Gerhard Müller
Institut für Informatik
Technische Universität München

Praktikum “Rechnernetze”

WANs am Beispiel von X.25

WS 97/98

Ausarbeitung vom 15. Januar 1998

Bearbeiter:

- Gerhard Müller (email: muellerg@informatik.tu-muenchen.de)
- Maximilan Fischer (email: fischmax@informatik.tu-muenchen.de)
- Viktor Schuppan (email: schuppan@informatik.tu-muenchen.de)
- Ramin Göttlich (email: goettlich@shs.de)

Dieses Dokument ist im Internet unter der URL
http://www.informatik.tu-muenchen.de/~muellerg/
docs/Praktika/RechnernetzeX25/ verfügbar

Übersicht

Diese Ausarbeitung beantwortet in Stichpunkten die in den Praktikumsaufgaben gestellten theoretischen Fragen und stellt die Ergebnisse der Praktikumsaufgaben dar.

Inhaltsverzeichnis

1 V.24 und Modems (Ebene 1)

1.1 Theorie

1.1.1 Modellbildung, Einordnung von V.24, Nullmodem

(1) **Erklärung des Aufbaus von Nachrichten:**

Dargestellt ist hier die Übertragung von der (Instanz der) Schicht N+1 zur Schicht N-1, also dem Sendevorgang. Der Empfangsvorgang läuft entsprechend umgekehrt ab.

Grundlage sind die Nutzdaten der (N+1)ten Schicht, die übertragen werden sollen.

Diese Daten werden dem Dienstzugangspunkt der (Instanz der) Schicht N (also (N)-SAP, *service access point*) übergeben, wobei Kommandos bzw. Steuerinformationen für die (Instanz der) Schicht N über die ICI (*interface control information*) übergeben werden. Die Nutzdaten werden in diesem Zusammenhang auch (N)-SDU (*service data unit*) genannt.

Die (Instanz der) Schicht N baut aus den PCI (*protocol control information*, die protokollsteuernde Informationen enthalten, und den Nutzdaten eine (N)-PDU (*protocol data unit*) auf. Diese Daten werden nun "virtuell" an die Partnerinstanz der Schicht N des empfangenden Gerätes gesendet. Dies wird realisiert, indem die (N)-PDU als Nutzdaten über die (N-1)-SAP zusammen mit den ICI übergeben wird.

Erklärung der Begriffe:

ICI: Über die ICI wird die Instanz mit Steuerungsinformationen versorgt, die der darunterliegenden Schicht übergeben werden. Realisiert üblicherweise über Funktionsparameter, oder eine Struktur mit entsprechenden Werten.

(N)-SDU: Bezeichnung für die Daten, die eine N-Instanz an der (N)-SAP von einer Schicht (N+1) entgegen nimmt.

(N)-SAP: Bezeichnung für die Schnittstelle der Instanz der Schicht N zur Schicht (N+1) (also der höheren Schicht)

PCI: Protokollinformationen für die Partnerinstanz des Schicht-N-Protokolls auf der Empfängerseite.

(N)-PDU: Datenpaket (bestehend aus PCI und (N)-SDU), die über die (N-1)-SAP (als Nutzdaten) übertragen werden sollen.

(2) **Beziehung Realisierung für Schichten 1-3 nach CCITT und ISO-OSI-Modell**
ISO-OSI und CCITT sind nur schwer zu vergleichen. Insbesondere Schicht 1 macht Probleme ([Ker95, Seite 100]): bei CCITT existiert nur eine Instanz der Schicht 1: ein Teil der Schicht 1 wird durch ein Gerät, das Modem, verkörpert; daher gibt es nur ein Vorkommnis pro Übertragungsmedium. Die Unterscheidung zwischen Instanz und Modem ist hier nicht sinnvoll.

Bei den Schichten 2 und 3 gibt es gewisse Entsprechungen (ISO-OSI $\leftrightarrow$ CCITT) [Ker95, Seite 100]:

– Dienstelement ICI $\leftrightarrow$ Steuerleitungen zwischen DDE und DÜE
– Dienstelement SDU $\leftrightarrow$ Bitfolgen auf den Sende- bzw. Empfangsleitungen
– Protokollelement PCI $\leftrightarrow$ Bitmuster in der PDU oder Signal einer ansonsten nicht genutzten Frequenz (Frequenzmultiplexing) oder ein Signal besonderer Länge oder zeitlicher Lage (Zeitmultiplexing)
– Protokollelement PDU $\leftrightarrow$ Bitfolgen

(3) **V.24-Schnittstellenanweisungen / Dienstelemente des OSI-Referenzmodells**
Die ICI (also die Protokollsteuerungsanweisungen) entsprechen bei V.24 RTS/CTS, bzw. DSR/DTR. Den SDU entsprechen die Daten auf den Leitungen TD und RD. Der Gesamtheit von ICI und SDU, also der PDU, entsprechen alle Daten auf den Leitungen zwischen zwei Geräten.

(4) **Definition Nullmodem und dessen Verwendung**
Unter einem Nullmodem versteht man eine Verbindung von zwei Rechnern, welche durch die Überkreuzung einiger Leitungen es ermöglicht, daß zwei DTEs trotz der asymmetrischen Schnittstelle V.24 miteinander kommunizieren können. Dies ist z.B. bei der direkten Koppelung zweier Rechner mit Hilfe eines Kabels nötig, im Gegensatz zur Verwendung von Modems.

1.1.2 Flußsteuerung, Hardware-Handshake, XON-XOFF-Protokoll

(1) **Definition Flußsteuerung, Verfahren, Hardware-Handshake, XON/XOFF, Unterschied Fluß- und Staukontrolle**
Flußsteuerung ist nötig, um die Überlastung eines Empfängers durch einen Sender zu verhindern. Dies kann auf verschiedene Arten geschehen (Sliding-Window-Protokoll, Start/Stop-Befehle (z.B. RTS/CTS), durch Kreditvergabe (selten) oder mit einer expliziten vorherigen Pufferreservierung).

Beim Sliding-Window-Protokoll wird dem Sender erlaubt, eine gewisse Anzahl von Paketen zu verschicken, die der Empfänger noch nicht bestätigt hat. Wenn diese erreicht ist, muß der Sender auf eine Empfangsbestätigung warten (bzw. nach Ablauf eines Timers die noch nicht bestätigten Pakete nochmals versenden). Die Start/Stop-Methode kann als ein Sliding-Window-Protokoll mit der Fenstergröße 1 verstanden werden, d.h. bevor ein weiteres Paket gesendet werden darf, muß das Vorherige bestätigt worden sein.

Bei der Kreditvergabe gibt es eine feste Anzahl von “Krediten”. Der Empfänger sendet dem Sender eine spezielle Nachricht, die aussagt, wieviele Pakete der Sender dem Empfänger ohne Empfangsbestätigung senden darf. Wenn die Kredite “verbraucht” sind, darf kein weiteres Paket gesendet werden.

Bei der expliziten Pufferreservierung weiß der Sender im Vorhinein, wieviel Platz der Empfänger für Nachrichten hat, und kann sich entsprechend darauf einrichten.

(2) **Beispiel für Notwendigkeit Flußsteuerung**
Druckersteuerung, Modemsteuerung, ...

(3) **Signale der V.24-Schnittstelle für Flußsteuerung in Hardware**
In-Band-Signalling: XON/XOFF

Out-of-Band-Signalling: CTS/RTS oder DSR/DTR

(4) **XON-XOFF-Protokoll**
Wenn der Empfänger kurz davor ist, keine Daten mehr empfangen zu können, schickt er dem Sender ein entsprechendes Zeichen (XOFF). Hierbei ist die Zeit zu berücksichtigen, die das Zeichen XOFF braucht, bis der Sender die Übertragung einstellt. Falls der Empfänger wieder in der Lage ist, Daten zu empfangen, sendet er das XON-Zeichen.

(5) **Vergleich XON-XOFF und Hardware-Handshake**
Vorteile Hardwaresteuerung:

- Unabhängig von den Daten (alle Daten können unverändert übertragen werden), bei SW-Steuerung müssen die Daten eventuell mit Fluchtsequenzen verändert werden
- schnelle Reaktion des Senders ist garantiert (keine Verarbeitung des empfangenen Zeichens nötig)
- Kann durch billige Hardware realisiert werden, besonders vorteilhaft bei Geräten, die wenig Eigenintelligenz besitzen.

Vorteile Softwaresteuerung:

- Geringerer Hardware-Aufwand
- Einfache Implementierung
- Eindeutige Steuerung (bei Hardware: wird nun RTS/CTS oder DSR/DTR verwendet?)

1.2 Praktische Versuche

1.2.1 Versuch I: V.24-Schnittstelle

(1) **Versuchsaufbau**
durchgeführt

(2) **Flußsteuerung**
Die Flußsteuerung wurde über das Auftrennen der Leitungen 5/CTS, 6/DSR (auf Seite des PCs) und Leitung 20/DTR durchgeführt. Als Daten wurden mit Hilfe des Programmes `te` “e”s vom PC an das Ampex-Terminal geschickt. Die Reaktion des Senders auf das

Auftrennen der Leitungen war (erstaunlicherweise) immer gleich: wenn die Unterbrechung kurz genug war, setzte der Sender(=PC) die Sendung fort, nach längerer Unterbrechung (ca. 3 Sekunden) wurde die Sendung mit einer Fehlermeldung abgebrochen.

Erwartet hätten wir, daß die Übertragung auch nach längerer Unterbrechung von DSR bzw. DTR wieder fortgesetzt würde, während die Unterbrechung von CTS die Sendung erwartungsgemäß unterbricht.

(3) **XON/XOFF**
Die Überschrift zu dem Versuch ist irreführend, nach Anleitung hat dieser Versuch nichts mit XON/XOFF zu tun. Die Auftrennung von Leitungen führt zu den unter (2) beschriebenen Ergebnissen, es wird kein besonderes Zeichen generiert.

Es war uns nicht möglich über Tastaturbefehle (CTRL-S/Q) die Übertragung vom Ampex-Terminal zu steuern.

1.2.2 Versuch II: Modemstrecke

(1) **Versuchsaufbau**
durchgeführt

(2) **Darstellung der modulierten Signale**
Die Unterschiede waren relativ gering (kurze Kabel), jedoch war eine leichte Überschwingung des Empfangssignals im Vergleich zum Sendesignal zu beobachten. Es ist anzunehmen, daß die Bandbreite der Leitung mit zunehmender Frequenz sinkt, da der ursprüngliche Signalverlauf immer mehr "verschmiert". Die Dämpfung der Leitung nimmt mit zunehmender Frequenz zu, d.h. Kabel erlauben nur die Übertragung von Signalen eines begrenzten Signalspektrums.

(3) **Modulationsverfahren**
Modulationsverfahren: Pulsbreitenmodulation

Das HDLC-Flag ist `0111 1110`, welches kontinuierlich übertragen wird, so lange keine Daten gesendet werden. Das Signal sieht wie folgt aus:

```
 |---| |-|   |-|   |-|   |-|   |-|   |-|   |---| |---|
 |   | | |   | |   | |   | |   | |   | |   |   | |   |
-|   |_| |___| |___| |___| |___| |___| |___|   |_|   |_ ...
   0     1     1     1     1     1     1     0      ...
```

(4) **Bitrate**
Die gemessene Bitrate lag zwischen 9300 Bit/s und 9900 Bit/s im Durchschnitt also ungefähr bei 9600 Bit/s, welches mit den 9600 Baud des Modems gut übereinstimmt (9600 Schritte pro Sekunde, übertragene Information jeweils 1 bit).
Die Spannungsdifferenz lag bei 2.3 Volt, die Länge eines Bits bei ungefähr 110 μs.

2 Sicherungsprotokolle (Ebene 2)

2.1 Theorie

2.1.1 Synchrone und Asynchrone Übertragung

(1) **Definition synchrone Übertragung, asynchrone Übertragung, Start-Stop-Übertragung**
Bei der *synchronen Übertragung* haben Sender und Empfänger den gleichen Takt, d.h. während der gesamten Übertragung (bzw. der Verbindung) besteht Schrittgleichlauf.

Bei der *asynchronen Übertragung* haben Sender und Empfänger nur während der aktuellen Übertragung (z.B. Frame oder Zeichen) den gleichen Takt. Bei verschiedenen Übertragungen muß kein Schrittgleichlauf vorliegen.

Bei der *Start-Stop-Übertragung* wird jedes einzelne Zeichen einzeln synchronisiert: durch das Startbit wird die Synchronisation während eines Zeichens "gestartet", durch das Stopbit beendet, der Empfang also bis zum nächsten Startbit beendet.

(2) **Prinzip Übertragung zwischen PC und Terminal**
Zwischen dem PC und externem Terminal wird üblicherweise die Start-Stop-Übertragung verwendet.

(3) **Vergleich synchrone/asynchrone Übertragung**
Vorteile synchroner Übertragung:

+ weniger Overhead (Start/Stop-Bits entfallen) = mehr Durchsatz

Vorteile asynchroner Übertragung:

+ einfach zu implementieren (Hardware-Aufwand), da keine Synchronisation während einer längeren Zeitdauer zu gewährleisten ist (Taktgeneratoren)
+ daher oft günstiger

2.1.2 Protokolle der OSI-Schicht 2, HDLC-LAP B, Sichere Übertragung über gestörte Kanäle

(1) **Klassen von Schicht-2-Protokollen**

a) bitorientierte Prozeduren
b) zeichenorientierte Prozeduren

Framing: bei a): durch Anfangs- und Endflags; in der Übertragung werden Flags durch Bit-Stopfung (Bit-Stuffing) vermieden
bei b): verschiedene Sonderzeichen werden verwendet (STX, ETX etc.)

Fehlerbehandlung: bei a): üblicherweise CRC (Cyclic Redundancy Check), verschiedene Längen/Polynome, kann wesentlich mehr Fehler erkennen als Parity/BCC, jedoch keine korrigieren.
bei b): üblicherweise Parity (pro Zeichen), BCC (Block Check Character), kann 1-Bit-Fehler korrigieren und 2-Bit-Fehler erkennen.

Quittierungsmechanismen: bei a): üblich: Fenstertechnik (Sliding Window Protocol) bei b): Numerierung der Datenblöcke, jeder Block wird einzeln quittiert

Vor/Nachteile von bitorientierten Prozeduren:

+ Codeunabhängigkeit der Übertragung, daher weniger Verwaltungsaufwand im Endgerät, Datenmenge unabhängiger von zu übertragenen Daten
+ CRC erkennt mehr Fehler (insbesondere bei Fehler-"Bursts")
+ bessere Auslastung der Leitungen (mehrere Pakete können bei der Fenstertechnik "unterwegs" sein)

Vor/Nachteile von zeichenorientierten Prozeduren:

+ recht einfach zu implementieren (heute eigentlich kein Thema mehr)
+ weniger Intelligenz erforderlich (heute eigentlich kein Thema mehr)
– nur für Terminals sinnvoll einsetzbar

Beispiele:

- bitorientiert: LLC (Logical Link Control, MAC-Schicht)
- zeichenorientiert: 3270-Terminals

(2) **Fehlererkennungsverfahren**

- BCC-Verfahren:
  * horizontale und vertikale Paritätsbits
  * damit Korrektur aller 1-Bit-Fehler und Erkennung aller 2-Bit-Fehler
  * Problematisch bei Fehler-Bursts
  * Beispiel: BSC (IBM)
- CRC-Verfahren:
  * Sender berechnet CRC:
    · zu übertragende Daten werden als Polynom betrachtet und mit dem Grad des Generatorpolynoms multipliziert (Effekt: Verschiebung nach links um n Bits, wobei n der Grad des Generatorpolynoms ist).
    · Das so veränderte Datenpolynom wird durch das Generatorpolynom dividiert

· Divisionsrest ist CRC

* Daten und CRC werden übertragen
* Empfänger berechnet CRC der empfangenen Daten (Daten werden als Polynom betrachtet und durch das bekannte Generatorpolynom dividiert)
* falls CRC = 0, wird vom korrekten Empfang ausgegangen
* ansonsten muß Paket noch mal übertragen werden

(3) **Generatorpolynom bei HDLC**
Bei HDLC wird das 16-Bit-CRC-CCITT-Standardgeneratorpolynom $x^{16} + x^{12} + x^5 + 1$ verwendet (Quelle: [Hal95, 240])

Erkannt werden damit folgende Fehler: (Begründung Literatur, z.B. [Tan96, Seite 190]

- alle 1-Bit-Fehler
- alle 2-Bit-Fehler
- alle ungeraden Bit-Fehler
- alle Fehlerbursts der Länger ≤ 16
- 99.997 % aller 17-Bit-Fehlerbursts
- 99.998 % aller $\geq$ 18-Bit-Fehlerbursts

(4) **CRC-Beispiel durchrechnen**
Verfahren analog [Hal95, Seite 132] bzw. [Tan96, Seite 189]

```
Frame: 1101011101   Generatorpolynom: 10011
                    (Frame wird um 4 0-Bits erweitert)

10011 | 1101011101 0000 = 1100001
        10011           => 1
        -----
         10011
         10011          => 1
         -----
          00001
          00000         => 0
          -----
           00011
           00000        => 0
           -----
            00110
            00000       => 0
            -----
             01101
             00000      => 0
             -----
```

```
         1101 0
         1001 1    => 1
         ---- -
          100 10
          100 11   => 1
          --- --
           00 010
           00 000  => 0
           -- ---
            0 0100
            0 0000 => 0
            - ----
              0100
              ====
```

Damit wird 1101011101 0100 übertragen.

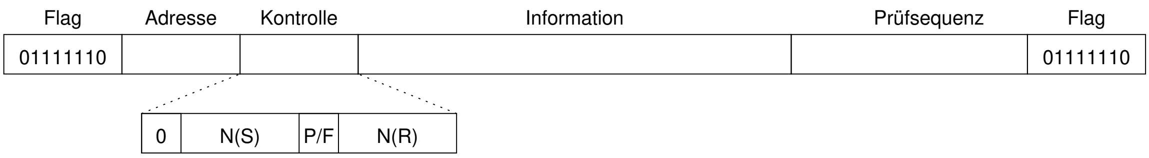
Überprüfung von 11101111001 0101:

```
Frame: 11101111001 0101   Generatorpolynom: 10011
10011 | 11101111001 0101 = 11111111001 Rest 1110
       10011           => 1
       -----
        11101
        10011          => 1
        -----
         11101
         10011         => 1
         -----
          11101
          10011        => 1
          -----
           11100
           10011       => 1
           -----
            11110
            10011      => 1
            -----
             11011
             10011     => 1
             -----
              1000 0
              1001 1   => 1
              ---- -
               001 11
               000 00  => 0
```

```
--- --
 01 110
 00 000 => 0
 -- ---
  1 1101
  1 0011=> 1
    ----
    1110
    ====
```

Der Rest ist von 0 verschieden, also ist die zweite Nachricht nicht korrekt übertragen worden.

(5) **Aufbau I-Frame bei HDLC, Bitstuffing**



Ein I(nformations)-Frame besteht aus einem besonderen Start-of-frame- und End-of-Frame-Bitmuster am Anfang und Ende (je 8 Bit, 01111110), einer Adresse (8 Bit), einem 0-Bit (Kennzeichnung für ein I-Frame), einer Sende-Sequenznummer (3 Bit, N(S)), einer Empfangssequenznummer (3 Bit, R(S)), einem Poll/Final Bit, sowie den zu übertragenden Informationen, und einer CRC-Checksumme.

Bit-Stuffing wird verwendet, um den Anfang und das Ende eines Frames eindeutig zu kennzeichnen. Dabei wird in HDLC ein spezielles Bitmuster verwendet (01111110), welches sechs 1er-Bits beinhaltet. Um nun in der Datenübertragung (also zwischen Start-of-frame und End-of-Frame) ein versehentliches Erkennen von End-of-Frame zu verhindern, und um die Übertragung der Daten transparent zu machen, wird nach jeweils fünf 1-Bits ein 0-Bit “eingeschoben”. So kann das Bit-Muster 01111110 nie innerhalb eines Frames gesendet werden. Der Empfänger muß innerhalb eines Frames jeweils nach fünf 1-Bits das folgende 0-Bit entfernen.

Bitstuffing ist bei BSC überflüssig, da hier spezielle Steuerzeichen reserviert sind, die nicht innerhalb der Daten vorkommen dürfen (außer durch die vorherige Benutzung von Escape-Sequenzen).

(6) **Flußsteuerung in HDLC**

In HDLC wird ein Sliding-Window-Protokoll zur Flußsteuerung verwendet. Desweiteren kann durch RNR/RR die Stop-and-Go-Technik benutzt werden.

(7) **Ungleichung**

Die angegebene Ungleichung beschreibt die Fenstergröße im Sliding-Window-Protokoll.

0 ist die (nicht erreichbare) untere Grenze für den Sequenznummernunterschied zwischen Empfänger und Sender, W die konfigurierbare obere Grenze (Fenstergröße).

V(S) beschreibt die Sendesequenznummer des Senders, die dem Empfänger in N(S) mitgeteilt wird. V(S) wird nach dem Senden eines I-Frames um eins (modulo 8) erhöht wird.

V(R) beschreibt beim Empfänger, welches I-Frames als nächstes von der Gegenseite erwartet wird, V(R)-1 Frames wurden also richtig empfangen. Dem Empfänger wird also durch N(R) mitgeteilt, welche Sequenznummer der Sender als nächstes vom Empfänger erwartet.

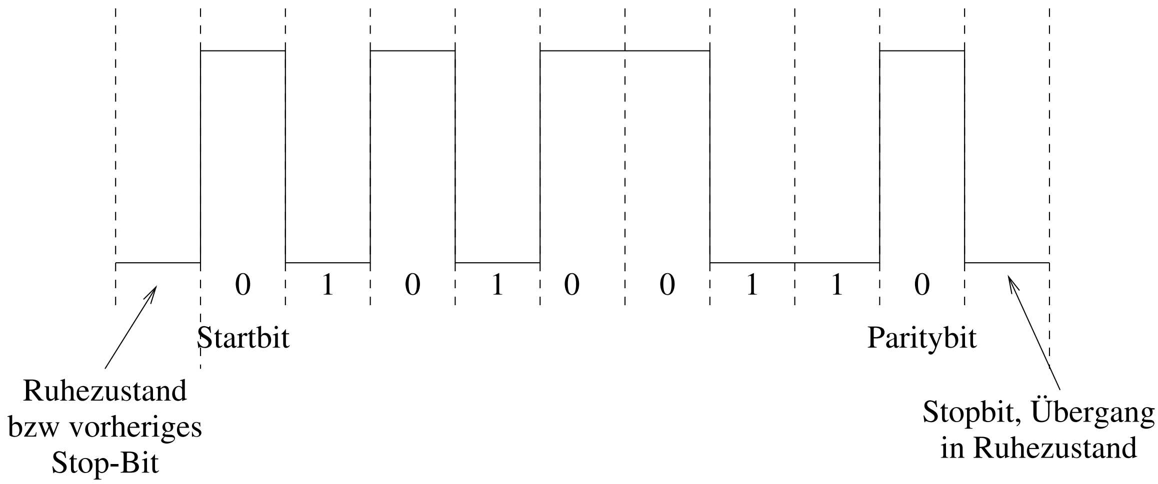
Falls die Sequenznummern zu weit auseinanderliegen (größer oder gleich W), werden die Pakete erneut übertragen.

2.2 Praktische Versuche

2.2.1 Versuch I: Asynchrone Übertragung

(1) **Versuchsaufbau**
durchgeführt

(2) **Aufzeichnung eines asynchronen Zeichenrahmens**
Mittels eines Oszilloskops wurde ein "e" (Binär 1100101 im ASCII-Code) dargestellt. Folgendes Bild war zu sehen:



Die Bits werden also "falsch herum" übertragen, also von hinten nach vorne. Eine 1 wird durch *keinen* Strom übertragen, eine 0 durch eine Spannung von 20 V.

2.2.2 Versuch II: HDLC-Protokollanalyse

(1) **Versuchsaufbau**
durchgeführt

(2) **Analyse eines HDLC-Normalablaufs**

Im Versuch wurde ein von “blue” ein Text “bbb” eingegeben. Der Ablauf konnte mittels des Protokollanalysators aufgezeichnet werden:

```
Verbindungs-  ADD  CODE      NS  PF  NR  DATA    Kommentar
  richtung

Verbindungsaufbau
-----------------

red->blue    01    SARM/DM        0              red w"unscht Verbindungsaufbau,
                                                  Set Asynchronous Response Mode
blue->red    01    SABM           1              blue will Asynchronous Balanced
                                                  Response Mode, will sofort Antwort
red->blue    01    UA             1              red akzeptiert und sendet einen
                                                  unnumbered acknowledge (es gibt noch
                                                  keine initialisierten Sequenznummern),
                                                  will ebenfalls sofort Antwort
red->blue    03    INFO      0    0   0          N(S) und N(R) werden gegenseitig
blue->red    01    INFO      0    0   0           auf 0 initialisiert
blue->red    03    RR             0   1          Der Gegenseite wird jeweils mitgeteilt,
red->blue    01    RR             0   1           da"s man bereit ist, Daten zu empfangen


Daten"ubertragung red->blue
--------------------------

red->blue    03    INFO      1    0   1  bbb     Nutzdaten von red an blue
blue->red    03    RR             0   2          blue best"atigt Empfang (N(R) hat sich erh"oht)
blue->red    01    INFO      1    0   1          blue sendet ``leere'' Daten an red
red->blue    01    RR             0   2          red best"atigt Empfang


Daten"ubertragung blue-> red
---------------------------

blue->red    01    INFO           0   2  ccc     Nutzdaten von blue an red
red->blue    01    RR        2    0   3          red best"atigt Empfang (N(R) hat sich erh"oht)
red->blue    03    INFO           0   2          red sendet ``leere'' Daten an blue
blue->red    03    RR        2    0   3          blue best"atigt Empfang
```

Kommandos (INFO z.B.) enthalten die Zieladressen (01 = DCE = red, 03 = DTE = blue), Meldungen die Absenderadressen (z.B. Receive Ready)

Es werden alle Kommandos durch Meldungen bestätigt.

Das Poll/Final Bit ist gesetzt, wenn der Sender sofort eine Antwort wünscht. Dies ist nur beim Verbindungsaufbau der Fall, alle anderen Pakete können implizit durch die Sequenznummern bestätigt werden.

Die Prüfsequenz, CRC, wurde hier weggelassen; sie dient zur Überprüfung aller Frames; fehlerhafte Frames werden verworfen.

Das verwendete Sequenznummernverfahren ist das Sliding-Window-Protokoll. Durch die beiden ersten INFO-Pakete werden die Sequenznummern auf 0 initialisiert.

Die Vorteile von endlichen Sequenznummernräumen sind die sehr einfache Implementierung, die Länge der Sequenznummern ist bekannt, die Protokolle werden dadurch einfacher; außerdem gibt es keine Probleme mit beschränkten Paketlängen. Nachteilig ist die Problematik der Eindeutigkeit: Sequenznummern dürfen nur nach einer bestimmten Zeit wiederverwendet werden, da es sonst zu Uneindeutigkeiten kommen kann. In der Praxis wird dem dadurch begegnet, daß Pakete eine maximale Lebensdauer bekommen und Sequenznummern erst nach einer Quittung bzw. der maximalen Lebensdauer wiederverwendet werden.

(3) **Analyse eines gestörten Ablaufs**

Mittels des Protokollanalysators wurde eine Verbindung gestört. Das Ergebnis (mit Kommentaren) sah wie folgt aus:

```
Verbindungs-  ADD  CODE      NS  PF  NR  DATA    Kommentar
  richtung

[...]

red->blue    03    INFO       2   0   3  eee...  Normaler Datenaustausch...
blue->red    03    RR             0   3          Best"atigung
blue->red    01    INFO       3   0   3          blue sendet ``leere'' Daten an red
red->blue    01    RR             0   4          red best"atigt Empfang

  -- ST"ORUNG --

red->blue    03    INFO       3   0   4  eee...  Daten werden gesendet
                                                 es kommt keine Best"atigung, da Leitung
                                                 gest"ort wurde.
red->blue    03    INFO       4   0   4  eee...  Sliding Window erlaubt weitere Pakete
red->blue    03    RR             1   4          red will Empfang best"atigt haben
red->blue    03    RR             1   4          red will Empfang best"atigt haben
red->blue    03    RR             1   4          red will Empfang best"atigt haben
red->blue    03    RR             1   4          red will Empfang best"atigt haben
red->blue    03    RR             1   4          red will Empfang best"atigt haben
red->blue    03    RR             1   4          red will Empfang best"atigt haben
[...]
  -- ST"ORUNG BESEITIGT --

blue->red    03    RR             1   3          blue best"atigt aber nur bis Paket 2, erwartet
                                                 also Paket mit Nr. 3
red->blue    03    INFO       3   0   4  eee...  red sendet Daten noch mal
red->blue    03    INFO       4   0   4  eee...  red sendet Daten noch mal
blue->red    03    RR             0   4          blue best"atigt 3
blue->red    03    RR             0   5          blue best"atigt 4
blue->red    01    INFO       4   0   5          blue sendet ``leere'' Daten an red
red->blue    03    RR             0   5          red best"atigt Leerdaten
red->blue    03    INFO       5   0   5  eee...  red sendet Daten wieder normal

[...]
```

Die Fenstergröße scheint hier nur auf 2 Pakete eingestellt gewesen zu sein, da direkt nach der Störung und dem Senden zweier Pakete eine explizite Quittung von der Gegenseite angefordert wurde (PF=1 bei RR).

3 Vermittlung von X.25 (Ebene 3)

3.1 Theorie

3.1.1 Vermittlungsverfahren in Rechnernetzen

(1) **Vermittlungsverfahren und Wirkungsweise**
Es wird üblicherweise zwischen Leitungsvermittlung und der “store-and-forward”-Vermittlung unterschieden. Die Store-and-Forward-Vermittlung kann dabei entweder ganze Nachrichten vermitteln, oder nur einzelne Pakete. Die Paketvermittlung kann wieder unterschieden werden in normale Paketvermittlung und einer Vermittlung, wo die Pakete sehr klein sind und eine feste Länge haben. Die Pakete werden dann Zellen genannt (Zellenvermittlung).

Leitungsvermittlung entspricht dem expliziten Schalten einer Leitung (Verbindungsorientierung), die dann exklusiv zur Übertragung genutzt werden kann. Beispiel: Telefonnetz.

Bei der Streckenvermittlung werden Übertragungseinheiten von einem System über ggf. mehrere andere Systeme zu einem anderen System übertragen. Die Übertragung in einem Zwischensystem wird fortgesetzt, wenn eine Ausgangsleitung frei ist.

Bei der Nachrichtenvermittlung werden ganze Nachrichten als Einheit verwendet. Dieses war bei Telex üblich, wird heute jedoch kaum noch eingesetzt.

Üblich ist hingegen die Paketvermittlung, hier wird eine Nachricht in mehrere Pakete (die Übertragungseinheiten) je nach Bedarf aufgespalten; jedes Paket kann dabei ggf. einen anderen Weg vom Sender zum Ziel nehmen.

Der Spezialfall der Zellenvermittlung ist z.B. ATM.

(2) **Beispiele**

Leitungsvermittlung:	Telefonleitungen
Nachrichtenvermittlung:	Telex
Paketvermittlung:	TCP/IP, Datex-P, ...
Zellenvermittlung:	ATM

(3) **Vor- und Nachteile, sinnvolle Anwendungen**

- Leitungsvermittlung

Vorteile:	garantierte Bandbreite (QoS), einmalige Routing-Entscheidung, leichte Abrechnung, kein Overhead (kurze Übertragungszeiten), Reihenfolgegarantie
Nachteile:	u.U. schlechte Ausnutzung (teuer), bei kleinen Nachrichten hoher Overhead durch Aufbau einer expliziten Verbindung, keine Redundanz (bei Ausfall einer Zwischenstrecke muß Verbindung neu aufgebaut werden)
Sinnvolle Anwendung:	alles, was garantierte Bandbreiten benötigt: Telefon, Standleitungen, Multimedia-Übertragung unkomprimiert, Echtzeitsystemsteuerung

- Nachrichtenvermittlung

Vorteile:	billiger als Leitungsvermittlung durch gute Leitungsausnutzung, mehr Redundanz
Nachteile:	Länge von Nachrichten begrenzt (da sonst Zwischenpuffer unendlich groß sein mußte)
Sinnvolle Anwendung:	Telex, Telegramm-Übertragung

- Paketvermittlung

Vorteile:	sehr gute Ausnutzung der Leitungen, variable Datenraten können sinnvoll kombiniert werden (statistisches Multiplexen), hohe Redundanz, kein Verbindungsaufbau/abbau-Overhead
Nachteile:	Routingentscheidung in jedem System für jedes Paket, Paket-Header ist Overhead, Nutzdaten müssen in Paketform gebracht werden, Reihenfolge der Pakete muß überprüft und gesichert werden
Sinnvolle Anwendung:	Datenübertragung

- Zellenvermittlung

Vorteile:	geringere Übertragungszeit, besserer QoS
Nachteile:	hoher Overhead durch prozentual größeren Header pro Paket/Zelle
Sinnvolle Anwendung:	Kombination von verschiedenen traditionellen Netzen (ATM)

(4) **Virtuelle Verbindung, Datagramm, Arten von virtuellen Verbindungen** Eine Virtuelle Verbindung ist eine Verbindung (verbindungsorientiert), die nicht wirklich dauerhaft und exklusiv zur Verfügung steht, sondern sich die vorhandenen realen Ressourcen teilen muß. Für die Benutzer einer virtuellen Verbindung ist dies (bis auf mögliche Verzögerungen) transparent. In der Realisierung wird die Routingentscheidung beim Aufbau der virtuellen Verbindung getroffen; die Zwischensysteme halten die benötigten Daten. Wie bei allen großen verteilten Systemen ergeben sich dadurch Probleme beim Ausfall von Teilsystemen bzw. Strecken.

Ein Datagramm-Dienst ist ein Dienst, wo Nachrichten in Form von Paketen verschickt werden (verbindungslos). Jede Nachricht ist von allen anderen Nachrichten unabhängig. Wenn also ein verbindungsorientierter Dienst über einen Datagramm-Dienst realisiert werden soll, so muß auf Überholvorgänge, Verlust von Paketen, etc. geachtet werden und geeignete Gegenmaßnahmen ergriffen werden.

Folgende Tabelle (aus [Sch88, Seite 120] vergleicht zwischen virtuellen Verbindungen und Datagrammen:

	virtuelle Verbindungen	Datagramme
Zieladresse	nur während der Erstellung einer virtuellen Verbindung	muß in jedem Datenpaket enthalten sein
Fehlerkontrolle (Verlust und Verdopplung)	wird vom Netz durchgeführt	muß explizit von der Datenendeinrichtung durchgeführt werden
Flußkontrolle	wird vom Netz unterstützt	muß explizit in den Datenendeinrichtungen durchgeführt werden
Einhalten der Sequenz der Datenpakete	wird vom Netz garantiert	muß explizit von den Datenendeinrichtungen besorgt werden
Aufbau einer Verbindung	immer erforderlich	nur durch Zusammenarbeit der Datenendeinrichtungen möglich

Siehe auch Hilfsblatt zu “Vermittlungsschicht”.

3.1.2 Protokolle der Schicht 3, X.25 Paketebene

(1) **Funktionalität in Vermittlungsschicht**
Die Vermittlungsschicht trägt der Existenz von Transitsystemen Rechnung und hat vor allem folgende Aufgaben:

- Wegewahl
- Vermittlung

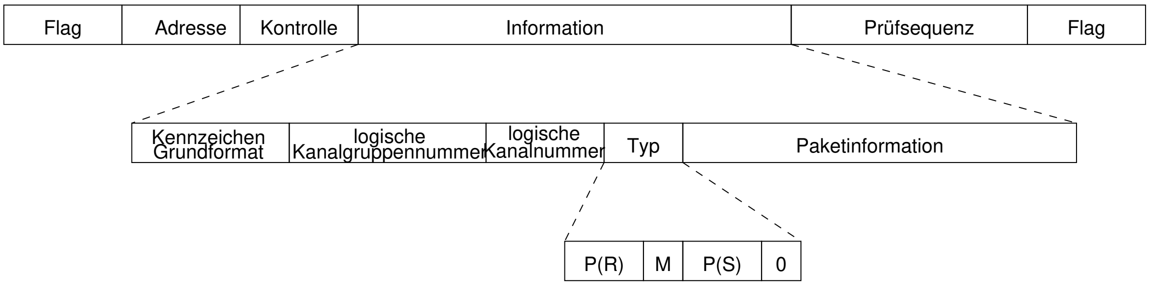
Desweiteren kümmert sie sich ggf. um Upward Multiplexing, Flußsteuerung, Reihenfolgesicherung und Dienstauswahl.

Eine genauere Beschreibung in Hinblick auf X.25 findet sich in [KB94, Seite 194].

(2) **Quality of Service, welche Parameter in Schicht 3** Verschiedene QoS-Parameter können ggf. in Schicht 3 festgelegt werden:

- Kosten
- Durchsatz (Geschwindigkeit)
- Übertragungsverzögerungszeit
- Fehlerrate
- Jitter
- Blockierwahrscheinlichkeit

(3) **Einbettung von Schicht-3-Paketen in Schicht-2-Pakete, Paketarten in Schicht 3**



Das Bild nach [KB94, Seite 200] zeigt ein Datenpaket in X.25.

In der Kennzeichnung des Grundformates kann angegeben werden, ob das Paket von der Gegenstelle bestätigt werden soll. Das Bit M gibt an, ob das Paket das letzte in einer Folge ist (M=0), oder nicht. P(S) und P(R) entsprechen den Folgezählern N(S) und N(R) in HDLC.

In X.25 werden folgende Gruppen von X.25-Paketen verwendet ([KB94, Seite 197]):

- Verbindungserstellung und -auslösung
- Daten und Unterbrechung
- Flußsteuerung und Rücksetzen
- Restart
- Diagnose

(4) **Mehrere Schicht-3-Verbindungen bzgl. Kanalbenennung**
In X.25 stehen

- Kanalgruppennummern, sowie
- Kanalnummern

zur Verfügung, die zur Realisierung des Multiplexens von Schicht-3-Verbindungen eingesetzt werden können.

(5) **Sequenznummernalgorithmus X.25-Schicht 3 vs. HDLC**
Die beiden Algorithmen funktionieren auf die gleiche Art und Weise.

3.1.3 X.25-Knoten, X.121-Adressierung

(1) **Funktion eines X.25-Knotens** Die Aufgaben eines X.25-Knotens sind Vermittlung und Wegewahl. Zusätzliche Funktionen (insbesondere in öffentlichen Netzen) sind Accounting und das Sammeln von statistischen Daten.

Die Funktionsweise eines X.25.Knotens entspricht zu großen Teilen denen eines ATM-Switches.

(2) **Komponenten des Addressierungsschemas nach X.121**
in mir zugänglicher gängiger Netzwerk-Literatur nicht zu finden

3.2 Praktische Versuche

3.2.1 Versuch I: PLP-Protokollanalyse

(1) **Versuchsaufbau**
durchgeführt

(2) **Protokollablauf-Analyse**

Der "Lebenszyklus" wird durchgesprochen:

- Verbindungsaufbau
  Call Request, wird bestätigt durch Call Confirm
  Eine logische Verbindung wird erstellt, indem Kanalnummern zugewiesen werden. Routing geschieht anhand der Adreßinformationen im Call Request-Paket durch den Knotenrechner. Ebene-3-Sende- und Empfangszähler werden initialisiert.
- Datenübertragung
  Flußsteuerung geschieht durch ein Sliding-Window-Protokoll, sowie explizit durch Receive Ready/Receive Not Ready, sowie implizit durch das Ausbleiben von Quittungen. Rücksetzten ggf. durch Reset Request/Reset Confirm, bewirkt ein Rücksetzen der Sequenznummern für die logische Verbindung, sowie Ignorierung aller noch im Netz befindlichen Pakete.
- Verbindungsabbau
  Wird durch Clear Request initiiert und durch Clear Confirm bestätigt. Alle an der Verbindung beteiligten Ressourcen werden freigegeben (Puffer, logische Kanalnummer auf allen Teilstrecken etc.)

(3) **Statistisches Multiplexen**
Wie bei ATM werden mehrere Schicht-3-Verbindung durch statistisches Multiplexen (mittels logischer Kanäle) auf eine Schicht-2-Verbindung abgebildet. Da beim statistischen Multiplexen die Aufteilung der Kanäle nicht fest ist (im Gegensatz zu z.B. SDH, Time Division Multiplexing), muß in jedem Paket die Kanalnummer enthalten sein.

3.2.2 Versuch II: Protokoll-Overhead

(1) **Versuchsaufbau**
durchgeführt

(2) **Overhead-Berechnung**
Die übertragenen Daten bestehen aus Verbindungsaufbau-Daten, Nutzdatenübertragung, sowie Verbindungsabbau-Daten.

Verbindungsaufbau:

Call Request	29 Bytes
Quittung (HDLC)	6 Bytes
Call Confirm	9 Bytes
Quittung (HDLC)	6 Bytes
Summe	50 Bytes

Datenübertragung:

– 20-Zeichen-Pakete

Info-Paket	30 Bytes
Quittung (HDLC)	6 Bytes
Quittung (Ebene 3)	9 Bytes
Quittung (HDLC)	6 Bytes
Summe pro Paket	51 Bytes

– 40-Zeichen-Pakete

Info-Paket	50 Bytes
Quittung (HDLC)	6 Bytes
Quittung (Ebene 3)	9 Bytes
Quittung (HDLC)	6 Bytes
Summe pro Paket	71 Bytes

– 300-Zeichen-Pakete (Übertragung als 3 Pakete, da maximale Informationsmenge 128 Byte)

Info-Paket	110 Bytes
Quittung (HDLC)	6 Bytes
Quittung (Ebene 3)	9 Bytes
Quittung (HDLC)	6 Bytes
Info-Paket	110 Bytes
Quittung (HDLC)	6 Bytes
Quittung (Ebene 3)	9 Bytes
Quittung (HDLC)	6 Bytes
Info-Paket	110 Bytes
Quittung (HDLC)	6 Bytes
Quittung (Ebene 3)	9 Bytes
Quittung (HDLC)	6 Bytes
Summe pro Paket	393 Bytes

Verbindungsabbau:

Clear Request	11 Bytes
Quittung (HDLC)	6 Bytes
Clear Confirm	9 Bytes
Quittung (HDLC)	6 Bytes
Summe	32 Bytes

Berechnung insgesamt:

Übertragungsbestandteil	**Anzahl**	**Bytes pro Anzahl**	**Summe**	**Nutzinformationen**
Verbindungsaufbau	1	50	50	0
20-Zeichen-Pakete	40	51	2040	800
300-Zeichen-Pakete	20	393	7860	6000
40-Zeichen-Pakete	20	71	1420	800
Verbindungsabbau	1	32	32	0
Summe			11402	7600

Insgesamt müssen also 11402 Bytes übertragen werden (Ebene 2 und 3), um 7600 Bytes an Nutzinformationen bei dem angegebenen Szenario zu übertragen. Dies entspricht einem Overhead von 3802 Bytes, oder ziemlich genau 50%.

3.2.3 Versuch III: Switch-Konfiguration

(1) **Versuchsaufbau**
Der Versuchsaufbau konnte wegen eines Defektes am Switch *nicht* durchgeführt werden.

(2) **Konfiguration der Wegewahl-Tabellen**
Die Konfiguration des Switches konnte wegen des defekten Switches *nicht* vorgenommen werden.

Literatur

[Bad97] BADACH, ANATOL: *Integrierte Unternehmensnetze*. Hüthig, Heidelberg, 1997. ISBN 3-7785-2562-X.

[HA93] HEGERING, HEINZ-GERD und SEBASTIAN ABECK: *Integriertes Netz- und Systemmanagement*. Addison-Wesley, Bonn, 1993. ISBN 3-89319-508-4.

[Hal95] HALSALL, FRED: *Data Communications, Computer Networks and Open Systems*. Addison-Wesley, Harlow, England, fourth edition, 1995. ISBN 0-201-42293-X.

[Hel97] HELD, GILBERT: *Virtual LANs: Construction, Implementation, and Management*. John Wiley & Sons, Bonn, 1997. ISBN 0-471-17732-6.

[KB94] KOWALK, WOLFGANG und MANFRED BURKE: *Rechnernetze*. B. G. Teubner, Stuttgart, 1994. ISBN 3-519-02141-2.

[Ker95] KERNER, HELMUT: *Rechnernetze nach OSI*. Addison-Wesley, Bonn, 3. Auflage, 1995. ISBN 3-89319-801-6.

[Sch88] SCHICKER, PIETRO: *Datenübertragung und Rechnernetze*. B. G. Teubner, Stuttgart, 3. Auflage, 1988. ISBN 3-519-22463-1.

[Tan96] TANENBAUM, ANDREW S.: *Computer Networks*. Prentice-Hall, third edition, 1996. ISBN 0-13-394248-1.

[WWW97a] *International Standards Organization ISO*, 1997. `http://www.iso.ch/`.

[WWW97b] *International Telecommunications Union ITU*, 1997. `http://www.itu.ch/`.