

BP 1	Verteilte Systeme: Nachrichtenmechanismen
2	Nachrichtenmechanismen
◆	<p>Spezielle Literatur</p> <p>Comer, D.: Operating System Design - Volume II, Internetworking with XINU. Prentice-Hall, Englewood Cliffs, New Jersey, 1987.</p> <p>Comer, Douglas and Stevens, David L.: Internetworking with TCP/IP Vol. II (3rd Edition) 1999 ISBN 0-13-973843-6</p> <p>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.</p> <p>◆ Einschlägige Links (überprüft am 4.10.2001)</p> <ul style="list-style-type: none"> • Protokolle http://www.protocols.com Einstieg zu Beschreibungen aller gängigen Protokolle • XINU-Hompage http://public.ise.canberra.edu.au/~chrisc/xinu.html
11.10.01	<p>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</p> <p>2-1</p>

BP 1

Verteilte Systeme: Nachrichtenmechanismen

◆ 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

11.10.01

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

2-2

BP 1	Verteilte Systeme: Nachrichtenmechanismen
□	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)
11.10.01	<p>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</p> <p>2-3</p>

BP 1	Verteilte Systeme: Nachrichtenmechanismen
2.1	<p>Strukturen der Kommunikation</p> <p>ISO/OSI-Referenzmodell</p> <p>The diagram illustrates the ISO/OSI Reference Model with two processes, Prozeß A and Prozeß B, each having seven layers: Anwendungsschicht (Application Layer), Darstellungsschicht (Presentation Layer), Sitzungsschicht (Session Layer), Transportschicht (Transport Layer), Netzwerkschicht (Network Layer), Sicherungsschicht (Data Link Layer), and Übertragungsschicht (Physical Layer). The layers are connected by dashed lines representing protocols: Anwendungsprotokoll, Darstellungsprotokoll, Sitzungsprotokoll, Transportprotokoll, Netzwerkprotokoll, Sicherungsprotokoll, and Übertragungsprotokoll. The Transport, Network, Data Link, and Physical layers are highlighted in blue.</p>
11.10.01	<p>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</p> <p>2-4</p>

BP 1

Verteilte Systeme: Nachrichtenmechanismen

- Übertragungsschicht
 - Ungesicherte Übertragung eines Bitstroms
 - Charakteristika der benutzten Signalformen sind Gegenstand der Standardisierung
- Sicherungsschicht
 - Übertragung von Bitblöcken zwischen direkt physisch verbundenen Einheiten
 - Synchronisation, Fehler- und Flußkontrolle, evtl. Nachrichtenwiederholung
- Netzwerkschicht
 - Virtuelle Verbindung zwischen zwei Einheiten, evtl. unter Zwischenschaltung anderer Einheiten
- Transportschicht
 - Zuverlässiger, transparenter Nachrichtentransport zwischen Endpunkten
 - Zerlegen von Nachrichten in Transportpakete und Wiederaussemblesetzen
 - Flußkontrolle und Fehlerbehandlung

11.10.01

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

2-5

BP 1

Verteilte Systeme: Nachrichtenmechanismen

◆ Beispiel: Windows 2000

API	application programming interface
DLL	dynamic-link libraries
HAL	hardware abstraction layer
NDIS	Network Driver Interface Specification
TDI	Transport Driver Interface

11.10.01

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

2-6

BP 1

Verteilte Systeme: Nachrichtenmechanismen

Nachrichtenformat geschichteter Protokolle

11.10.01

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

2-7

BP 1

Verteilte Systeme: Nachrichtenmechanismen

□ 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

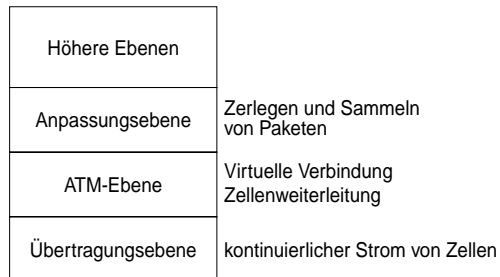
11.10.01

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

2-8

BP 1 Verteilte Systeme: Nachrichtenmechanismen

- 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

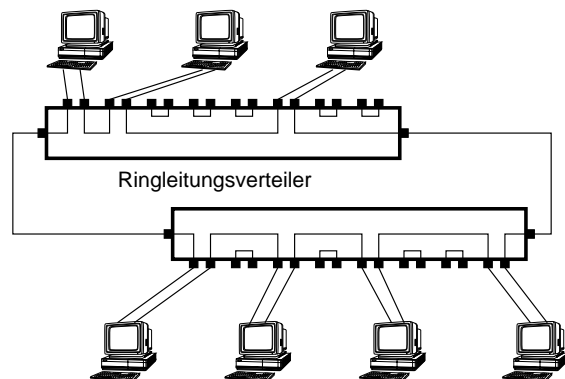


BP 1 Verteilte Systeme: Nachrichtenmechanismen

- 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

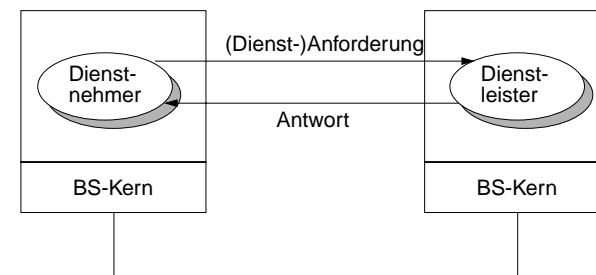
BP 1 Verteilte Systeme: Nachrichtenmechanismen

- ◆ Hardwarestruktur des 'Token Bus'



BP 1 Verteilte Systeme: Nachrichtenmechanismen

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



- ◆ Problem: Adressierung

BP 1

Verteilte Systeme: Maschinensprachebene

Ethernet packet

01631

Präambel (0 - 31)

Präambel (32 - 63)

Ziel-Adresse (0 - 31)

Ziel-Adresse (32 - 47)

Quell-Adresse (0 - 15)

Quell-Adresse(16 - 47)

Paket-Typ

Daten (Bytes 0 - 1)

Daten (Bytes 2 - 5)

...

CRC-Prüfsumme

Ethernet Link-Level Protocol Format

11.10.01

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

2-13

BP 1

Verteilte Systeme: Maschinensprachebene

```

class EtherAddress {
    public static final int    EMINPAK = 64; // minimum packet length in bytes
    public static final int    EMAXPAK = 1266; // maximum packet length in bytes
    public static final int    EHLEN  = 14; // size of ether header
    public static final int    EDLEN  = EMAXPAK - EHLEN;
    public static final int    EPADLEN = 6; // length of phys. addresss
    public static final byte[] EBCAST = {(byte)0377, (byte)0377, (byte)0377,
                                         (byte)0377, (byte)0377, (byte)0377};
                                         // Ethernet broadcast address

    byte[] address = new byte[EPADLEN];
}

class EtherPacket {
    // Structure of ethernet header
    byte[] preamble = new byte[8];
    EtherAddress e_dest; // destination host address
    EtherAddress e_src;  // source host address
    short e_ptype; // packet type
    public static final int EP_IP   = 0x00000800; // DARPA Internet protocol
    public static final int EP_ARP  = 0x00000806; // address resolution protocol
    public static final int EP_RARP = 0x00000803; // reverse addr. resol. protocol
    byte[] ep_data = new byte[1500]; // 46 <= len <= 1522
    unsigned crc;
}

```

11.10.01

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

2-14

BP 1

Verteilte Systeme: Maschinensprachebene

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

11.10.01

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

2-15

BP 1

Verteilte Systeme: Maschinensprachebene

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

11.10.01

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

2-16

BP 1 Verteilte Systeme: Maschinensprachebene

Bedeutung des Kontroll- und Statusregisters

Bit	Bezeichnung	Bedeutung
15	DQ_RINT	Eingabe-Interrupt (auf 1 gesetzt, wenn ein Eingabevorgang abgeschlossen ist, muß vom Betriebssystem 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 Betriebssystem 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)

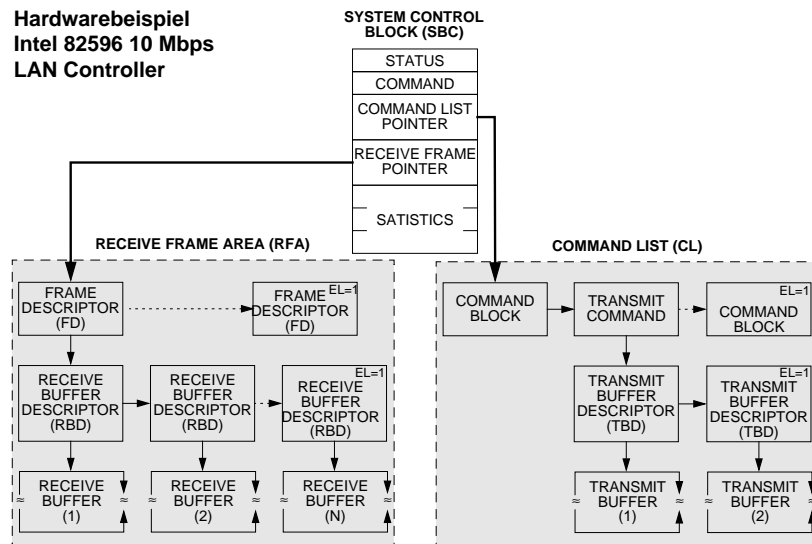
BP 1 Verteilte Systeme: Maschinensprachebene

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 (??)

BP 1 Verteilte Systeme: Maschinensprachebene

Hardwarebeispiel Intel 82596 10 Mbps LAN Controller



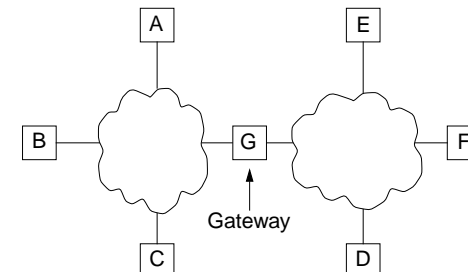
BP 1 Verteilte Systeme: Internet

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

H. Gilbert, Yale

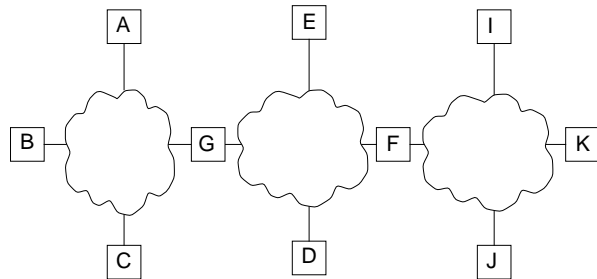
Although the OSI model is useful, the TCP/IP protocols don't match its structure exactly.

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



BP 1 Verteilte Systeme: Internet

Verknüpfung von drei Netzwerken



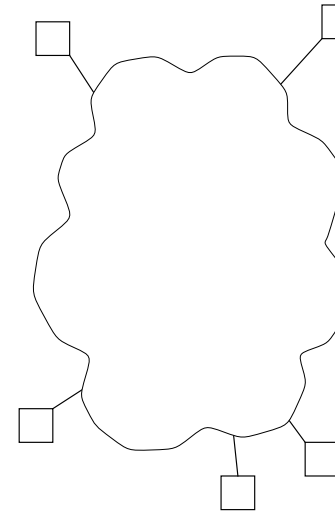
11.10.01

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

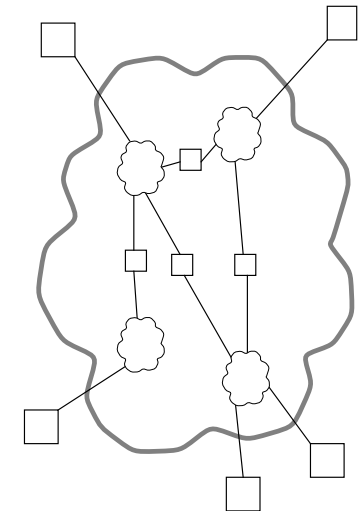
2-21

BP 1 Verteilte Systeme: Internet

Benutzersicht



Systemansicht



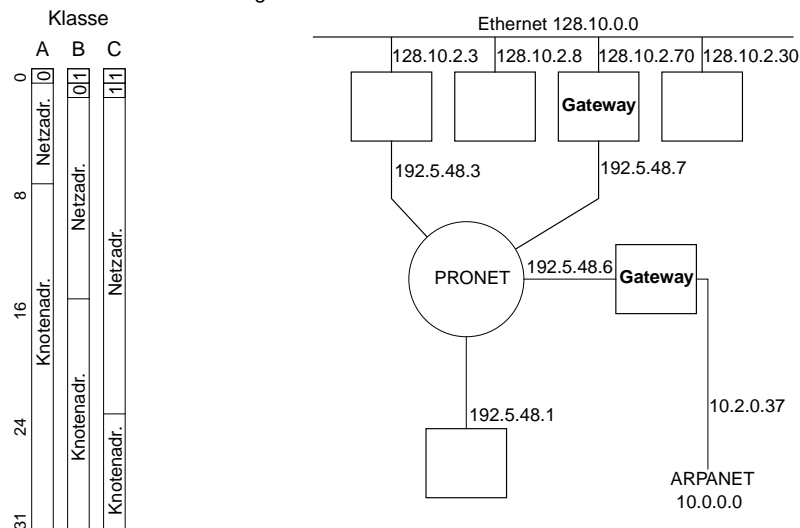
11.10.01

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

2-22

BP 1 Verteilte Systeme: IP-Adressen und ihre Darstellung

IP-Adressen und ihre Darstellung



11.10.01

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

2-23

BP 1 Verteilte Systeme: Aufbau von IP-Paketen

Aufbau von IP-Paketen

Version (0x40), Kopflänge (5)		Diensttyp
Paketlänge (in Bytes)		Fragmentidentifikator
Fragmentadresse		Max. Zahl an 'hops'
Prüfsumme		Protokoll
IP-Adresse der Quelle (16 - 31)		IP-Adresse der Quelle (0 - 15)
IP-Adresse des Ziels (16 - 31)		IP-Adresse des Ziels (0 - 15)
Optionen		Optionen
Füllbyte		Daten (Bytes 0 - 1)
Daten (Bytes 2 - 5)		

11.10.01

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

2-24

BP 1 Verteilte Systeme: Aufbau von IP-Paketen

IP packet

```
public class IPPacket {
    static public final short IPHLEN = 20;
    static public final int IVERLEN = 0x45;
    static public final int ISVCTYP = 0;
    static public final int IFRAGOFF = 0;
    static public final int ITIM2LIV = 10;

    byte    verlen;        // IP vers. (0x40) + hdr len in longs (5)
    byte    svctyp;        // service type (0 => normal service)
    short    paclen;        // packet length in octets (bytes)
    short    id;            // datagram id (to help gateways frag.)
    short    fragoff;       // fragment offset (0 for 1st fragment)
    byte    tim2liv;        // time to live in gateway hops (10)
    byte    proto;          // IP protocol (UDP is assigned 17)
    short    cksum;         // 1s compl. of sum of shorts in header
    IPAddress src;          // IP address of source
    IPAddress dest;         // IP address of destination

    byte[]    data = new byte[paclen - 20]; // IP datagram data area
}
```

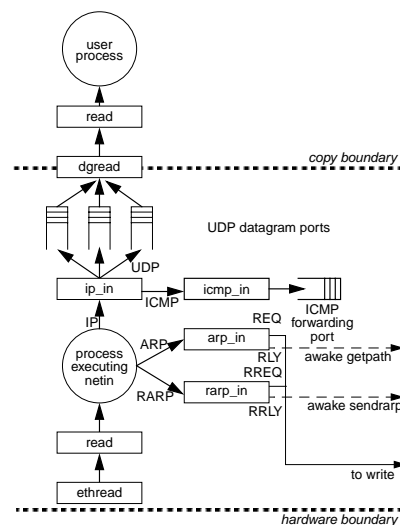
BP 1 Verteilte Systeme: Aufbau von IP-Paketen

In Ethernet-Paket eingebettetes Internet-Paket

0	16	31
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), Kopflänge (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	Daten (Bytes 0 - 1)
Daten (Bytes 2 - 5)		
■ ■ ■		
CRC-Prüfsumme		

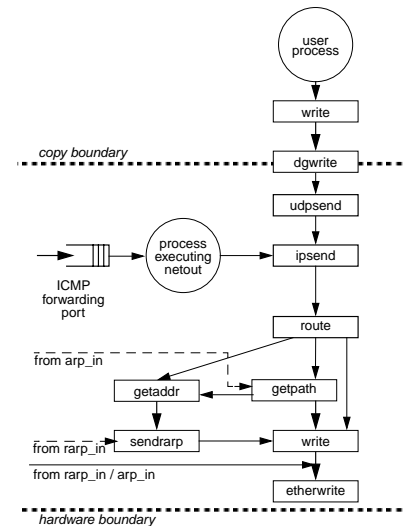
BP 1 Verteilte Systeme: Aufbau von IP-Paketen

Struktur der Empfangs-Seite

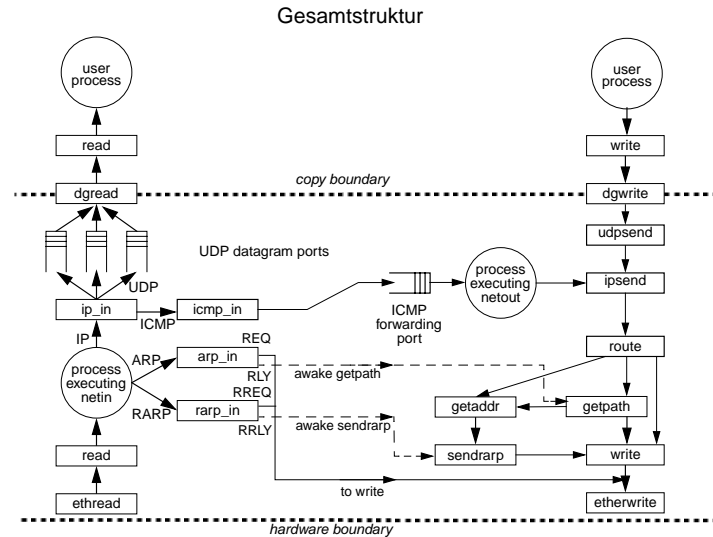


BP 1 Verteilte Systeme: Aufbau von IP-Paketen

Struktur der Sende-Seite



BP 1 Verteilte Systeme: Aufbau von IP-Paketen



BP 1 Verteilte Systeme: Methoden der Ethernet-Ebene

Methoden der Ethernet-Ebene

EtherControlBlock

```
public EtherControlBlock()

public int read(EtherPacket f_packet)

public int write(short type,
                  EtherAddress destination,
                  byte[] message)

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

BP 1 Verteilte Systeme: Methoden der Ethernet-Ebene

```
public int read(EtherPacket f_packet) {
    int returncode = 0;
    etrsem.p();
    startreading(f_packet);
    etrpid = Thread.currentThread().getName();
    degna.scheduler.suspend();
    Dcmd dcmptr = ercmd[0];
    if ((dcmptr.dc_st1 & Dcmd.DC_LUSE) == Dcmd.DC_ERRU) {
        returncode = Xinu.SYSERR;
    } else {
        returncode = 0;
        int msglen = (L->dcmptr->dc_st1 & DC_HLEN) |
                    (L->dcmptr->dc_st2 & DC_LLEN);
    }
    etrsem.v();
    return returncode;
}
```

BP 1 Verteilte Systeme: Methoden der Ethernet-Ebene

```
public void startreading(EtherPacket f_packet) {
    Dcmd dcmptr;
    DgRegs dqptr;
    dcmptr = ercmd[0];
    dcmptr.dc_bufh = Dcmd.DC_VALID;
    dcmptr.dc_buf = f_packet;
    dcmptr.dc_len = f_packet.ep_length;
    dcmptr.dc_st1 = dcmptr.dc_st2 = Dcmd.DC_INIT;
    dcmptr.dc_flag = Dcmd.DC_NUSED;
    dqptr = degna.eioaddr;
    degna.setRcmd(dcmptr);
}

public void setRcmd(Dcmd dcmptr) {
    eioaddr.d_csr = (short)
        ((eioaddr.d_csr | DgRegs.DQ_RLI) ^ DgRegs.DQ_RLI);
    eioaddr.d_rcmd = dcmptr;

    // the next two lines describe (simulate) the effect of
    // reading from bytes 6/7 of the DEQNA registers
    eioaddr.d_rcmdh = 0;
    scheduler.resume(myName);
}
```


BP 1	Verteilte Systeme: Methoden der Ethernet-Ebene
	<pre> // Procedure ethwrite on page 38; write a single packet to the ethernet public int etherwrite(short type, EtherAddress destination, byte[] message) { EtherPacket packet = new EtherPacket(etpaddr, destination, message); packet.ep_hdr.e_ptype = type; if (packet.ep_length > EtherAddress.EMAXPAK - 18) { return SYSERR; } if (packet.ep_length < EtherAddress.EMINPAK - 18) { return SYSERR; } etwsem.p(); startwriting(packet, Dcmd.DC_NORM); } </pre>
11.10.01	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 2-33

BP 1	Verteilte Systeme: Methoden der Ethernet-Ebene
	<pre> public void startwriting(EtherPacket packet, int setup) { Dcmd dcmptr; DqRegs dqptr; dqptr = degna.eioaddr; while ((dqptr.d_csr & DqRegs.DQ_XLI) == 0) ; dcmptr = ewcmd[0]; dcmptr.dc_bufh = Dcmd.DC_VALID Dcmd.DC_ENDM (etsetup = setup); if ((packet.ep_length % 2) == 1) dcmptr.dc_bufh = Dcmd.DC_LBIT; dcmptr.dc_buf = packet; dcmptr.dc_len = packet.ep_length; dcmptr.dc_st1 = dcmptr.dc_st2 = Dcmd.DC_INIT; dcmptr.dc_flag = Dcmd.DC_NUSED; degna.setWcmd(dcmptr); return ; } public void setWcmd(Dcmd dcmptr) { eioaddr.d_csr = (short) ((eioaddr.d_csr DqRegs.DQ_XLI) ^ DqRegs.DQ_XLI); eioaddr.d_wcmd = dcmptr; // the next two lines describe (simulate) the effect of // writing to bytes 10/11 of the DEQNA registers eioaddr.d_wcmdh = 0; scheduler.resume(myName); } </pre>
11.10.01	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 2-34

BP 1	Verteilte Systeme: Methoden der Ethernet-Ebene
	<pre> public void ethinter() { Dcmd dcmptr = null; DqRegs dqptr = degna.eioaddr; short csr = dqptr.d_csr; dqptr.d_csr = (short)((csr DqRegs.DQ_RINT DqRegs.DQ_XINT Dcmd.DC_ERRU) ^ (DqRegs.DQ_RINT DqRegs.DQ_XINT Dcmd.DC_ERRU)); boolean doresch = false; if ((csr & DqRegs.DQ_RINT) != 0) { // interrupt after reading dcmptr = ercmd[0]; if ((dcmptr.dc_st1 & Dcmd.DC_LUSE) != Dcmd.DC_ERRU) { // received packet correct doresch = true; } else { // error, so retry dcmptr.dc_st1 = dcmptr.dc_st2 = DC_INIT; dcmptr.dc_flag = DC_NUSED; degna.setRcmd(dcmptr); } } } </pre>
11.10.01	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 2-35

BP 1	Verteilte Systeme: Methoden der Ethernet-Ebene
	<pre> if ((csr & DqRegs.DQ_XINT) != 0) { // interrupt after sending dcmptr = ewcmd[0]; if ((dcmptr.dc_st1 & Dcmd.DC_LUSE) != Dcmd.DC_ERRU) { // packet successfully delivered to the ethernet etwsem.v(); } else { // sending erroneous, retry ... } } if (doresch) { degna.scheduler.resume(etrpid); } </pre>
11.10.01	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 2-36

BP 1

Verteilte Systeme: Adreßauflösung

Adreßumsetzung Ethernet ↔ Internet

(reverse) address resolution packet

```

class ArpPacket {
    short  hrd;// type of hardware (Ethernet = 1)
    short  prot;// format of proto. (IP=0x0800)
    byte   hlen;// hardware address length (6 for Ether)
    byte   plen;// protocol address length (4 for IP)
    short  op;// arp operation
//   ARP request to resolve address
//   internet addr. --> ethernet addr.
//   reply to a resolve request
//   reverse ARP request (RARP packet)
//   ethernet addr. --> internet addr.
//   reply to a reverse request
//   (RARP pack.)
    EtherAddress sha;// sender's physical hardware address
    IPAddress    spa;// sender's protocol address (IP addr.)
    EtherAddress tha;// target's physical hardware address
    IPAddress    tpa;// target's protocol address (IP)
}

```

11.10.01

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

2-37

BP 1

Verteilte Systeme: Adreßauflösung

Aufbau von "address resolution packets"

	Hardwaretyp (1 = Ethernet)	
Format der Protokolladresse	Länge der Hardwareadresse (6)	Länge der Protokolladresse (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)		

11.10.01

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

2-38

BP 1

Verteilte Systeme: Adreßauflösung

```

void rarp_in(ArpPacket packet)
    // handle RARP packet coming from Ethernet network

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)

```

11.10.01

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

2-39

BP 1

Verteilte Systeme: Adreßauflösung

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)

```

11.10.01

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

2-40

BP 1 **Verteilte Systeme: IP-Paket**

IP packet

```
class IPPacket {
    byteverlen;// IP vers.(0x40) + hdr len in longs (5)
    bytesvctyp;// service type (0 => normal service
    shortpaclen;// packet length in octetts
    shortid;// datagram id (to help gateways frag.)
    shortfragoff;// fragment offset (0 for first fragment)
    bytetim2liv;// time to live in gateway hops (10)
    byteproto;// IP protocol (ICMP is assigned 1
                // UDP is assigned 17)
    shortchksum;// 1s compl. of sum of shorts in header
    IPAddresssrc;// IP address of source
    IPAddressdest;// IP address of destination
    byte[]data = new byte[paclen - 20];// IP datagram data area
}
```

BP 1 **Verteilte Systeme: IP-Paket**

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

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

BP 1 **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

BP 1 **Verteilte Systeme: ICMP-Paket**

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

BP 1

Verteilte Systeme: UDP-Paket

UDP packet

```

class UDPPacket
  shortsport;// Source UDP port number
  shortdport;// Destination UDP port number
  shortdplen;// Length of UDP data
  shorttucksum;// UDP checksum (0 => no checksum)
  byte[]data;// Data in UDP message
}

```

11.10.01

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

2-45

BP 1

Verteilte Systeme: UDP-Paket

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), Kopflänge (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		

11.10.01

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

2-46

BP 1

Verteilte Systeme: UDP-Paket

```

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

void netin()
  // thread

void netout(int icmp)
  // thread

void ip_in(EtherPacket packet, int icmp, int lim)

```

11.10.01

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

2-47

BP 1

Verteilte Systeme: TCP

TCP (Transmission Control Protocol)

Verbindungsaufbau (3-way handshake), entnommen aus RFC 793

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>

-->

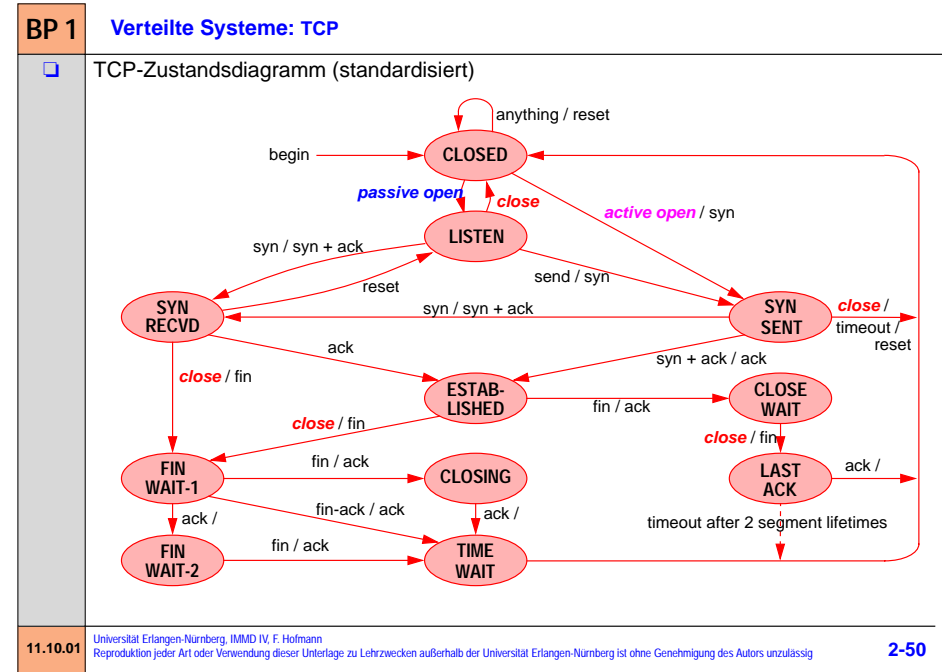
ESTABLISHED

11.10.01

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

2-48

BP 1 Verteilte Systeme: TCP	
♦	Gleichzeitig in beiden Richtungen
	1. 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
11.10.01	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 2-49

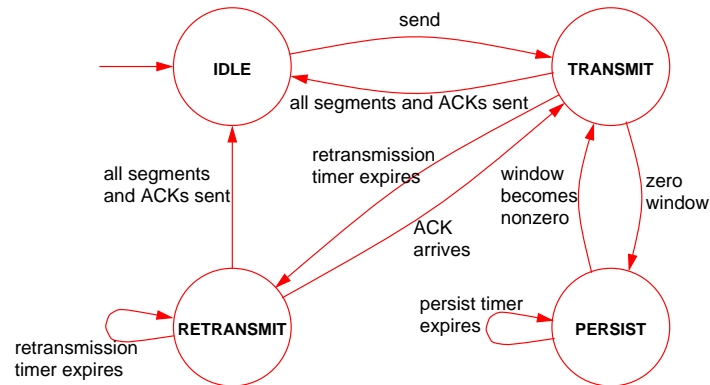


BP 1 Verteilte Systeme: TCP	
♦	Verbindungsabbau
	Einseitig
	1. ESTABLISHED
	2. (close) FIN-WAIT-1 --> <SEQ=100><ACK=300><CTL=FIN,ACK> --> CLOSE-WAIT
	3. FIN-WAIT-2 <-- <SEQ=300><ACK=101><CTL=ACK> <-- CLOSE-WAIT
	4. (Close) TIME-WAIT <-- <SEQ=300><ACK=101><CTL=FIN,ACK> <-- LAST-ACK
	5. TIME-WAIT --> <SEQ=101><ACK=301><CTL=ACK> --> CLOSED
	6. (2 MSL) CLOSED
11.10.01	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 2-51

BP 1 Verteilte Systeme: TCP	
♦	Gleichzeitig von beiden Seiten
	1. ESTABLISHED
	2. (close) FIN-WAIT-1 --> <SEQ=100><ACK=300><CTL=FIN,ACK> ... FIN-WAIT-1 <-- <SEQ=300><ACK=100><CTL=FIN,ACK> <-- FIN-WAIT-1 ... <SEQ=100><ACK=300><CTL=FIN,ACK> -->
	3. CLOSING --> <SEQ=101><ACK=301><CTL=ACK> ... CLOSING
	4. TIME-WAIT <-- <SEQ=301><ACK=101><CTL=ACK> <--
	5. (2 MSL) ... <SEQ=101><ACK=301><CTL=ACK> -->
	6. CLOSED
11.10.01	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 2-52

BP 1 Verteilte Systeme: TCP

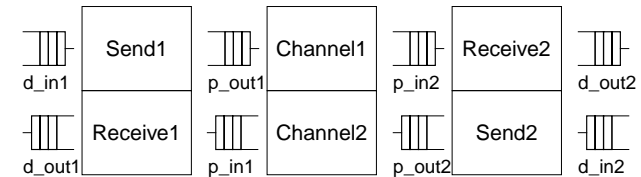
- Laufende Übertragung: Nicht formal standardisiert
Basiert auf Zustandsdiagramm



BP 1 Verteilte Systeme: ALP

- 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

BP 1 Verteilte Systeme: ALP

- ◆ 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.

BP 1 Verteilte Systeme: ALP

- ◆ Algorithmus

Variable

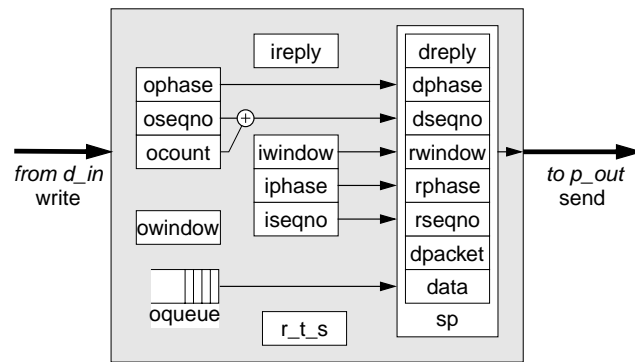
- dseqno Sequenznummer zur Numerierung abgehender Pakete.
- iseqno nächste akzeptable Sequenznummer.
- dpacket gibt an, ob das Paket Daten enthält.
- dreply gibt an, ob der Absender eine Antwort fordert.

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.

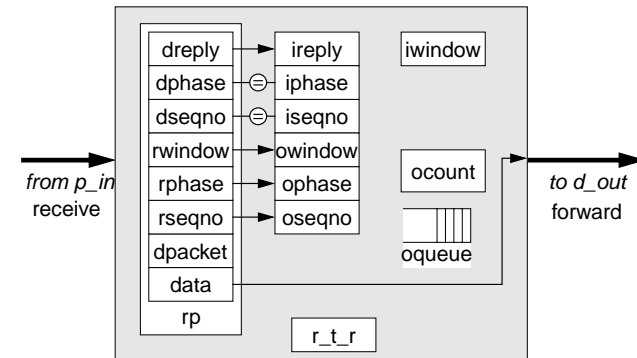
BP 1 Verteilte Systeme: ALP

Sender



BP 1 Verteilte Systeme: ALP

Empfänger



BP 1 Verteilte Systeme: ALP

```
class Packet {
    boolean dreply; // indicates that receiver should send a packet
    int dphase; // phase in which packet is sent
    int dseqno; // sequence number of packet
    int rwindow; // receive window of sender
    int rphase; // phase in which receiver should be
    int rseqno; // sequence number for next packet of the receiver
    boolean dpacket; // indicates that message contains user data
    String data; // user data
}
```

BP 1 Verteilte Systeme: ALP

```
/*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(queue); 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*/ }
```

```
/*1*/ // This algorithm should be executed
/*2*/ // (1) after a packet is pushed on oqueue and olength is incremented and
/*3*/ // (2) after execution of the preceding algorithm for receipt
/*4*/ // of a non-erroneous 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*/ }
```