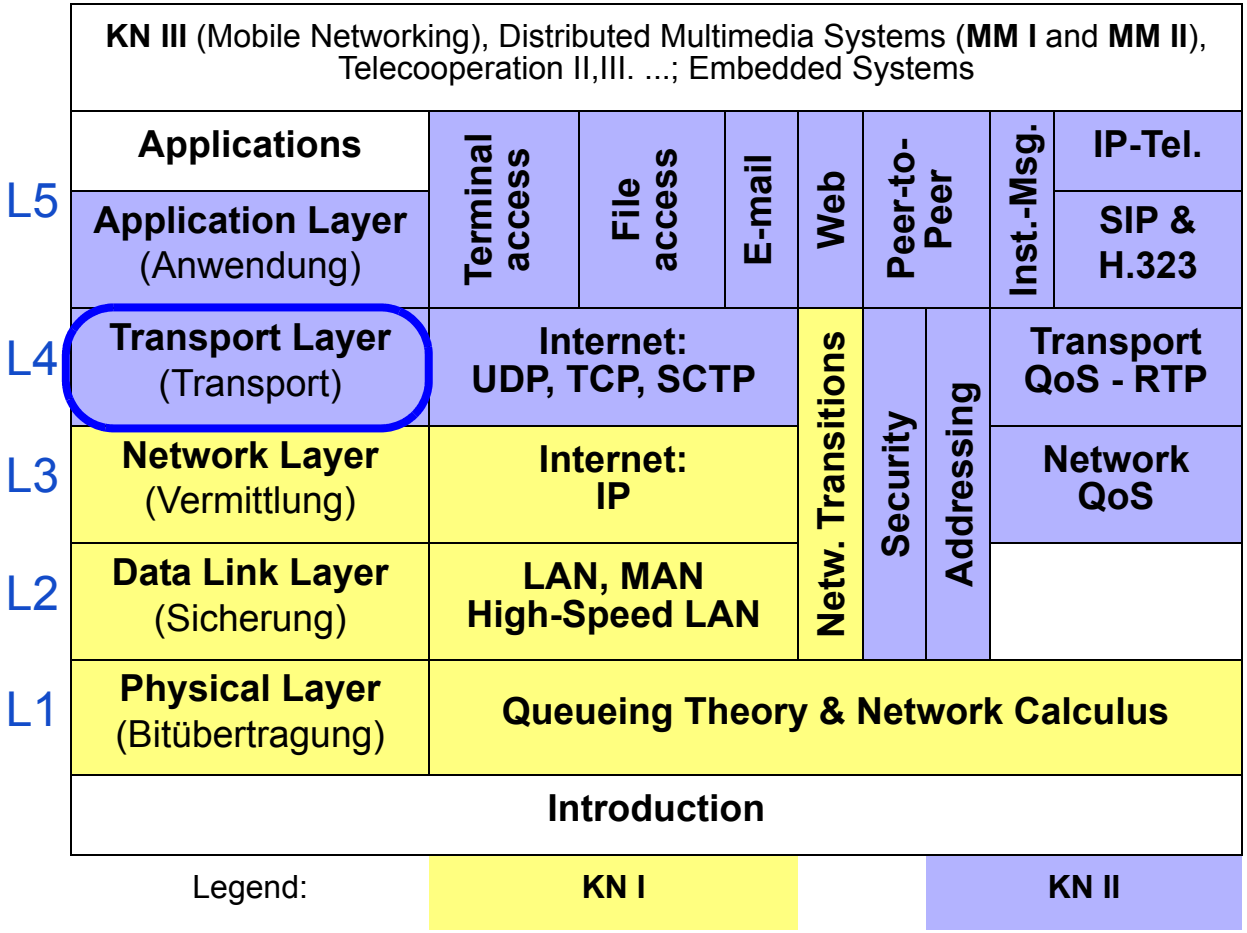
Communication Networks II
Transport Layer

Prof. Dr.-Ing. **Ralf Steinmetz**

TU Darmstadt - Technische Universität Darmstadt,
Dept. of Electrical Engineering and Information Technology, Dept. of Computer Science
KOM - Multimedia Communications Lab
Merckstr. 25, D-64283 Darmstadt, Germany, Ralf.Steinmetz@KOM.tu-darmstadt.de
Tel.+49 6151 166151, Fax. +49 6151 166152

httc - Hessian Telemedia Technology Competence-Center e.V
Merckstr. 25, D-64283 Darmstadt, Ralf.Steinmetz@httc.de

Scope


KN III (Mobile Networking), Distributed Multimedia Systems (MM I and MM II), Telecooperation II,III. ...; Embedded Systems
Applications
Terminal access
File access
E-mail
Web
Peer-to-Peer
Inst.-Msg.
IP-Tel.
L5
Application Layer (Anwendung)
SIP & H.323
L4
Transport Layer (Transport)
Internet: UDP, TCP, SCTP
Netw. Transitions
Security
Addressing
Transport QoS - RTP
L3
Network Layer (Vermittlung)
Internet: IP
Network QoS
L2
Data Link Layer (Sicherung)
LAN, MAN High-Speed LAN
L1
Physical Layer (Bitübertragung)
Queueing Theory & Network Calculus
Introduction
Legend:
KN I
KN II

Overview

1. Transport Layer Function
2. Quality of Service (QoS) at Layer 4
3. Addressing (at Transport Layer)
4. Duplicates (at Data Transfer Phase)
5. Reliable Connection Establishment
6. Disconnect
7. Flow Control on Transport Layer
8. Memory Management / Cache Administration
9. Multiplexing / Demultiplexing
10. Some Familiar Internet Protocols

1. Transport Layer Function

To provide data transport

- **reliably**
- **efficiently**
- **at low-cost**

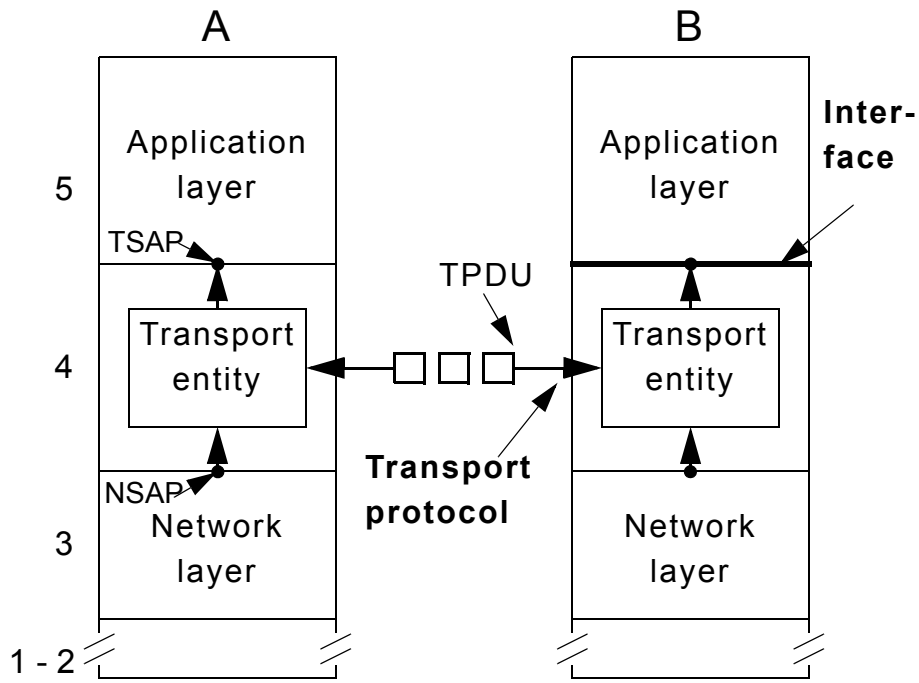
for

- **process-to-process (applications)**
- **i.e. at endsystem-to-endsystem**

(if possible) independent from

- **particularities of the networks used**

1.1 Transport Service



Connection oriented service

- 3 phases: connection set-up, data transfer, disconnect

Connectionless service

- transfer of isolated units

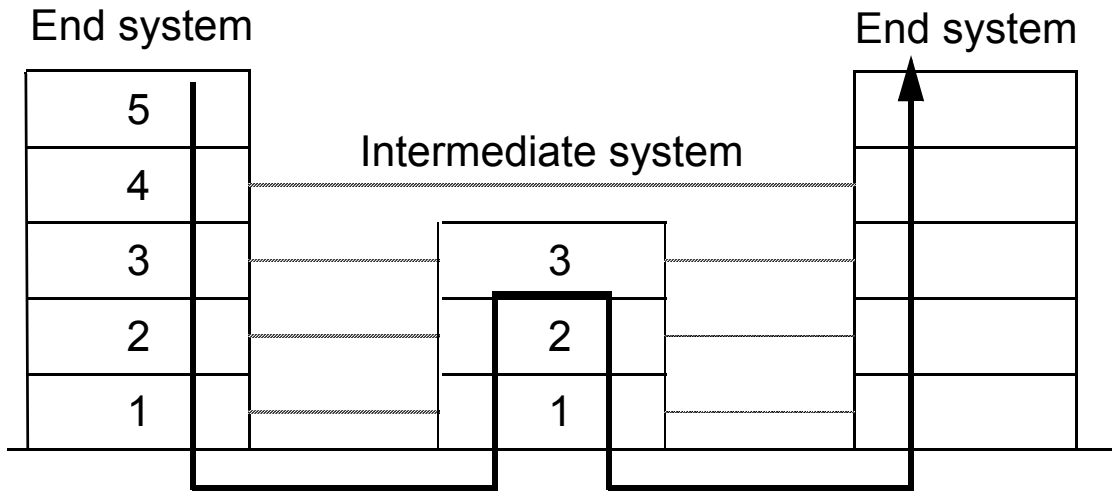
Realization: transport entity

- software and/or hardware
- software part usually contained within the kernel (process, library)

Transport Service (2)

Similar services of

- **network layer and transport layer:**
  WHY 2 LAYERS?



Network service

- **not to be self-governed or influenced by the user**
- **independent from application & user**
  - enables compatibility between applications
- **provides for example**
  - "only" connection oriented communications
  - or "only" unreliable data transfer

Transport service: TO IMPROVE THE NETWORK SERVICE QUALITY

- **users and layers want to get from the network layer, e.g.**
  - reliable service
  - necessary time guarantees

Transport Service (3)

Transport layer:

- **isolates upper layers from technology, design and imperfections of subnet**

Traditionally distinction made between

- **layers 1 - 4**
  - transport service provider
- **layers above 4**
  - transport service user

Transport layer has key role:

- **major boundary between**
  - provider and
  - user of reliable data transmission service

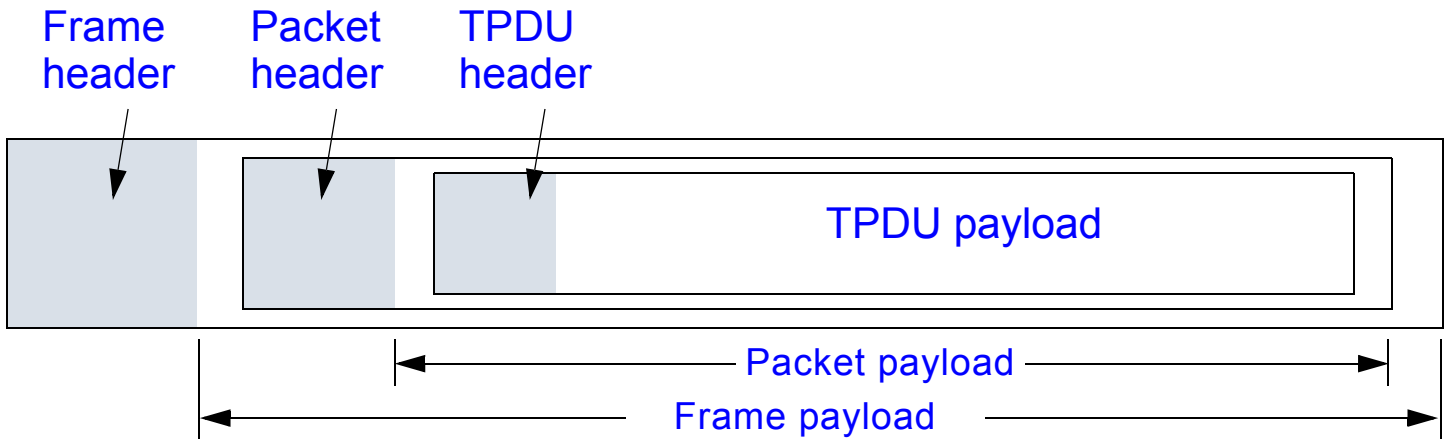
Transport Service: Terminology

Entities exchanged:

Layer	Data Unit
Transport	TPDU / Message (TPDU: Transport Protocol Data Unit)
Network	Packet
Data Link	Frame
Physical	Bit/Byte (bitstream)

TPDU: Transport Protocol Data Unit

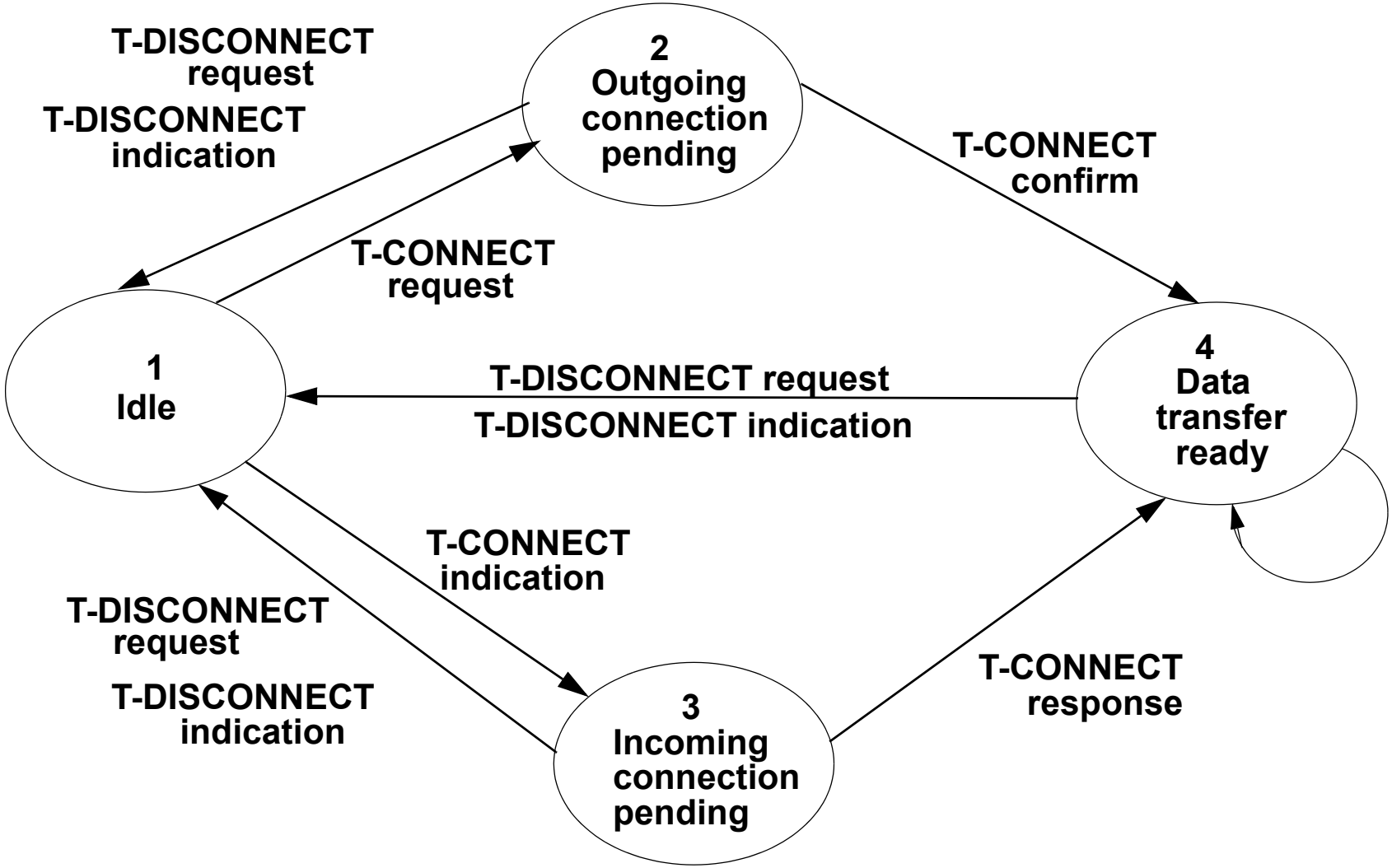
Nesting of TPDUs, packets, and frames:



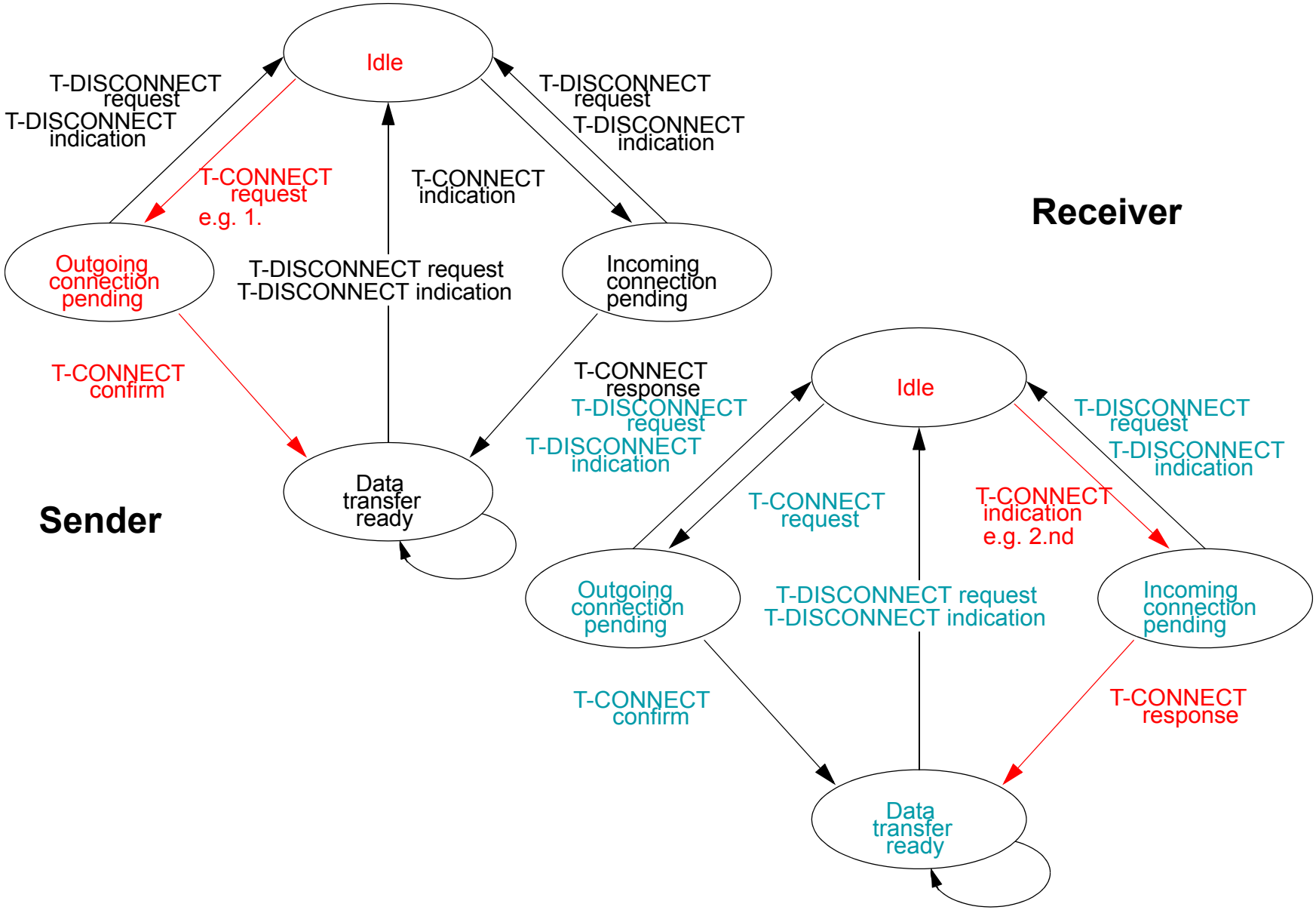
1.2 Connection Oriented Service: State Transition Diagr.

Example: ISO-OSI nomenclature

- **state transition diagram**



Connection Oriented Service: State Transition Diagr. (2)



Example: Parameters for Disconnect

Reason for any "T-Disconnect"

Reason	Notes
Normal disconnect initiated by session entity	1, 4
Remote congestion at transport entity during CC	1, 4
Failed connection negotiation	1, 3
Duplicated source reference for same NSAP pairs	1, 4
References are mismatched	1, 4
Protocol error	1, 4
Reference overflow	1, 4
Connection request refused	1, 4
Header or parameter length invalid	1, 4
No reason specified	2, 4
Congestion at TSAP	2, 4
TSAP and session entity not attached	2, 3
Unknown address	2, 3
(Note 1) Used for classes 1 to 4	
(Note 2) Used for all classes	
(Note 3) Reported to TS-user as persistent	
(Note 4) Reported to TS-user as transient	

1.3 Transport Service Primitives: More Practical

Primitives for a simple transport service:

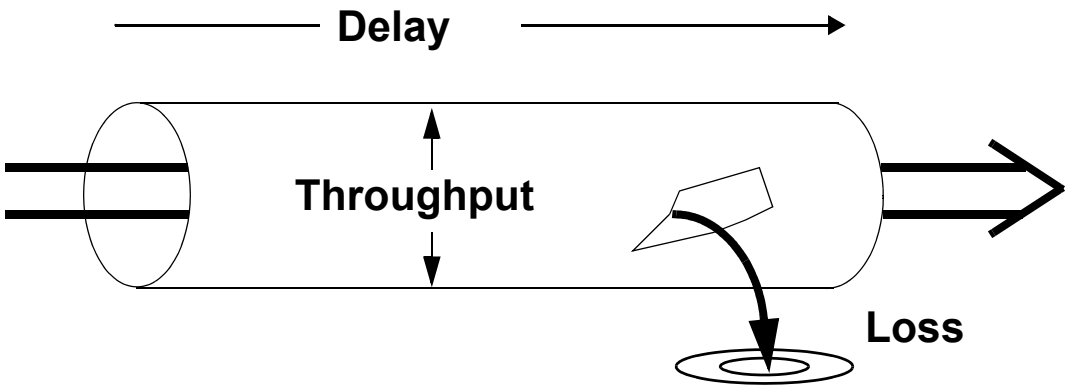
Primitive	Packet sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ	Actively attempt to establish a connection
SEND	DATA	Send Information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ	Request to release the connection

2. Quality of Service (QoS) at Layer 4

Quality of Service

Characterizes the well defined, controllable behavior of a system with regard to quantitatively measurable parameters

Fundamental parameters - e.g. network



- **delay:** **“end-to-end“ speed**
- **throughput:** **rate**
- **error:** **probability, error handling**

QoS - Some Parameters in More Detail

QoS defined at and/or for

- **set-up phase**
- **data transfer phase**
- **disconnect phase**

Parameters

- **error probability at connection set-up phase**
- **throughput**
- **transfer delay**
- **remaining error rate ("Restfehlerrate")**
  - error probability at data transfer
- **duration of time to disconnect**
  - i.e. at disconnect phase
- **failure probability of disconnect**
  - "Ausfallwahrscheinlichkeit beim Verbindungsabbau"
- **security**
  - with regard to “listening in” ("Mithören")
- **priority**
- **resilience ("Störausgleichverhalten")**
  - against errors within the transport layer itself

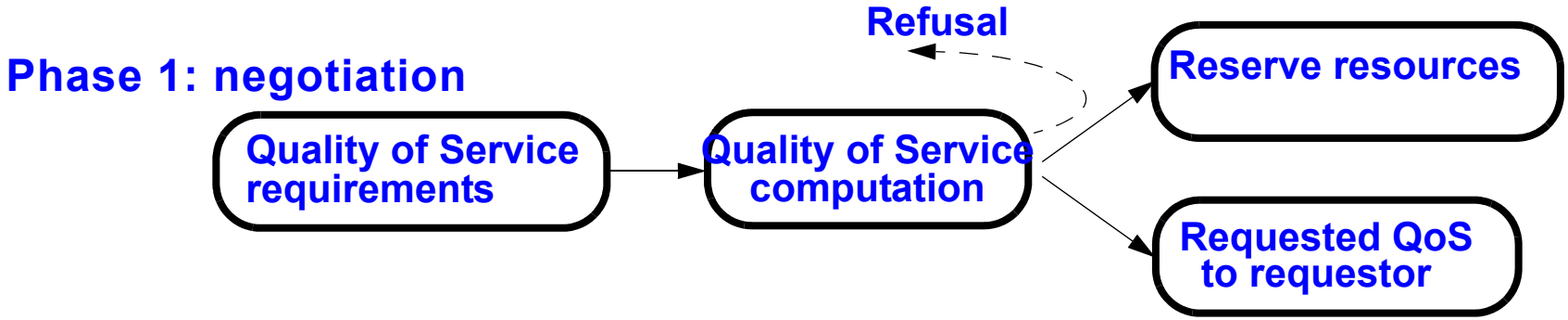
QoS at Various Communication Architectures

Variation: connectionless service - at the Internet: DIFFSERV ARCHITECTURE

- **QoS specifications in every packet**
- **actual guarantee**
  - only possible on average

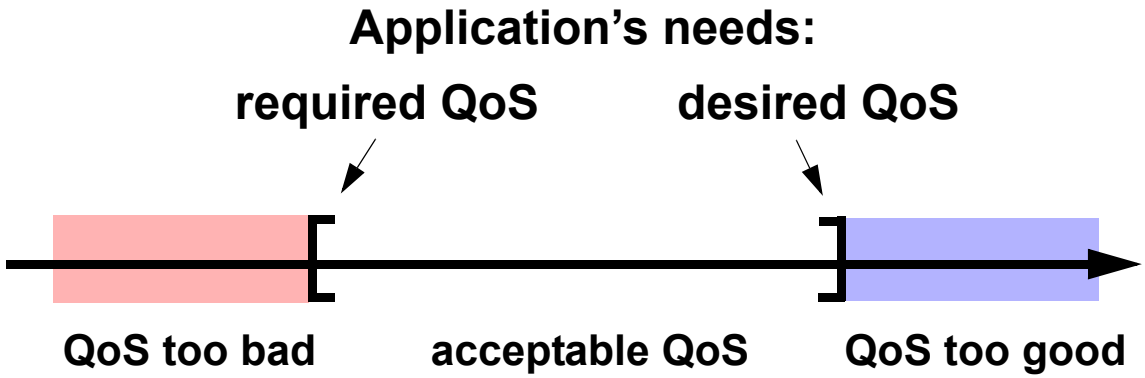
Variation: connection oriented service - at Internet: INTSERV ARCHITECTURE

- **more typical for QoS**
- **phases of resource reservation and data transfer**



QoS Negotiation: Required - Desired

QoS negotiations take place during set-up (negotiation of options)



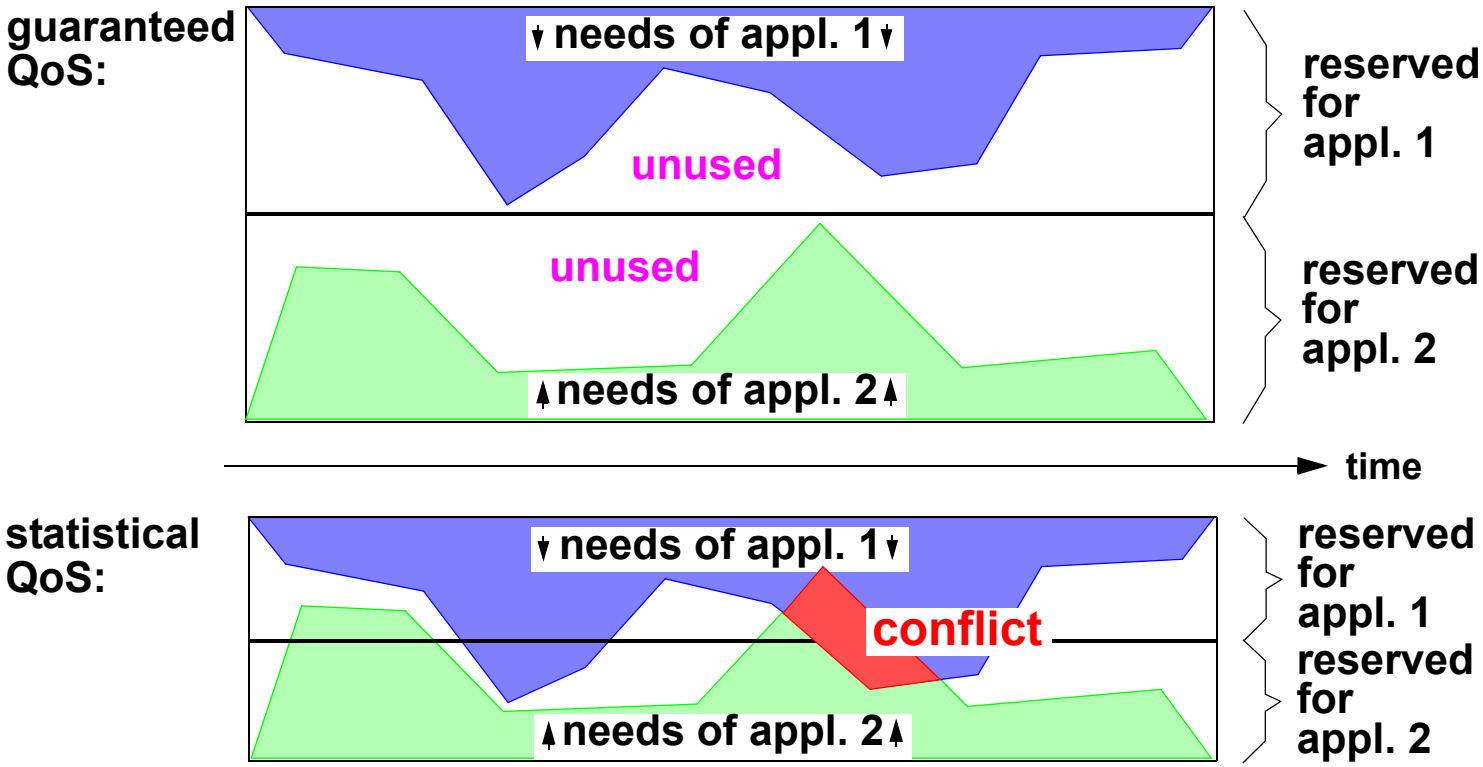
Specification of the application's QoS

- **interval between *desired* and *required* QoS**
  - system is to deliver QoS guarantee within this interval
- ***required QoS* = the application's minimum quality of service requirement**
  - lower QoS value: service does not make sense
- ***desired QoS* = the application's maximum quality of service requirement**
  - higher QoS value: unnecessary reservation / costs

Comment

- **negotiation often means "only" stating the parameters (without any actual negotiations)**

QoS Negotiation: QoS Classes



QoS classes

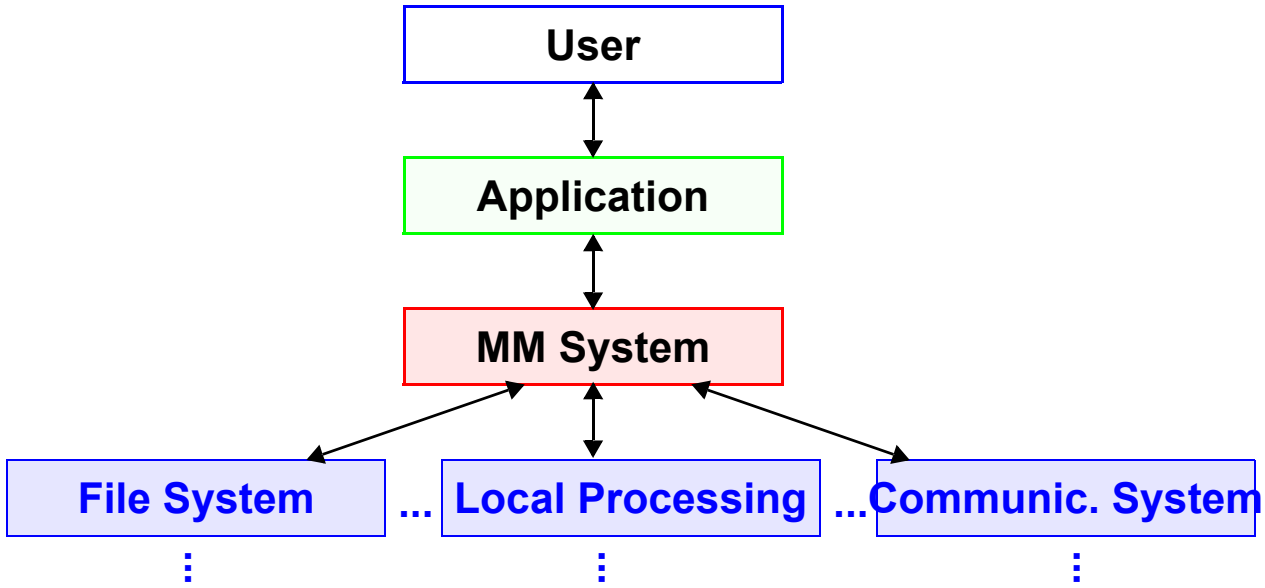
- guaranteed QoS
- statistical QoS

QoS class specifies

- QoS reliability
- efficient utilization of resources

QoS - Layers

Levels with varying degrees of detail



Examples:

- **user:** **show movie in MPEG-2 quality**
- **MM system:** **throughput (compressed video images)**
- **network:** **throughput (of the protocol data units)**

Here we concentrate on

- **transport layer**
  - as part of the communication system

Example: QoS of Several Data Flows

Synchronizing the QoS of one data flow vs. several data flows

- **lateral off-set influences acceptance**
- **dependency of media and applications**

Media		Type, Application	QoS
video	animation	correlates	+/- 120 ms
	audio	lip synchronization	+/- 80 ms
	image	as an overlay	+/- 240 ms
		not as an overlay	+/-500 ms
	text	as an overlay	+/- 240 ms
		not as an overlay	+/-500 ms
audio	animation	coherence of events	+/- 80 ms
	audio	stereo	+/- 10 µs
		closely linked (conference)	+/- 120 ms
		loosely linked (background music)	+/- 500 ms
	image	closely linked (music and score)	+/- 5 ms
		loosely linked (slide presentation)	+/- 500 ms
	text	comments about the text	+/- 240 ms
	pointer	audio refers to pointed object	- 500 +750 ms

Example: QoS Parameter in the Internet Protocol IPv4

Type of service

- **contained as 8 bit code in the header of each IP packet**
- **simple QoS**
- **combination of reliability and delay**

Statements in the field: precedence (3 bit):

- **priority 0 (normal) ... 7 (network control)**
- **influences queue (not routing)**
  - D (1 bit): Delay, e.g. no satellite transmission
  - T (1 bit): Throughput, e.g. no phone line
  - R (1 bit): Reliability, e.g. no radio channels
  - C (1 bit): low Cost, e.g. not well specified ...
- **1 bit unused**
- **comment: if C & D are set - actually invalid**

Comments (NO RESERVATION and NO GUARANTEE for QoS)

- **in practice: ignored by routers**
- **only few combinations of QoS values (QoS Tuple) possible**
  - as IP is connectionless
- **no QoS negotiation and optimization possible**
  - only yes/no decision

3. Addressing (at Transport Layer)

Why identification?

- **sender (process) wants to address receiver (process)**
  - for connection setup or individual message
- **receiver (process) can be approached by the sender (process)**

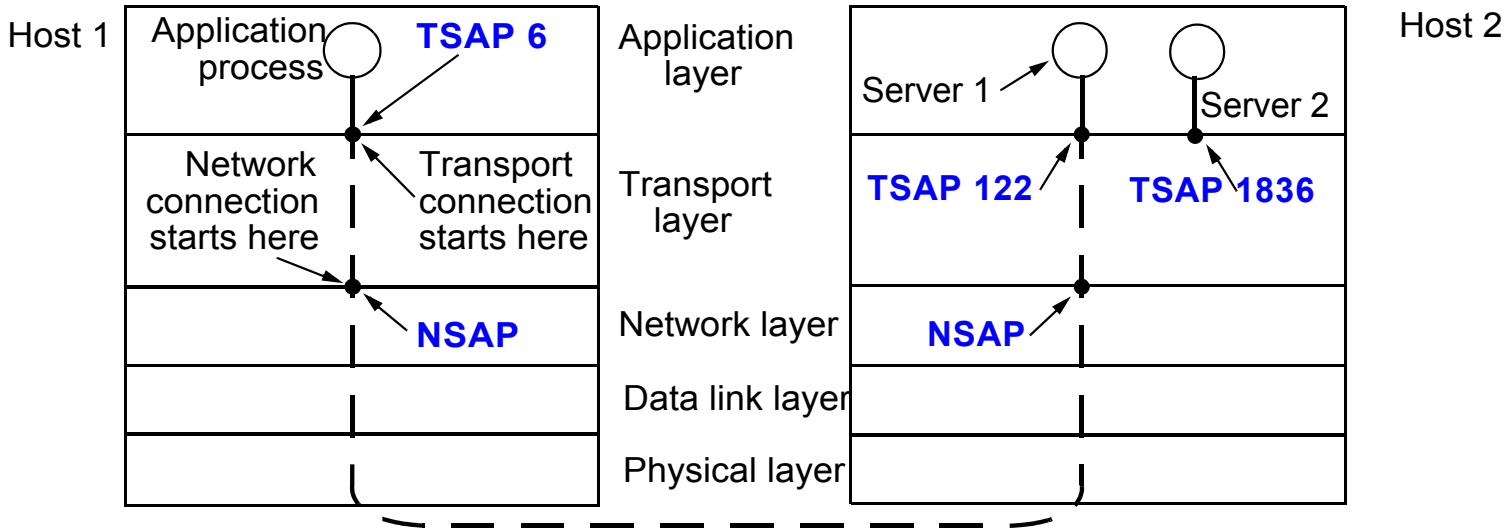
Define transport addresses:

- **generic term: (Transport) Service Access Point *TSAP***
- **Internet: port**
- **or e.g. ATM: AAL-SAP**

Reminder: analogous end points in network layer: NSAP

- **e.g., IP addresses**

Model

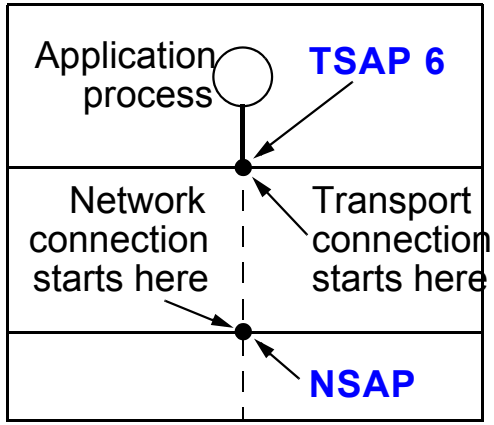


Steps

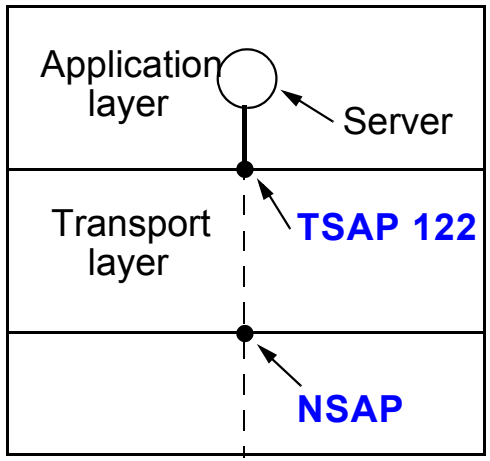
In general

1. **Server (service provider)**
   - connects itself to TSAP 122
   - waits for service request (polling, signalling, ..)
2. **Client (application)**
   - initiates connection via TSAP 6 as source and TSAP 122 as destination
     - i.e. connect.req
3. **Transport system on host 1**
   - identifies dedicated NSAP
   - initiates communication at network layer
   - communicates with transport entity on host2
   - informs TSAP 122 about desired connection
4. **Transport entity on host 2**
   - addresses the server
   - requests acceptance for the desired connection
     - i.e. connect.ind
5. **etc.**

Host 1: Sender - Client

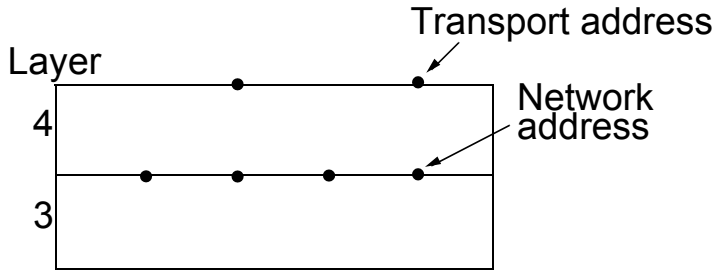


Host 2: Receiver - Server



3.1 Determination of Appropriate Service Provider TSAP

How does the specific address of a service becomes known?



1. **Approach: TSAP known implicitly**
   - services that are well known and often used have pre-defined TSAPs
     - as "well known ports" of a transport protocol
   - e.g., stored in /etc/services file at UNIX systems

example: service 'time of day'

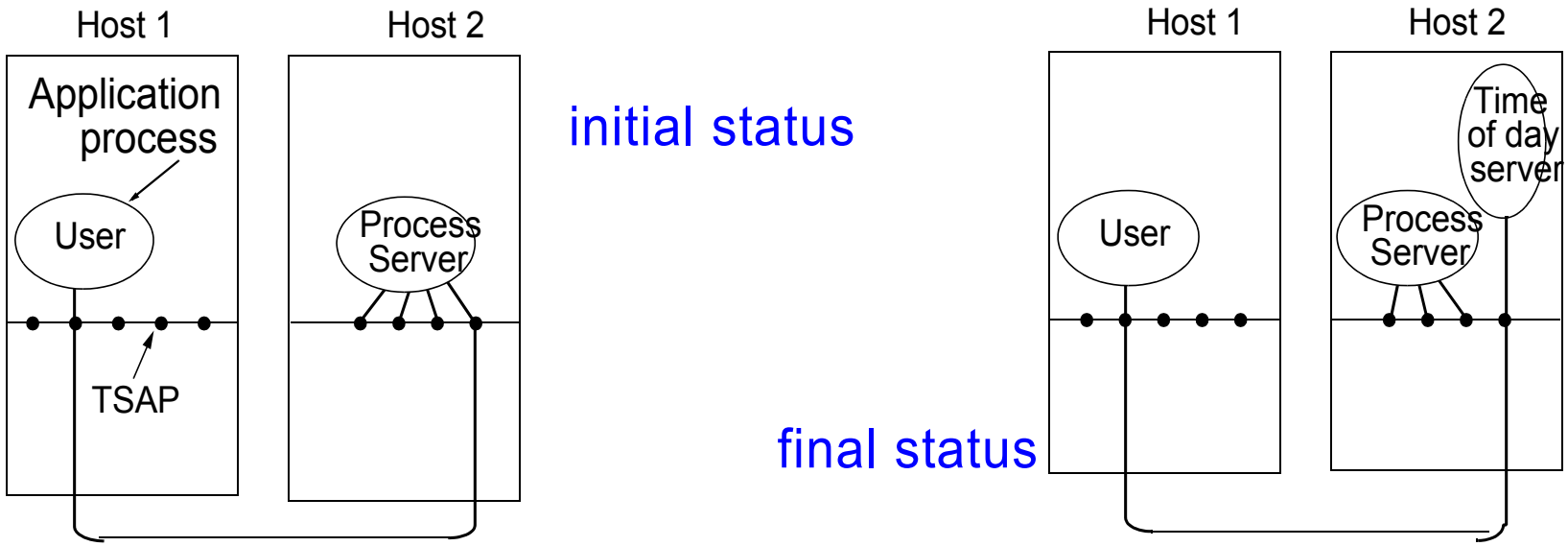
Characteristics:

- **works well for small number of stable services**
- **not suitable for user specific processes**
  - existing for short time, no known TSAP addr
- **waste of resources; seldom used servers active and listening**

Determination of Appropriate Service Provider TSAP (2)

2. Approach: "initial connection protocol"

1. *process server* acting as proxy for less often used servers
2. process server listens to set of ports at same time, waiting for connection requests
3. creates the appropriate service provider process
4. transfers connection and desired service
5. waits for further requests



Characteristics:

- **works well for servers which can be created on demand**
- **not suitable if service exists independently of process server at another machine (e.g., file server)**

Determination of Appropriate Service Provider TSAP (3)

3. Approach: Name Server (directory server)

- **context**
  - server process already exists
- **procedure**
  1. client addresses **NAME SERVER** (establishing connection)
  2. client specifies the service as an ASCII data set
     - example “name of day”
  3. name server supplies TSAP
  4. client disconnects from name server
  5. client addresses TSAP provided by name server

     .....
- **comments**
  - new services
    - have to register at the name server
  - name server
    - adds corresponding information at the database

3.2 Determination of Appropriate NSAP

How to localize the respective endsystem NSAP?

- **TSAP known**
- **i.e. how to determine the appropriate NSAP?**

1. approach: hierarchic addressing

- **TSAP contains this information**

```
example: <country>.<network>.<port>
```

2. approach: “flat” addressing

- **dedicated “name server”**
  - entry

```
TSAP address: address of the endsystem + port
```

- **request via broadcast**
  - e.g. as correlation of ethernet address and internet address
  - i.e. possible in geographically and topologically limited spaces

4. Duplicates (at Data Transfer Phase)

Initial Situation: Problem

- **network has**
  - varying transit times for packets
  - certain loss rate
  - storage capabilities
- **packets can be**
  - manipulated
  - duplicated
  - resent by the original system after timeout

In the following, uniform term: "DUPLICATE"

- **a duplicate originates due to one of the above mentioned reasons and**
- **is at a later (undesired) point in time passed to the receiver**

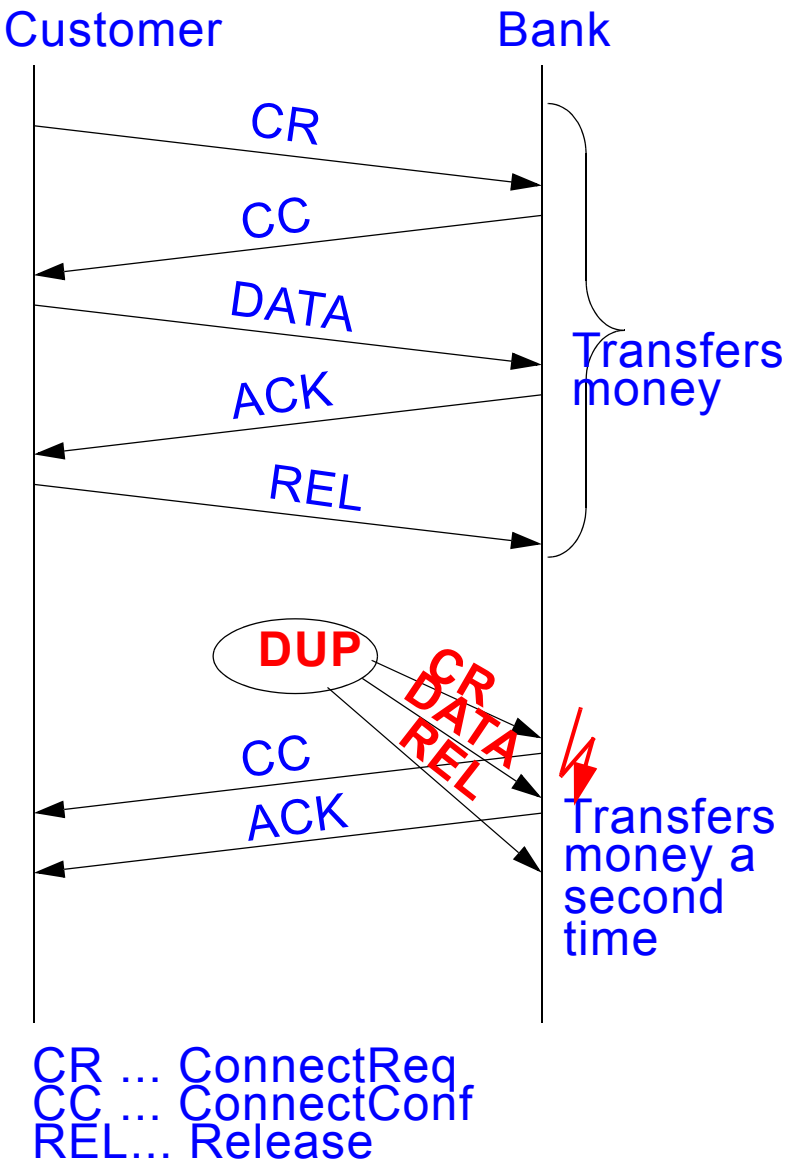
Example: Duplicates

E.g. description of possible error causes and their possible consequences (5 transactions)

- **due to network capabilities**
  - duplication of sender's packets
  - subsequent to the first 5 packets duplicates are transferred in correct order to the receiver
  - also conceivable is that an old delayed DATA packet (with faulty contents) from a previous session may appear;
    this packet might be processed instead of or even in addition to the correct packet

Result:

- without additional means the receiver cannot differentiate between correct data and duplicated data
- would re-execute the transaction



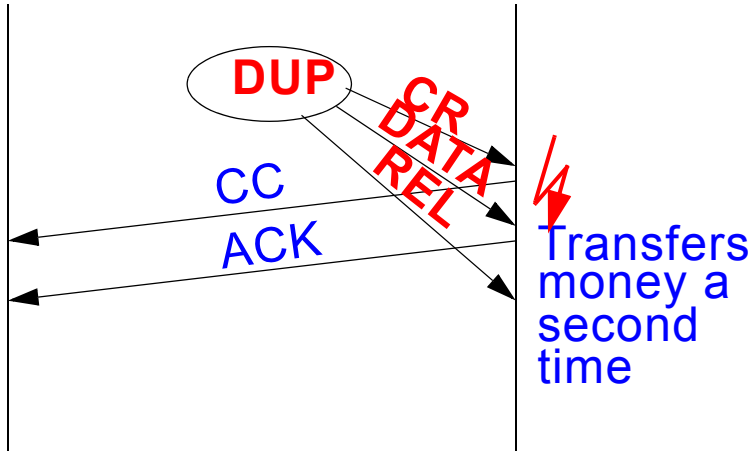
Duplicates - Description of Problematic Issues

3 somehow disjoint problems

1. **how to handle duplicates WITHIN A CONNECTION?**

2. **what characteristics have to be taken into account regarding**
   - **CONSECUTIVE CONNECTIONS** or
   - **CONNECTIONS** which are being re-established after a crash?

3. **what can be done to ensure that a connection that has been established:**
   - has actually been initiated by and **WITH THE KNOWLEDGE OF BOTH COMMUNICATING PARTIES?**
   - see also the lower part of the previous illustration



4.1 Duplicates - Methods of Resolution

1. to use temporary valid TSAPs

- **method:**
  - TSAP valid for one connection only
  - to generate always new TSAP
- **evaluation**
  - in general not always applicable:
    process server addressing method not possible,
    because
    - server is reached via a designated/known TSAP
    - some TSAPs always exist as “well known”

2. to identify connections individually

- **method**
  - each individual connection is assigned a new SeqNo and
  - endsystems remember already assigned SeqNo
- **evaluation**
  - endsystems must be capable of storing this information
  - prerequisite:
    - connection oriented system (what if connection-less?)
  - endsystems, however, will be switched off and
    it is necessary that the information is reliably available whenever needed

Duplicates - Methods of Resolution (2)

3. to identify PDUs individually:
individual sequential numbers for each PDU

- **method**
  - SeqNo basically never gets reset
  - e.g. 48 bit at 1000 msg/sec: reiteration after 8000 years

- **evaluation**
  - higher usage of bandwidth and memory
  - sensible choice of the sequential number range depends on
    - the packet rate
    - a packet's probable "lifetime" within the network

⇒ **discussed in more detail in the following**

4.2 Duplicates: Approach to Limit Packet Lifetime

Enabling the above method

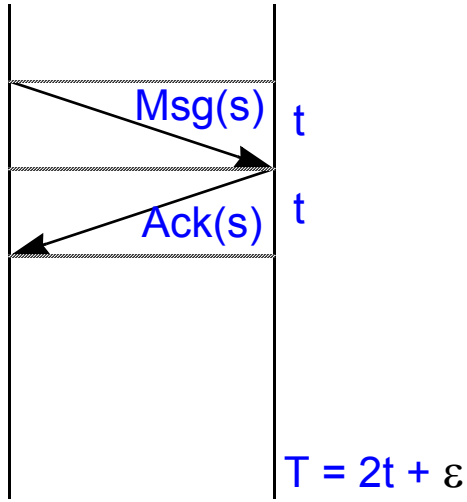
> **'3. to identify PDUs individually: individual sequential numbers for each PDU'**

- **SeqNo only reissued if**
  - all PDUs with this SeqNo or references to this SeqNo are extinct
- **i.e., ACK (N-ACK) have to be included**
  - otherwise new PDU may be
    - wrongfully confirmed or non-confirmed by delayed ACK (N-ACK).

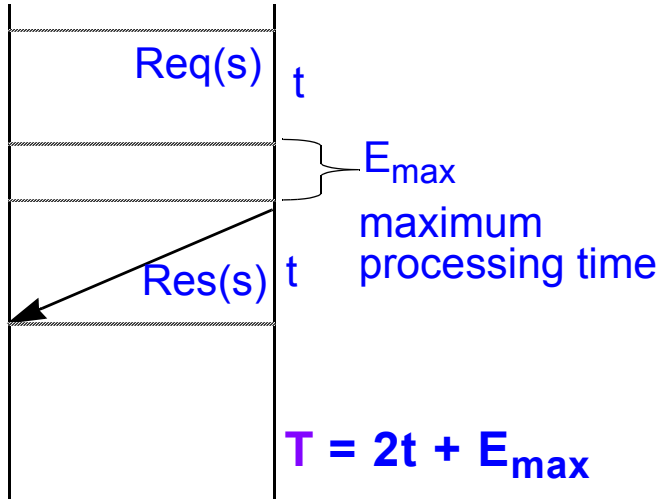
Mandatory PREREQUISITE for this solution

- **limited packet lifetime**
- **i.e. introduction of a respective parameter T**

example 1 (in principle)



example 2:Request/Response
taking processing time into account)



Duplicates: Approach to Limit Packet Lifetime (2)

Methods:

1. **Limitation by appropriate network design**
   - inhibit loops
   - limitation of delays in subsystems & adjacent systems

2. **Hop-counter / time-to-live in each packet**
   - counts traversed systems
   - if counter exceeds maximum value
     - => packet is discarded

3. **Time marker in each packet**
   - packet exceeds maximum predefined / configured lifetime
     - => packet is discarded
   - ***NOTICE:*** requires “consistent” network time

Duplicates: Approach to Limit Packet Lifetime (3)

Determining maximum time T, which a packet may remain in the network

- T is a small multiple of the (real maximal) packet lifetime t
- T time units after sending a packet
  - the packet itself is no longer valid
  - all of its (N)ACKs are no longer valid

TCP/IP term: Maximum Segment Lifetime (MSL)

- to be imposed by IP layer
- defined by and referenced by other protocol specifications
  - 2 minutes

4.3 Initial Sequence Number Allocation and Handling of Consecutive Connections

Problem (wrt. "3. to identify PDUs individually: individual sequential numbers for each PDU")

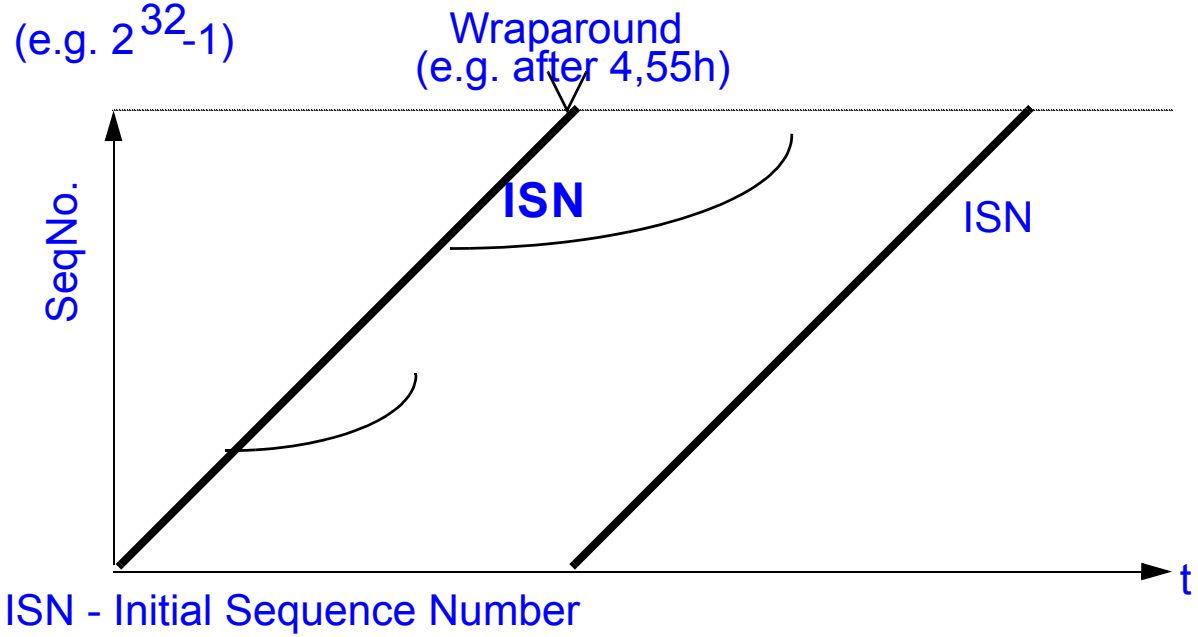
- **to be considered**
  packets from connections which can otherwise not be distinguished
  - hence at TCP that is:
    - same source and destination address and
      same source and destination port
  - this is always unique at one point in time
- **method: to use consecutive sequential numbers**
  from sufficiently large sequential number range

  ⇒ **RESOLVES PROBLEMS WITH DUPLICATES WITHIN A SINGLE CONNECTION**
  - duplicates are all other packets with the same sequential number
    - irrelevant is origin of packets, sequence of creation

Problems:

- restart after crash
- (very fast) reconnect between exactly the same communication entities
  - (addr./port see above), information about previous connections do not exist anymore after crash/restart, generally all conn. have to be reconsidered
- complete usage of the range of available sequential numbers

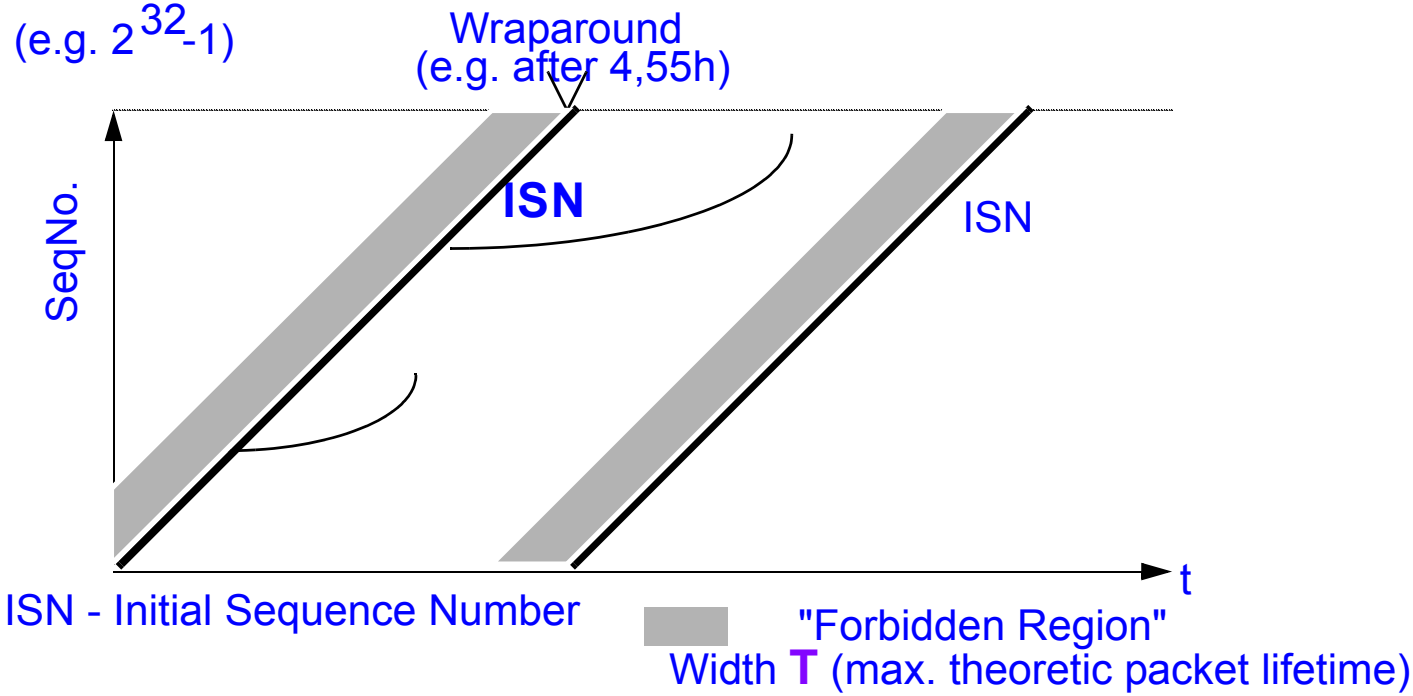
Initial Sequence Number Allocation and Handling of Consecutive Connections (2)



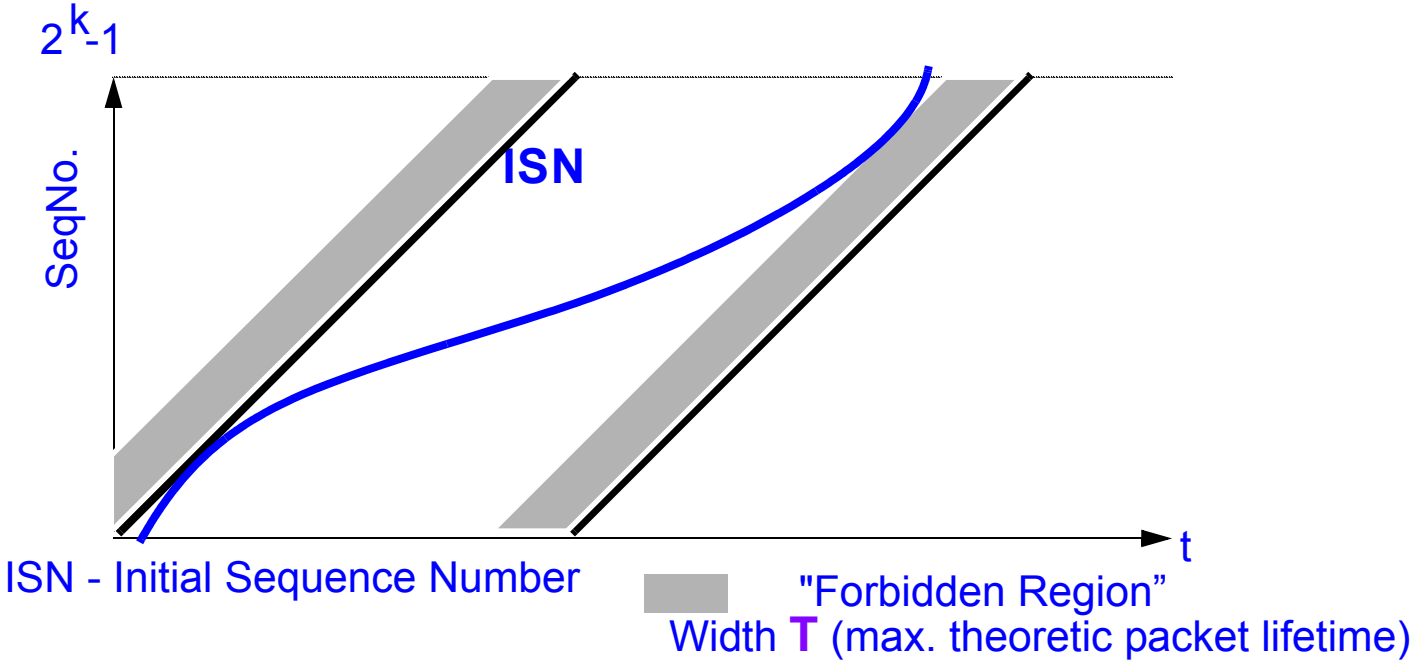
Method

- **endsystems**
  - timer continues to run at switch-off / system crash
- **allocation of initial SeqNo (ISN) depends on**
  - time markers (linear or stepwise curve because of discrete time)
- **SeqNos can be allocated consecutively within a connection**
  - curve consisting out of discrete points may have any gradient form depending on the method used for sending the data

Initial Sequence Number Allocation and Handling of Consecutive Connections (3)



Initial Sequence Number Allocation and Handling of Consecutive Connections (4)



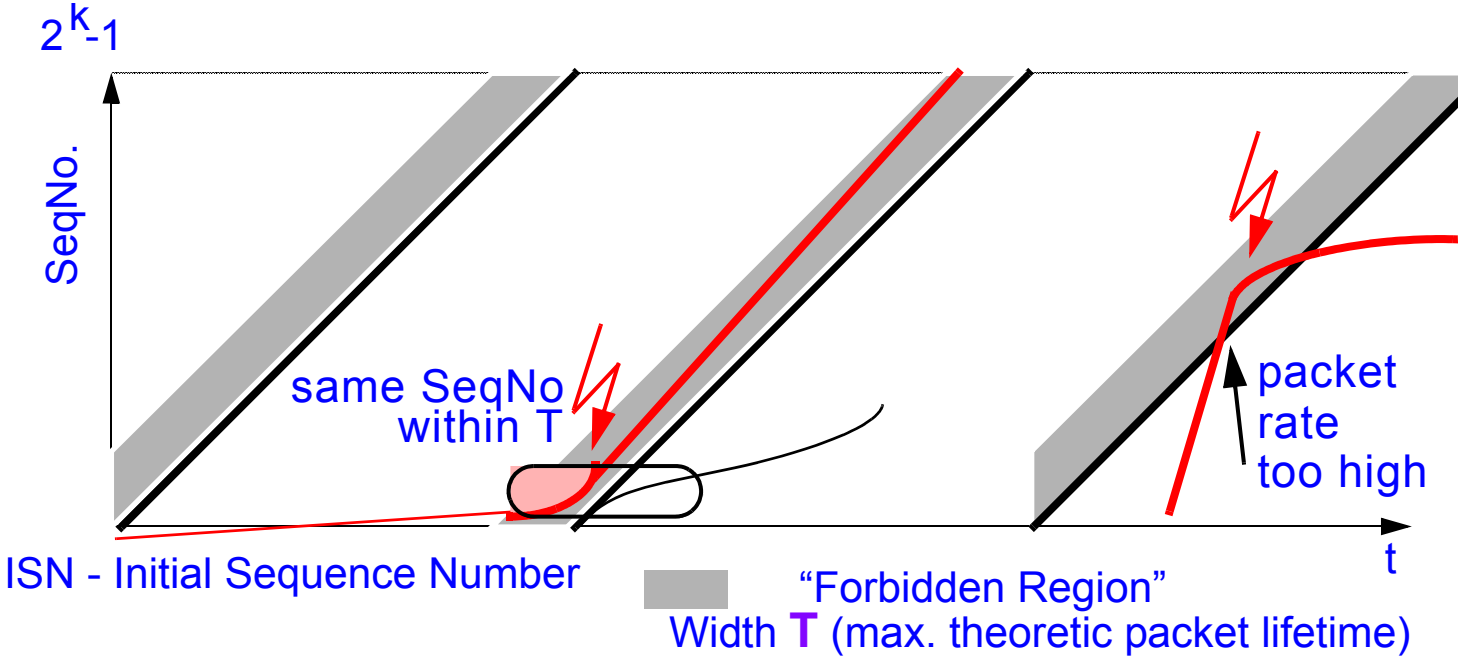
No problem, if

- **"normal lived" session (shorter than wrap-around time) with data rate smaller than ISN rate (ascending curve less steep)**

Then, after crash

- **reliable continuation of work always ensured**

Initial Sequence Number Allocation and Handling of Consecutive Connections (5)



Problems possible, if

1. **SeqNo is used within time period T before it is being used as initial SeqNo**
   - =>"Forbidden Region" - begins T ***BEFORE*** Initial SeqNo (ISN) is generated
   - i.e. endsystem has to check if the PDU is in the forbidden region before it is sent (during the actual data phase)´
2. **"long lived" session (longer than wrap-around time)**
3. **high data rate**
   - curve of the consecutively allocated sequence numbers steeper than ISN curve

Initial Sequence Number Allocation and Handling of Consecutive Connections (6)

Note:

- **32 bit sequence numbers with technology considered as sufficient when designing TCP/IP**
- **sequence number range exploitation**
  - 10 Mbit/sec in ca. 57 min
  - 1 Gbit/sec in ca. 17 sec

⇒ **Using timestamps in**
- "TCP extensions for highspeed paths"
  - PAWS "Protect Against Wrapped Sequence Numbers"

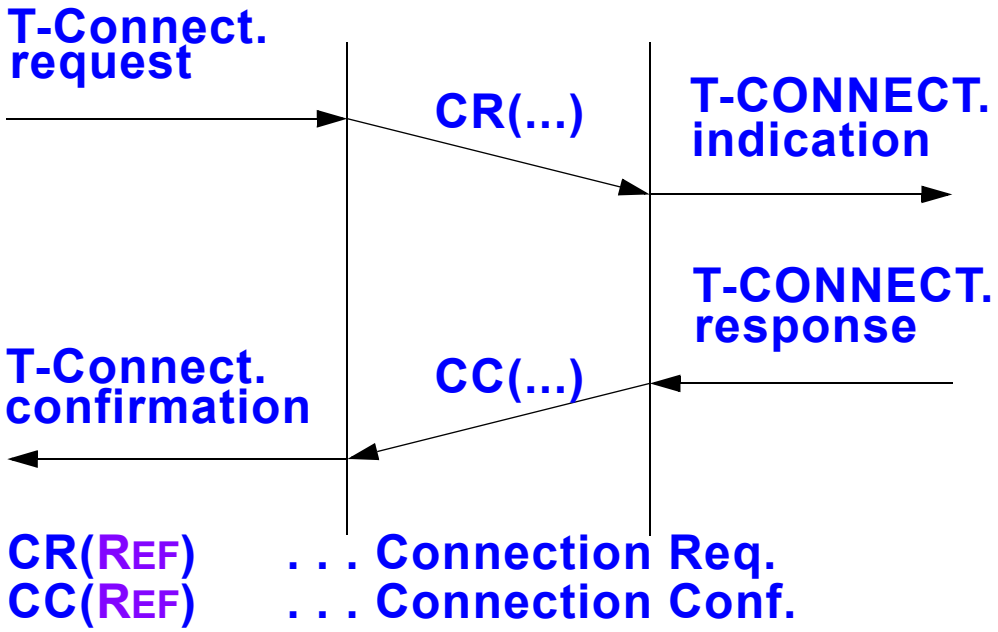
Further literature in addition to Tanenbaum

- **RFC 793 (TCP) / Sequence Numbers; "When to keep quiet"**
- **RFC 1185 / Appendix - Protection against Old Duplicates**
- **RFC 1323 / PAWS**
  - Protect Against Wrapped Sequence Numbers
  - Appendix B - Duplicates from Earlier Connection Incarnations

5. Reliable Connection Establishment

Connection (see also Connection Oriented Service: State Transition Diagr.)

- **by simple protocol**



- approach using 2 messages (2 phases)
  - problems may occur due to delayed duplicates
  - compare with previous example (bank transaction)

Connect: Three-way Handshake Protocol

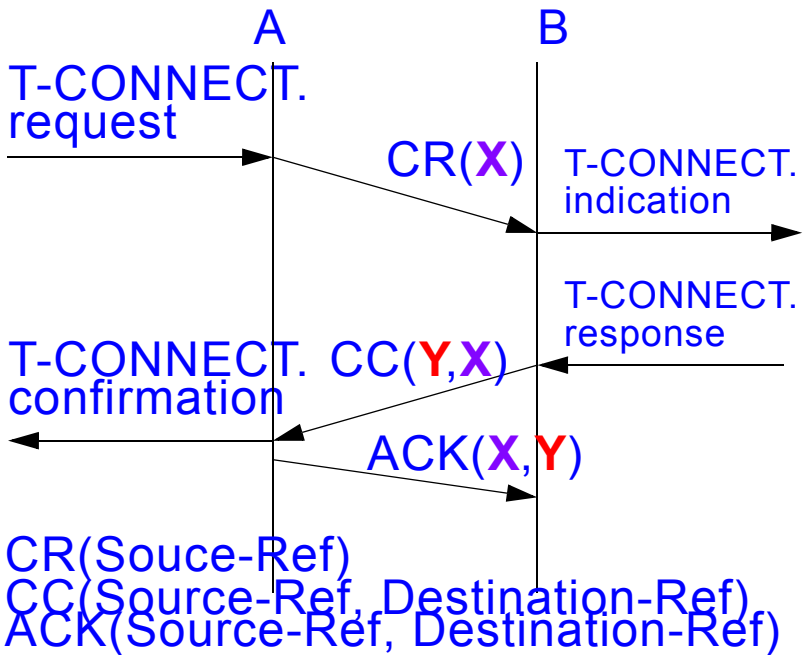
Principle

1. CR: Connect Request

- **initiator (A) sends request with**
  - **SEQUENCENO (X)** selected by sender

2. CC: Connect Confirmation

- **receiver (B) responds with**
  - sequence number transmitted by the initiator (**X**) and
  - (randomly) selected sequence number (**Y**) by receiver
    - while observing the previously discussed criteria for selection, in order to avoid a collision with delayed duplicates



3. Acknowledgment

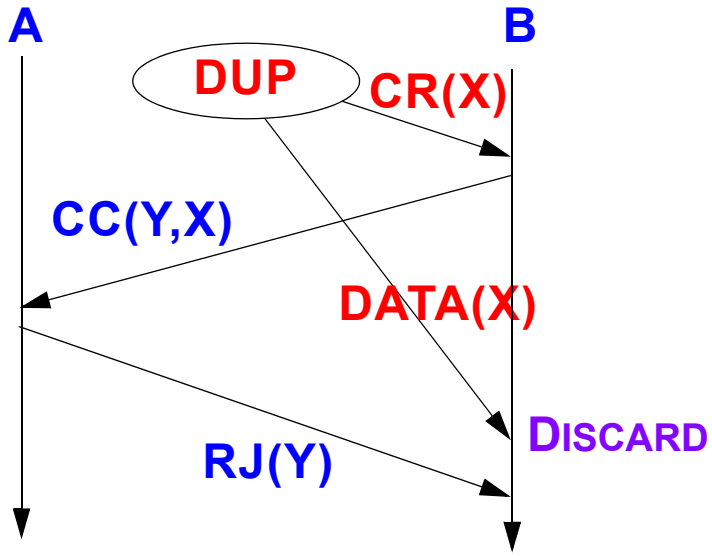
- **initiator (A) acknowledges**
  - sequence numbers **X**, **Y** (as received before)
- **after receiving a valid ACK, receiver (B) accepts data**

Note:

- **some protocols (including TCP) acknowledge the next byte expected**
  - (ACK X+1,Y+1), not the last byte received

Three Way Handshake Protocol: Results

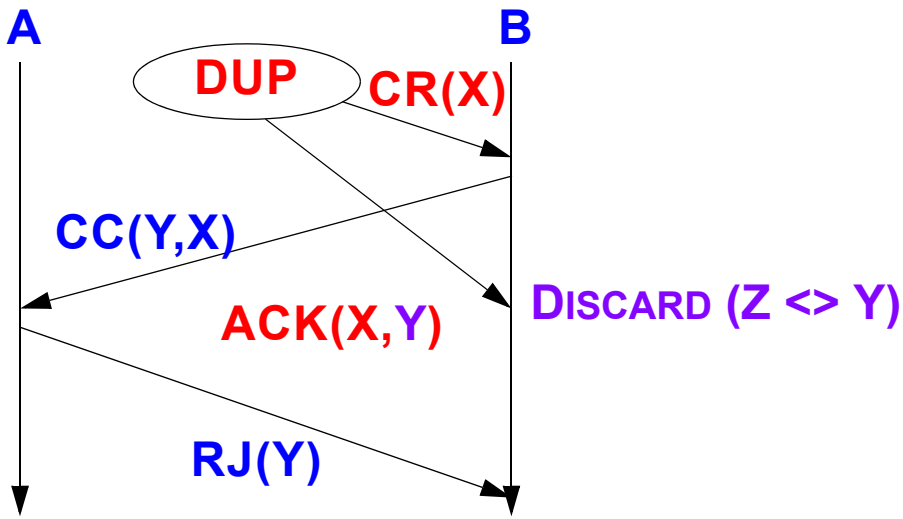
CR and data duplicate



- **duplicated data is discarded**
  - for success
    - should have occurred an ACK (X) before

Three Way Handshake Protocol: Results (2)

Connect Request CR Duplicate and Acknowledgment AK Duplicates



- **AK (X,Z) discarded because**
  - AK (X, Y) expected
  - AK (X, Z) received, **Z <> Y**
    - B will be ensured by a premise of a maximum packet lifetime by selecting the initial sequence number according to the described algorithms

6. Disconnect

Two variants:

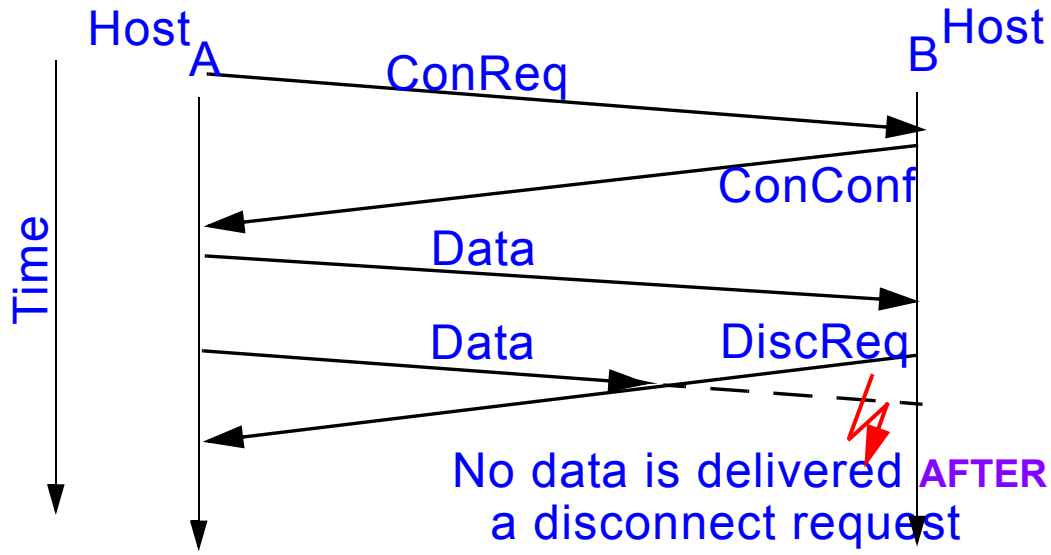
- **asymmetric disconnect**
- **symmetric disconnect**

6.1 Asymmetric Disconnect

Approach

- **to disconnect in one direction implies to disconnect in both “directions”**
  - analog to telephone
- **e.g. may result in data losses**

Example



⇒ **approach for a solution:**

- 3 phase-handshake-protocol
  - to implement a disconnect
  - like a connect

6.2 Symmetric Disconnect

Idea:

to avoid data loss incurred by asymmetric disconnect by using symmetric disconnect

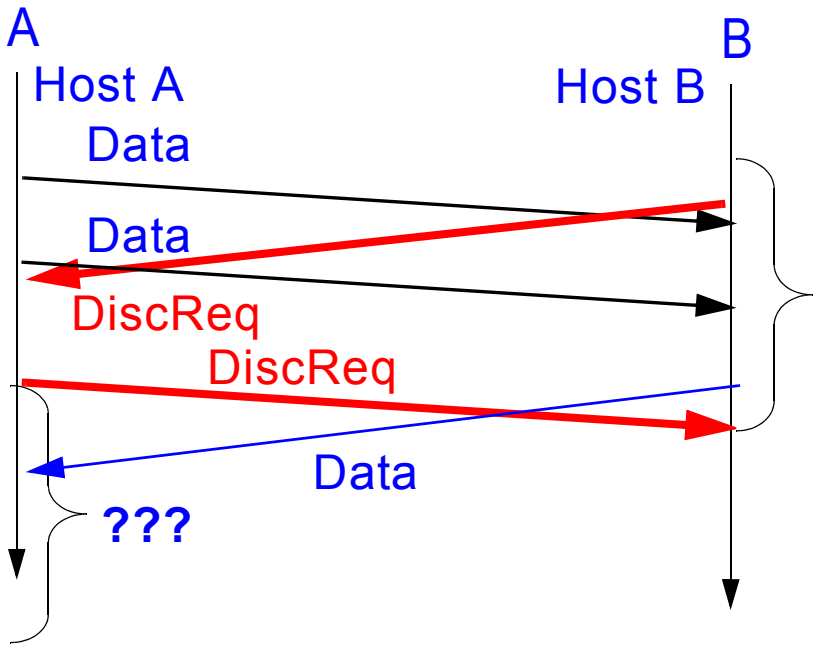
- **i.e.**
  - both sides have to issue a disconnect
  - host received DISCONNECT -> stops to send data
  - host sent DISCONNECT -> may continue to receive data

Properties

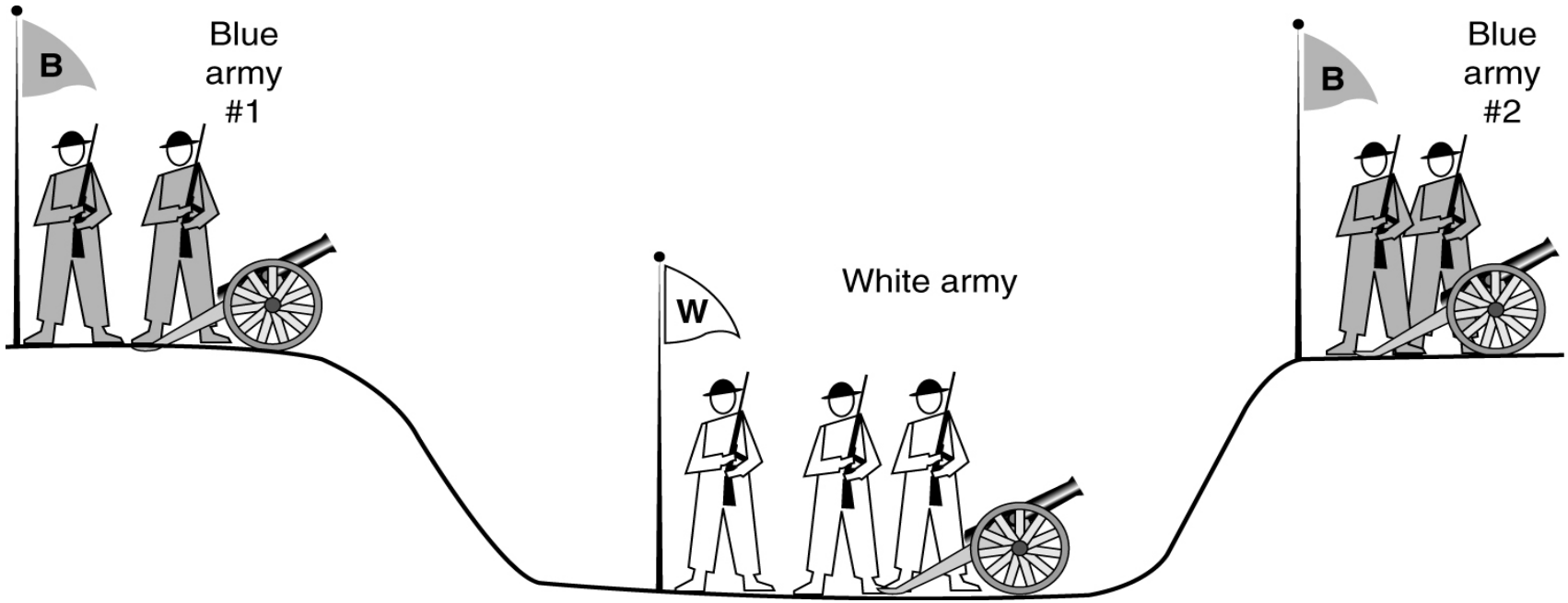
- **if host knows**
  - after having send a disconnect
  - how much data (or how long data) will be issued by the partner
  - i.e. how much data will arrive

  ⇒ **works well**
- **if host does not know about**
  - data to be received after having send a disconnect
  - after having send a disconnect

  ⇒ **???**

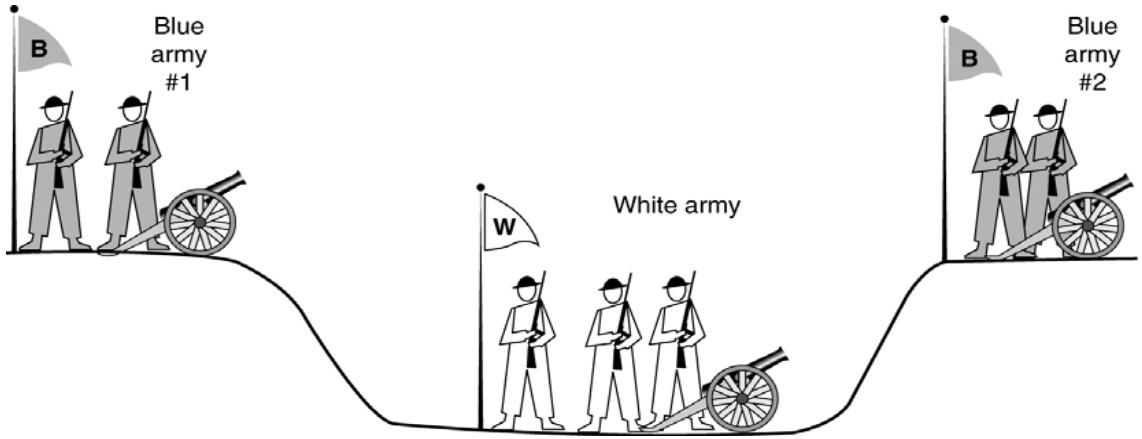


Two-Army Problem



- **army/ies win/s which at a single attach has/ve more soldiers**
  - **⇒ 2 blue armies need to be synchronized**
  - **⇒ 2 blue armies have to agree on exact time of attack**

Two-Army Problem (2)



- **army/ies win/s which at a single attach has/ve more soldiers**
  - ⇒ **2 blue armies need to be synchronized**
  - ⇒ **2 blue armies have to agree on exact time of attack**

Notices

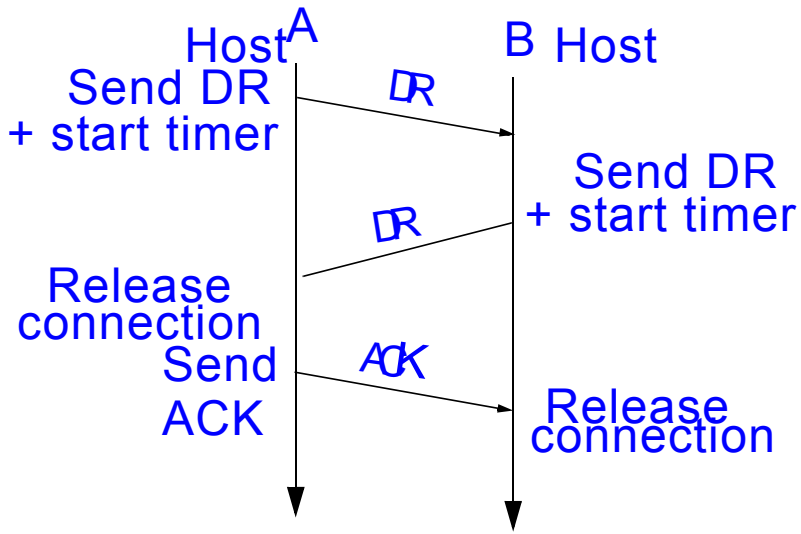
- **blue1 ---> blue2: let us attack at 11:11**
- **blue1 <--- blue2: OK**
- **blue1 ---> blue2: OK**
- **blue1 <--- blue2: OK**

Problem: when to stop?

- all messages need acknowledgements to be sure that other commander agrees
- but 'final' ACK can always be lost
- no perfect protocol exists

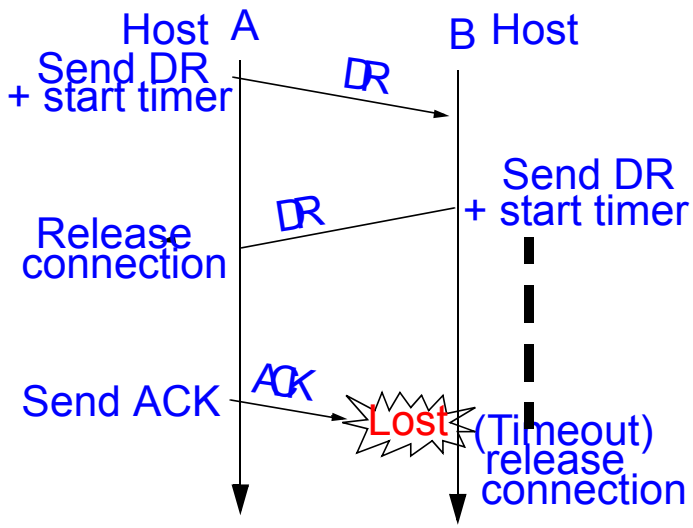
Disconnect with Three-Way Handshake

Regular disconnect with Three-Way handshake



Disconnect with Three-Way Handshake (2)

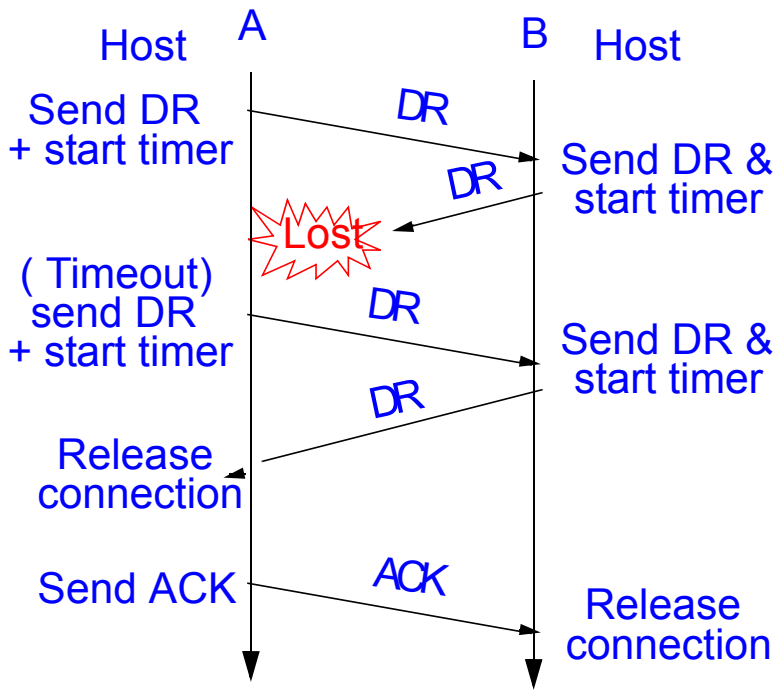
Last acknowledgment lost



- **timer disconnects from host 2**
- **therefore no further problems**

Disconnect with Three-Way Handshake (3)

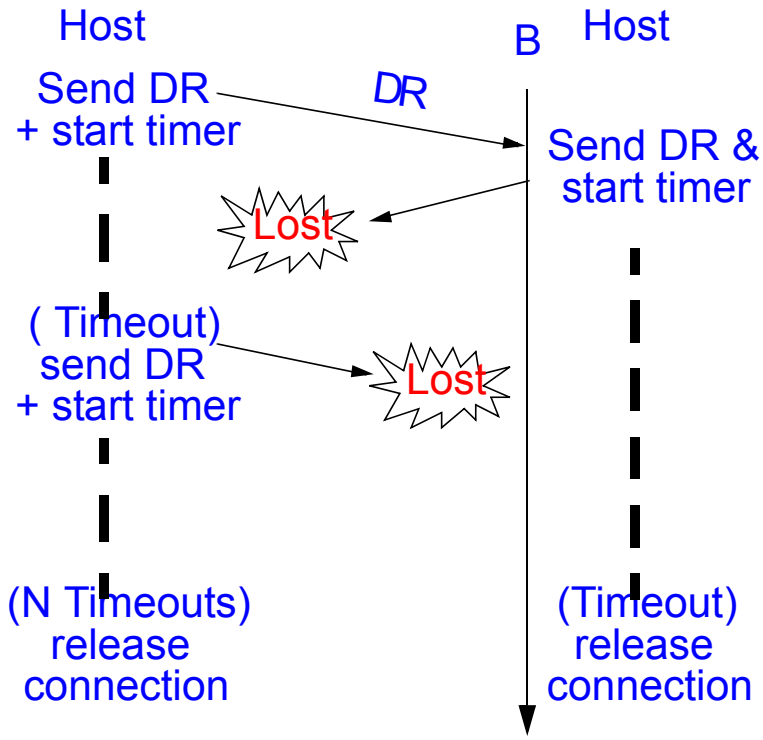
Second “disconnect request” lost



- **repeat to send “disconnect release”**
  - because response was an unexpected DR and ACK
- **loss is repaired**
  - otherwise procedure as described using timer

Disconnect with Three-Way Handshake (4)

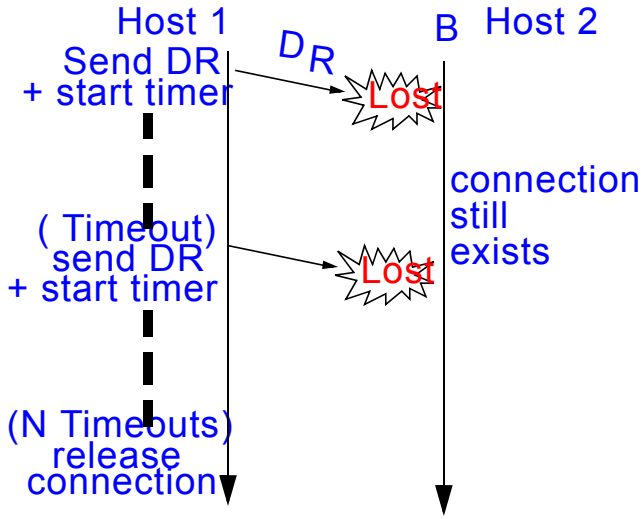
Second and all further “disconnect releases” lost



- **disconnects by timeout**

Disconnect with Three-Way Handshake (5)

All “disconnect releases” lost



- **resulting problem:**
  - Host1 disconnects,
    but Host2 retains inconsistent information: “semi-open” connections
- **prevented by: activity strategy**
  - TPDUs have to arrive within a certain time
    - otherwise automatic disconnect
  - implementation
    - after TPDU has been sent: re-initiate timer
    - when timeout before data has been sent:
      send “Dummy-TPDU” to retain connection
      (“keep-alive” packets without actual data)

7. Flow Control on Transport Layer

Joint characteristics (flow control on data link layer)

- **fast sender shall not flood slow receiver**
- **sender shall not have to store all not acknowledged packets**

Differences (flow control on data link layer)

- **L2-DLL: router serves few “bagpipes”**
- **L4-TL: endsystem contains a multitude of**
  - connections
  - data transfer sequences
- **L4-TL: receiver may (but does not always have to) store packets**

Strategies

1. **sliding window / static buffer allocation**
2. **sliding window / no buffer allocation**
3. **credit mechanism / dynamic buffer allocation**

7.1 Sliding Window / Static Buffer Allocation

Flow control

- **sliding window - mechanism with window size w**

Buffer reservation

- **receiver reserves 2*w buffers per duplex connection**

Characteristics

+ receiver can accept all PDUs

- buffer requirement may be very high
  - proportional to #transp.-connections

- poor buffer utilization for low throughput connections

i.e.

⇒ **good for traffic with high throughput**
  - (e.g. data transfer)

⇒ **poor for bursty traffic with low throughput**
  - (e.g. interactive applications)

7.2 Sliding Window / No Buffer Allocation

Flow control

- sliding window (or no flow control)

Buffer reservation

- receivers do not reserve buffers
- buffers allocated buffer space upon arrival of TPDU
- TPDU will be discarded if there are no buffers available
- sender maintains TPDU buffer until ACK arrives from receiver

Characteristics

\+ optimized storage utilization

\- possibly high rate of ignored TPDUs during high traffic load

i.e.

$\Rightarrow$ good for bursty traffic with low throughput

$\Rightarrow$ poor for traffic with high throughput

7.3 Credit Mechanism

Flow control

- **credit mechanism**

Buffer reservation

- **receiver allocates buffers dynamically for the connections**
- **allocation depends on the actual situation**

Principle

- **sender requests required buffer amount**
- **receiver reserves as many buffers as the actual situation permits**
- **receiver returns ACKs and buffer-credits separately**
  - ACK: confirmation only (does not imply buffer release)
  - CREDIT: buffer allocation
- **sender will be blocked, when all credits have been used up**

Credit Mechanism

Example: with dynamic buffer allocation

- **4 bit SeqNo (0..15) and "..." corresponds to data loss**

Comments		A	Message	B	Comments (buffers located at B)
A wants 8 buffers	1	→	< request 8 buffers>		
	2		<ack = 15, cred = 4>	←	B grants messages 0-3 only
A has 3 buffers left now	3	→	<seq = 0, data = m0>		
A has 2 buffers left now	4	→	<seq = 1, data = m1>		
Message lost but A thinks it has 1 left	5	→	<seq = 2, data = m2>	•••	Message lost
	6		<ack = 1, cred = 3>	←	B acknowledges 0 and 1, permits 2-4
A has buffer left	7	→	<seq = 3, data = m3>		
A has 0 buffers left, and must stop	8	→	<seq = 4, data = m4>		
A times out and retransmits	9	→	<seq = 2, data = m2>		
but A still blocked	10		<ack = 4, cred = 0>	←	Everything acknowledged, A still blocked
A may now send next msg.	11		<ack = 4, cred = 1>	←	
	12		<ack = 4, cred = 2>	←	B found a new buffer somewhere
A has 1 buffer left	13	→	<seq = 5, data = m5>		A has 1 buffer left
A is now blocked again	14	→	<seq = 6, data = m6>		A is now blocked again
A is still blocked	15		<ack = 6, cred = 0>	←	A is still blocked
	16	•••	<ack = 6, cred = 4>	←	

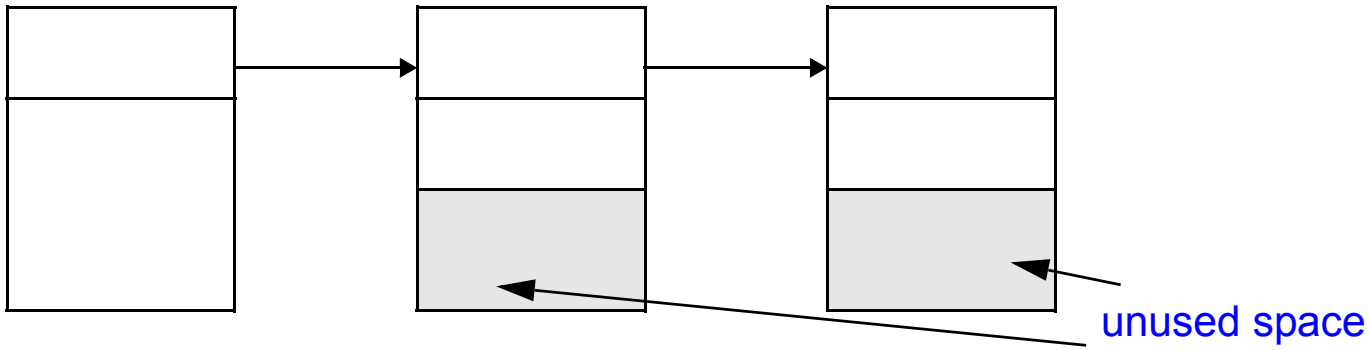
Dynamic adjustment to

- **buffer situation**
- **number of open connections**
- **type of connections**
  - high throughput: many buffers
  - low throughput: few buffers

8. Memory Management / Cache Administration

Interlinked buffer pool per endsystem,

constant buffer size



+ simple because of consistent buffer size

- buffer size difficult to determine

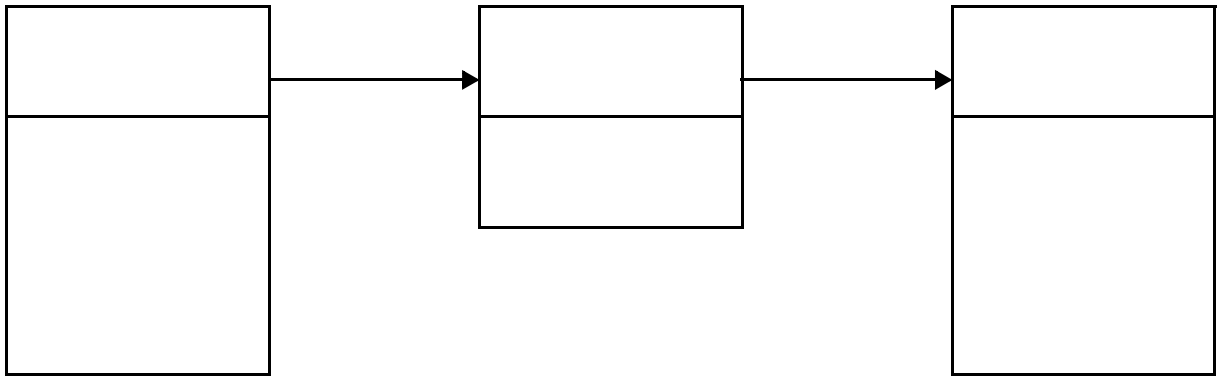
- too large: high fragmentation for small TPDUs
- too small: TPDU distributed over several caches

i.e. increasing complexity, buffer space wasted

Memory Management / Cache Administration (2)

Interlinked buffer pool per endsystem,

variable buffer size



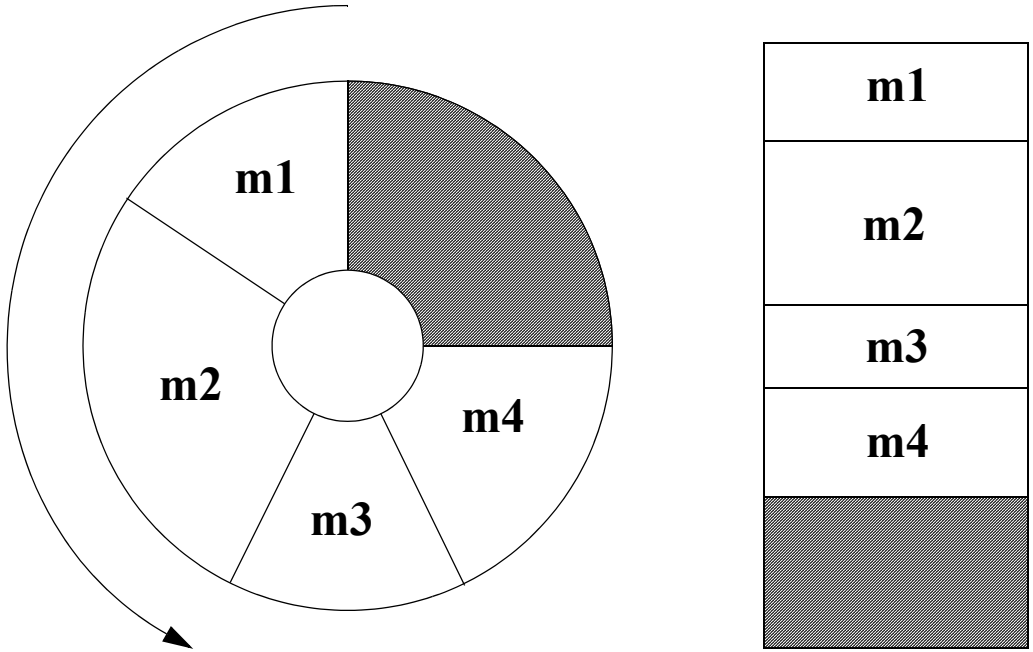
+ good storage use

- buffer management with increased complexity

⇒ more time-consuming

Memory Management / Cache Administration (3)

For each Transport layer connection 1 circular buffer is allocated



+ good storage usage during high utilization of a connection
- poor storage usage during low utilization

Memory Management / Cache Administration (4)

Strategies are traffic dependent

- **low data rate, bursty**
  - example: interactive terminal
    - => interlinked with variable buffer size,
      dynamic allocation both on sender's and receiver's side
    - => sender should buffer

- **high constant data rate**
  - example: file transfer
  - provide sufficient buffer space
    - => interlink with constant size
      static allocation on the receiver's side
    - => receiver should provide more buffer space

Memory Management / Cache Administration (5)

Strategies

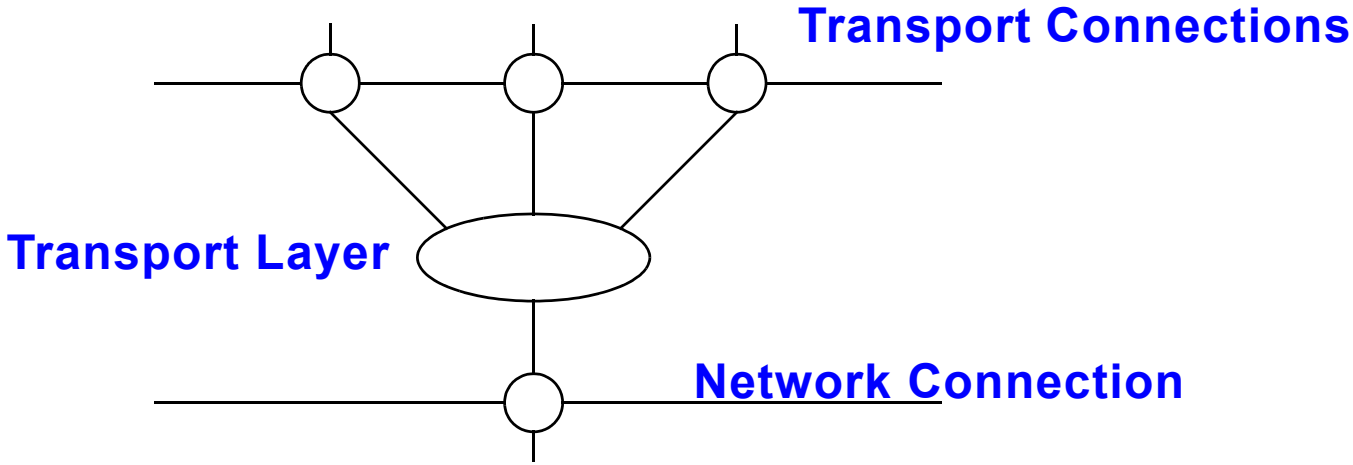
Interaction sender - receiver

- **sender**
  - knows traffic particularities
  - should also be able to “control” buffer reservation on receiver’s end
- **receiver**
  - can inform sender about reserved buffer space
  - sender can modify/control traffic

Allocation

- **per connection**
- **for all connections between 2 endsystems**

9. Multiplexing / Demultiplexing



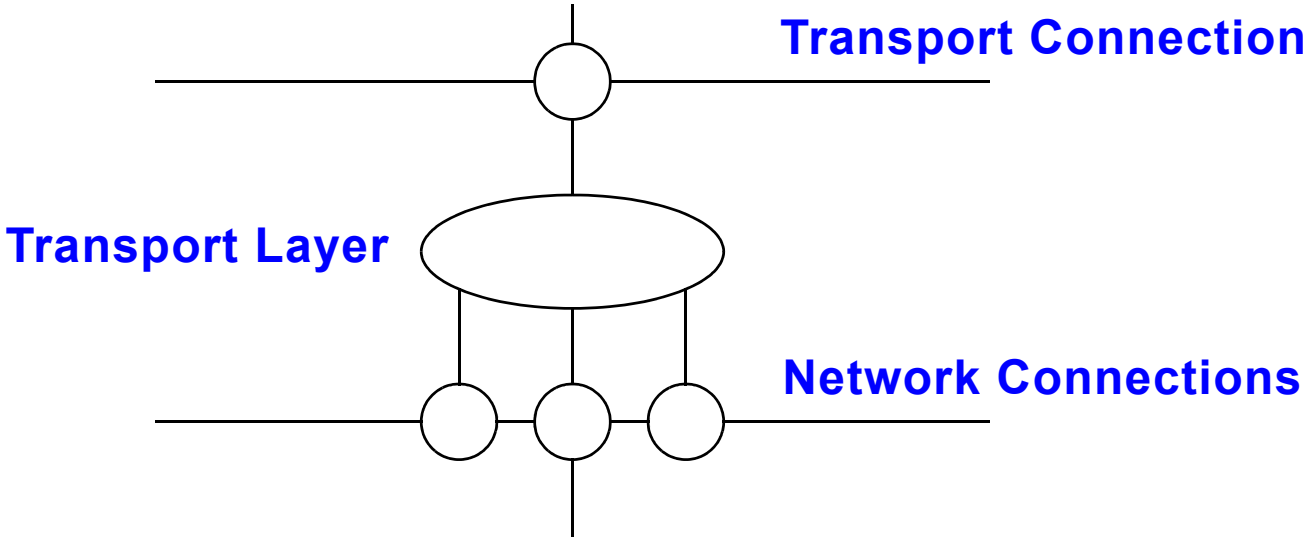
Application

- **minimizing costs when num. of connections / connection time represents the main cost factor**

Multiplexing function

- **grouping of T connections by destination address**
- **each group is mapped to the minimum number of network connections**
  - too many L4-T connections per L3-V connection
    - ⇒ possibly poor throughput
  - too few T connections per V connection
    - ⇒ possibly transfer costs too high

Splitting / Recombination



Application:

- implementation of T connections with high bandwidth

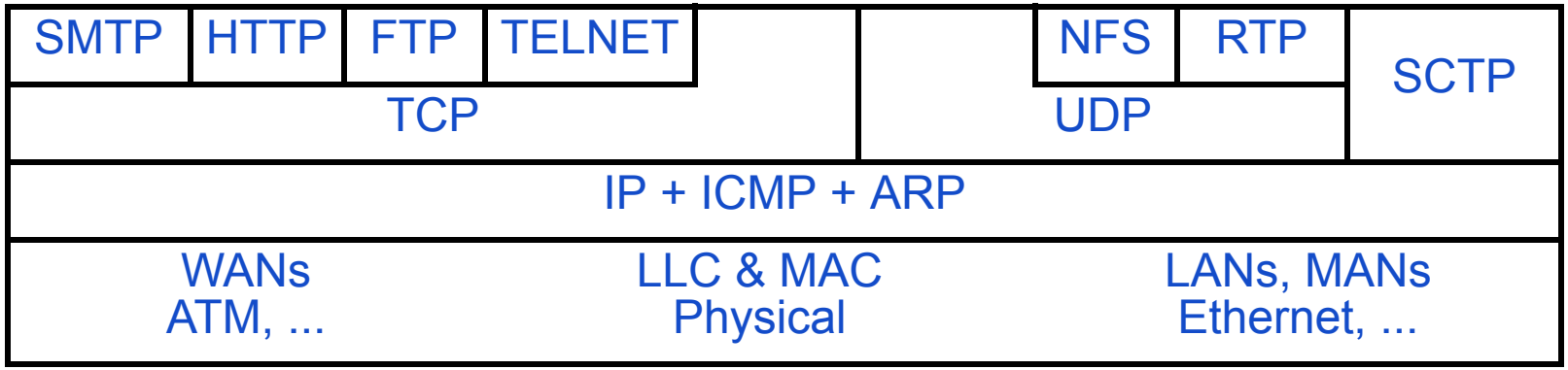
Splitting function

- distributing the TPDUs onto the various network connections
- usual algorithm: Round Robin

Comment

- also known as “upward” multiplexing

10. Some Familiar Internet Protocols



ARP	**=**	**Address Resolution Protocol**
FTP	**=**	**File Transfer Protocol**
HTTP	**=**	**Hypertext Transfer Protocol**
IP	**=**	**Internet Protocol**
ICMP	**=**	**Internet Control Message Protocol**
LLC	**=**	**Logical Link Control**
MAC	**=**	**Media Access Control**
NFS	**=**	**Network File System**
SMTP	**=**	**Simple Mail Transfer Protocol**
TELNET	**=**	**Remote Login Protocol**
TCP	**=**	**Transmission Control Protocol**
UDP	**=**	**User Datagram Protocol**
SCTP	**=**	**Stream Control Transmission Protocol**