknoten.

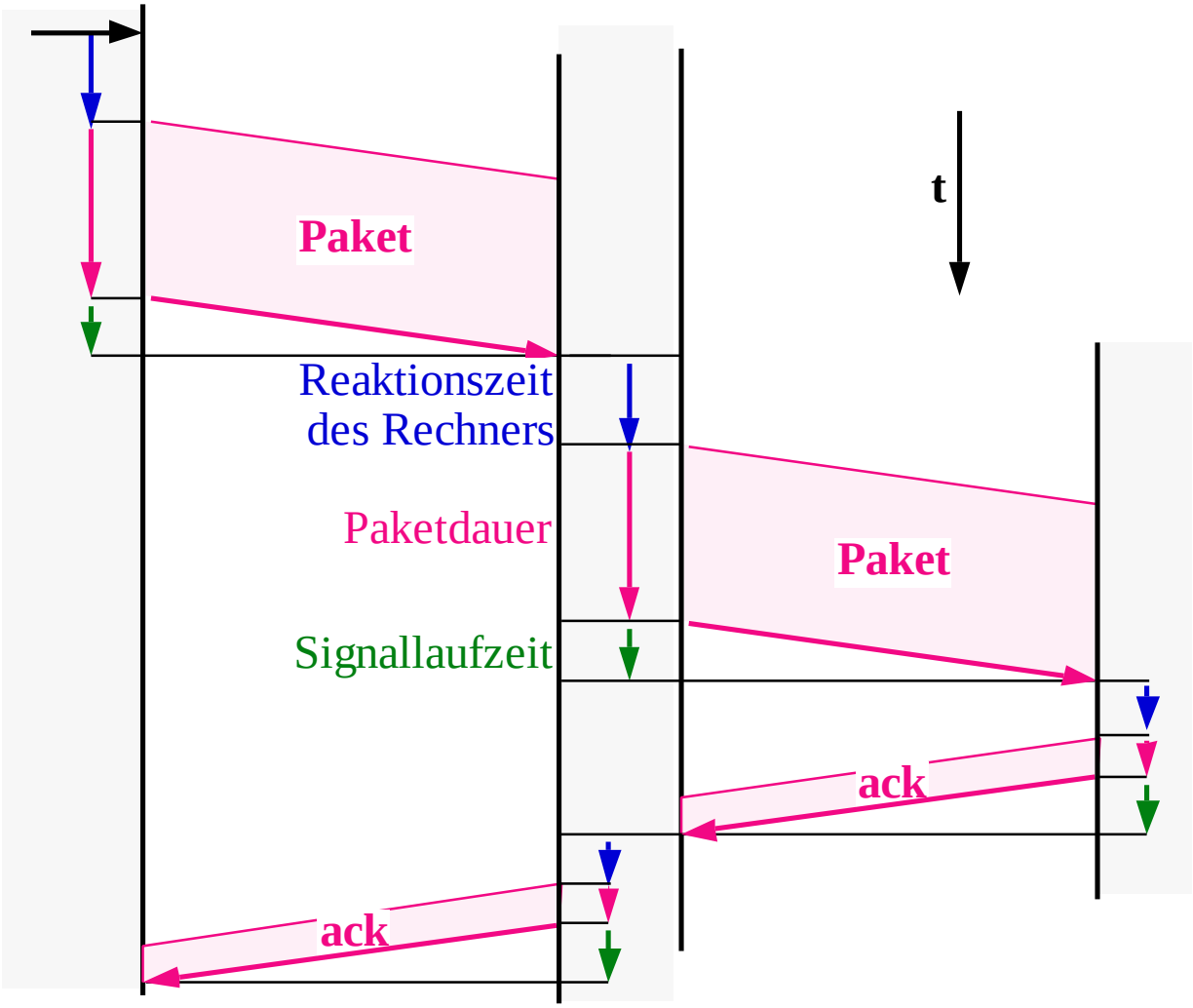
6.2.4. Laufzeiten von Nachrichten:

- Genauere Darstellung der Zeitverhältnisse beim quittierten Nachrichtenaustausch.
- Direkte Leitung zwischen zwei Knoten:



- Kurze Nachrichten und Bestätigungen belegen die Leitung weniger lang.
- Oft dominiert die Verzögerung im Rechner.

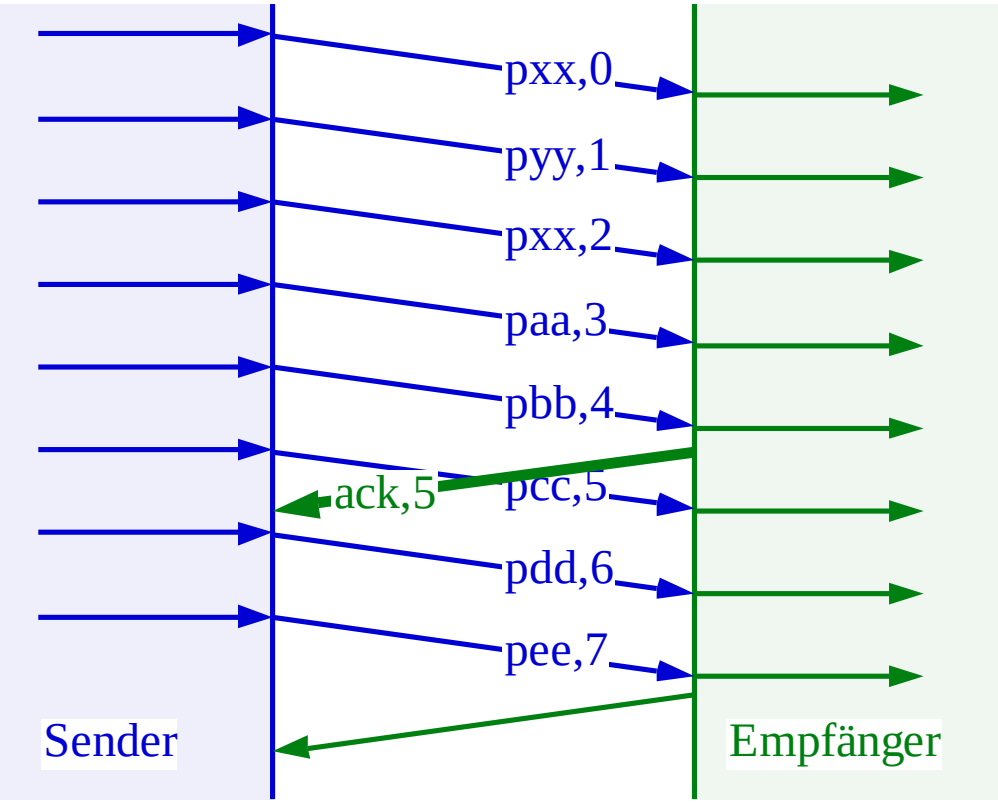
- Übertragung via Zwischenknoten:
  - zusätzlich "Store-and-Forward" delay:



- In normalen Netzen führt der Weg von einem Endsystem zum anderen oft über viele Zwischenknoten.
- Die Prüfsumme muss abgewartet werden.
- Normalerweise kein "Cut-through" Routing in den Zwischenknoten.

6.3. Fenstermechanismen

- Bessere Auslastung der Leitung.
- Es wird zugelassen, dass zu einem Zeitpunkt mehrere Nachrichten noch unbestätigt sind.
- Grösserer Bereich für die Sequenznummern der Nachrichten:

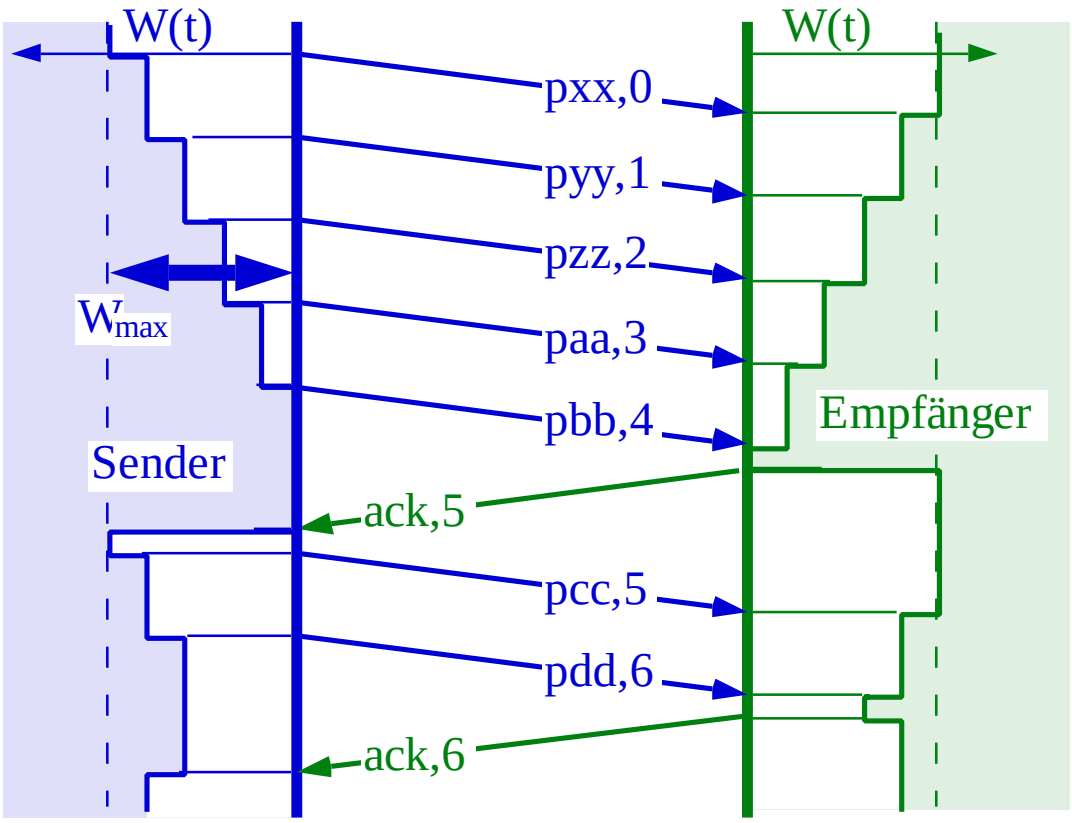


- Nicht mehr nach jeder Nachricht warten.
- Mehrere Nachrichten mit einer Antwort bestätigen.
- Meist wird die nächste erwartete Nummer in der Bestätigung genannt (z.B. ack,5).

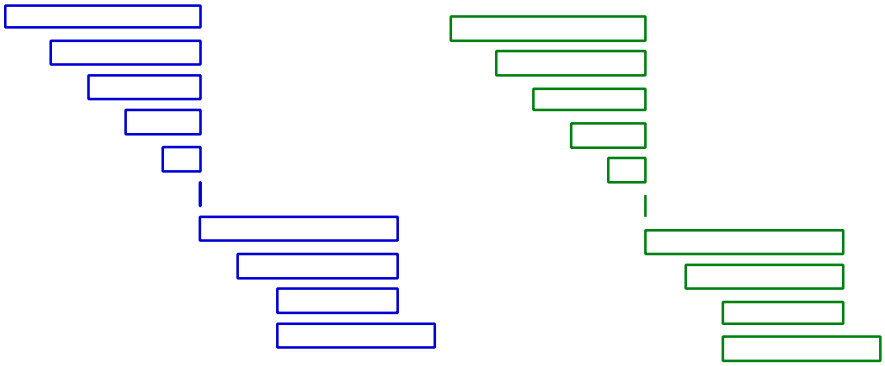
6.3.1. Fensteröffnung

- Maximale Fensteröffnung W_{max} Pakete:
  - mindestens um 1 kleiner als Nummernbereich,
  - meist als fest vereinbart.
- Aktuelle Fensteröffnung W (=Window):
  - Fenster W variiert während der Übertragung,
  - Empfänger hält zumindest W Paketpuffer bereit,
  - Sender bewahrt unbestätigte Pakete auf,
  - Sender stoppt nach W unbestätigten Paketen.
- Die Puffer für bestätigte Pakete werden vom Sender freigegeben.
- Empfänger bestätigt Paket P, wenn er genügend Puffer für die Pakete P+1 bis P+W hat.
- Wrap-Around der Sequenznummern.
- Nummernbereich 0..n erlaubt n-1 unbestätigte Nachrichten.
- Grosse Fensteröffnungen z.B. für Satellitenstrecken.

Ablauf der Fensteröffnung:



- Unterschiedliche Sicht beim Sender und beim Empfänger.
- Sender stoppt bei ausgeschöpftem Fenster (=Flusskontrolle).
- Wiederholung falls länger keine Bestätigung.
- Sicht als Fenster:

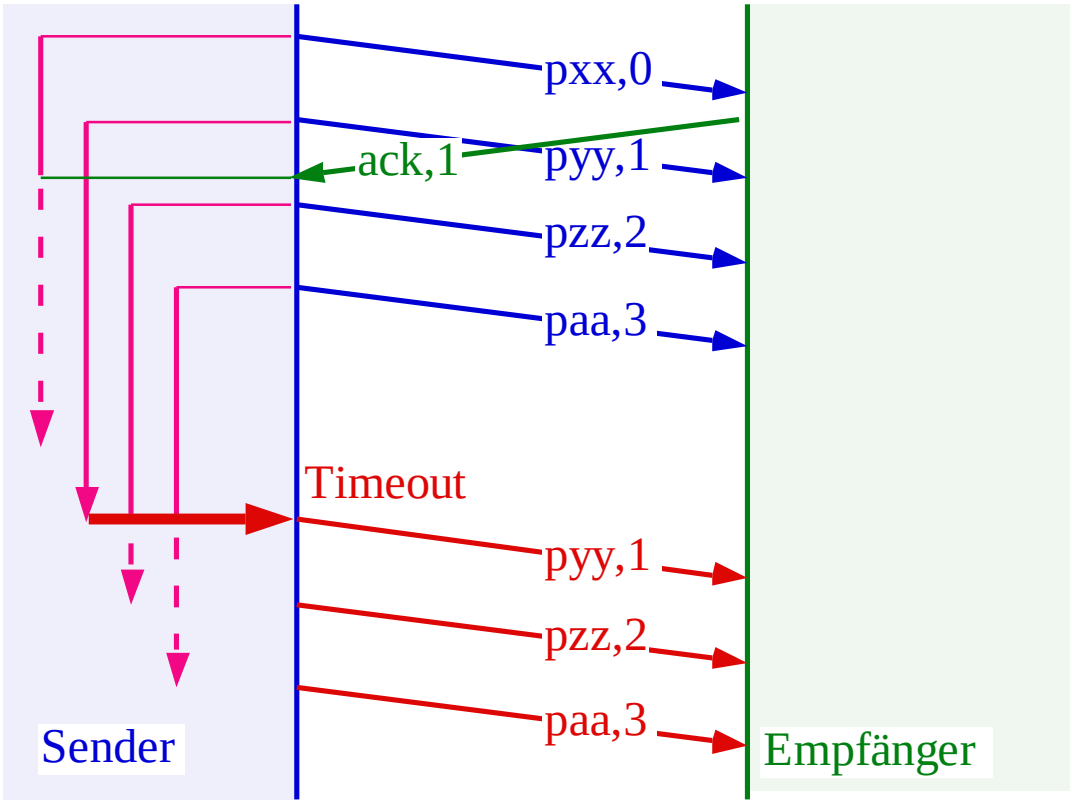


6.3.2. Fehlerbehandlung

- Timeout muss so gross eingestellt werden, daß normalerweise vorher eine Antwort eintrifft.
- Trifft eine Quittung vor Ablauf des Timer ein, so wird einfach das Fenster wieder geöffnet.
- Am besten für jedes Paket ein Timer.

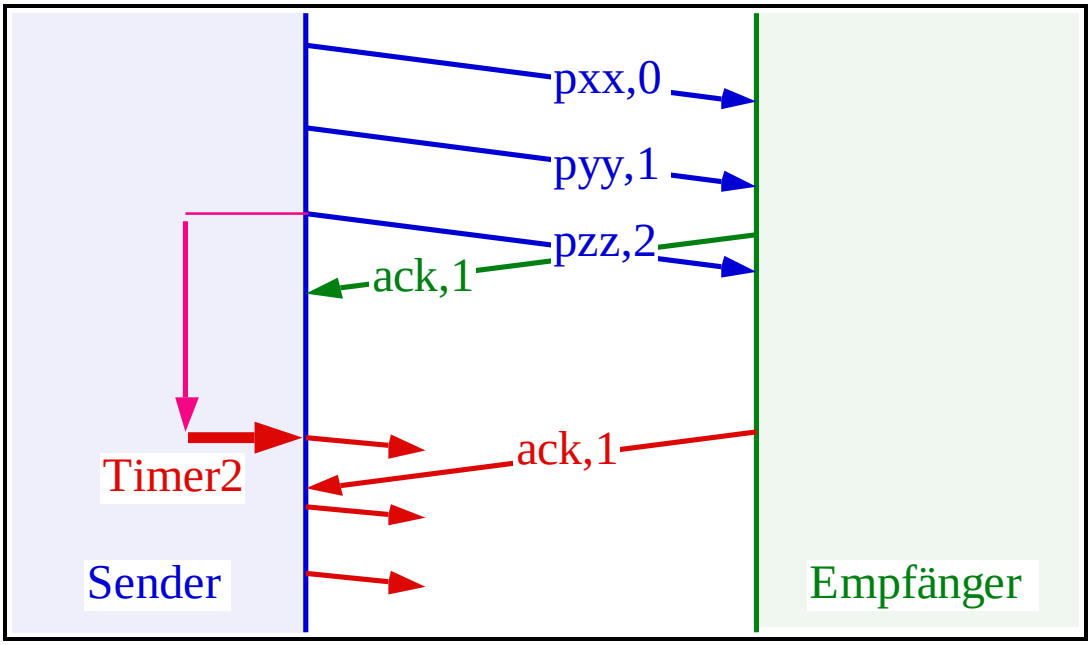
Implizierte Paketwiederholung Fall A:

- Läuft der Timer ab, so werden alle unbestätigten Pakete wiederholt.
- Timer für das Paket (pyy,1) läuft ab.
- Pakete ab (pyy,1) werden wiederholt.



Implizierte Paketwiederholung Fall B:

- Eventuell bestimmt ein zweiter Timer, ab welchem Zeitpunkt eine Quittung gleichzeitig als "Reject" interpretiert wird.
- Wird "nach" dem Senden eines Paketes eine Quittung für ein altes Paket erhalten, so werden die noch unbestätigten Pakete wiederholt.



Negative Bestätigung als "Go-back-to-N":

- Eine besondere Kontrollnachricht (REJ) fordert Wiederholung ab Paket N.

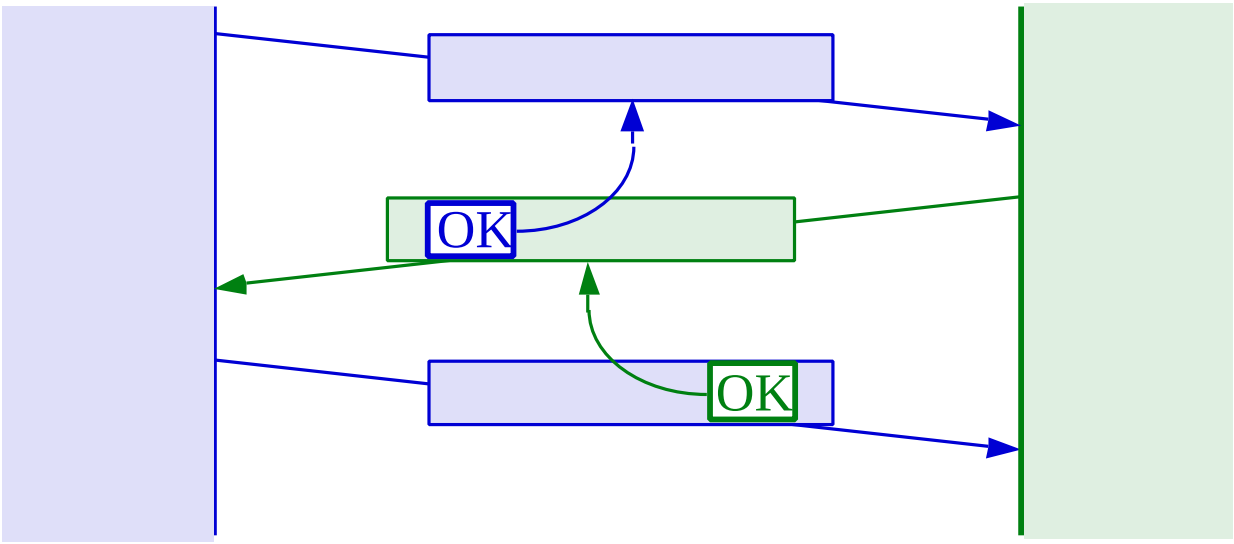
Selektives Reject:

- Nur das verlangte Paket wird wiederholt.
- SREJ als Kontrollnachricht.

6.4. Huckepack-Transport

6.4.1. Fenstergröße 1:

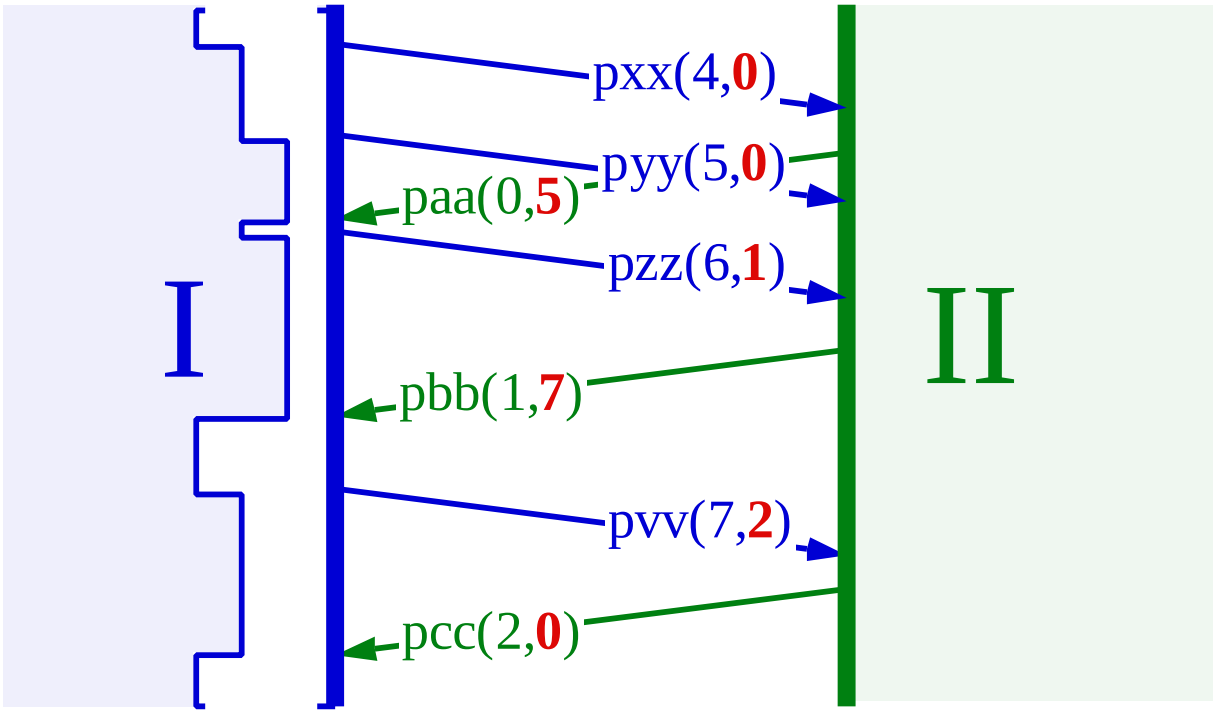
- Englisch "Piggy Back"-Transport.
- Die Bestätigung wird in den Kopf einer Nachricht in Gegenrichtung verpackt:



- Wenn sowieso eine Nachricht in Gegenrichtung ansteht, so sind die Kosten für die Bestätigung klein.
- Anderenfalls separate Kontrollnachricht.

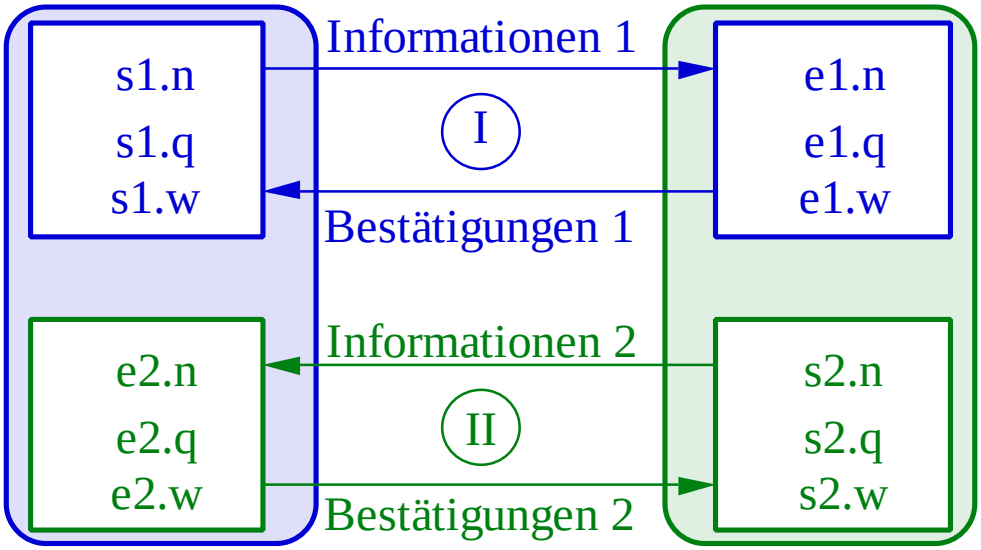
6.4.2. Transport in beide Richtungen:

- Huckepack-Quittungen rot bzw. **fett** markiert.
- Separate Nachrichtennummern in beide Richtungen z.B. jeweils 0..7 .
- Nachrichten 0..3 zur Station II seien schon übertragen und bestätigt.



6.5. Zustandsautomaten

- Protokolle werden meist mithilfe von Zustandsvariablen implementiert.
- In jedem Knoten läuft ein endlicher Automat.
- Das Ziel ist eine Übereinstimmung der Zustände in den kommunizierenden Knoten.



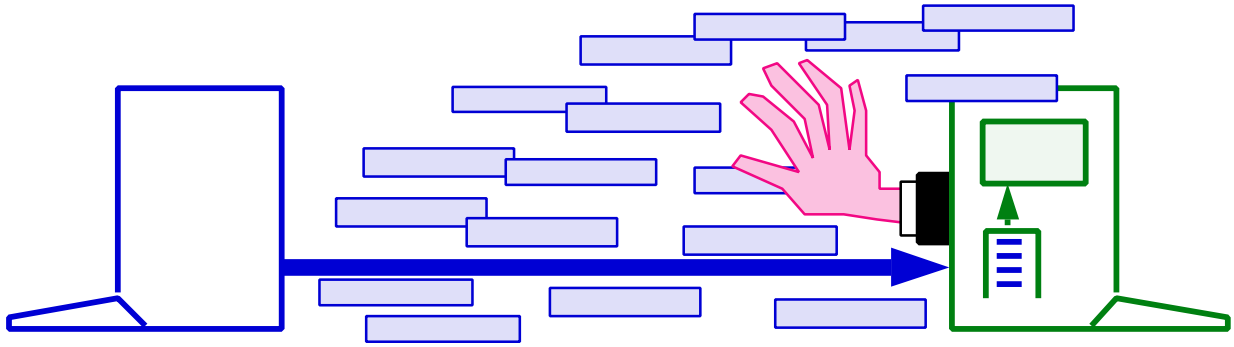
- Legende:

s1, s2	-	Sender
e1, e2	-	Empfänger
w	-	Fensteröffnung
q	-	Quittungsnummer
n	-	Nachrichtensequenznummer

6.6. Flusskontrolle

6.6.1. Weshalb Flusskontrolle?

- Schutz eines Empfängers vor einer Überflutung durch Datenpakete:



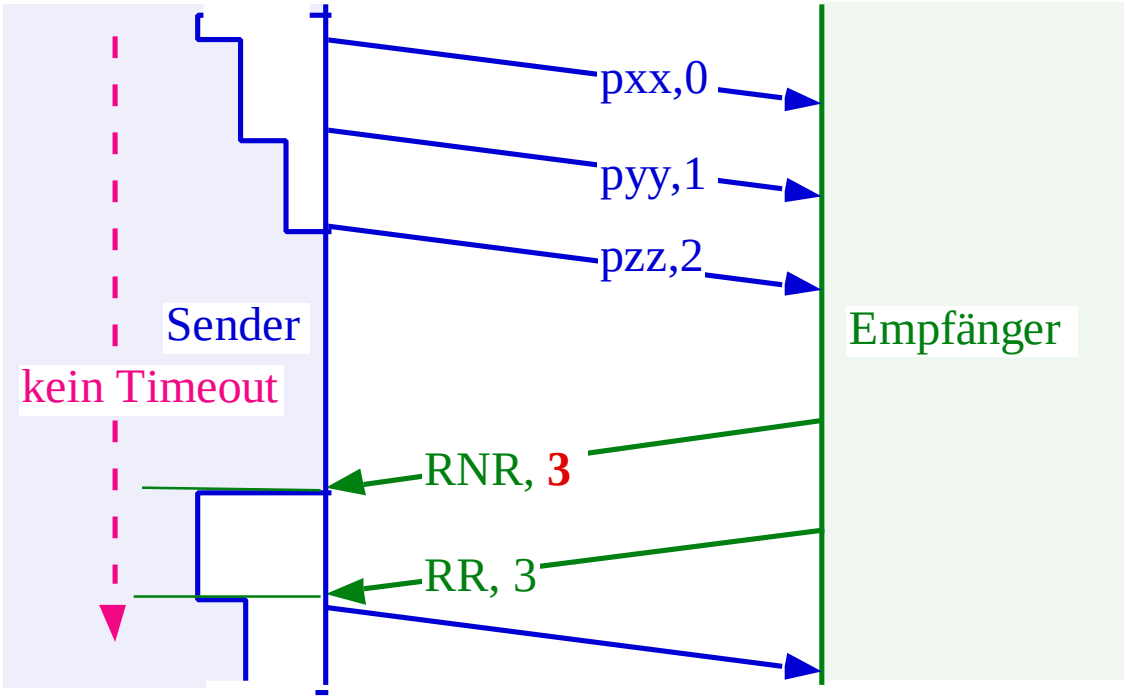
- Suspendieren des Datenflußes wegen Überlastung im Empfanger.
- Empfänger schreibt nicht schnell genug:
  - auf die weiterführende Datenleitung,
  - in den Bildschirmspeicher,
  - Drucken auf Papier,
  - auf Festplatte.
- Anwenderprogramm rechnet zu langsam.
- Keine Puffer mehr für Meldungen.

6.6.2. Explizite Flusskontrolle:

- Für asynchrone Datenströme oft mit Sonderzeichen:
  - Datenstrom anhalten: cntl-S, x-off, DC3,
  - Datenstrom weiter: cntl-Q, x-on, DC1,
  - N.B. das cntl-Präfix erzeugt Zeichen <32.
- Hardware-Flusskontrolle an der seriellen Schnittstelle mit:
  - RTS: Request to Send (Signal zum Modem),
  - CTS: Clear to Send (Signal vom Modem).
- Handshake-Protokoll an der parallelen Druckerschnittstelle.
- Bei paketorientierten Protokollen oft besondere Kontrollnachricht:
  - RR: Receive Ready,
  - RNR: Receive not Ready.

6.6.3. Fensterbasierte Flusskontrolle:

- Ist das Fenster ausgeschöpft und hält der Sender die Bestätigung zurück, so entsteht eine Flusskontrollwirkung.
- Bestätigung nicht zu lange zurückhalten, sonst geschieht Timeout und erneute Übertragung.
- Vor Ablauf des Timers bestätigen und expliziten Flusskontrollbefehl schicken:

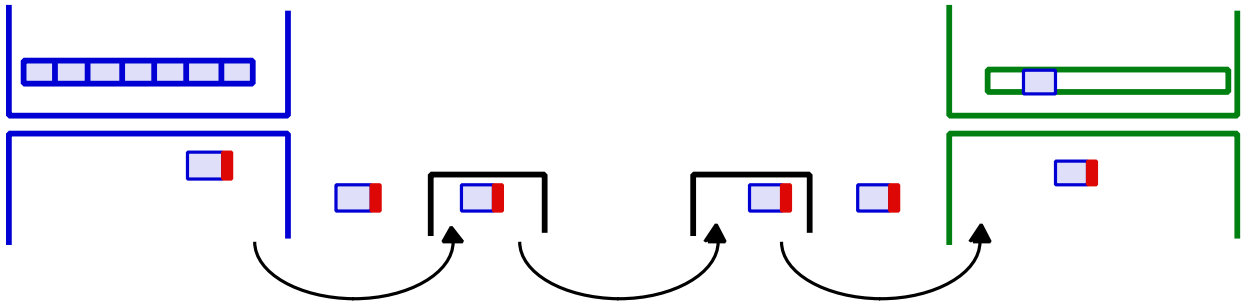


6.6.4. Übertragungsratensteuerung:

- Englisch: "Rate Control".
- Der Sender ist dafür verantwortlich, die vereinbarte Übertragungsrate einzuhalten:
  - mittlere Übertragungsrate,
  - Dauer der Spitzenlast,
  - Spitzenbelastung,
  - garantierte Rate.
- Für Transportsysteme mit langen Bestätigungszeiten. Dann kommt die explizite Flusskontrolle zu spät.
- Für Transportsysteme mit sehr vielen Nachrichten im Transit.
- Teil der ATM-Dienstgüteverhandlung (QoS) ist die Verhandlung der zulässigen Datenrate.
- Wird die Bandbreiten-Allozierung überzogen, so darf das Netz Datenpakete verwerfen.

6.7. Paketisierung / Segmentierung

= Aufspaltung einer längeren Nachricht in kleinere Pakete bzw. Segmente.



- Vorteile:
  - reduzierter Pufferbedarf in den Zwischenknoten,
  - weniger Paketverluste bei schlechten Leitungen,
  - reduzierter "Store & Forward"-Delay.
- Nachteile:
  - zusätzlicher Header für die Pakete,
  - Paketreihenfolge muss sichergestellt werden,
  - erhöhter Kopieraufwand beim Empfang,
  - Reassembly Puffer beim Empfänger.
- Keine Monopolisierung der Leitung durch einen Nutzer.

6.8. Adressierung und Gruppierung

"Machines use adresses, people prefer names"

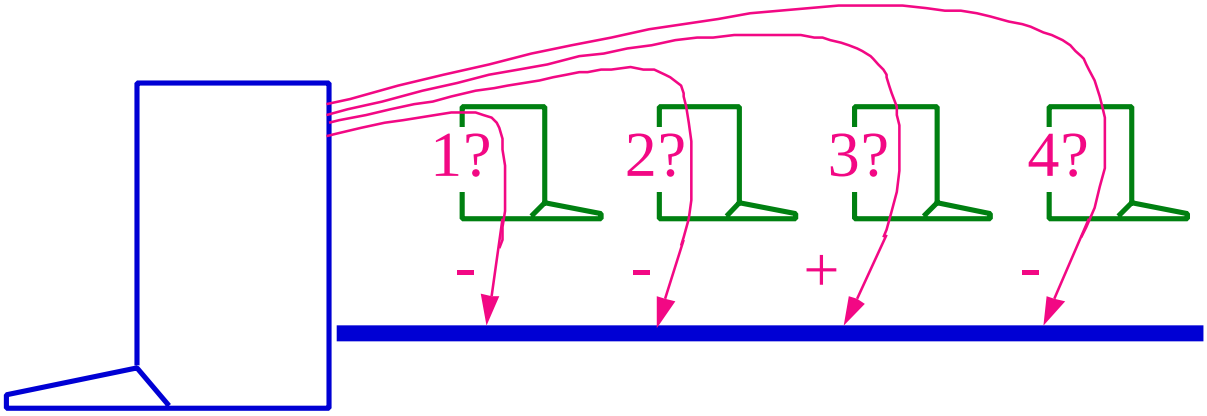
- Namen (mnemonische Funktion):
  - Peter Schulthess (Personenname)
  - pschulth (Benutzername)
  - Vertsys_D (Netzbereichsname)
  - vs.informatik.uni-ulm.de (Internetname)
  - inputChar (Variablenname)

  ...
- Adressen (Auswahlfunktion):
  - 00 01 02 A4 B3 C5 (Adapterkarte)
  - 134.60.77.74 (Internet adresse)
  - 0049 731/502-4140 (Telefonnummer !)
  - ($0040:$001A) (Speicheradresse)
  - $03f8 (I/O Port)
- Gruppen-Adressen:
  - 80 01 02 A4 B3 C5 (Gruppe im Ethernet)
  - FF FF FF FF FF FF (Rundspruch)
  - 0049 731 502 2428 (Modemgruppe am RZ)
- Namensbindung bedeutet das Verbinden eines Namens mit einer Adresse:
  - statisch Binden(z. Start- oder Übersetzungszeit),
  - dynamisch Binden (= zur Laufzeit),

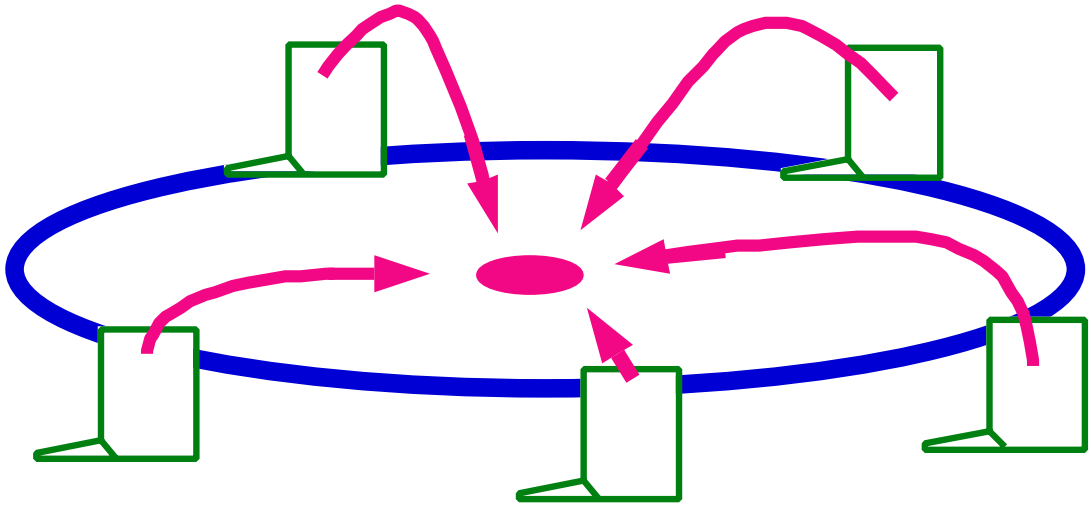
  => Ernst.informatik.uni-ulm.de -> 134.60.77.56

6.9. Zuteilungsprotokolle

- Wichtig, wenn sich mehrere Stationen eine Leitung teilen.
- Station wird über ihre Adresse angesprochen.
- vgl. Mehrpunktverbindung im Kapitel "Betriebsarten und Verkehrsrichtungen".
- Entweder Abfrage durch zentrale Station:



- Oder dezentrale Zugangsteuerung:
  - mit Kollisionsrisiko,
  - oder mit Token:



6.10. Verbindungsauf- & Abbau

- Eine Verbindung wird aufgebaut, um den Paketstrom als Gesamtheit zu behandeln:
  - Sequenznummern,
  - Flusskontrolle,
  - Dienstgüte.
- Beim Verbindungsaufbau:
  - Sequenznummern beidseitig initialisieren,
  - Kommunikationspartner identifizieren,
  - Übertragungsweg im Netz suchen,
  - maximale Paketgröße aushandeln,
  - Puffer und Resourcen allozieren,
  - Dienstgüte aushandeln.
- Beim Verbindungsabbau:
  - Abgrenzung von der nächsten Verbindung,
  - Resourcen freigeben.
- Eventuell Konferenzverbindungen.
- Eventuell verbindungsloser Dienst:
  - keine Übertragungsgarantien,
  - keine Aufbauverzögerung,
  - keine Sequenznummern.