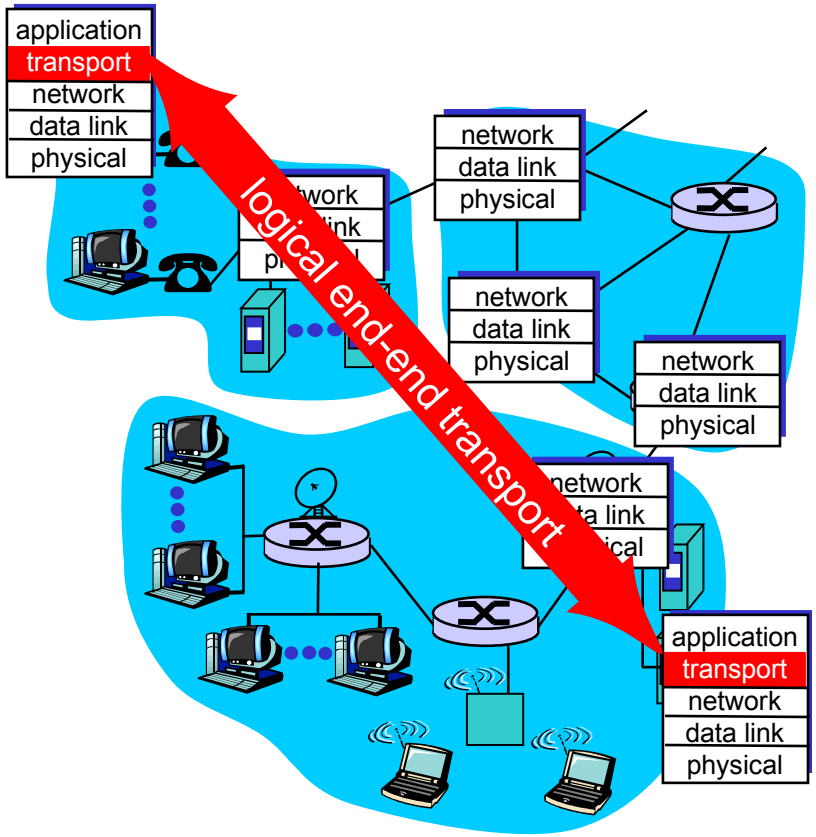
Ausgewählte Kapitel der Rechnernetze

- Protokoll-Port-Konzept
- TCP/UDP

Transport Services and Protocols

- Provide *logical communication* between app processes running on different hosts
- Transport protocols run in end systems
  - Sending side: breaks app messages into segments, passes to network layer
  - Receiving side: reassembles segments into messages, passes to app layer
- More than one transport protocol available to applications
  - Internet: TCP and UDP



Transport vs. Network Layer

- *Network layer:* logical communication between hosts
- *Transport layer:* logical communication between processes
  - relies on, enhances, network layer services

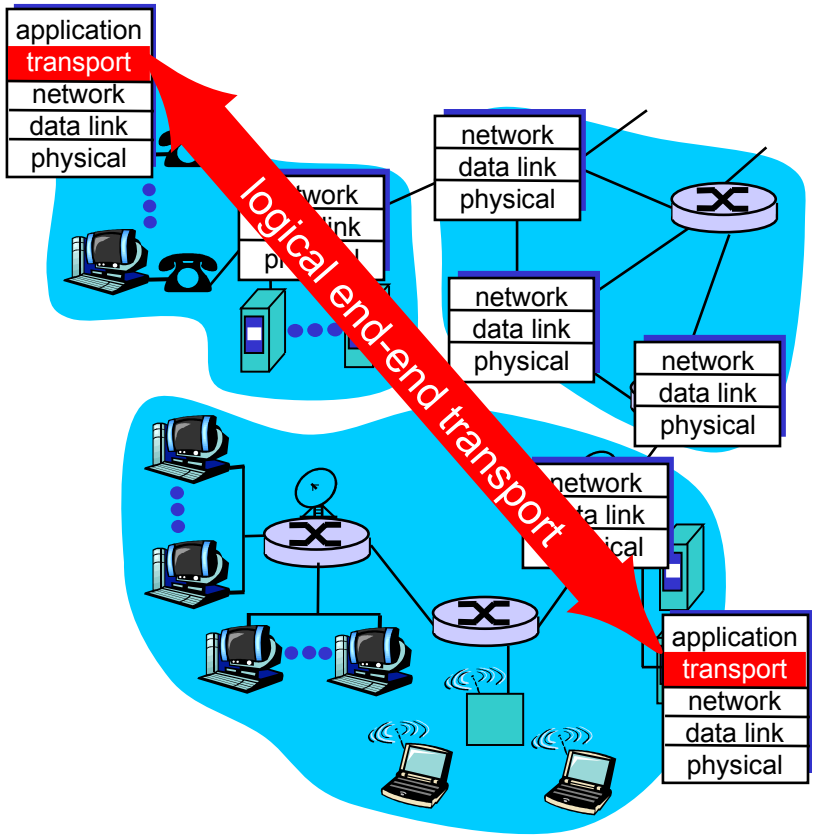
Household analogy:

12 kids sending letters to 12 kids

- processes = kids
- app messages = letters in envelopes
- hosts = houses
- transport protocol = Ann and Bill
- network-layer protocol = postal service

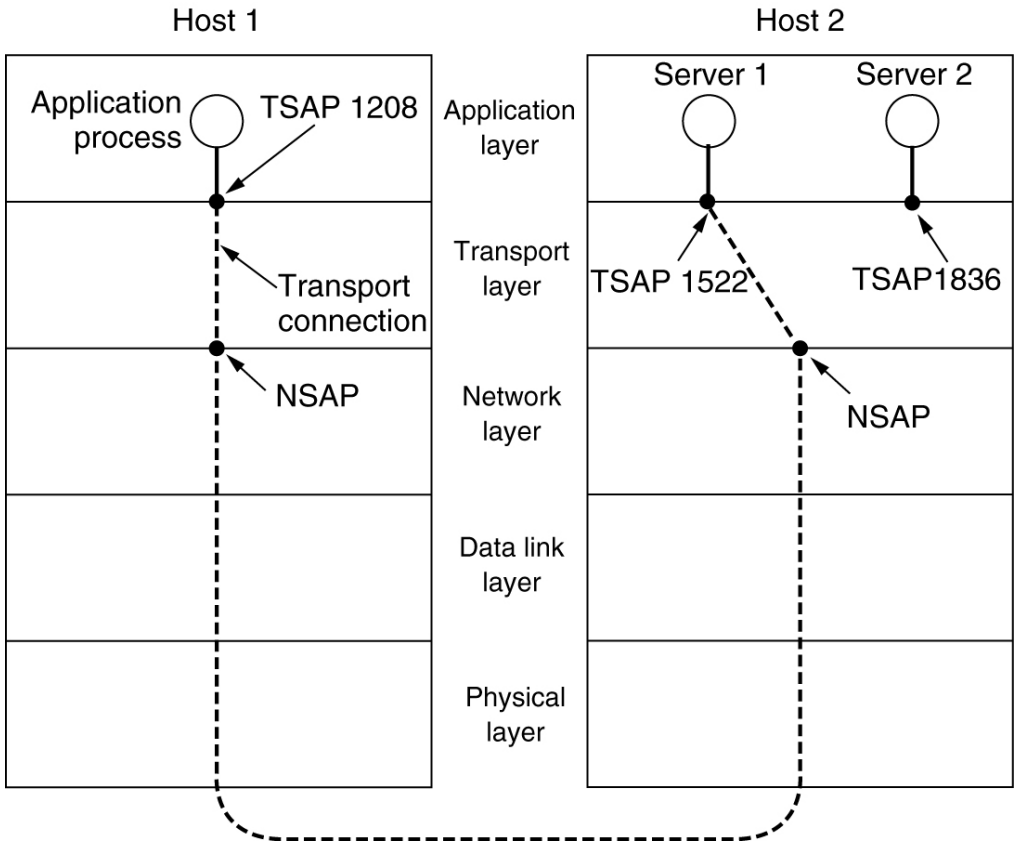
Internet Transport-Layer Protocols

- Reliable, in-order delivery (TCP)
  - Congestion control
  - Flow control
  - Connection setup
- Unreliable, unordered delivery: UDP
  - No-frills extension of “best-effort” IP
- Services not available:
  - Delay guarantees
  - Bandwidth guarantees



Addressing and Multiplexing

- Provide multiple *service access points (SAP)* to multiplex several applications
  - SAPs can identify connections or data flows



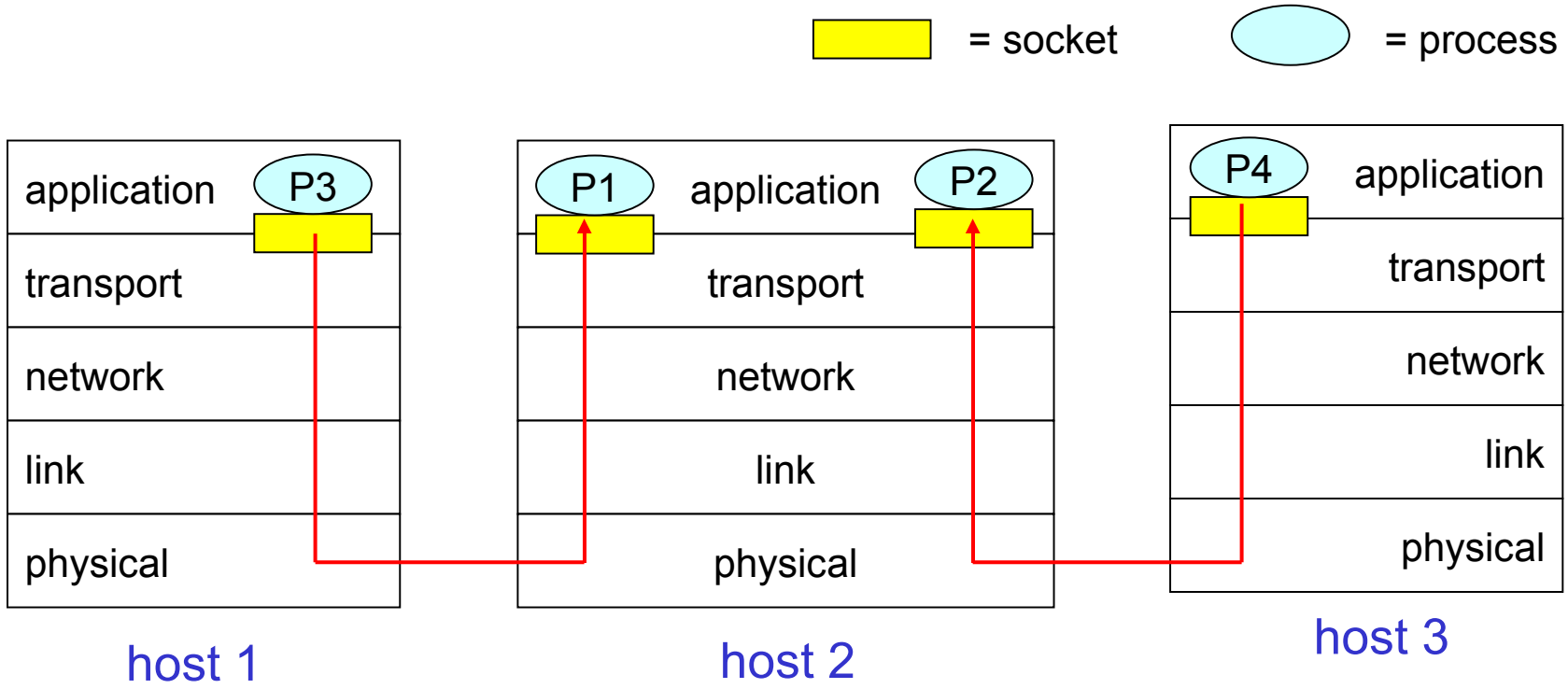
Multiplexing/Demultiplexing

Multiplexing at send host:

Gathering data from multiple sockets, enveloping data with header

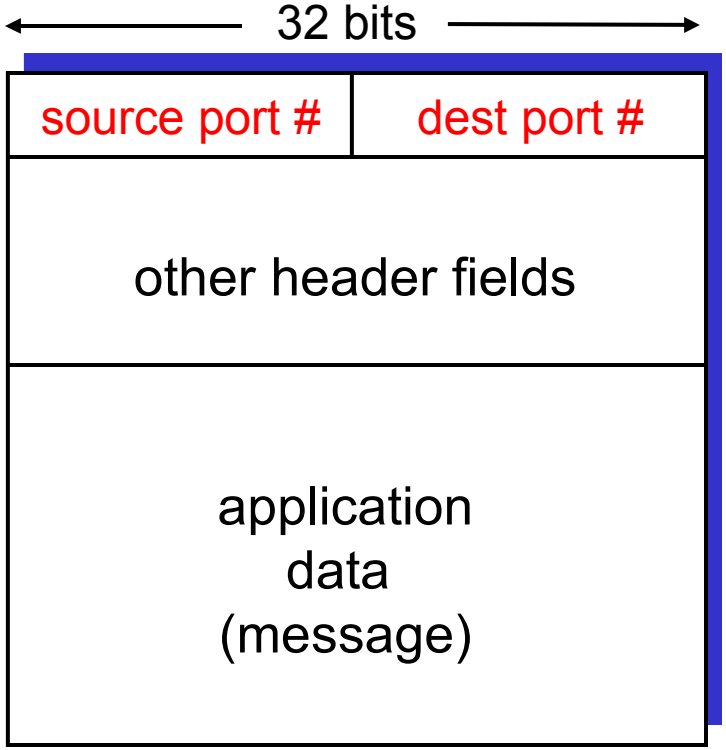
Demultiplexing at rcv host:

Delivering received segments to correct socket



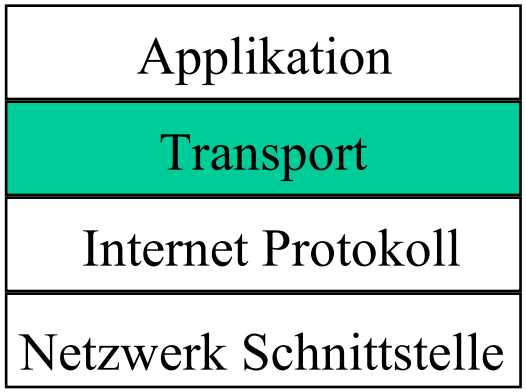
How Demultiplexing Works

- Host receives IP datagrams
  - Each datagram has source IP address, destination IP address
  - Each datagram carries 1 transport-layer segment
  - Each segment has source, destination port number (recall: well-known port numbers for specific applications)



TCP/UDP segment format

User Datagram Protocol - UDP



- Motivation:
  - Beim Internet Protocol (IP) definiert eine Zieladresse einen Rechner.
  - Es fehlt die Möglichkeit einen bestimmten Benutzer oder Prozeß auf dem Rechner anzusprechen.
  - Das ***User Datagram Protocol*** (UDP) ermöglicht es, zwischen verschiedenen Zielen innerhalb eines Rechners zu unterscheiden.

Protocol-Port-Konzept

- Die intuitive Lösung als Ziele Prozesse zu nehmen, hat Nachteile:
  - Prozesse werden dynamisch erzeugt (fork) und vernichtet (exit): Die Absender wissen selten, welcher Prozeß zu einem bestimmten Zeitpunkt existiert oder nicht.
  - Prozesse sollten ersetzt werden können, ohne daß alle Sender benachrichtigt werden müssen. (Ein Reboot kann alle Prozeßnummern ändern!).
  - Ziele müssen aufgrund der Dienste (Funktionen), die sie anbieten, identifiziert werden (Z.B. Ziel: Fileserver, egal welcher Prozeß sich dahinter verbirgt).
  - Prozesse, die mehr als einen Dienst anbieten, müssen entscheiden können, welcher Dienst gewünscht ist.

Protocol-Port-Konzept /2

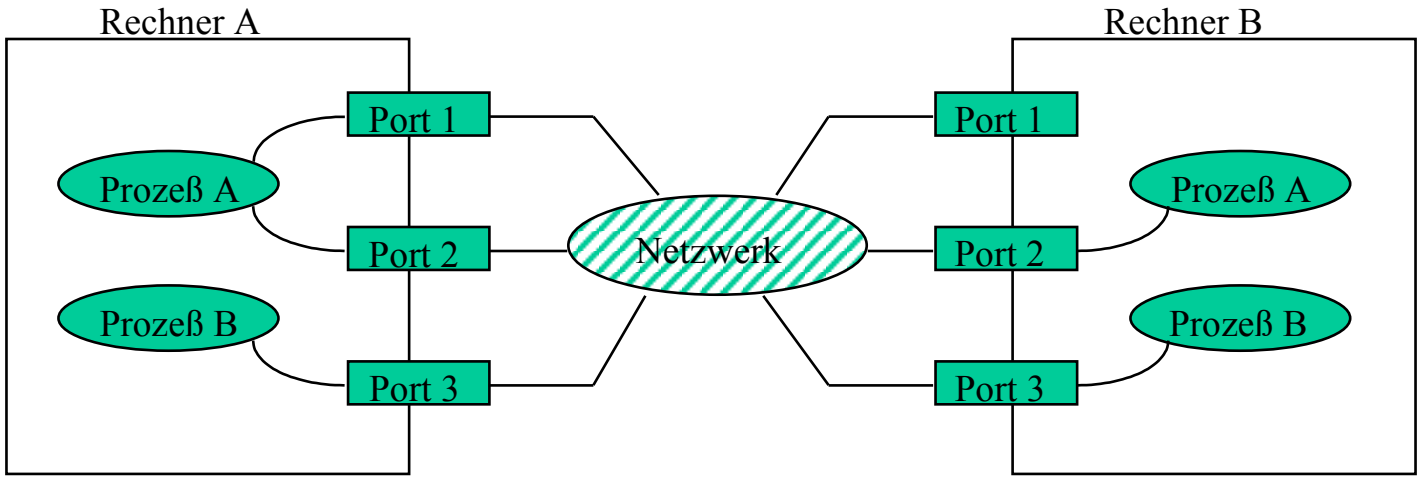
- Anstelle von Prozessen werden abstrakte Zielpunkte, ***Protocol Ports***, verwendet.

- Das Betriebssystem stellt Mechanismen zur Verfügung, um
  - Ports festzulegen und
  - auf Ports zuzugreifen.

- Die Protokoll-Software innerhalb des Betriebssystems synchronisiert den Zugriff auf einen bestimmten Port:
  - Pakete, die an den Port gehen werden in eine Warteschlange für den Port gestellt, bis ein Prozeß auf den Port zugreift.
  - Prozesse, die auf einen Port zugreifen, werden blockiert, bis ein Paket für den Port angekommen ist.

Protocol-Port-Konzept /3

- Zur Kommunikation mit einem fremden Port muß der Sender sowohl
  - die Internet-Adresse als auch
  - die Port-Nummer wissen.



- In jedem Paket werden Internet-Adresse und Port-Nummer des Empfängers und des Senders angegeben (Antwort einfach möglich).

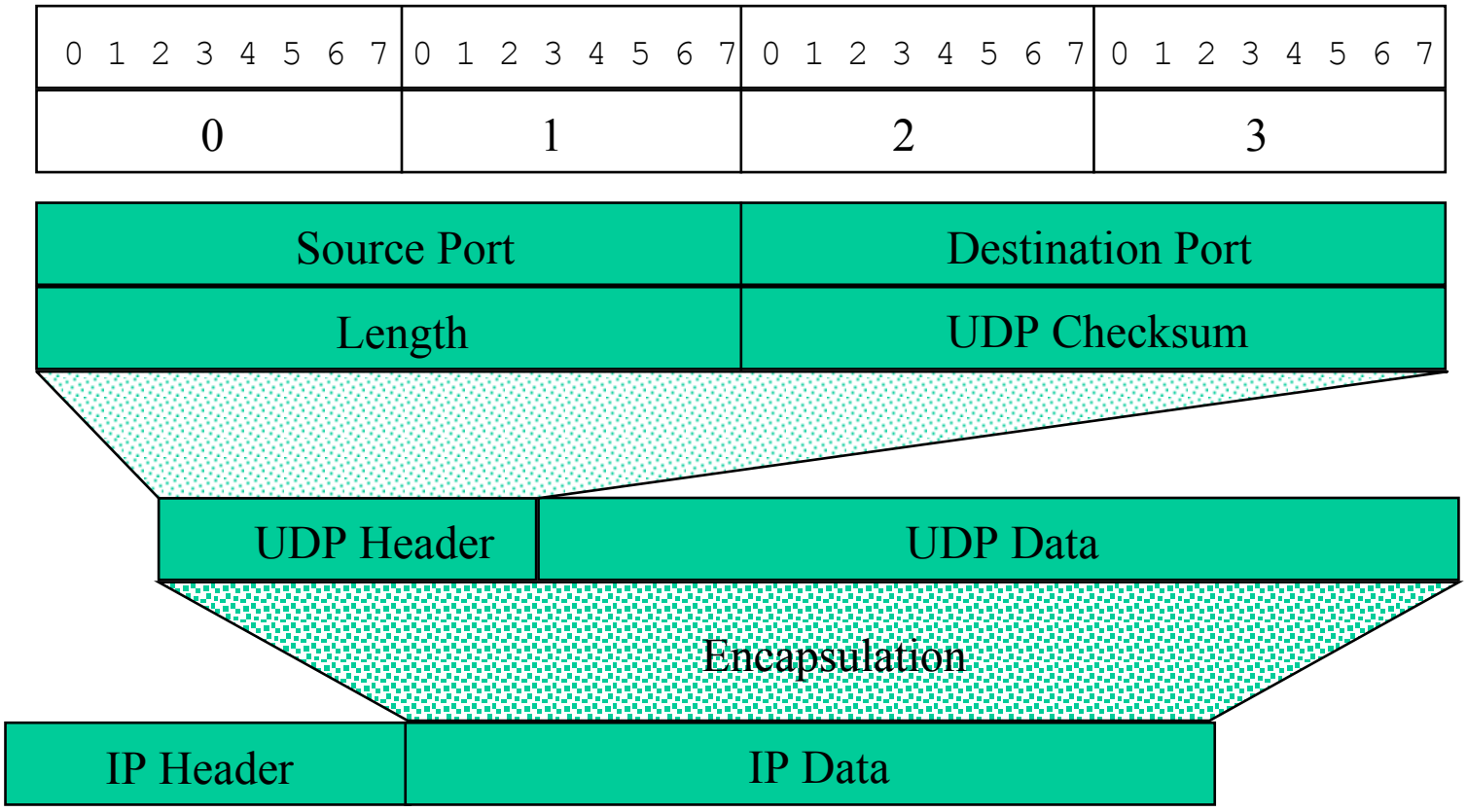
UDP-Port-Nummern

- Bevor zwei Rechner miteinander kommunizieren können, müssen sie sich auf die entsprechenden Portnummern einigen.
- Es gibt zwei Möglichkeiten, Port-Nummern zu vergeben:
  - zentrale Vergabe der Portnummern (***universal assignment, well-known port assignment***)
  - dynamisches Zuordnen (***dynamic binding***)
- TCP/IP, also auch UDP, verwendet eine Hybridlösung:
  - einige Port-Nummern werden a priori vergeben (generelle Anwendungsprogramme);
  - die restlichen Port-Nummern stehen für Benutzerprogramme zur Verfügung.

Ausgewählte UDP-Port-Nummern

Service	Port-Nummer	Kommentar
echo	7/udp	
time	37/udp	timeserver
name	42/udp	nameserver
tftp	69/udp	
sunrpc	111/udp	
who	513/udp	whod
talk	517/udp	

UDP-Header

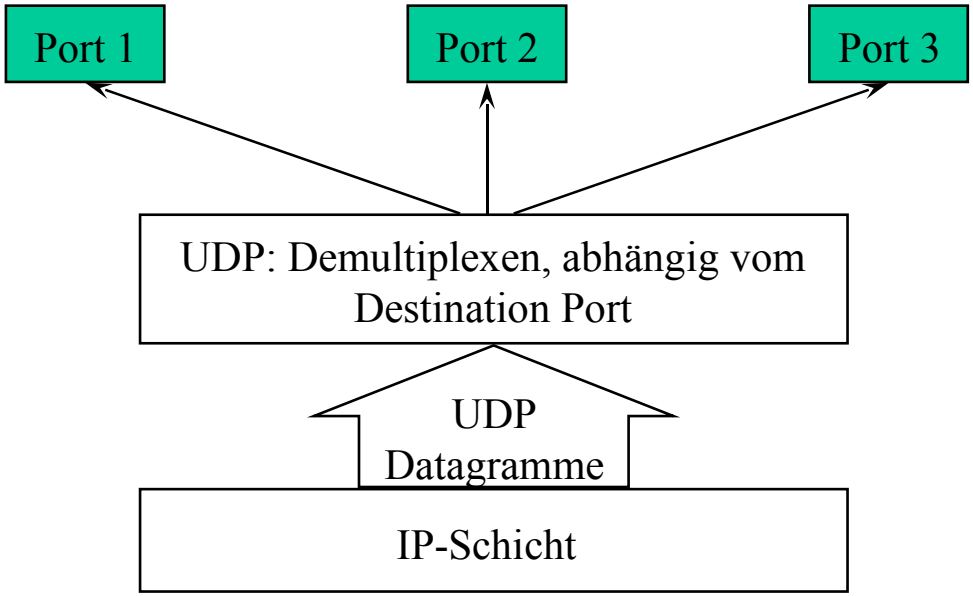

0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0
1
2
3
Source Port
Destination Port
Length
UDP Checksum
UDP Header
UDP Data
Encapsulation
IP Header
IP Data

Eigenschaften des User Datagram Protocol's

- UDP verwirklicht das Protokoll-Port-Konzept für IP.
- UDP bietet denselben unzuverlässigen, verbindungslosen Übertragungs-Service wie IP.
- UDP-Pakete können
  - verloren gehen,
  - in falscher Reihenfolge ankommen,
  - zu schnell beim Empfänger ankommen.
- UDP-Pakete heißen ***User Datagram***.

Demultiplexen

- UDP-Datagramme werden in IP-Datagramme gekapselt.
- Das UDP-Protokoll erhält vom IP-Protokoll alle UDP-Datagramme.
- Aufgrund des Destination Port's müssen die UDP-Datagramme demultiplext werden.



Connection oriented transport protocols

- Recall the two types of communication services to be distinguished:
  - **Connection-oriented service**
  - **Connectionless service**

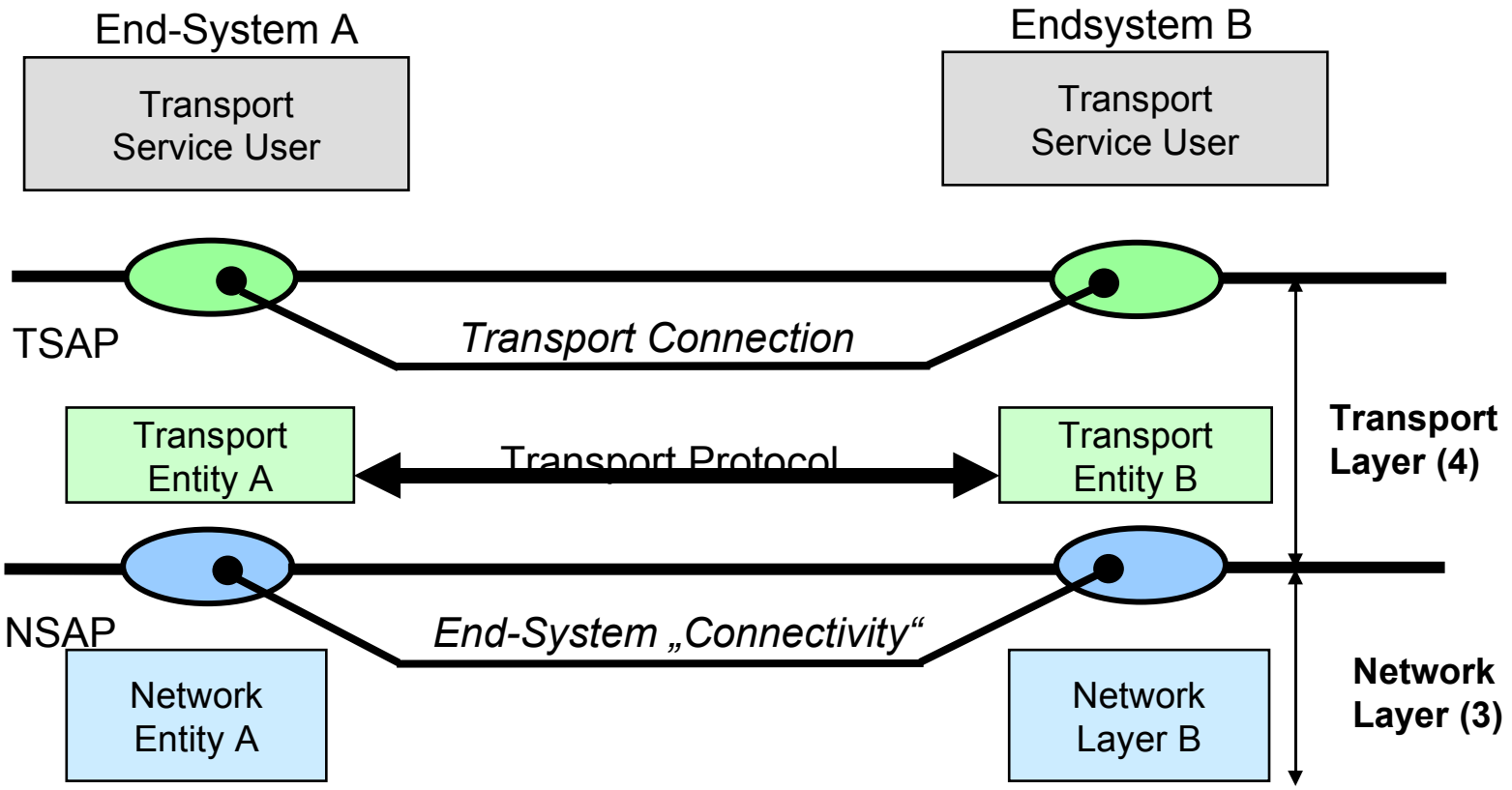
In the following, we will deal with connection-oriented services

- In principle, there are three phases of a connection:
  - Connection establishment phase (Connect)
  - Data transfer phase (Data)
  - Connection release phase (Disconnect)

For every phase there are specific service primitives

- When talking about the service of a specific layer, we usually add a ***layer specific prefix*** to the primitives, e.g.:
  - Transport Layer: T-Connect, T-Data, T-Disconnect
  - Network Layer: N-Connect, N-Data, N-Disconnect
    (***but***: network layer of the ***Internet*** provides a connectionless service)

Transport Connections and End-System „Connectivity“



Transport Connection Establishment (OSI Terminology)

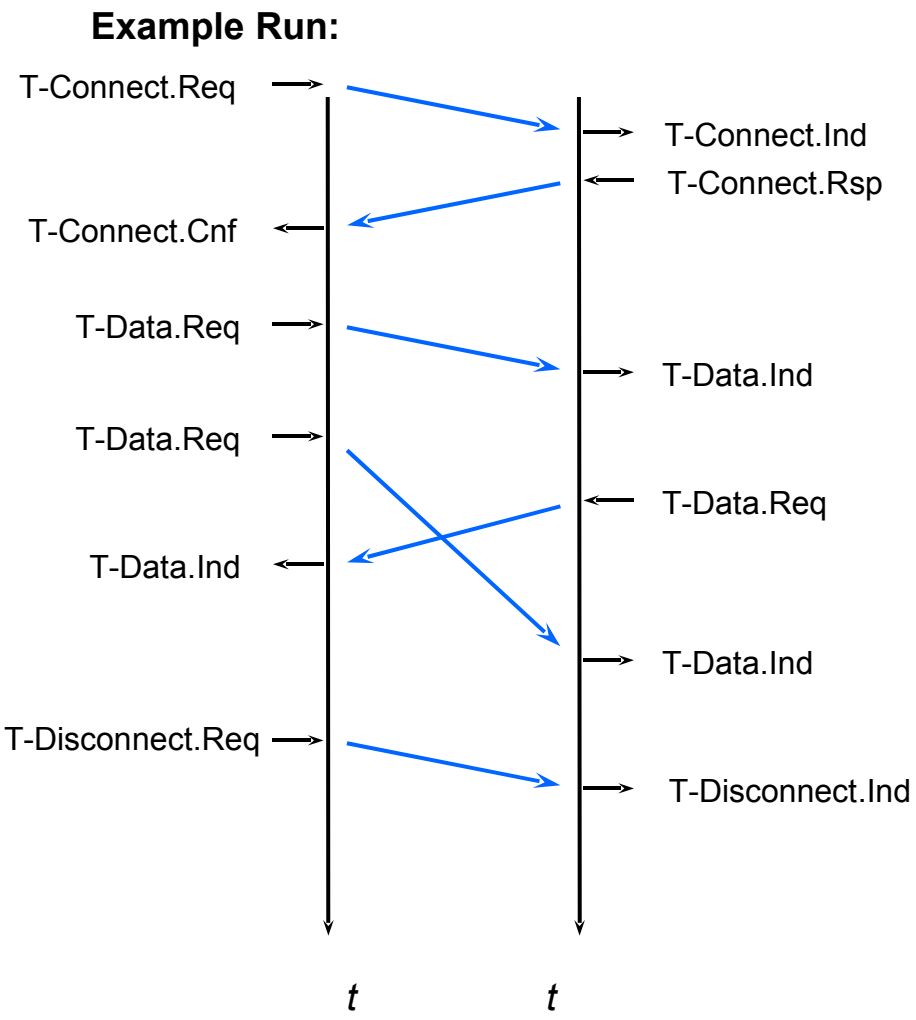
- Confirmed service primitive: T-Connect

- Primitive:
  - T-Connect.Request (Destination Address, Source Address)
  - T-Connect.Indication (Destination Address, Source Address)
  - T-Connect.Response (Responding Address)
  - T-Connect.Confirmation (Responding Address)

- Parameters:
  - Destination Address: Address of the called transport service user
    (= Application)
  - Source Address: Address of the calling service user
  - Responding Address: Address of the responding service user (in general, this is the address of the called service user)

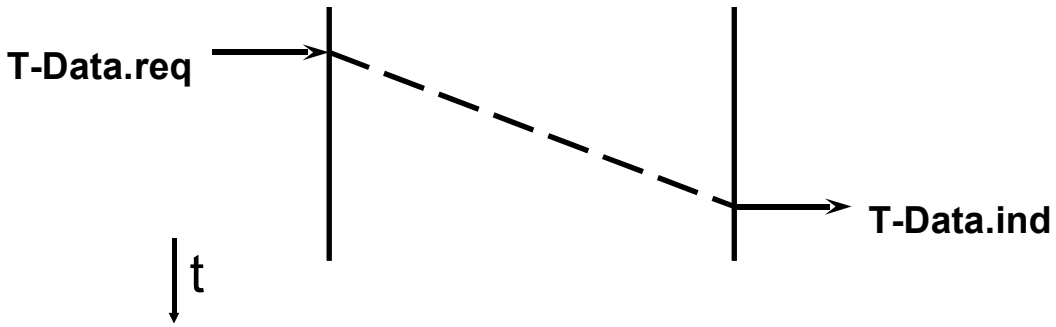
Transport Layer Services in a Message Sequence Chart

Primitive	Type
T-Connect	confirmed
T-Data	***unconfirmed***
T-Disconnect	unconfirmed (or confirmed)



Data Transfer Service

- Data Transfer Service: T-Data
  - unconfirmed service
- Primitive:
  - T-Data.req (userdata)
  - T-Data.ind (userdata)
- Parameter:
  - Userdata: transport service data unit to be transfered (TSDU, can have arbitrary length)

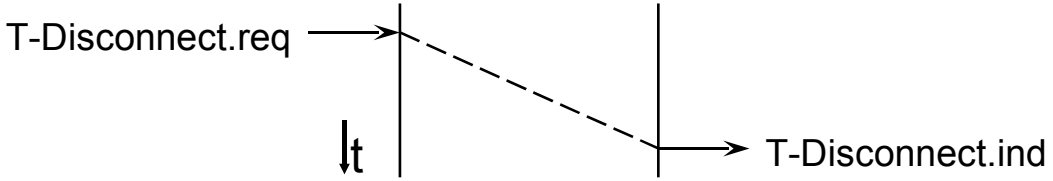


Connection Release (1)

- Unconfirmed release service: T-Disconnect
- Usage:
  - Abrupt teardown of a connection, loss of TSDUs is possible
  - Rejection of a connection establishment request
- Primitives:
  - T-Disconnect.req (userdata)
  - T-Disconnect.ind (cause, userdata)
- Parameters:
  - Cause of the teardown, e.g.:
    - unknown
    - requested by remote user
    - lack of local or remote resources for the transport service provider
    - Quality of service below minimal level
    - error occured in transport service provider
    - can not reach remote transport service user
  - User Data: TSDU to be transfered (max. length e.g. 64 Byte)

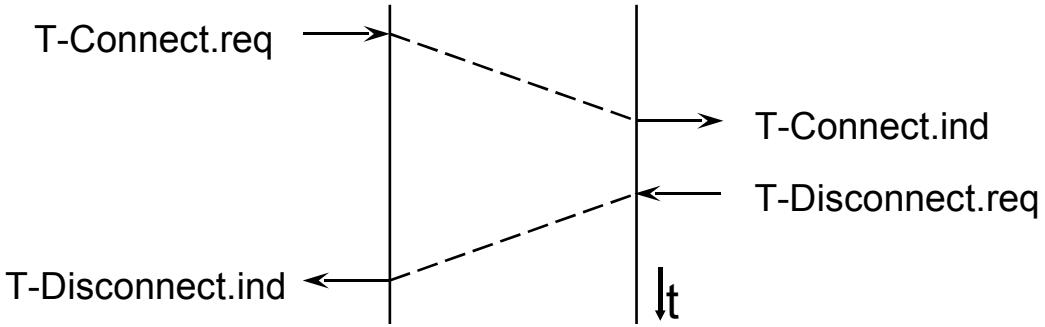
Connection Release (2)

Teardown of a connection by a *service user*:

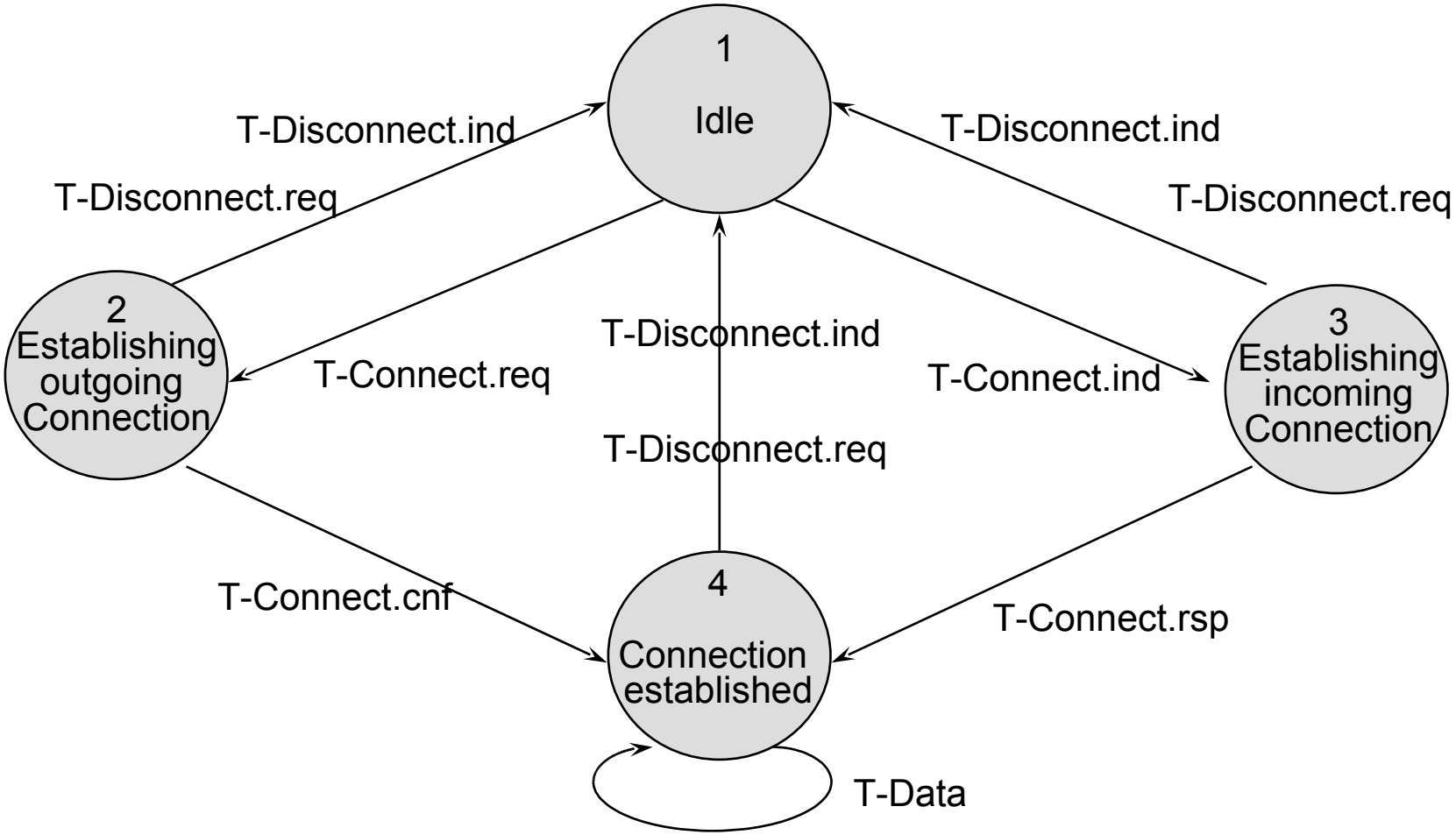


Teardown of a connection by a *service provider*:

Rejection of connection establishment:

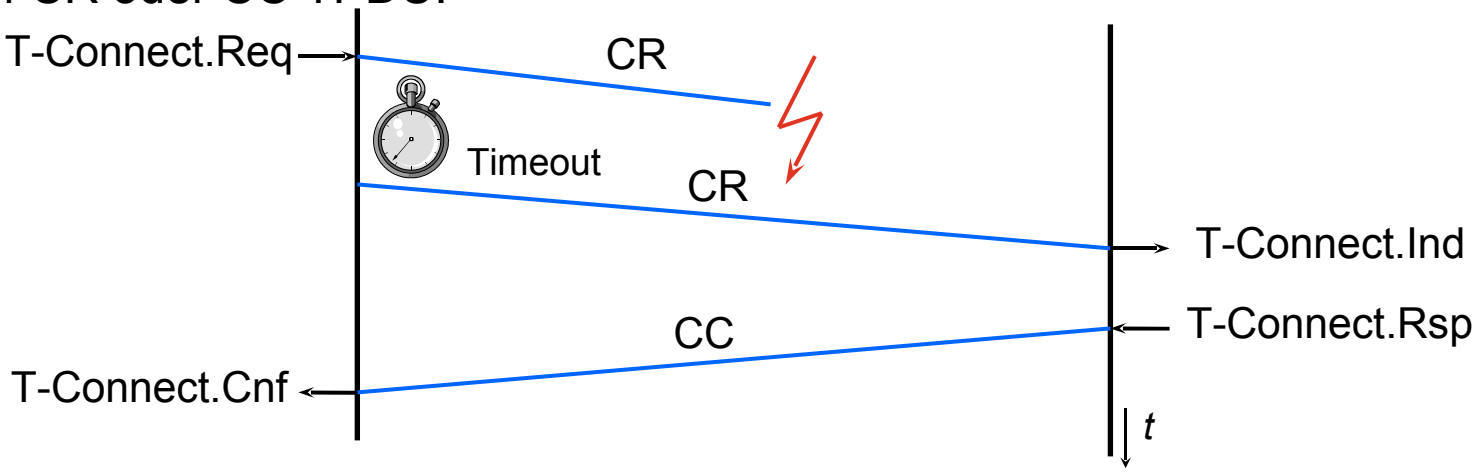


State Diagram for a Transport Service Access Point

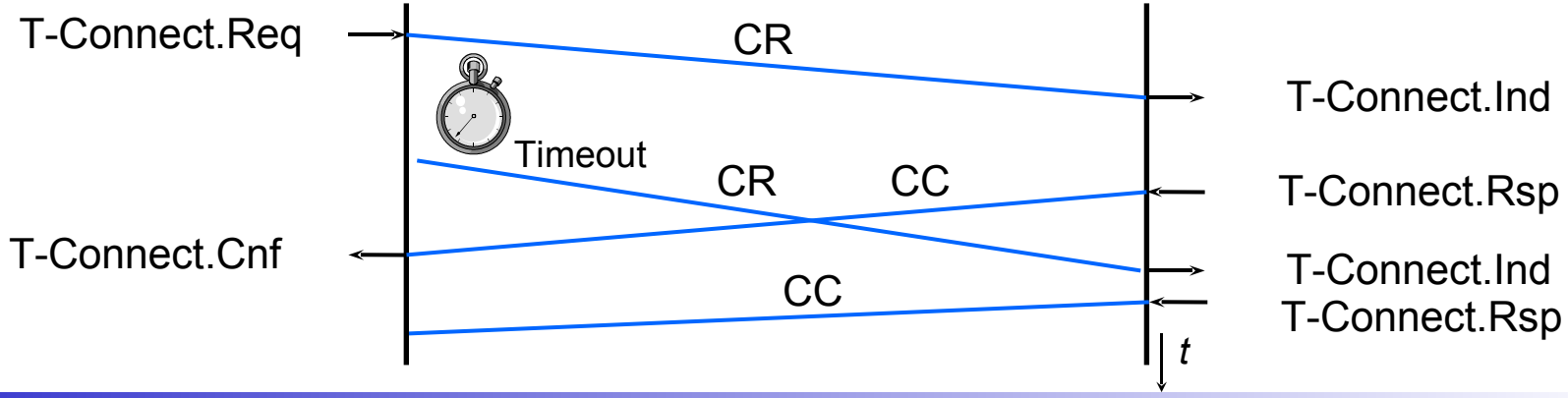


Errors during Connection Establishment

- Loss of CR oder CC TPDU:

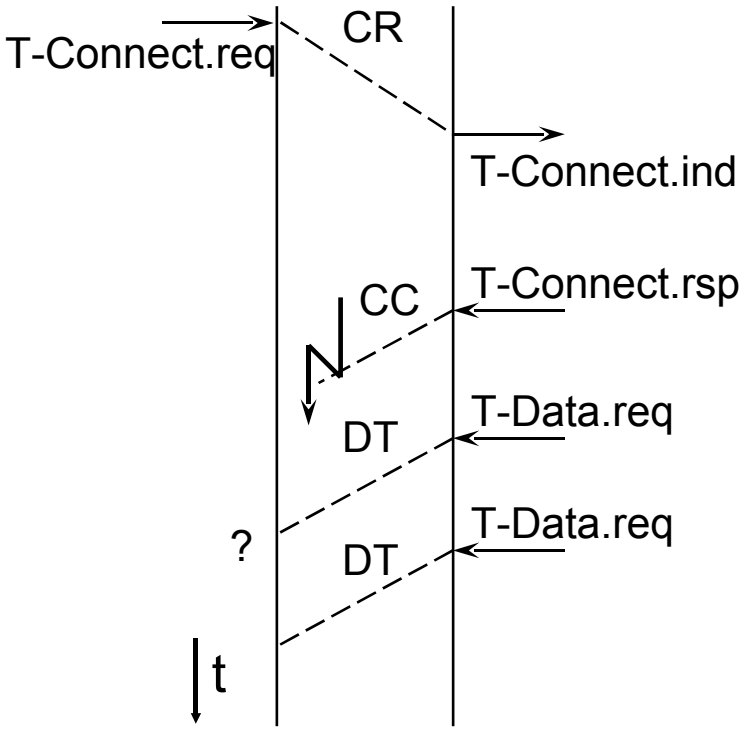


- Duplication of TPDUs:

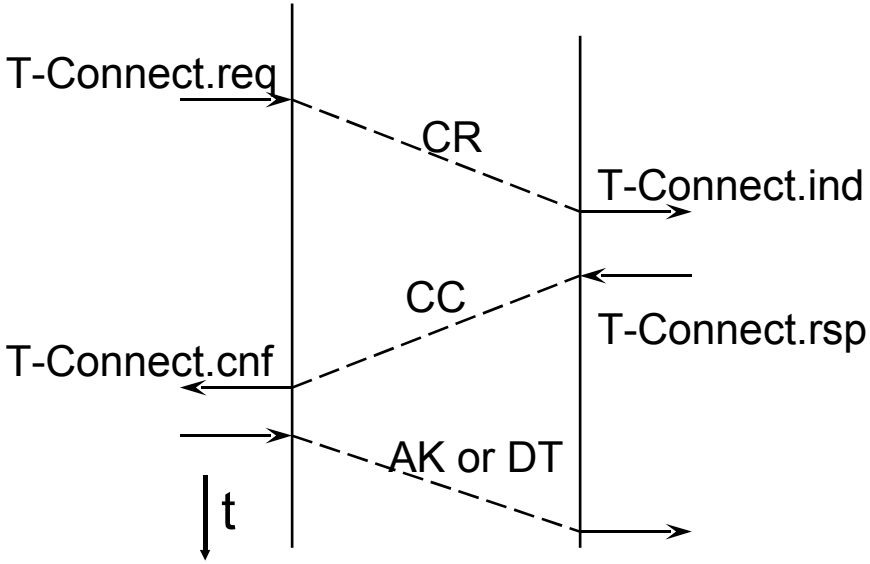


Three-Way Handshake

Problem: Loss of CC TPDU



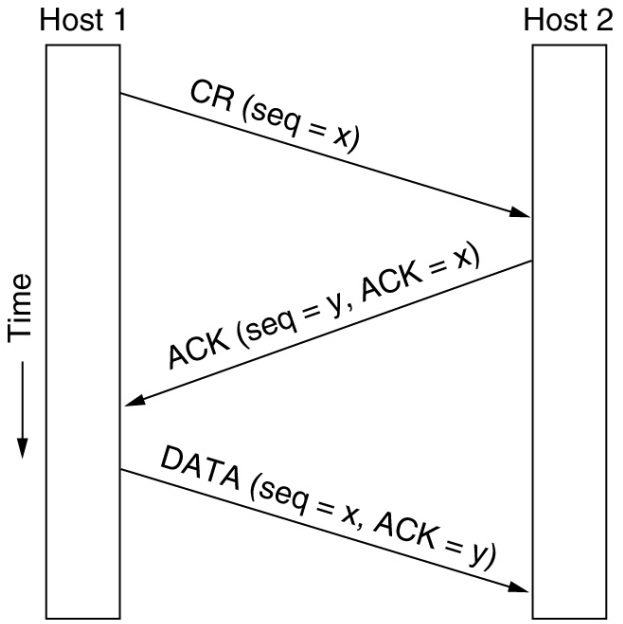
- Solution - Three-Way Handshake during connection establishment:
  - Connection is established, when both connection establishment TPDUs (CR and CC) have been acknowledged
  - Requires an additional AK (Acknowledge) or DT (Data) TPDU



Is Three-Way Handshake Sufficient?

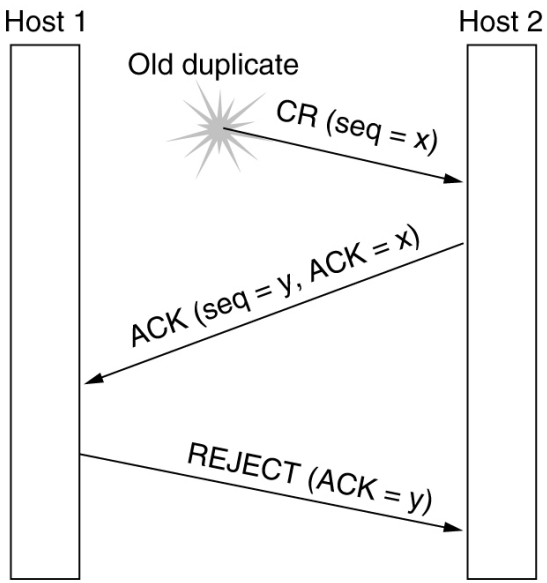
- No, it does not protect against delayed duplicates!
  - Problem: If both the connection request and the connection confirmation are duplicated and delayed, receiver again has no way to ascertain whether this is fresh or an old copy

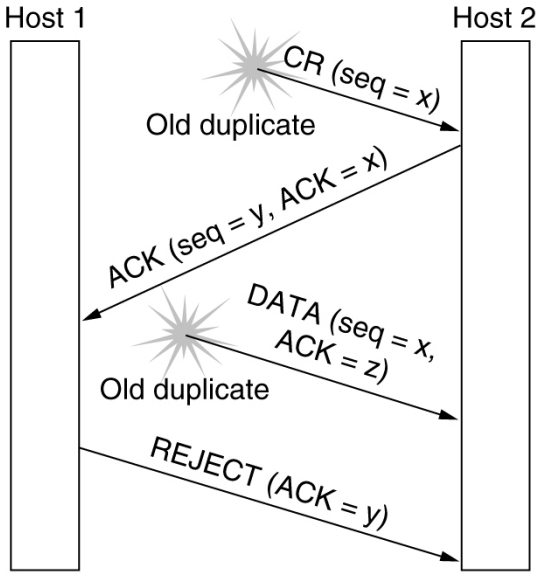
- Solution: Have the sender answer a question that the receiver asks!
  - Actually: Put ***sequence numbers*** into
    - connection request
    - connection acknowledgement,
    - and connection confirmation
  - Have to be copied by the receiving party to the other side
  - Connection only established if the correct number is provided
  - Sequence numbers should not be re-used too quickly (start with number higher than in last connection; wrap-around)



Three-Way Handshake + Sequence Numbers

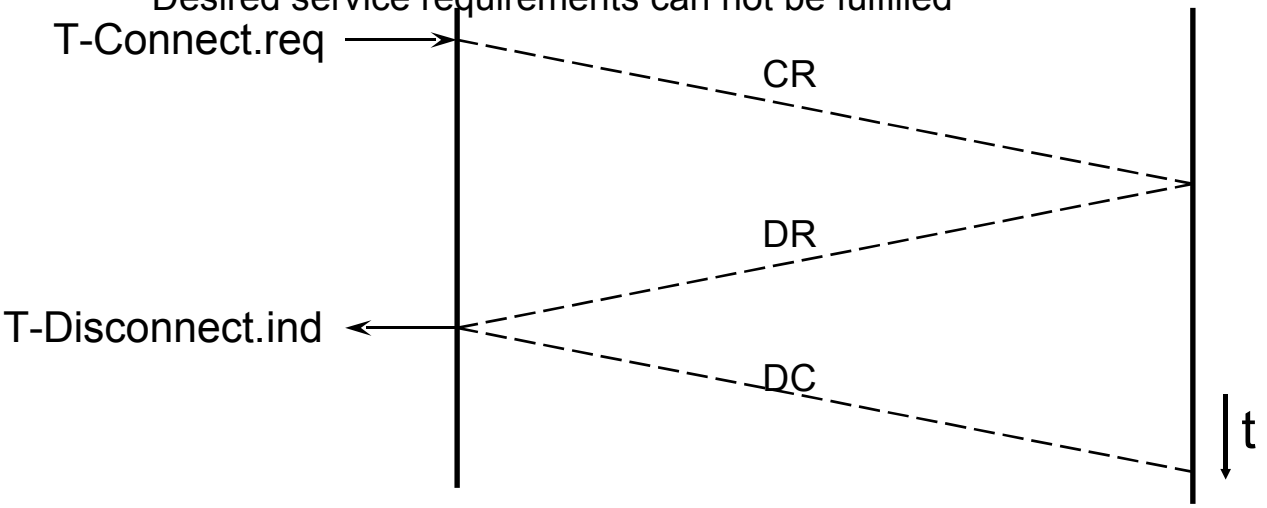
- Connection request appears as an old duplicate:
- Connection request & confirmation appear as old duplicates:
  - Two examples for critical cases (which are handled correctly):





Connection Rejection

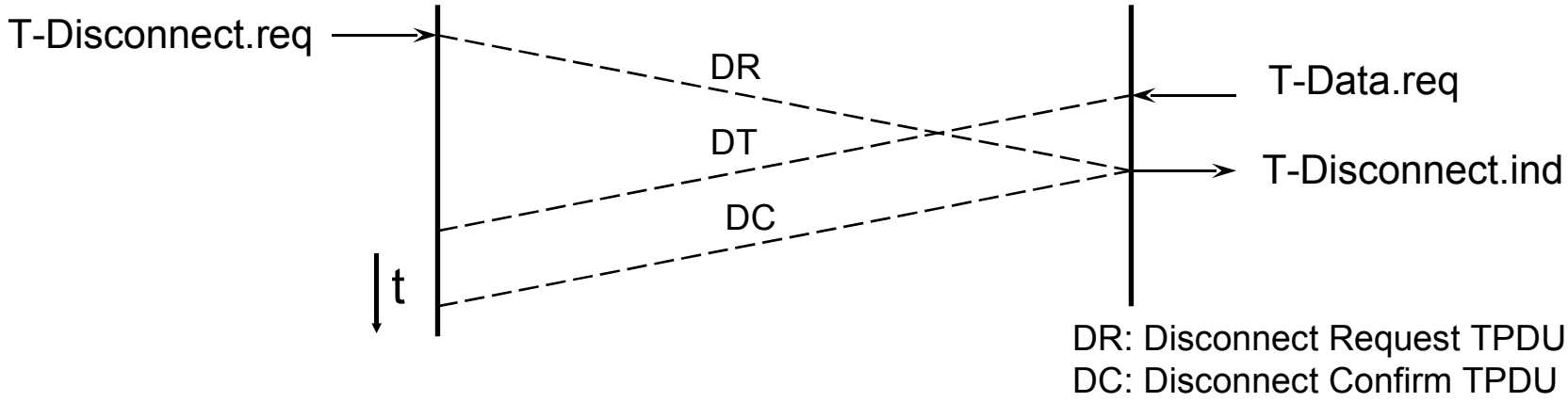
- Refusing an incoming connection request with a Disconnect-Request (DR) or Error-TPDU (reasons for this can be communicated)
  - Reasons:
    - Rejection by transport service user
    - Desired service requirements can not be fulfilled



DR: Disconnect Request TPDU
DC: Disconnect Confirm TPDU

Connection Release (1)

- Normal Release:
  - Teardown of an existing transport connection
  - This can cause loss of data that has not yet been acknowledged
    - The Internet transport protocol TCP avoids loss of data by requiring all sent PDUs to be acknowledged before a connection is closed
  - Variants:
    - Implicit: Teardown of network layer connection when remote peer entity suddenly is unreachable
    - Explicit: connection release with Disconnect-TPDUs

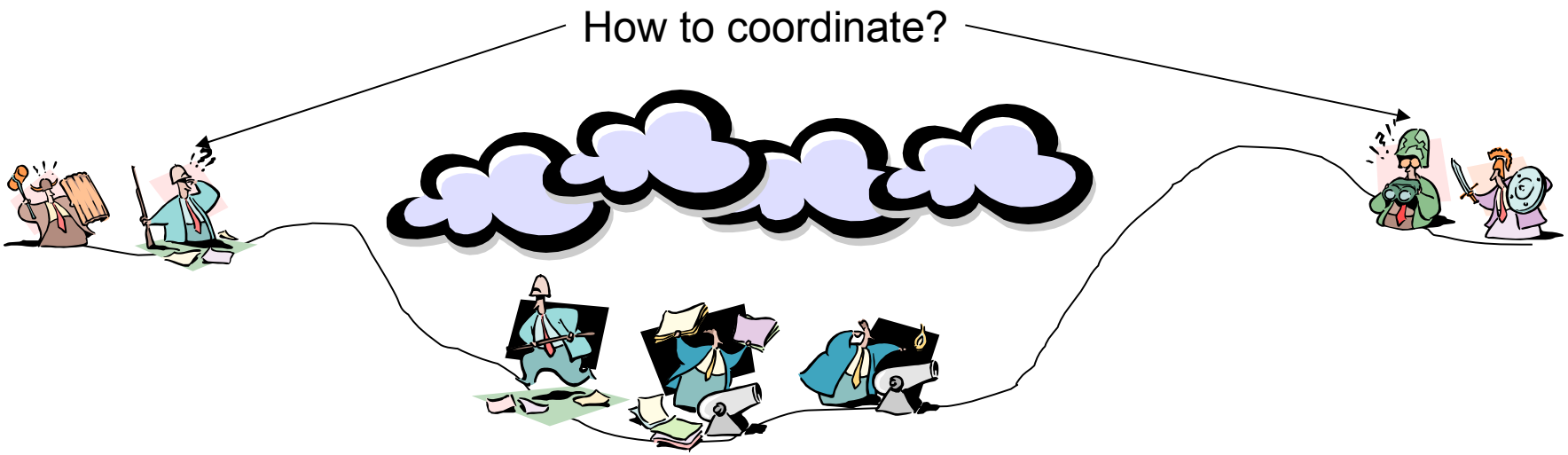


Connection Release (2)

- Once connection context between two peers is established, releasing a connection should be easy
  - Goal: Only release connection when both peers have agreed that they have received all data and have nothing more to say
  - I.e., both sides must have invoked a “Close”-like service primitive
- It fact, it is impossible
  - Problem: How to be sure that the other peer knows that you know that it knows that you know … that all data have been transmitted and that the connection can now safely be terminated?
- Analogy: Two army problem

Two Army Problem

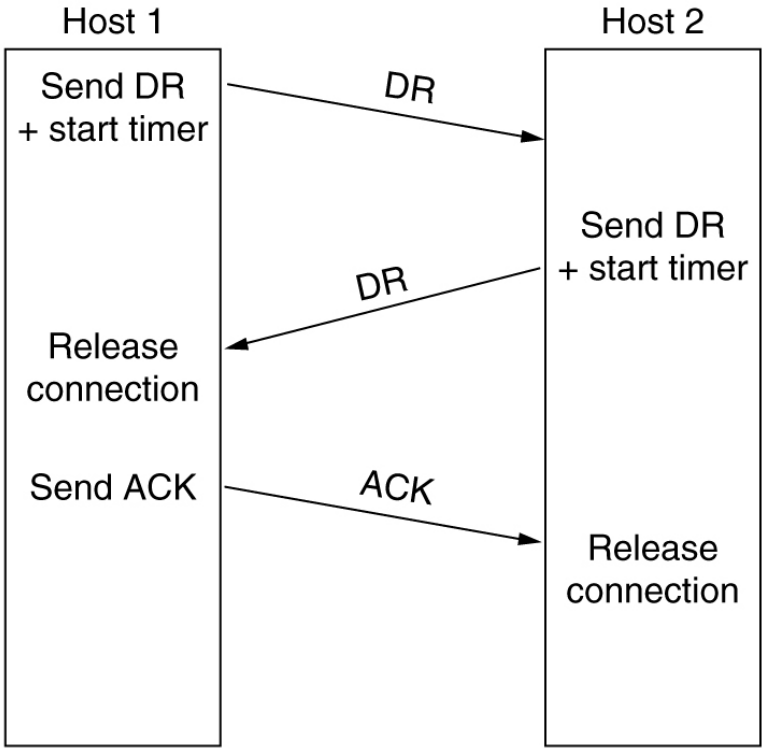
- Coordinated attack
  - Two armies form up for an attack against each other
  - One army is split into two parts that have to attack together – alone they will lose
  - Commanders of the parts communicate via messengers who can be captured
- Which rules shall the commanders use to agree on an attack date?
- Provably unsolvable if the network can loose messages



Connection Release in Practice

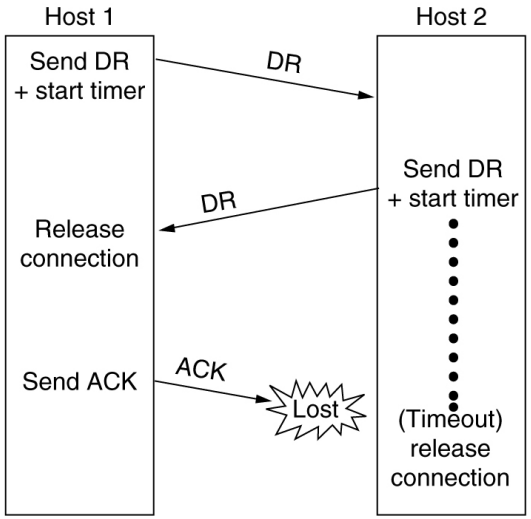
- Two army problem equivalent to connection release
- But: when releasing a connection, bigger risks can be taken

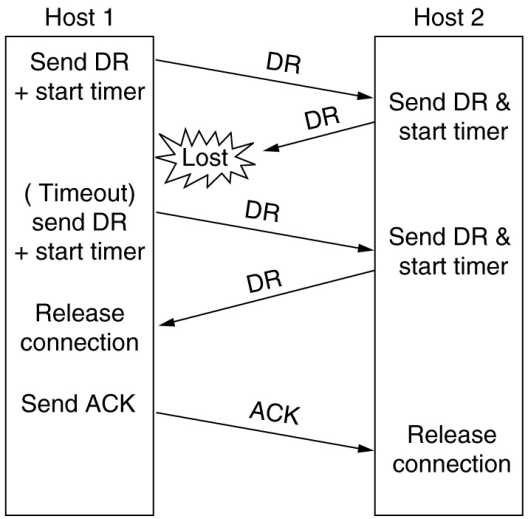
- Usual approach: Three-way handshake again
  - Send disconnect request (DR), set timer, wait for DR from peer, acknowledge it

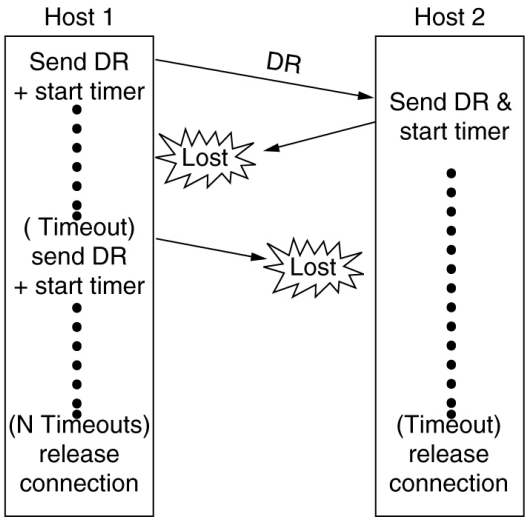


Problem Cases for Connection Release with 3WHS

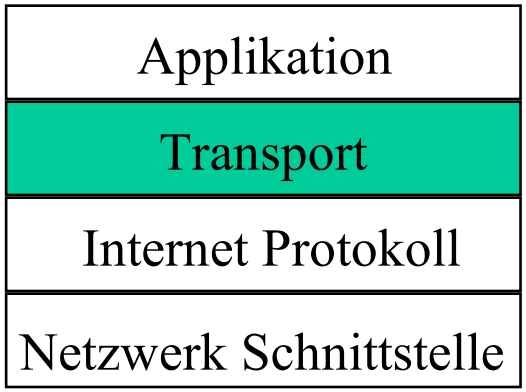
- Lost ACK solved by (optimistic) timer in Host 2
- Lost second DR solved by retransmission of first DR
- Timer solves (optimistically) case when 2nd DR and ACK are lost







Transmission Control Protocol - TCP



- Das ***T**ransmission/**T**ransport **C**ontrol **P**rotocol* (TCP) ermöglicht eine zuverlässige Datenübertragung.

- Die unteren Schichten stellen unzuverlässige Datenübertragungsmechanismen zur Verfügung.
  - Pakete können verloren gehen.
    - Übertragungsfehler zerstören die Datenbereiche, Checksummen sind falsch, Netzwerk-Hardware fällt aus, Netzwerk ist überlastet, ...
  - Das dynamische Routing kann falsche Datenübertragungen verursachen.
    - die Reihenfolge kann durcheinander geraten, Pakete kommen zu spät an, Pakete kommen zweimal an, ...

TCP

- TCP ist ein Protokoll, nicht eine bestimmte Software.
  (Protokoll <=> Implementierung)

- TCP definiert bzw. spezifiziert
  - das Format der Daten und Quittungen (Flußsteuerung),
  - wie zwischen Empfängern auf einer bestimmten Maschine unterschieden wird,
  - wie Fehler behoben werden,
  - wie Verbindungen aufgebaut werden.

Eigenschaften des Transport Control Protocol's

- TCP-Verbindungen lassen sich durch die folgenden fünf Eigenschaften charakterisieren:
  - Stream-Orientierung
  - Virtual Circuit Connection
  - Gepufferte Übertragung
  - Unstrukturierte Übertragung
  - Full-Duplex-Verbindung

Eigenschaften des Transport Control Protocol's

- TCP-Verbindungen lassen sich durch die folgenden fünf Eigenschaften charakterisieren:
  - Stream-Orientierung
  - Virtual Circuit Connection
  - Gepufferte Übertragung
  - Unstrukturierte Übertragung
  - Full-Duplex-Verbindung

2. Stream-Orientierung
   - Tauschen zwei Anwendungsprogramme Datenmengen aus,
     - bezeichnet man die Datenmenge insgesamt als Bit-Stream (Fluß),
     - wird der Stream in Octets (8 Bit) unterteilt,
     - erhält der Empfänger die Octets in genau derselben Reihenfolge, wie sie der Sender abgeschickt hat.

Eigenschaften des Transport Control Protocol's /2

1. Virtual Circuit Connection

- Für die Übertragung wird eine Verbindung aufgebaut.
  - Anwendungsprogramme müssen ihr Kommunikationsvorhaben dem Betriebssystem mitteilen. (Eine Maschine führt einen "Anruf" bei der Partnermaschine durch.)
  - Protokoll-Software auf beiden teilnehmenden Rechnern kommuniziert miteinander. (Verifikation, daß die Übertragung gestattet ist, daß beide Seiten übertragungsbereit sind, ...)
  - Die Protokoll-Software informiert die jeweiligen Anwendungsprogramme, daß eine Verbindung (Connection) aufgebaut ist und die Übertragung beginnen kann.
  - Während der Übertragung wird fortlaufend kontrolliert, ob die Daten in der richtigen Reihenfolge ankommen.
  - Wird die Übertragung unterbrochen (Netzwerk- oder Hardware-Ausfall), so werden beide kommunizierenden Anwendungsprogramme von der Protokoll-Software darüber informiert.
- Die Verbindung (Connection) scheint für die Anwendungssoftware eine eigene Hardware-Verbindung (Circuit) zu sein, ist aber nur virtuell (Virtual).

Eigenschaften des Transport Control Protocol's /2

3. Gepufferte Übertragung

- TCP-Implementierungen sammeln Übertragungsdaten, bis ein Paket damit gefüllt werden kann (effiziente Übertragung).

4. Unstrukturierte Stream-Verbindung

- Analog zu Unix-Dateien sind Stream-Verbindungen nicht strukturiert. (*Das Satzende wird nicht vom TCP markiert*)

5. Full-Duplex-Verbindung

- TCP-Verbindungen erlauben eine gleichzeitige Übertragung in beiden Richtungen (Full Duplex).

TCP-Algorithmen

- Nagle-Algorithmus (Small Packet Avoidance Algorithm)
  - Durch Zwischenpuffern beim Sender wird das Erzeugen kleiner Datensegmente verhindert.
- Slow-Start-Algorithmus
  - Nach Zusammenbruch (Durchsatz) wird versucht, die Datenmenge langsam zu erhöhen, bis das Optimum eines gleichmäßigen Datenflusses erreicht ist.
- Congestion-Avoidance-Algorithmus
  - Nachdem das Optimum eines gleichmäßigen Datenflusses gefunden wurde, kann ansteigende Netzlast ein Verringern des Datenflusses erfordern, um das Optimum auf niedrigerem Durchsatzniveau zu finden.
- Karn-Algorithmus
  - Der Karn-Algorithmus dient zur Ermittlung der Round Trip Time eines Datensegments zwischen Sender und Empfänger.
- Silly-Window-Avoidance-Algorithmus
  - Die Fensterfreigabe erfolgt nur, wenn 1/4 des Empfangspuffers frei ist, da häufige Freigabe kleiner Fensterbereiche die Netzlast durch übermäßigen Anteil von Steuernachrichten erhöht.

TCP-Timer

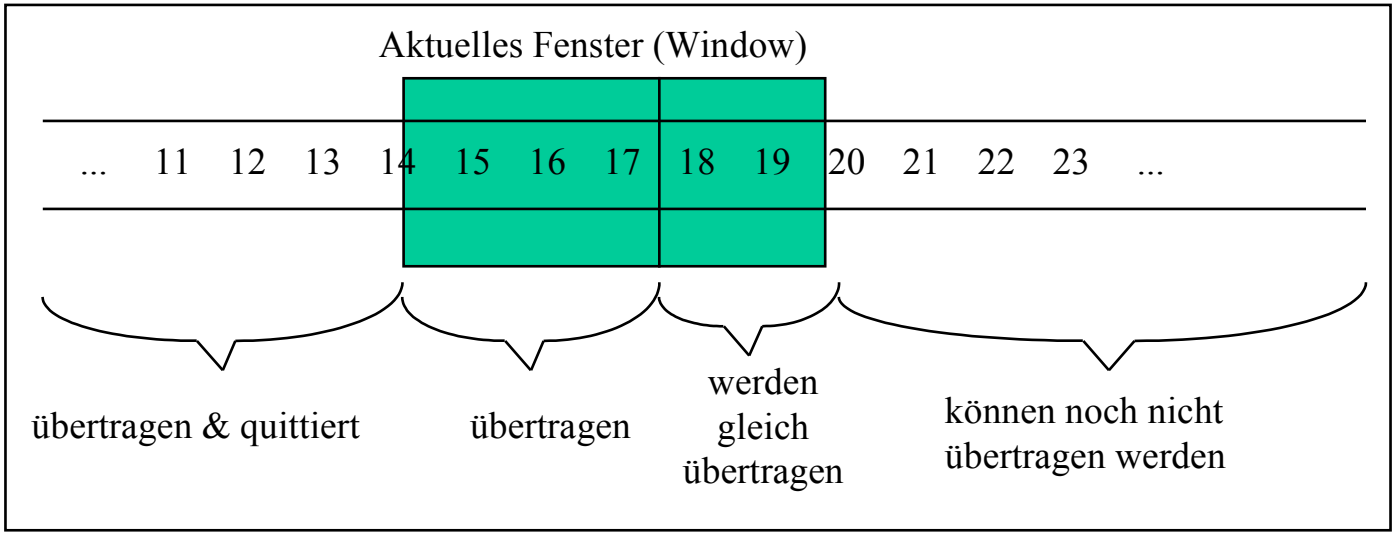
- *Retransmission Timer*
  - Nach Ablauf des Timers werden unbestätigte Datensegmente erneut gesendet.
- *Reconnection Timer*
  - minimale Zeit, die gewartet werden muß, um nach dem Abbau einer Verbindung erneut eine Verbindung zum gleichen Ziel aufzubauen
- *Retransmit-Syn-Timer*
  - minimale Zeit, die gewartet werden muß, um nach erfolglosem Verbindungsaufbauwunsch erneut eine Verbindungsaufbauanfrage zu senden

TCP-Timer /2

- *Persistence Timer*
  - Wenn Window = 0 wird die Empfangsbereitschaft des Senders überprüft, damit der Give Up Timer nicht unberechtigt zur Wirkung kommt.
- *Give Up Timer*
  - Zeit, nach der nach dem letzten Empfang eines Datensegments/Bestätigung die Verbindung abgebaut wird
- *Quiet Timer*
  - Timer, der die Portfreigabe nach Abbau einer Verbindung verzögert

TCP-Windowing

- TCP teilt den Datenfluß (Stream) für die Übertragung in Segmente ein.
- Jedes Segment geht - normalerweise - als IP-Datagramm über das Netz.
- TCP verwendet Windowing, um eine effiziente Übertragung der Flußsteuerung zu ermöglichen.

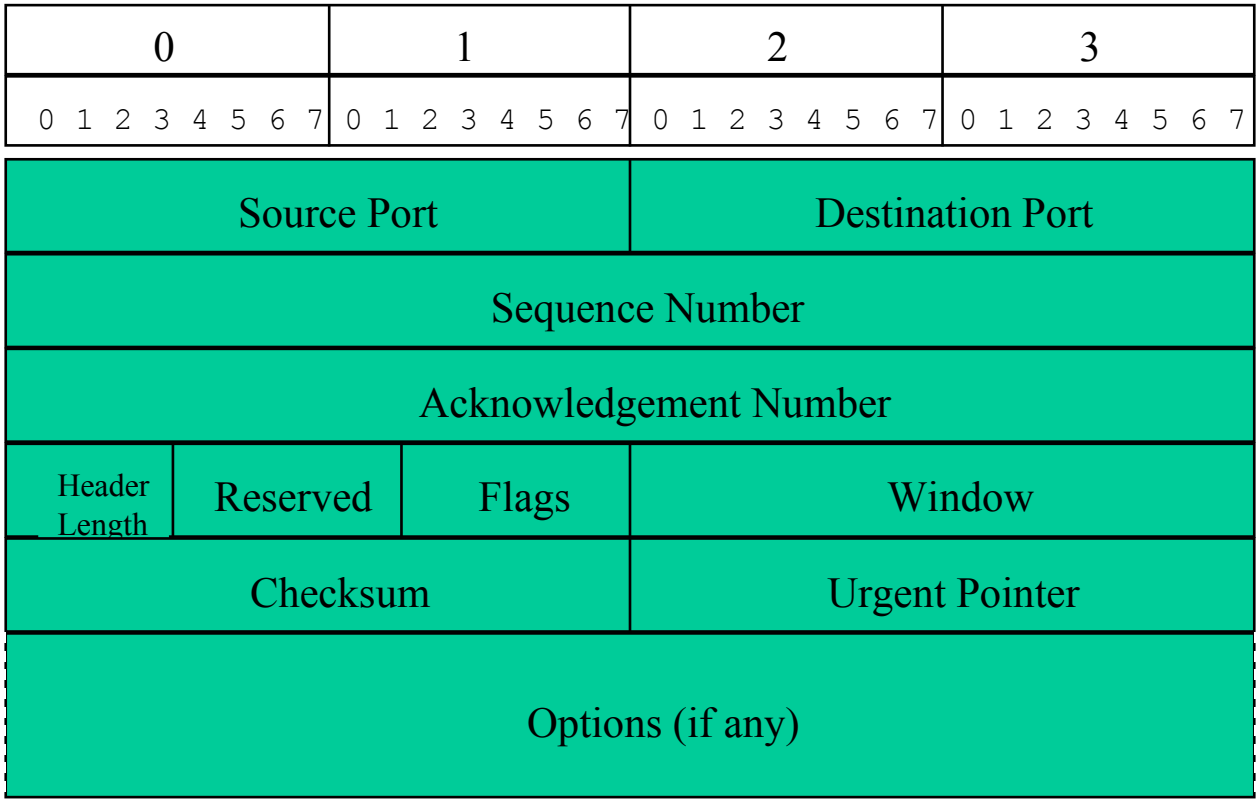


- Der Windowing-Mechanismus arbeitet auf der Byte-Ebene.
- TCP kann die Fenstergröße vergrößern oder verkleinern (Flußsteuerung).

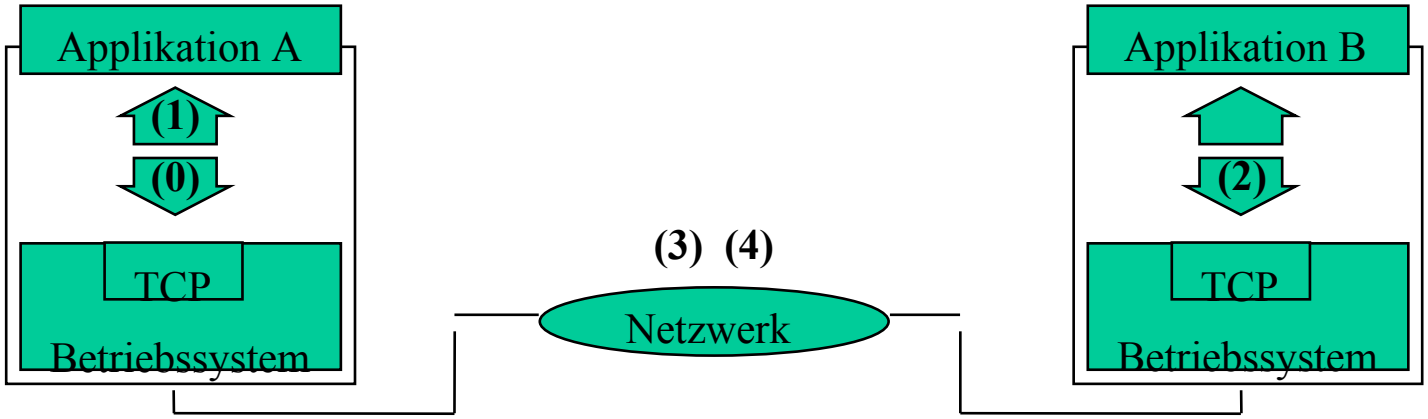
TCP-Ports und Verbindungen

- TCP Ports – analog zu UDP, zur Identifikation von Zielprozessen
- Jedem Port ist ein Integer als Identifikator zugeordnet.
- In IP-Datagrammen wird verwendetes Protokoll angegeben →
  Kein Konflikt mit UDP-Port-Nummern
- Bevor eine TCP-Verbindung etabliert wird, müssen beide Partner der Verbindung zustimmen.
- Begriffe:
  - *Port*: TCP-Paket und zugehörige Anwendung
  - *Socket*: Adresse der TCP-Verbindung (Internet-Adresse + Portnummer)
  - *Well known Ports*: Portnummern für Standard-Applikationen
    - Beispiele: FTP - 21, Telnet - 23, SMTP - 25, Rlogin - 513

TCP-Header


0
1
2
3
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
0 1 2 3 4 5 6 7
Source Port
Destination Port
Sequence Number
Acknowledgement Number
Header Length
Reserved
Flags
Window
Checksum
Urgent Pointer
Options (if any)

TCP-Verbindungsaufbau



(0) Partner A führt ein *Passive Open* durch, indem er dem Betriebssystem mitteilt, daß er Verbindungswünsche akzeptiert.

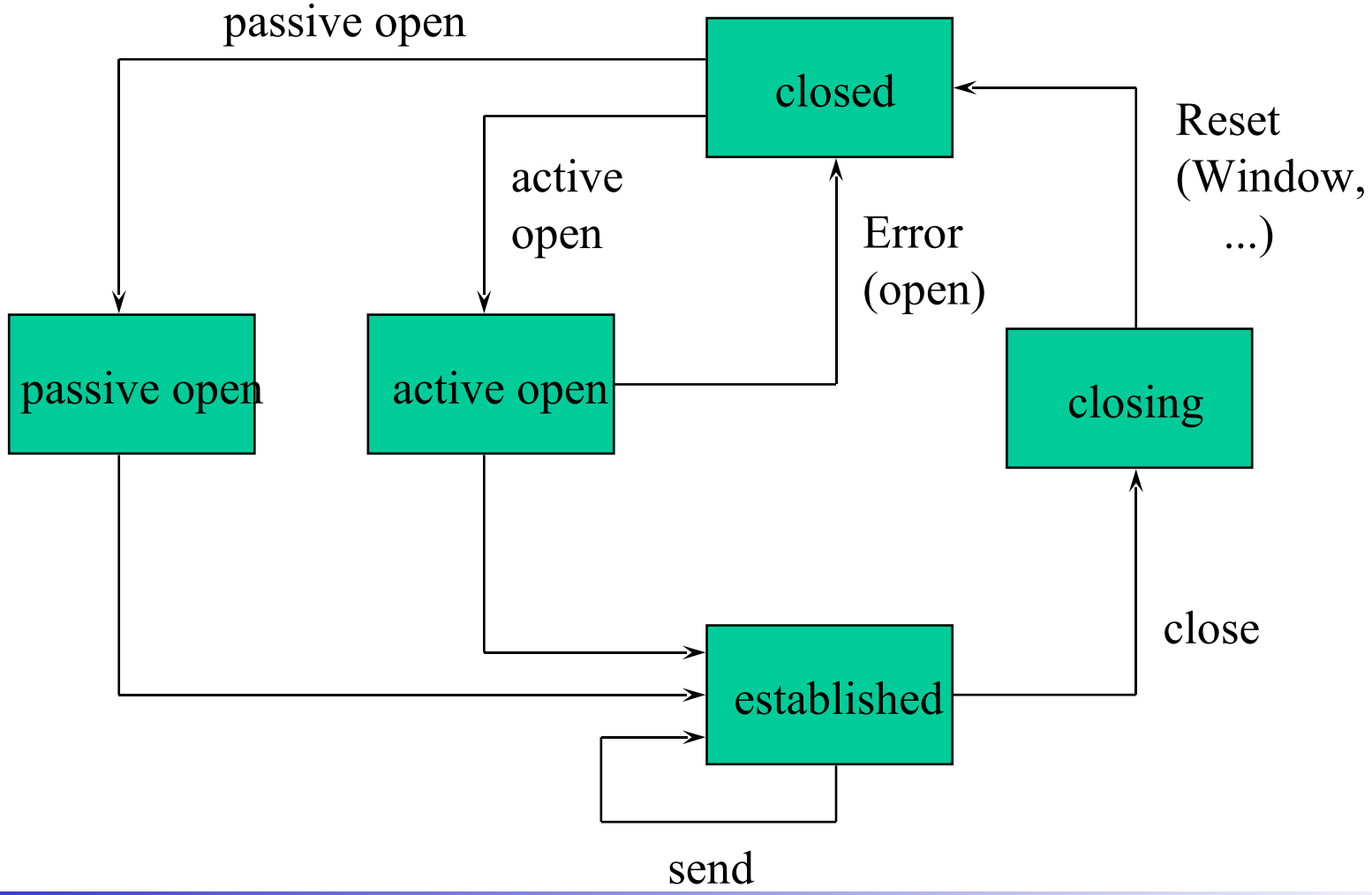
(1) Betriebssystem A weist dem Verbindungsendpunkt eine Portnummer zu.

(2) Partner B informiert sein Betriebssystem, daß er ein Active Open durchführen will.

(3) Beide TCP-Module kommunizieren miteinander und verifizieren, daß die Verbindung aufgebaut ist.

(4) Sobald die Verbindung besteht, beginnen die TCP-Module mit der Datenübertragung.

TCP-Zustände (vereinfacht)



TCP-Zustandsgraph