2 Nachrichtenmechanismen

- Spezielle Literatur

Comer, D.: Operating System Design - Volume II, Internetworking with Xinu. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

Comer, Douglas and Stevens, David L.: Internetworking with TCP/IP Vol. II (2nd Edition) 1994 ISBN 0-13-125527-4

Fletcher, J. G.: Serial Link Protocol Design: A Critique of the X.25 Standard, Level 2. Computer Communication Review, Vol. 14, No. 2, 1984, pp. 26-33.

- Einschägige Links
  - Protokolle
    http://www.protocols.com
    Enthält unter anderem ein Poster mit standardisierten Protokollen
  - XINU-Hompage
    http://willow.canberra.edu.au/~chrisc/xinu.html

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Standards

  Niedergelegt in mehreren “Request for Comment (RFC)"
  http://www.faqs.org/rfcs/rfc-index.html

IP	RFC 791	September	1981	51 Seiten
UDP	RFC 768	August	1980	4 Seiten
ARP	RFC 826	November	1982	10 Seiten
RARP	RFC 903	Juni	1984	4 Seiten
ICMP	RFC 792	September	1981	22 Seiten
TCP	RFC 793	September	1981	85 Seiten

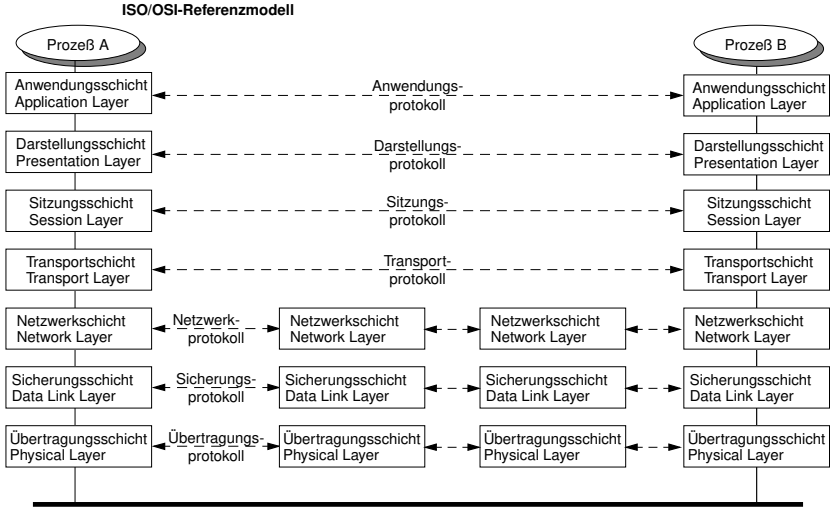
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Untergliederung

2.1 Strukturen der Kommunikation
2.2 Beispiel einer Implementierung
2.2.1 Maschinensprachebene
2.2.2 Das Internet
2.2.3 Struktur der Systemsoftware und ihre Realisierung
2.2.4 Systemanlauf, Adreßauflösung
2.2.5 Internet-Protokoll (IP)
2.2.6 Internet-Kontrollnachrichten (ICMP)
2.2.7 Datagrammdienst (UDP)
2.2.8 Verschiedene höhere Protokolle (TCP, FTP, SMTP)

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

2.1 Strukturen der Kommunikation



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

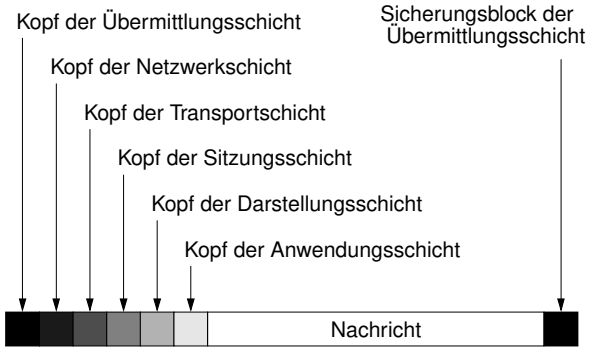
1. Übertragungsschicht
   - Ungesicherte Übertragung eines Bitstroms
   - Charakteristika der benutzten Signalformen sind Gegenstand der Standardisierung

2. Sicherungsschicht
   - Übertragung von Bitblöcken zwischen direkt physisch verbundenen Einheiten
   - Synchronisation, Fehler- und Flußkontrolle, evtl. Nachrichtenwiederholung

3. Netzwerkschicht
   - Virtuelle Verbindung zwischen zwei Einheiten, evtl. unter Zwischenschaltung anderer

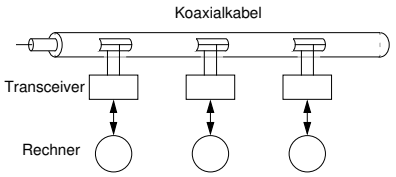
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Nachrichtenformat geschichteter Protokolle



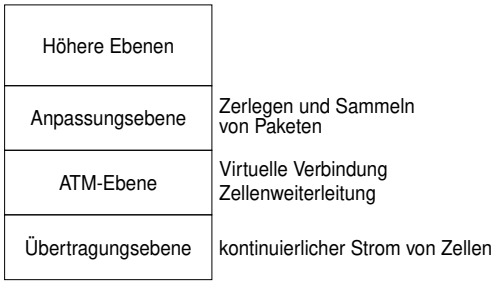
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Ethernet
  - Entwickelt von Xerox PARC ab etwa 1970
  - Standardisiert von Xerox, Intel und DEC 1978
  - Transaktionsorientiert
  - Zugriffsverfahren CSMA/CD (Carrier Sense Multiple Access with Collision Detection)
  - Pakete mit 46 bis 1522 Byte Nutzinformation
  - Übertragungsgeschwindigkeit
    - Standard: 10 MBit/s
    - Fast Ethernet: 100 MBit/s
    - Gigabit Ethernet: 1 GBit/s



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- ATM: Asynchronous Transfer Mode
  - Transaktionsorientiert
  - Zellen fester Größe (53 Byte) werden über virtuelle Verbindungen übertragen
  - Geeignet zur Übermittlung von Sprache, Video und digitalen Daten
  - Übertragungsraten reichen von 155 MBit/s bis 622 MBit/s
  - ATM-Referenzmodell

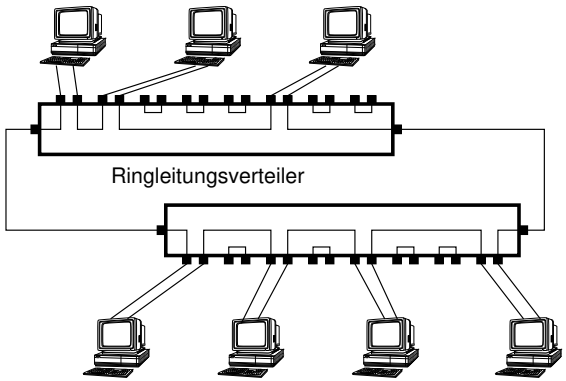


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Token-Bus
  - Token durchläuft ständig einen Ring und kann Daten mit sich führen.
    - Ein Knoten kann 'freies' Token belegen durch Anfügen von Daten.
    - Knoten, die ein 'belegtes' Token erhalten, können die Daten entnehmen; auf jeden Fall reichen sie es weiter, wenn sie es nicht belegt haben.
    - Wenn der belegende Knoten das Token erhält, gibt er es wieder frei.
  - Übertragungsraten
    - 4 Mbit/s
    - 16 MBit/s

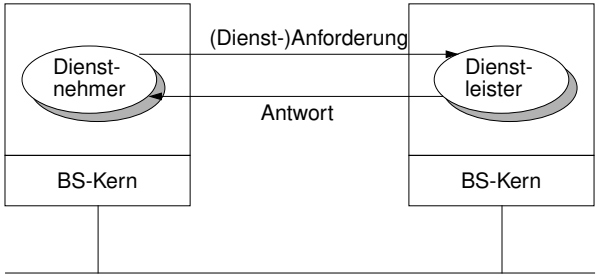
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Hardwarestruktur des 'Token Bus'



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

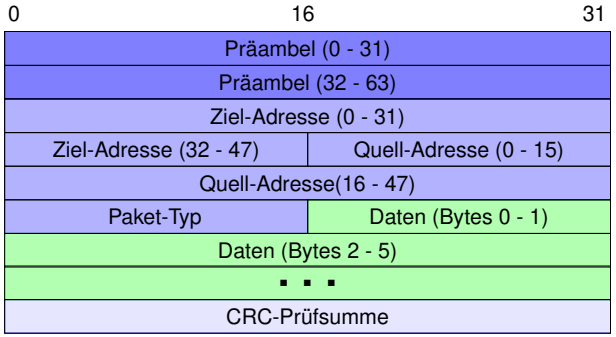
- Das Dienstnehmer-/Dienstleister-Modell (Client/Server Model)



- Problem: Adressierung

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Ethernet packet



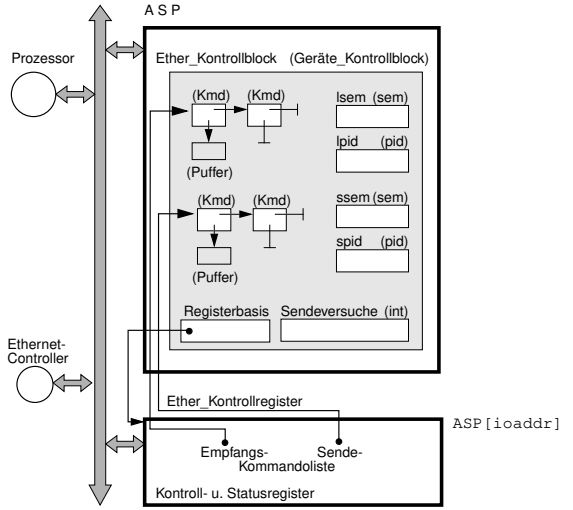
Ethernet Link-Level Protocol Format

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

```
/* 1 */    class EtherAddress {
/* 2 */      public static final int    EMINPAK =   64;     // minimum packet length in bytes
/* 3 */      public static final int    EMAXPAK = 1266;     // maximum packet length in bytes
/* 4 */      public static final int    EHLEN   =   14;     // size of ether header
/* 5 */      public static final int    EDLEN   = EMAXPAK - EHLEN;
/* 6 */      public static final int    EPADLEN =    6;     // length of phys. addresss
/* 7 */      public static final byte[] EBCAST  =   {(byte)0377, (byte)0377, (byte)0377,
/* 8 */                                              (byte)0377, (byte)0377, (byte)0377};
/* 9 */                                             //   Ethernet broadcast address
/*10*/      byte[] address = new byte[EPADLEN];
/*11*/    }
/*12*/
/*13*/   class EtherPacket {
/*14*/     // Structure of ethernet header
/*15*/     byte[]       preample = new byte[8];
/*16*/     EtherAddress e_dest;                          // destination host address
/*17*/     EtherAddress e_src;                           // source host address
/*18*/     short        e_ptype;                         // packet type
/*19*/       public static final int EP_IP    = 0x00000800; //  DARPA Internet protocol
/*20*/       public static final int EP_ARP   = 0x00000806; //  address resolution protocol
/*21*/       public static final int EP_RARP = 0x00008035; //  reverse addr. resol. protocol
/*22*/     byte[]       ep_data = new byte[len];         // 46 <= len <= 1522
/*23*/     unsigned crc;
/*24*/    }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Der Ethernet-Anschluß des Rechners am Beispiel des Digital Equipment Q-bus Network Adapter



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Die Register des DEQNA (Digital Equipment Q-bus Network Adapter)

Byte-Nr.	Bedeutung beim Lesen	Bedeutung beim Schreiben
0/1	Byte 0 der eigenen phys. Ethernetadr.	bedeutungslos
2/3	Byte 1 der eigenen phys. Ethernetadr.	bedeutungslos
4/5	Byte 2 der eigenen phys. Ethernetadr.	Bits 0 bis 15 der Adresse der Eingabebefehlsliste
6/7	Byte 3 der eigenen phys. Ethernetadr.	Bits 16 bis 21 der Adresse der Eingabebefehlsliste
8/9	Byte 4 der eigenen phys. Ethernetadr.	Bits 0 bis 15 der Adresse der Ausgabebefehlsliste
10/11	Byte 5 der eigenen phys. Ethernetadr.	Bits 16 bis 21 der Adresse der Ausgabebefehlsliste
12/13	Adresse des Interruptvektors	Adresse des Interruptvektors
14/15	Kontroll- und Statusregister	Kontroll- und Statusregister

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Bedeutung des Kontroll- und Statusregisters

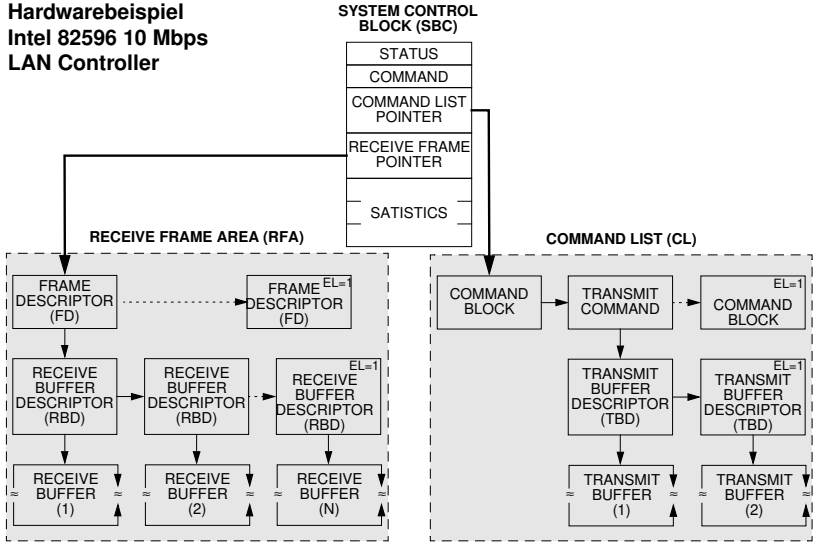
Bit	Bezeich-nung	Bedeutung
15	DQ_RINT	Eingabe-Interrupt (auf 1 gesetzt, wenn ein Eingabevorgang abgeschlossen ist, muß vom BS auf 0 gesetzt werden)
13	DQ_CARR	Trägersignal vorhanden (hat Wert 1, wenn Trägersignal vorhanden)
7	DQ_XINT	Ausgabe-Interrupt (auf 1 gesetzt, wenn ein Ausgabevorgang abgeschlossen ist, muß vom BS auf 0 gesetzt werden)
6	DQ_IEN	Interrupts erlaubt (Eingabe- und Ausgabe-Interrupt werden nur gesetzt, wenn dieses Bit 1 ist)
5	DQ_RLI	Eingabeliste ungültig (wird von der HW auf 1 gesetzt, wenn die Eingabeliste abgearbeitet ist)
4	DQ_XLI	Ausgabeliste ungültig (wird von der HW auf 1 gesetzt, wenn die Ausgabeliste abgearbeitet ist)
1	DQ_REST	Rücksetzen (mit 1 beschreiben, wenn die HW in Anfangszustand versetzt werden soll)
0	DQ_ENBL	Empfang erlaubt (muß nach dem Einschalten auf 1 gesetzt werden, um den Empfang von Paketen zu erlauben)

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Befehlsaufbau

Bytes 0/1	Indikator
Bytes 2/3	Bits 8 bis 15: Festlegung der Bedeutung der nachfolgenden Adresse (Pufferadresse, Befehlsadresse) bzw. Kennzeichnung als letzter Befehl Bits 0 bis 7: Bits 16 - 21 der Adresse
Bytes 4/5	Bits 0 bis 15 der Adresse
Bytes 6/7	Pufferlänge
Bytes 8/9	Zustandswort 1 (u. a. Fehleranzeigen)
Bytes 10/11	Zustandswort 2 (??)

Hardwarebeispiel
Intel 82596 10 Mbps
LAN Controller

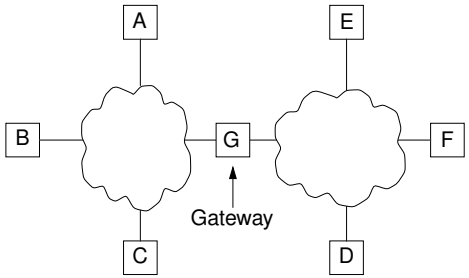


The textbook definitions of Ethernet have little to do with current practice.

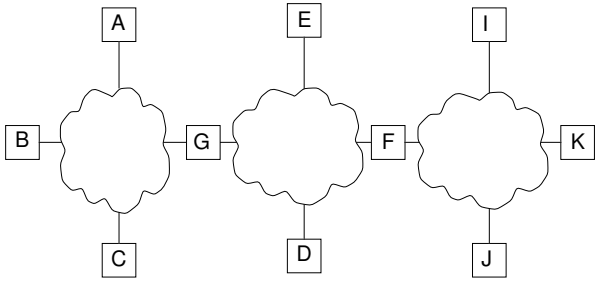
H. Gilbert, Yale

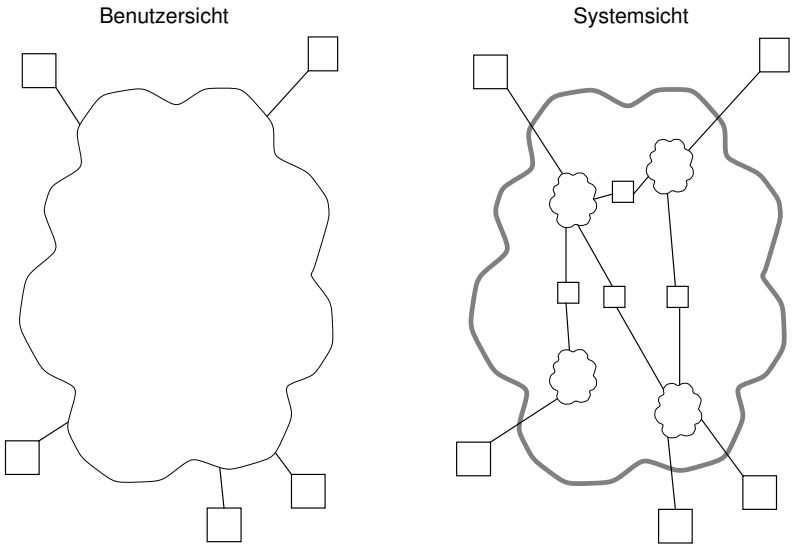
Although the OSI model is useful, the TCP/IP protocols don't match its structure exactly.

C. Hunt, In: TCP/IP, Network Administration.

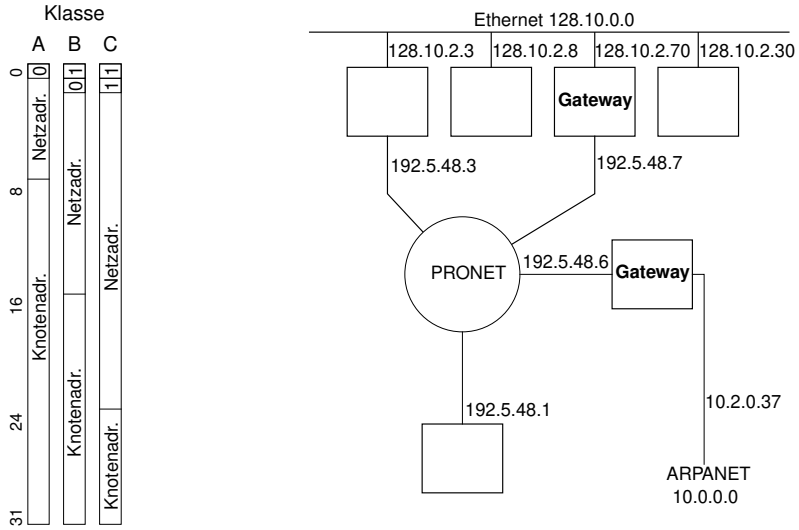


Verknüpfung von drei Netzwerken





IP-Adressen und ihre Darstellung



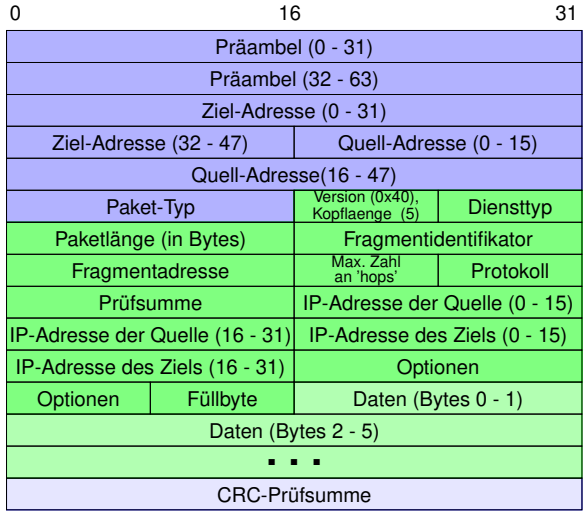
Aufbau von IP-Paketen

<table>
<tr><td colspan="2"></td><td>Version (0x40), Kopflaenge (5)</td><td>Diensttyp</td></tr>
<tr><td colspan="2">Paketlänge (in Bytes)</td><td colspan="2">Fragmentidentifikator</td></tr>
<tr><td colspan="2">Fragmentadresse</td><td>Max. Zahl an 'hops'</td><td>Protokoll</td></tr>
<tr><td colspan="2">Prüfsumme</td><td colspan="2">IP-Adresse der Quelle (0 - 15)</td></tr>
<tr><td colspan="2">IP-Adresse der Quelle (16 - 31)</td><td colspan="2">IP-Adresse des Ziels (0 - 15)</td></tr>
<tr><td colspan="2">IP-Adresse des Ziels (16 - 31)</td><td colspan="2">Optionen</td></tr>
<tr><td>Optionen</td><td>Füllbyte</td><td colspan="2">Daten (Bytes 0 - 1)</td></tr>
<tr><td colspan="4">Daten (Bytes 2 - 5)</td></tr>
<tr><td colspan="4">▪ ▪ ▪</td></tr>
</table>

IP packet

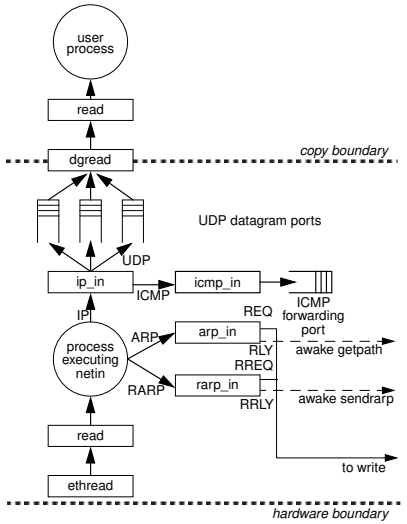
```
/* 1 */  public class IPPacket {
/* 2 */      static public final short IPHLEN = 20;
/* 3 */      static public final int IVERLEN = 0x45;
/* 4 */      static public final int ISVCTYP = 0;
/* 5 */      static public final int IFRAGOFF = 0;
/* 6 */      static public final int ITIM2LIV = 10;
/* 7 */
/* 8 */      byte      verlen;           // IP vers. (0x40) + hdr len in longs (5)
/* 9 */      byte      svctyp;           // service type (0 => normal service)
/*10*/      short     paclen;           // packet length in octets (bytes)
/*11*/      short     id;               // datagram id (to help gateways frag.)
/*12*/      short     fragoff;          // fragment offset (0 for 1st fragment)
/*13*/      byte      tim2liv;          // time to live in gateway hops (10)
/*14*/      byte      proto;            // IP protocol (UDP is assigned 17)
/*15*/      short     cksum;            // 1s compl. of sum of shorts in header
/*16*/      IPAddress src;              // IP address of source
/*17*/      IPAddress dest;             // IP address of destination
/*18*/
/*19*/      byte[]    data = new byte[paclen - 20];  // IP datagram data area
/*20*/  }
```

In Ethernet-Paket eingebettetes Internet-Paket



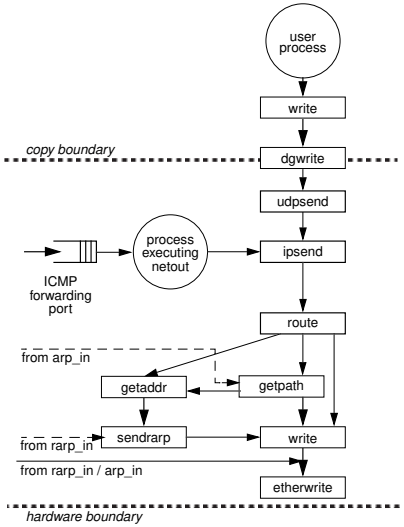
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Struktur der Empfangs-Seite



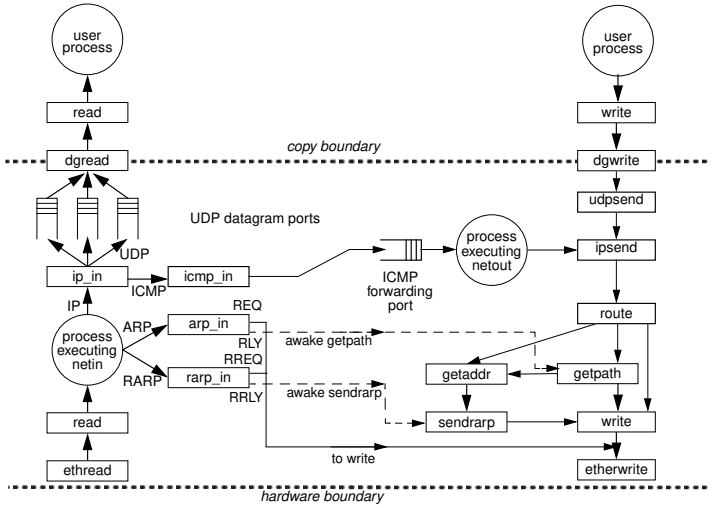
Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Stuktur der Sende-Seite



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Gesamtstruktur



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Methoden der Ethernet-Ebene

EtherControlBlock

```
public EtherControlBlock()

public int read(EtherPacket f_packet)

public int write(short type,
                 EtherAdress destnation,
                 byte[] message)

public void ethinter()
  // Systemthread!
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/* 1 */   public int read(EtherPacket f_packet) {
/* 2 */     int returncode = 0;
/* 3 */     etrsem.p();
/* 4 */     startreading(f_packet);
/* 5 */     etrpid = Thread.currentThread().getName();
/* 6 */     deqna.scheduler.suspend();
/* 7 */     Dcmd dcmptr = ercmd[0];
/* 8 */     if ((dcmptr.dc_st1 & Dcmd.DC_LUSE) == Dcmd.DC_ERRU) {
/* 9 */       returncode = Xinu.SYSERR;
/*10*/     } else {
/*11*/       returncode = 0;
/*12*/       int msglen = (L->dcmptr->dc_st1 & DC_HLEN) |
/*13*/                    (L->dcmptr->dc_st2 & DC_LLEN);
/*14*/     }
/*15*/     etrsem.v();
/*16*/     return returncode;
/*17*/   }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/* 1 */   public void startreading(EtherPacket f_packet) {
/* 2 */     Dcmd dcmptr;
/* 3 */     DqRegs dqptr;
/* 4 */     dcmptr = ercmd[0];
/* 5 */     dcmptr.dc_bufh = Dcmd.DC_VALID;
/* 6 */     dcmptr.dc_buf = f_packet;
/* 7 */     dcmptr.dc_len = f_packet.ep_length;
/* 8 */     dcmptr.dc_st1 = dcmptr.dc_st2 = Dcmd.DC_INIT;
/* 9 */     dcmptr.dc_flag = Dcmd.DC_NUSED;
/*10*/     dqptr = deqna.eioaddr;
/*11*/     deqna.setRcmd(dcmptr);
/*12*/   }
/*13*/
/*14*/
/*15*/   public void setRcmd(Dcmd dcmptr) {
/*16*/     eioaddr.d_csr = (short)
/*17*/                     ((eioaddr.d_csr | DqRegs.DQ_RLI) ^ DqRegs.DQ_RLI);
/*18*/     eioaddr.d_rcmd = dcmptr;
/*19*/
/*20*/     // the next two lines describe (simulate) the effect of
/*21*/     // reading from bytes 6/7 of the DEQNA registers
/*22*/     eioaddr.d_rcmdh = 0;
/*23*/     scheduler.resume(myName);
/*24*/   }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
    // Procedure ethwrite on page 38; write a single packet to the ethernet
/* 1 */   public int etherwrite(short type,
/* 2 */                         EtherAddress destination,
/* 3 */                         byte[] message) {
/* 4 */     EtherPacket packet = new EtherPacket(etpaddr, destination, message);
/* 5 */     packet.ep_hdr.e_ptype = type;
/* 6 */     if (packet.ep_length > EtherAddress.EMAXPAK - 18) {
/* 7 */       return SYSERR;
/* 8 */     }
/* 9 */     if (packet.ep_length < EtherAddress.EMINPAK - 18) {
/*10*/       return SYSERR;
/*11*/     }
/*12*/     etwsem.p();
/*13*/     startwriting(packet, Dcmd.DC_NORM);
/*14*/   }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

```
/* 1 */  public void startwriting(EtherPacket packet, int setup) {
/* 2 */      Dcmd dcmptr;
/* 3 */      DqRegs dqptr;
/* 4 */      dqptr = deqna.eioaddr;
/* 5 */      while ((dqptr.d_csr & DqRegs.DQ_XLI) == 0) ;
/* 6 */      dcmptr = ewcmd[0];
/* 7 */      dcmptr.dc_bufh = Dcmd.DC_VALID | Dcmd.DC_ENDM | (etsetup = setup);
/* 8 */      if ((packet.ep_length % 2) == 1) dcmptr.dc_bufh |= Dcmd.DC_LBIT;
/* 9 */      dcmptr.dc_buf = packet;
/*10*/      dcmptr.dc_len = packet.ep_length;
/*11*/      dcmptr.dc_st1 = dcmptr.dc_st2 = Dcmd.DC_INIT;
/*12*/      dcmptr.dc_flag = Dcmd.DC_NUSED;
/*13*/      deqna.setWcmd(dcmptr);
/*14*/      return ;
/*15*/    }
/*16*/
/*17*/    public void setWcmd(Dcmd dcmptr) {
/*18*/      eioaddr.d_csr = (short) ((eioaddr.d_csr | DqRegs.DQ_XLI) ^ DqRegs.DQ_XLI);
/*19*/      eioaddr.d_wcmd = dcmptr;
/*20*/      // the next two lines describe (simulate) the effect of
/*21*/      // writing to bytes 10/11 of the DEQNA registers
/*22*/      eioaddr.d_wcmdh = 0;
/*23*/      scheduler.resume(myName);
/*24*/    }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/* 1 */  public void ethinter() {
/* 2 */    Dcmd dcmptr = null;
/* 3 */    DqRegs dqptr = deqna.eioaddr;
/* 4 */    short csr = dqptr.d_csr;
/* 5 */    dqptr.d_csr = (short)((csr | DqRegs.DQ_RINT | DqRegs.DQ_XINT | Dcmd.DC_ERRU)
/* 6 */                          ^ (DqRegs.DQ_RINT | DqRegs.DQ_XINT | Dcmd.DC_ERRU));
/* 7 */    boolean doresch = false;
/* 8 */    if ((csr & DqRegs.DQ_RINT) != 0) {
/* 9 */      // interrupt after reading
/*10*/      dcmptr = ercmd[0];
/*11*/      if ((dcmptr.dc_st1 & Dcmd.DC_LUSE) != Dcmd.DC_ERRU) {
/*12*/        // received packet correct
/*13*/        doresch = true;
/*14*/      } else {              // error, so retry
/*15*/        dcmptr.dc_st1 = dcmptr.dc_st2 = DC_INIT;
/*16*/        dcmptr.dc_flag = DC_NUSED;
/*17*/        deqna.setRcmd(dcmptr);
/*18*/      }
/*19*/    }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/*20*/    if ((csr & DqRegs.DQ_XINT) != 0) {
/*21*/      // interrupt after sending
/*22*/      dcmptr = ewcmd[0];
/*23*/      if ((dcmptr.dc_st1 & Dcmd.DC_LUSE) != Dcmd.DC_ERRU) {
/*24*/        // packet sucessfully delivered to the ethernet
/*25*/        etwsem.v();
/*26*/      } else {
/*27*/        // sending erronous, retry
/*28*/        ...
/*29*/      }
/*30*/    }
/*31*/    if (doresch) {
/*32*/      deqna.scheduler.resume(etrpid);
/*33*/    }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Adreßumsetzung Ethernet ↔ Internet
- (reverse) address resolution packet

```
/* 1 */  class ArpPacket {
/* 2 */    short  hrd;                    // type of hardware (Ethernet = 1)
/* 3 */    short  prot;                   // format of proto. (IP=0x0800)
/* 4 */    byte   hlen;                   // hardware address length (6 for Ether)
/* 5 */    byte   plen;                   // protocol address length (4 for IP)
/* 6 */    short  op;                     // arp operation
/* 7 */                                   //  arp request to resolve address
/* 8 */                                   //     internet addr. --> ethernet addr.
/* 9 */                                   //  reply to a resolve request
/*10*/                                   //  reverse ARP request (RARP packet)
/*11*/                                   //     ethernet addr. --> internet addr.
/*12*/                                   //  reply to a reverse request
/*13*/                                   //     (RARP pack.)
/*14*/    EtherAddress  sha;             // sender's physical hardware address
/*15*/    IPAddress     spa;             // sender's protocol address (IP addr.)
/*16*/    EtherAddress  tha;             // target's physical hardware address
/*17*/    IPAddress     tpa;             // target's protocol address (IP)
/*18*/  }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Aufbau von "address resolution packets"

	Hardwaretyp (1 = Ethernet)	
Format der Protokolladresse	Länge der Hard-wareadresse (6)	Länge der Proto-kolladresse (4)
Operation	Ethernetadr. der Quelle (0 - 15)	
Ethernetadresse der Quelle (16 - 47)		
IP-Adresse der Quelle (0 - 31)		
Ethernetadresse des Ziels (0 - 31)		
Ethernetadr. des Ziels (32 - 47)	IP-Adresse des Ziels (0 - 15)	
IP-Adresse des Ziels (16 - 31)		


```
void rarp_in(ArpPacket packet)

receive_timed(int maxwait)

void send(int pid, int message)

void make_arp(short type, short op,
              IPAddress source_process_addr,
              IPAddress target_process_addr)
```


Cache for address resolution protocol

```
int getpath(IPAddress addr)
    // Find route table index for
    // a given IP address

arpfind(IPAddress faddr)
    // Find or insert entry in ARP cache
    // and return its index

arp_in(ArpPacket packet, int device)
```


IP packet

```
/*19*/  class IPPacket {
/*20*/    byte      verlen;                    // IP vers.(0x40) + hdr len in longs (5)
/*21*/    byte      svctyp;                    // service type (0 => normal service
/*22*/    short     paclen;                    // packet length in octetts
/*23*/    short     id;                        // datagram id (to help gateways frag.)
/*24*/    short     fragoff;                   // fragment offset (0 for first fragment)
/*25*/    byte      tim2liv;                   //time to live in gateway hops (10)
/*26*/    byte      proto;                     // IP protocol (ICMP is assigned 1
/*27*/                                         // UDP  is assigned 17)
/*28*/    short     chksum;                    // 1s compl. of sum of shorts in header
/*29*/    IPAddress src;                       // IP address of source
/*30*/    IPAddress dest;                      // IP address of destination
/*31*/    byte[]    data = new byte[paclen - 20]; // IP datagram data area
/*32*/  }
```

Verteilte Systeme: IP-Paket

```
void ipsend(IPAddress faddr,
            EtherPacket packet, int datalen)

void route(IPAddress faddr,
           EtherPacket packet, int totlen)
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Verteilte Systeme: ICMP-Paket

Type ICMP message type

- 0 Echo Reply
- 3 Destination unreachable
- 4 Source Quench
- 5 Redirect (change a route)
- 8 Echo Request
- 11 Time Exceeded for a Datagram
- 12 Parameter Problem on a Datagram
- 13 Timestamp Request
- 14 Timestamp Reply
- 15 Information Request
- 16 Information Reply

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Verteilte Systeme: ICMP-Paket

```
void icmp_in(EtherPacket packet,
             int icmpp, int lim)
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Verteilte Systeme: UDP-Paket

UDP packet

```
/* 1 */  class UDPPacket
/* 2 */    short       sport;          // Source UDP port number
/* 3 */    short       dport;          // Destination UDP port number
/* 4 */    short       dplen;          // Length of UDP data
/* 5 */    short       ucksum;         // UDP checksum (0 => no checksum)
/* 6 */    byte[]      data;           // Data in UDP message
/* 7 */  }
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Präambel (0 - 31)		
Präambel (32 - 63)		
Ziel-Adresse (0 - 31)		
Ziel-Adresse (32 - 47)	Quell-Adresse (0 - 15)	
Quell-Adresse(16 - 47)		
Paket-Typ	Version (0x40), Kopflaenge (5)	Diensttyp
Paketlänge (in Bytes)	Fragmentidentifikator	
Fragmentadresse	Max. Zahl an 'hops'	Protokoll
Prüfsumme	IP-Adresse der Quelle (0 - 15)	
IP-Adresse der Quelle (16 - 31)	IP-Adresse des Ziels (0 - 15)	
IP-Adresse des Ziels (16 - 31)	Optionen	
Optionen / Füllbyte	Quell-Port	
Ziel-Port	Länge des UDP-Datagramms	
UDP-Prüfsumme	UDP-Daten (Bytes 0 - 1)	
UDP-Daten (Bytes 2 - 5)		
▪ ▪ ▪		
CRC-Prüfsumme		


```
void udpsend(IPAddres faddr,
             short fport, short lport,
             EtherPacket packet, int datalen)

void netin()
  // thread

void netout(int icmpp)
  // thread

void ip_in(EtherPacket packet, int icmpp, int lim)
```

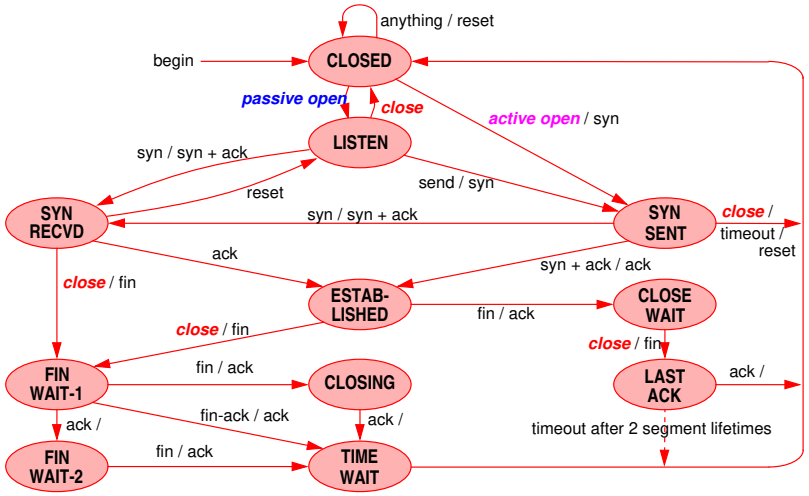

- TCP (Transmission Control Protocol)
- Verbindungsaufbau (3-way handshake)
  - Einseitig

1. CLOSED				CLOSED
2. CLOSED				(passive open) LISTEN
3. (active open) SYN-SENT	-->	<SEQ=100><CTL=SYN>	-->	SYN-RECEIVED
4. ESTABLISHED	<--	<SEQ=300><ACK=101><CTL=SYN,ACK>	<--	SYN-RECEIVED
5. ESTABLISHED	-->	<SEQ=101><ACK=301><CTL=ACK>	-->	ESTABLISHED
6. ESTABLISHED	-->	<SEQ=101><ACK=301><CTL=ACK><DATA>	-->	ESTABLISHED

- Gleichzeitig in beiden Richtungen

1. CLOSED				CLOSED
2. (active open) SYN-SENT	-->	<SEQ=100><CTL=SYN>	...	
3.				(active open)
SYN-RECEIVED	<--	<SEQ=300><CTL=SYN>	<--	SYN-SENT
4.	...	<SEQ=100><CTL=SYN>	-->	SYN-RECEIVED
5. SYN-RECEIVED	-->	<SEQ=100><ACK=301><CTL=SYN,ACK>	...	
6. ESTABLISHED	<--	<SEQ=300><ACK=101><CTL=SYN,ACK>	<--	SYN-RECEIVED
7.	...	<SEQ=101><ACK=301><CTL=ACK>	-->	ESTABLISHED

❑ TCP-Zustandsdiagramm (standardisiert)



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ Verbindungsabbau

◆ Einseitig

1. ESTABLISHED				ESTABLISHED
2. (close) FIN-WAIT-1	-->	<SEQ=100><ACK=300><CTL=FIN,ACK>	-->	CLOSE-WAIT
3. FIN-WAIT-2	<--	<SEQ=300><ACK=101><CTL=ACK>	<--	CLOSE-WAIT
4.				(Close)
TIME-WAIT	<--	<SEQ=300><ACK=101><CTL=FIN,ACK>	<--	LAST-ACK
5. TIME-WAIT	-->	<SEQ=101><ACK=301><CTL=ACK>	-->	CLOSED
6. (2 MSL) CLOSED				

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

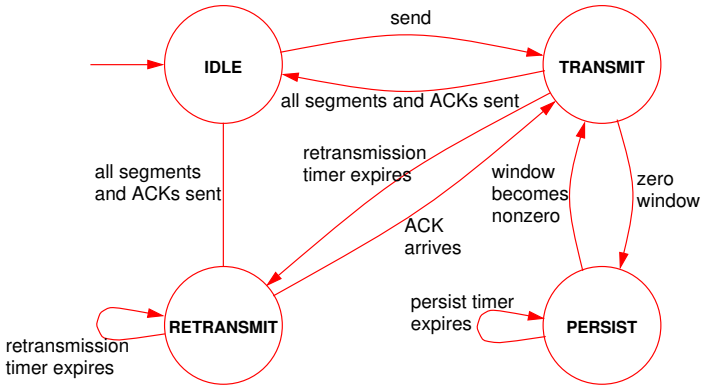
◆ Gleichzeitig von beiden Seiten

1. ESTABLISHED				ESTABLISHED
2. (close) FIN-WAIT-1	-->	<SEQ=100><ACK=300><CTL=FIN,ACK>	...	(close) FIN-WAIT-1
	<--	<SEQ=300><ACK=100><CTL=FIN,ACK>	<--	
	...	<SEQ=100><ACK=300><CTL=FIN,ACK>	-->	
3. CLOSING	-->	<SEQ=101><ACK=301><CTL=ACK>	...	CLOSING
	<--	<SEQ=301><ACK=101><CTL=ACK>	<--	
	...	<SEQ=101><ACK=301><CTL=ACK>	-->	
4. TIME-WAIT				TIME-WAIT
5. (2 MSL)				(2 MSL)
6. CLOSED				CLOSED

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ Laufende Übertragung:Nicht formal standardisiert

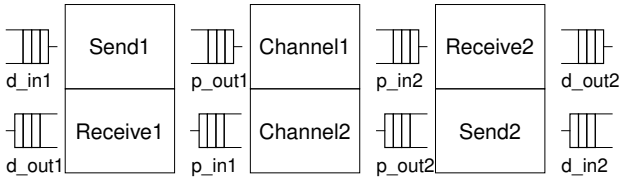
Basiert auf Zustandsdiagramm



Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Ein Verbindungsprotokoll (ALP, A Link Protocol)
Basiert auf Zustandsaustauschmethode

Gesamstruktur



Vereinfachende Annahme:
Der Kanal kann keine Pakete speichern

- Trivialerweise gegeben bei Ethernet
- Verletzt bei Vernetzung mittels Gateway-Rechnern

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Zustandsaustauschmethode

- Paket enthält vollständige Sicht des Senders
- Paketaustausch bis zum Angleich der Sichten
  Dazu versendet der Sender ein Paket, wenn

(1) der Sender seine Sicht ändert,

(2) der Empfänger veranlaßt werden soll den Zustand zu ändern,

(3) ein Paketempfang Unstimmigkeiten der Sichten erkennen läßt,

(4) ein fehlerhaftes Paket empfangen wird,

(5) der Sender mit dem Zustand unzufrieden ist.

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Algorithmus

Variable

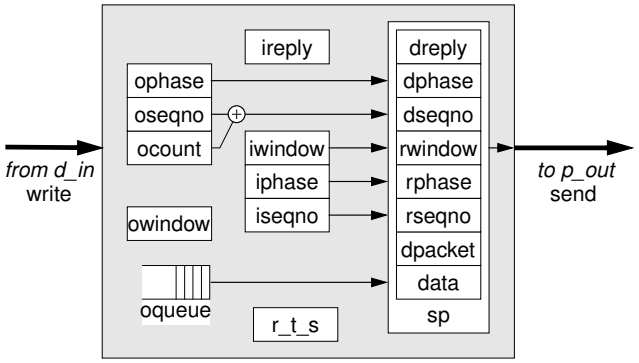
dseqno	Sequenznummer zur Numerierung abgehender Pakete.
iseqno	nächste akzeptable Sequenznummer.
dpacket	gibt an, ob das Paket Daten enthält.
dreply	gibt an, ob der Absender eine Antwort fordert.

Bei Feststellung eines Paketverlustes, wird die "Übertragungsphase" gewechselt. Der Phasenwechsel veranlaßt, daß noch nicht bestätigte Pakete erneut übermittelt werden. Die Phasenangabe muß erlauben, alte und neue Pakete unterscheiden zu können. Es zeigt sich, daß es ausreichend ist, die Phasen modulo 2 zu zählen.

dphase	Phase aus der Sicht des Senders.
rphase	Phase aus der Sicht des Empfängers.
owindow	Fensterbreite beim Sender.
iwindow	Fensterbreite beim Empfänger.

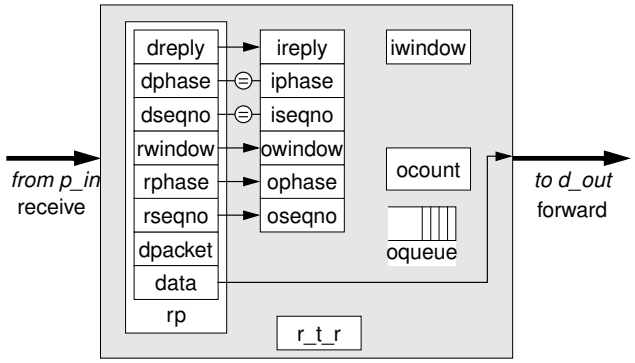
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Sender



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Empfänger



Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/* 1 */  class Packet {
/* 2 */    boolean dreply;          // indicates that receiver should send a packet
/* 3 */    int     dphase;          // phase in which packet is sent
/* 4 */    int     dseqno;          // sequence number of packet
/* 5 */    int     rwindow;         // receive window of sender
/* 6 */    int     rphase;          // pase in which receiver should be
/* 7 */    int     rseqno;          // sequence number for next packet of the receiver
/* 8 */    boolean dpacket;         // indicates that message contains user data
/* 9 */    String  data;            // user data
/*10*/  }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/*1 */   // This algorithm should be executed after receipt of each
/*2 */   //  non-erroneous frame.
/*3 */  { f = receive();
/*4 */    if (f->dseqno == iseqno && f->dphase == iphase) {
/*5 */      if (f->dpacket == 1 && iwindow > 0) {
/*6 */        // An input packet has arrived in sequence. Accept it
/*7 */        iseqno = (iseqno + 1) % m; forward(f->data); }
/*8 */    } else {
/*9 */      // An input packet has been lost. Prepare to accept retransmission.
/*10*/      iphase = 1 - f->dphase; ireply = 1; }
/*11*/    if ((f->rseqno - oseqno) % m <= ocount) {
/*12*/      // The received reverse sequence number is not anomalous.
/*13*/      while (f->rseqno != oseqno) { // Discard accepted output packets.
/*14*/        pop(oqueue); oseqno = (oseqno + 1) % m; ocount--; olength--; }
/*15*/      owindow = f->rwindow; }
/*16*/    if (f->rphase != ophase) { // Prepare to retransmit rejected output packets
/*17*/      ophase = f->rphase; oseqno = f->rseqno; ocount = 0; }
/*18*/    if (f->dreply == 1 || olength > 0) { // state is unsatisfactory.
/*19*/      ireply = 1; }
/*20*/    discard(f);
/*21*/  }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig


```
/*1 */   // This algorithm should be executed
/*2 */   // (1) after a packet is pushed on oqueue and olength is incremeted and
/*3 */   // (2) after execution of the preceding algorithm for receipt
/*4 */   //     of a non-erronous frame.
/*5 */
/*6 */   // Also ireply should be set to 1 and this algorithm then executed
/*7 */   // (3) after receipt of an erroneous frame,
/*8 */   // (4) after iwindow is changed,
/*9 */   // (5) after a give-up timeout and any associated purging of the output queue,
/*10*/   // (6) after initialization, and perhaps
/*11*/   // (7) periodically so long as olength > 0.
/*12*/   { while (ocount < min(owindow, olength) || ireply == 1) {
/*13*/       f = new packet;
/*14*/       ireply = 0; f->dphase = ophase; f->dseqno = (oseqno + ocount) % m;
/*15*/       if (ocount < min(owindow, olength)) {
/*16*/         // A packet should be included in the frame.
/*17*/         f->dpacket = 1; f->data = oqueue[ocount++];
/*18*/       }
/*19*/       if (olength > 0) { // Not all output packets have as yet been accepted.
/*20*/         f->dreply = 1;
/*21*/       }
/*22*/       f->rphase = iphase; f->rseqno = iseqno; f->rwindow = iwindow; send(f);
/*23*/     }
/*24*/   }
```


Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig