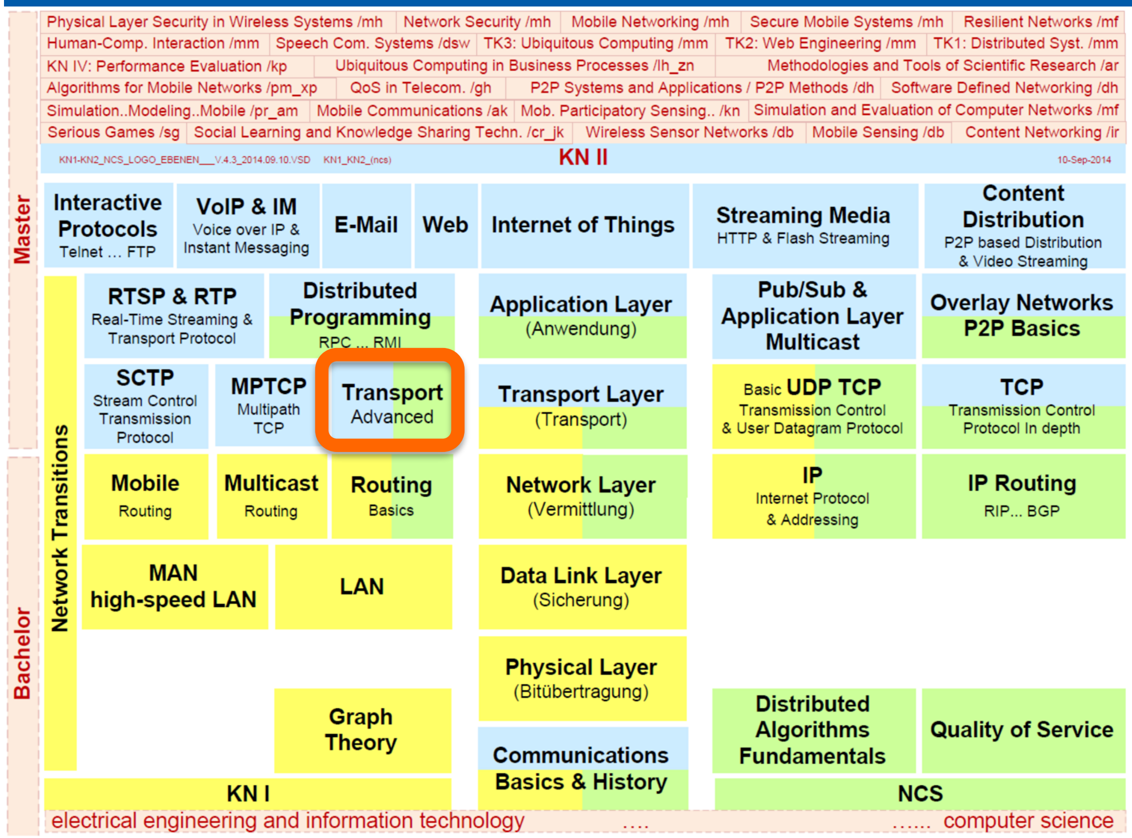
TECHNISCHE UNIVERSITÄT DARMSTADT

Communication Networks II

L4 Transport Layer – Advanced



Prof. Dr.-Ing. **Ralf Steinmetz**
KOM - Multimedia Communications Lab

Overview

1 Transport Layer Function
2 Addressing (at Transport Layer)
3 Duplicates (at Data Transfer Phase)
  3.1 Basic Challenges - Example
  3.2 Basic Methods of Resolution
  3.3 Advanced Methods of Resolution
  3.4 Initial Sequence Number Allocation and Handling of Consecutive Connections
4 Connect - Reliable Connection Establishment
5 Disconnect
  5.1 Asymmetric Disconnect
  5.2 Symmetric Disconnect
  5.3 Two-Army Problem
6 Flow Control on Transport Layer
  6.1 Sliding Window / Static Buffer Allocation
  6.2 Sliding Window / No Buffer Allocation
  6.3 Credit Mechanism
7 Multiplexing / Demultiplexing

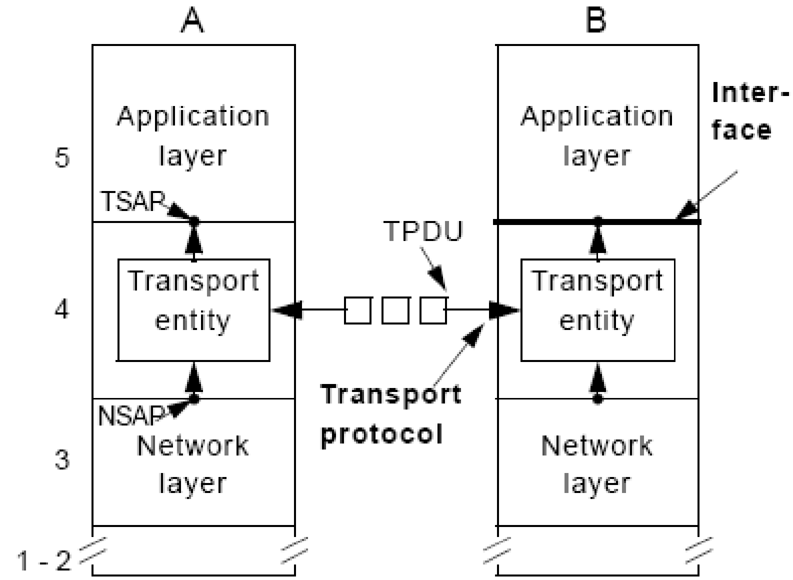
1 Transport Layer Function

To provide data transport

- reliably
- efficiently
- at low-cost

for

- process-to-process (applications)
- i.e. at end system-to-end system

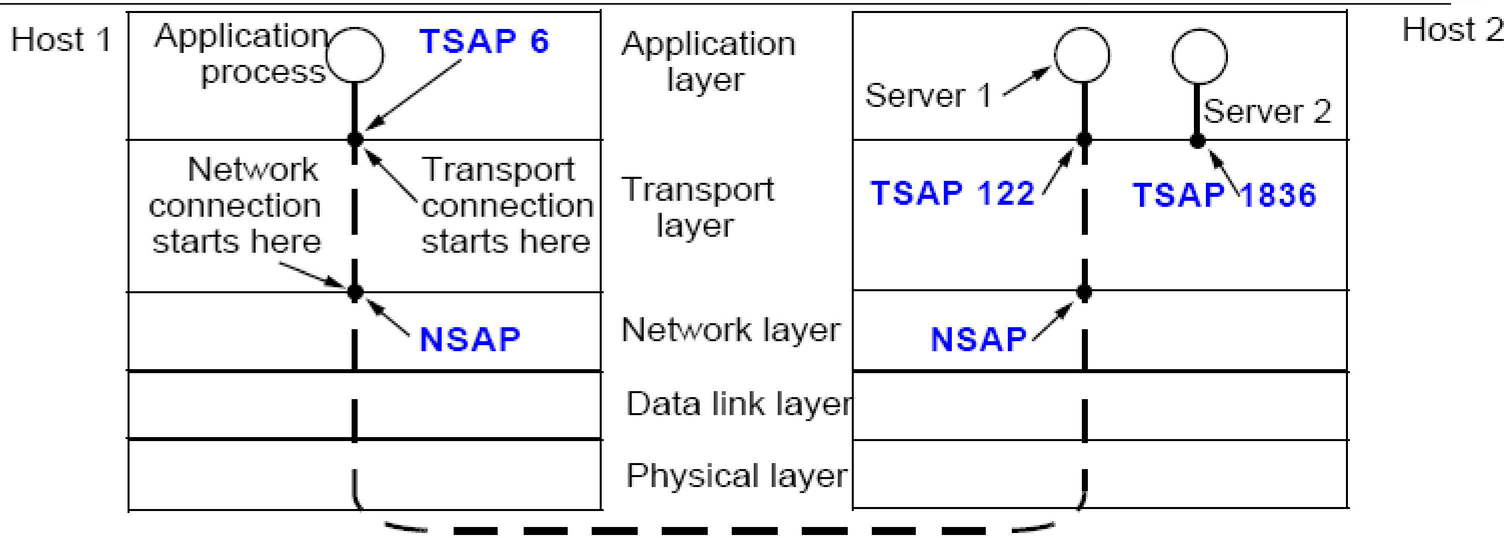


(if possible) independent from

- particularities of the networks (lower layers) used

2 Addressing (at Transport Layer)

Model:



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

Reminder: analogous end points in network layer: NSAP

- e.g., IP addresses

TECHNISCHE UNIVERSITÄT DARMSTADT

Initial Situation:

- 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

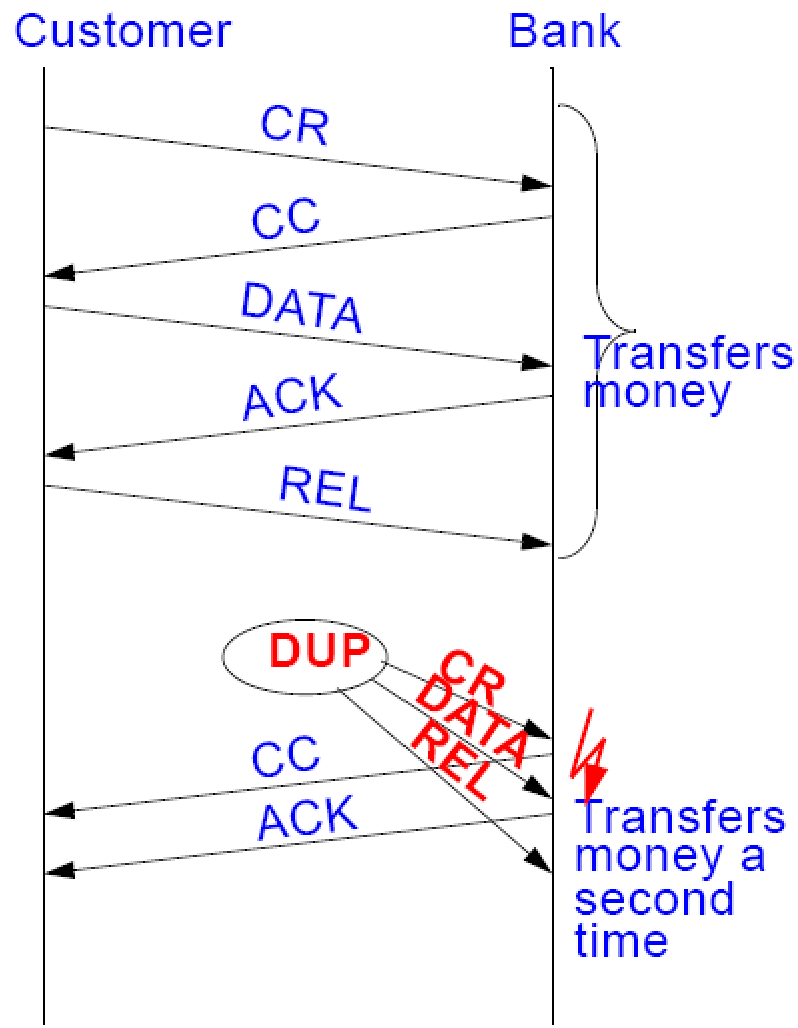
3.1 Basic Challenges - Example

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

- 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



CR ... ConnectReq
CC ... ConnectConf
REL... Release

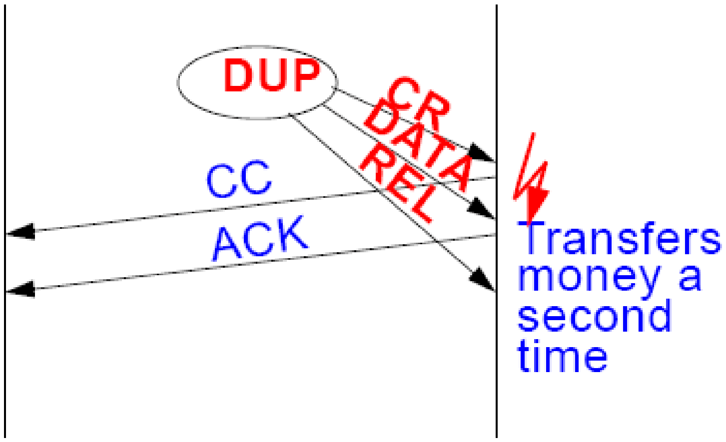
Duplicates – Description of Problematic Issues

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



3.2 Basic Methods of Resolution

1. to use temporarily valid TSAPs

- method:
  - TSAP valid for one connection only
  - generate always new TSAPs
- 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

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

…

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

→ to be considered in the following

Duplicates: Approach to Limit Packet Lifetime

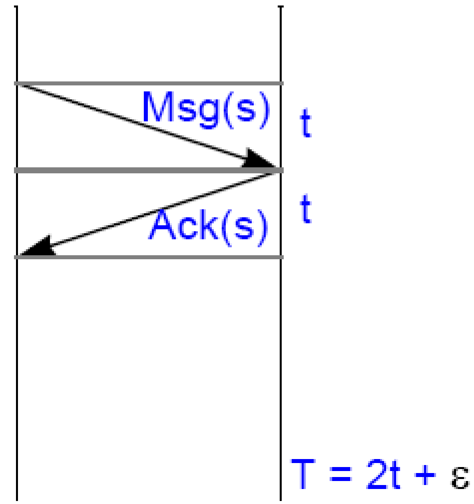
Enabling the 3rd method

- ’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) has to be included
  - otherwise new PDU may be
    - wrongfully confirmed by delayed ACK (N-ACK) or
    - non-confirmed

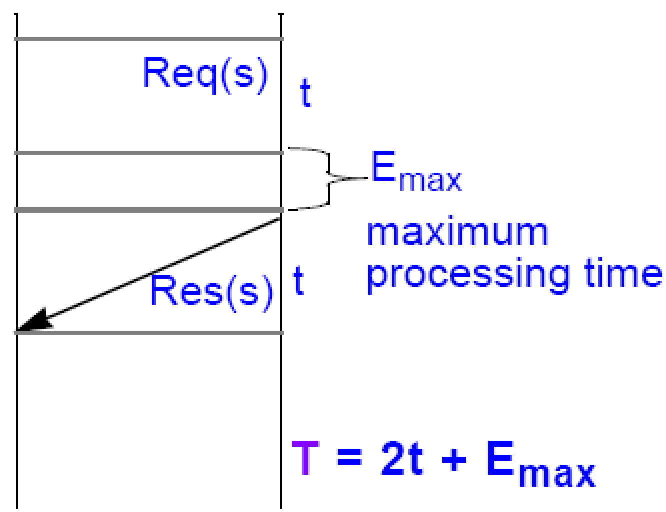
Mandatory prerequisite for this solution

- 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

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

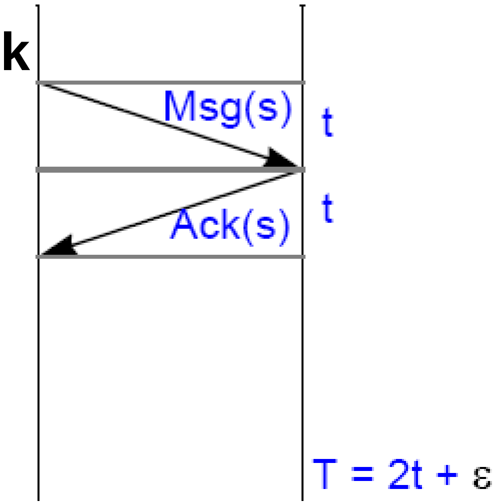
3. Time marker in each packet
- packet exceeds maximum predefined / configured lifetime
  → packet is discarded
- notice: requires “consistent” network time

Duplicates: Approach to Limit Packet Lifetime

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

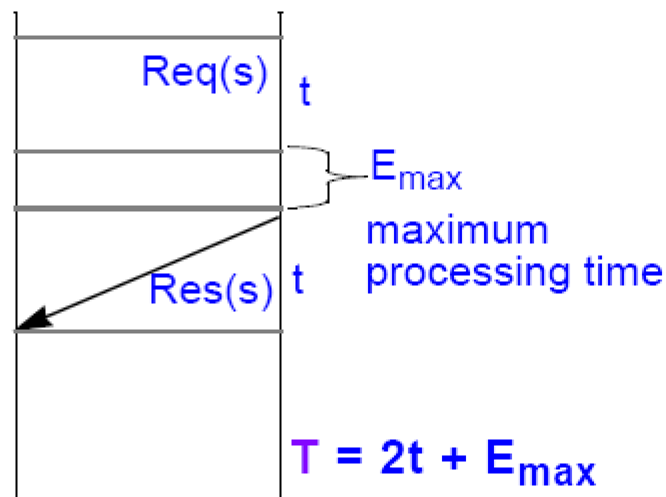
example 1 (in principle)



TCP/IP term:
Maximum Segment Lifetime (MSL)

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

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



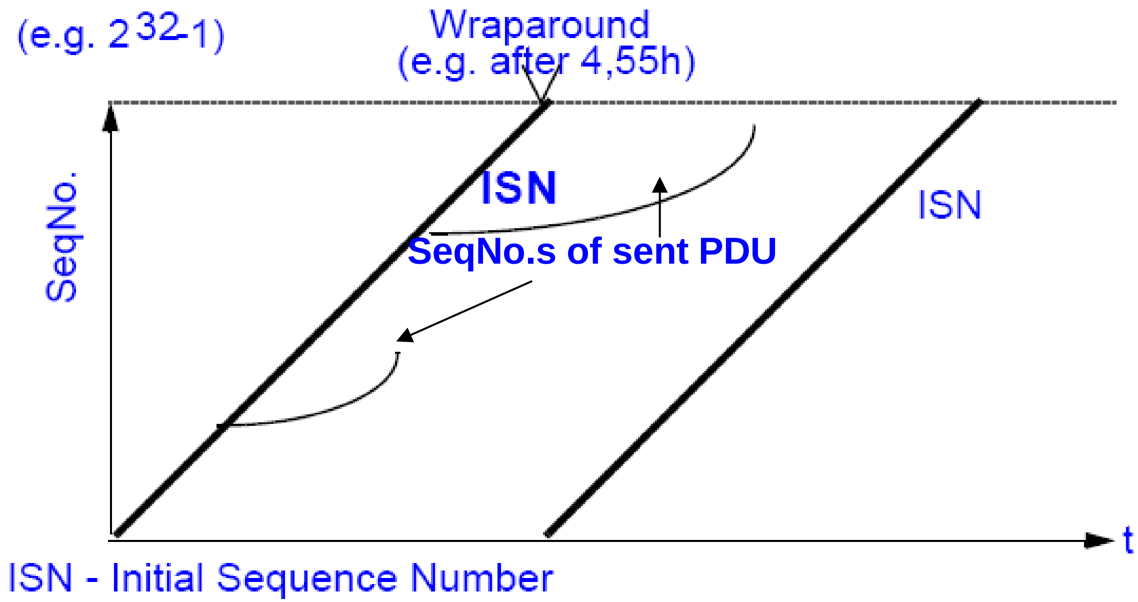
Problem (wrt. "identify PDUs individually: individual sequential numbers for each PDU")

- consider packets from connections which can otherwise not be distinguished
  - hence at TCP:
    - 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:

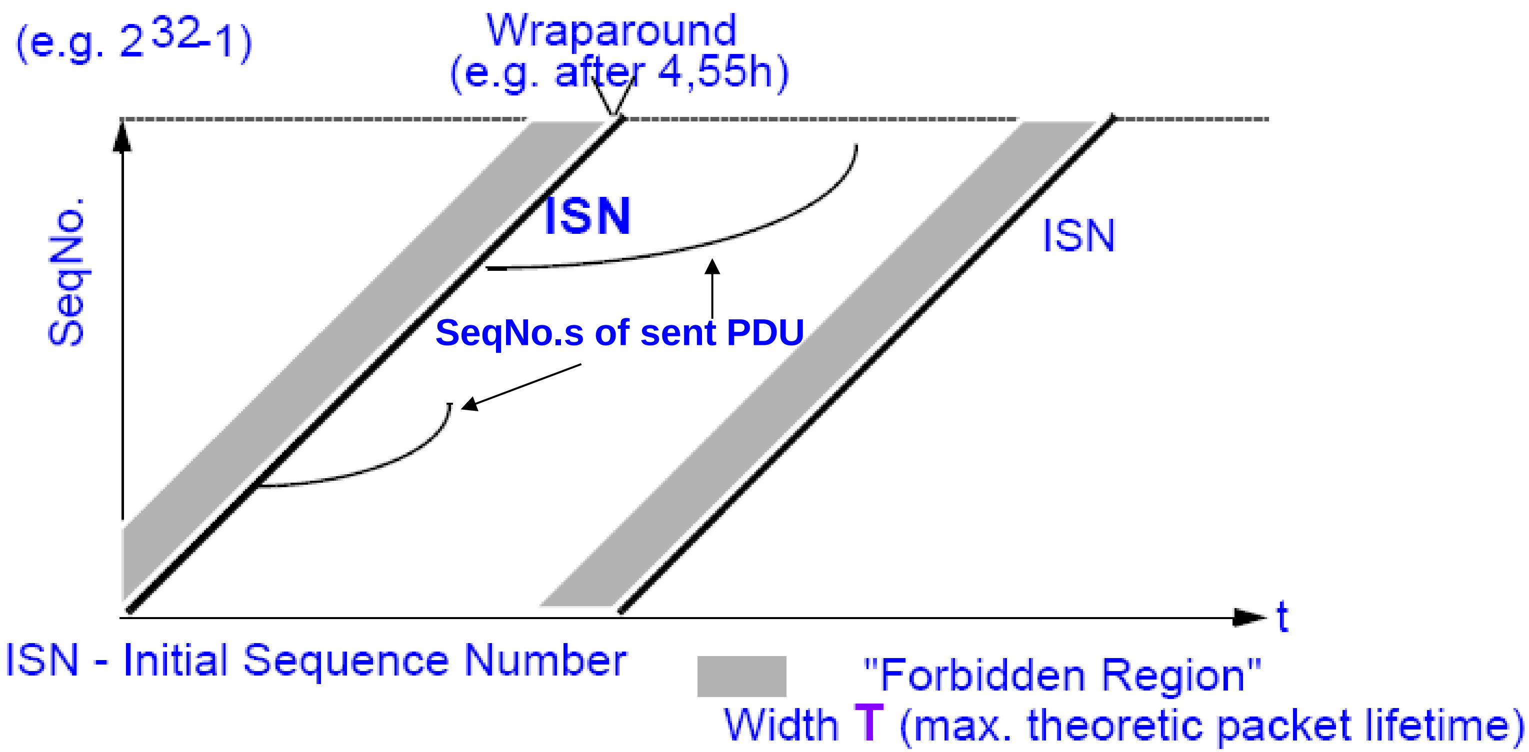
- restart after crash
- (very fast) reconnect between exactly the same communication entities
  - (addr./port see above), information about previous connections do not exist anymore after crash/restart, generally all connections have to be reconsidered
- complete usage of the range of available sequential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

Initial Sequence Number Allocation and Handling of Consecutive Connections

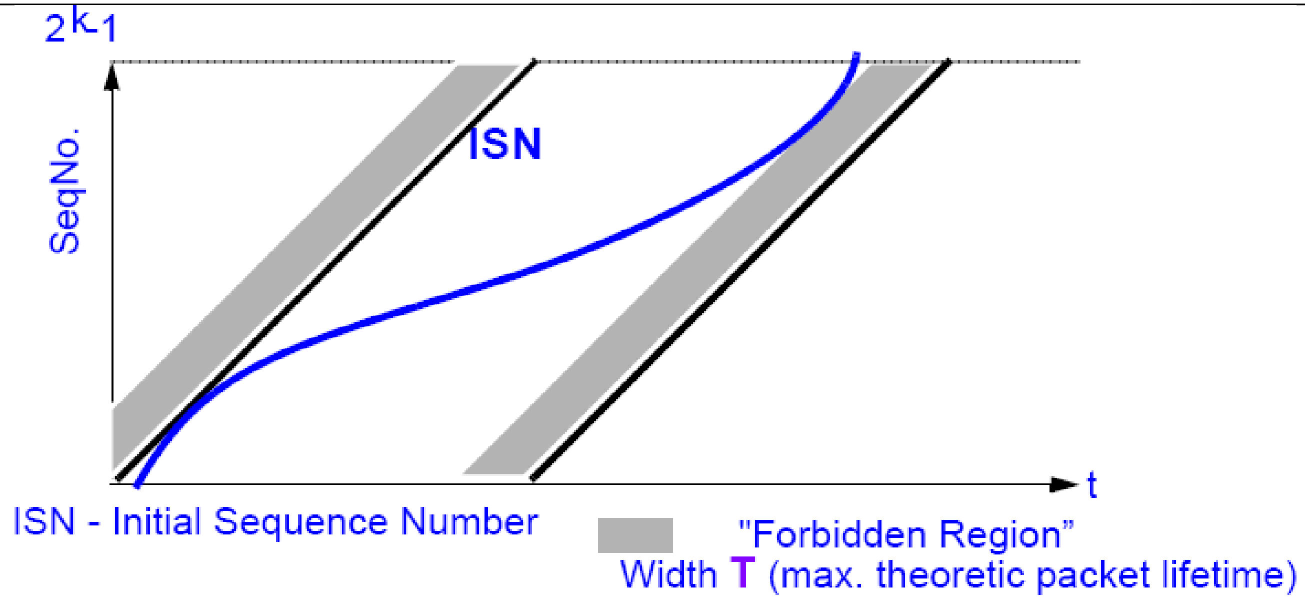


Method

- 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


(e.g. 2^{32}-1)
Wraparound
(e.g. after 4,55h)
SeqNo.
ISN
ISN
SeqNo.s of sent PDU
t
ISN - Initial Sequence Number
"Forbidden Region"
Width T (max. theoretic packet lifetime)

Initial Sequence Number Allocation and Handling of Consecutive Connections



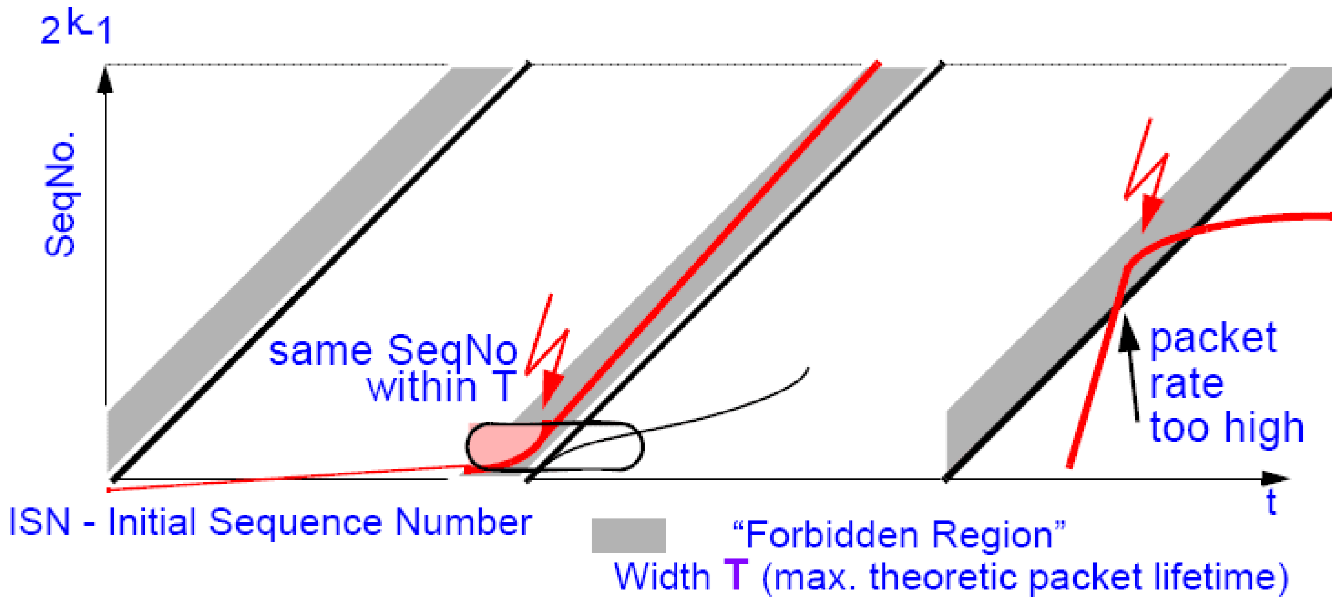
No problem, if

- “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



Problems possible, if

- 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 current data phase)
- "long lived" session (longer than wrap-around time)
- high data rate
  - curve of the consecutively allocated sequence numbers steeper than ISN curve

Initial Sequence Number Allocation and Handling of Consecutive Connections

Note:

- 32 bit sequence numbers with technology considered as sufficient when designing TCP/IP
- sequence number range exploitation (PDU = 1 byte)
  - 10 Mbit/sec in ca. 57 min
  - 1 Gbit/sec   in ca. 34 sec

→ Using timestamps in

- "TCP extensions for high speed 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

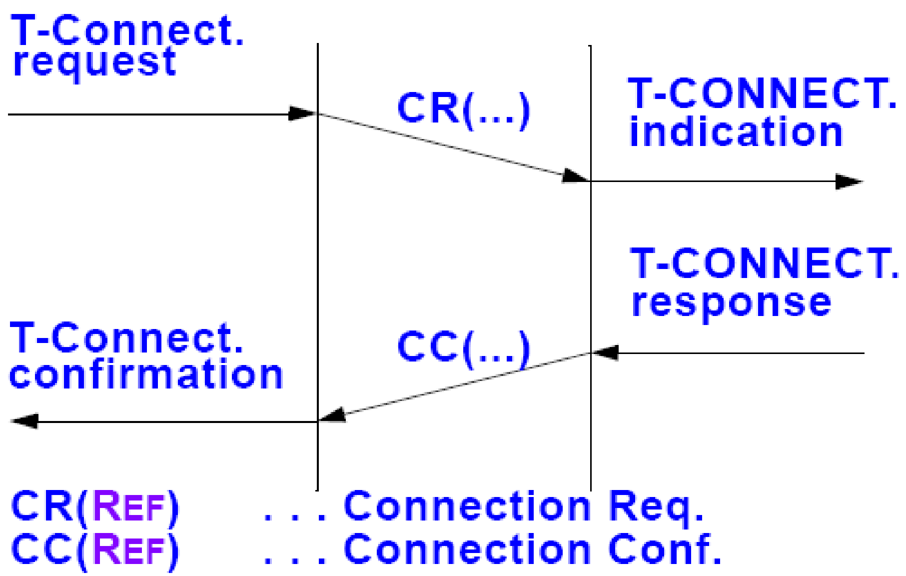
TECHNISCHE UNIVERSITÄT DARMSTADT

Connection

- see also
  Connection Oriented Service:
  State Transition Diagram

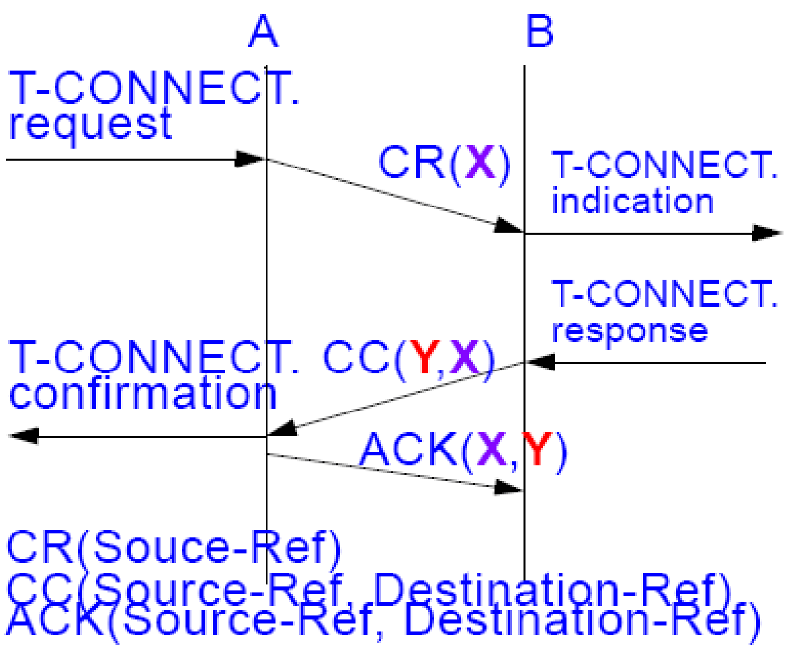
by

- simple protocol



or by

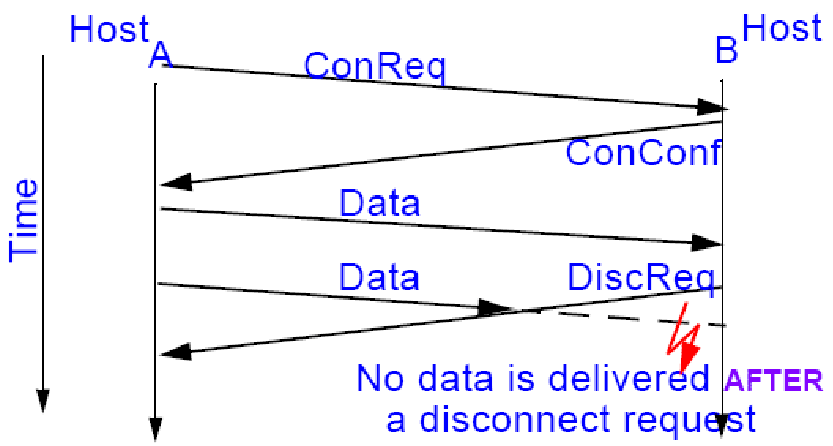
- three-way Handshake Protocol



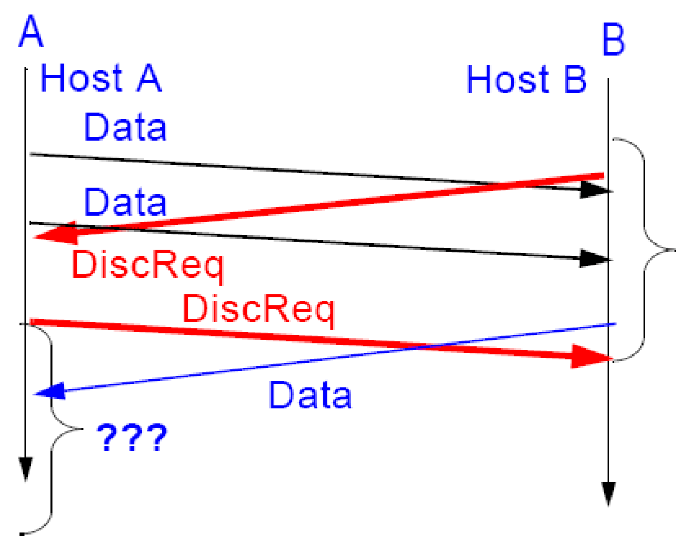
5 Disconnect

Two alternatives

asymmetric disconnect



symmetric disconnect

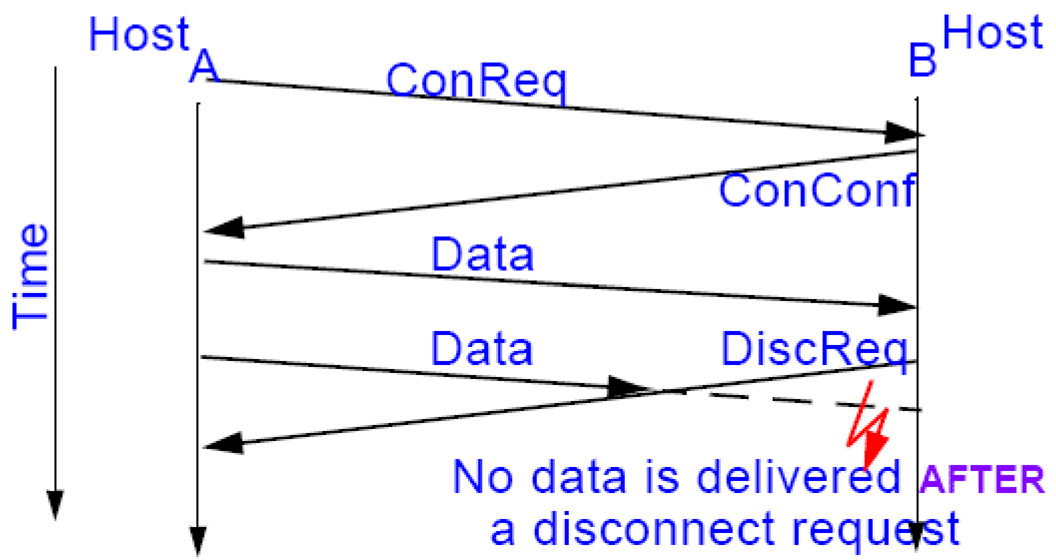


5.1 Asymmetric Disconnect

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

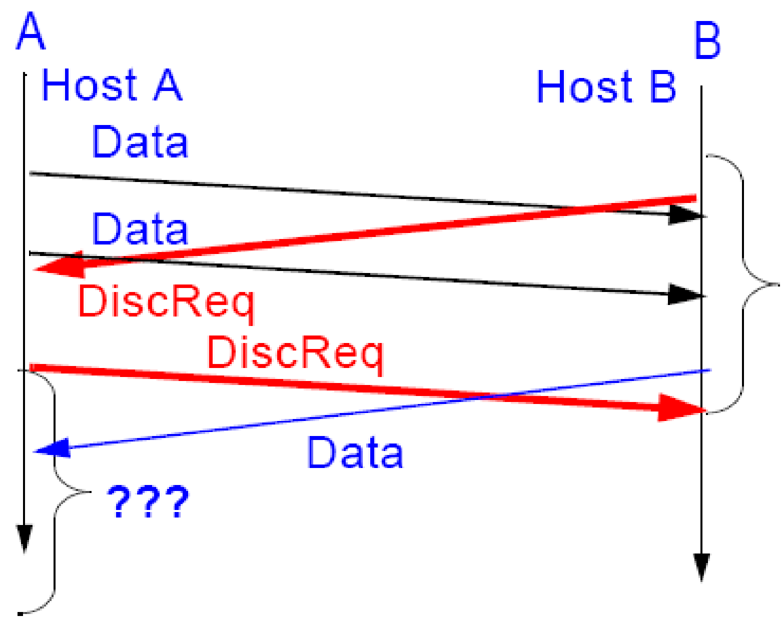
5.2 Symmetric Disconnect

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

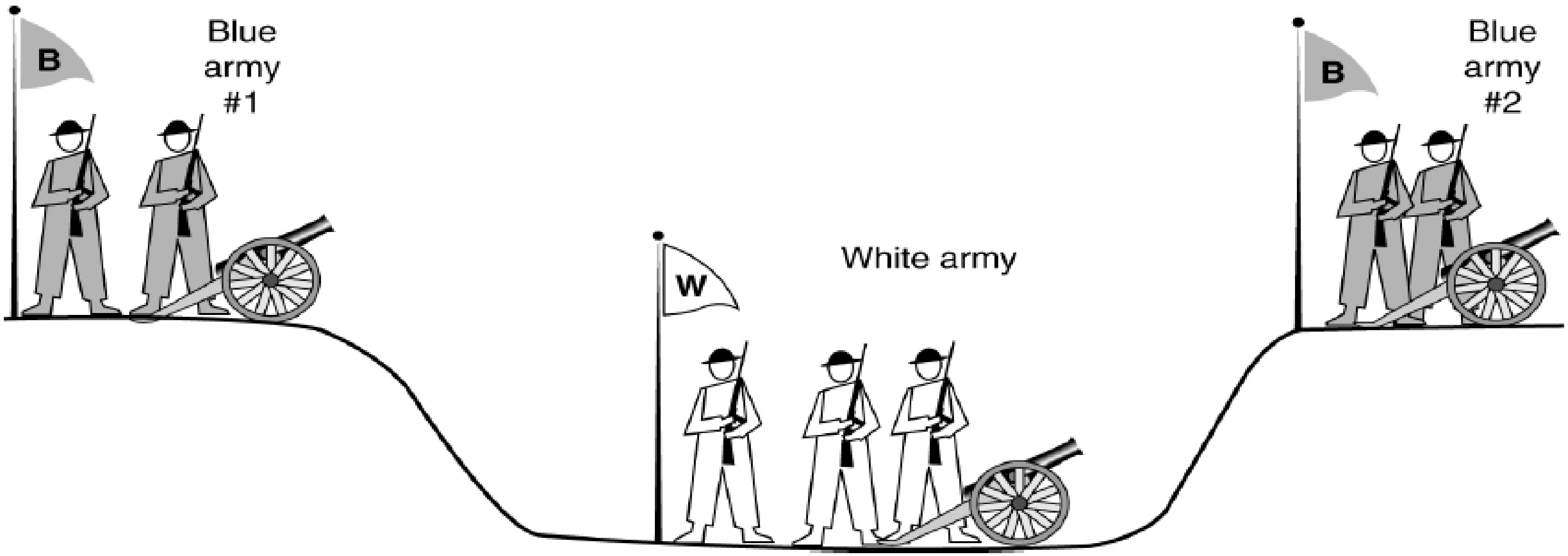
- i.e.
  - both sides have to issue a disconnect
  - host received DISCONNECT → stops to send data
  - host sent DISCONNECT → may continue to receive data

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 sent a disconnect
- ➔ ???

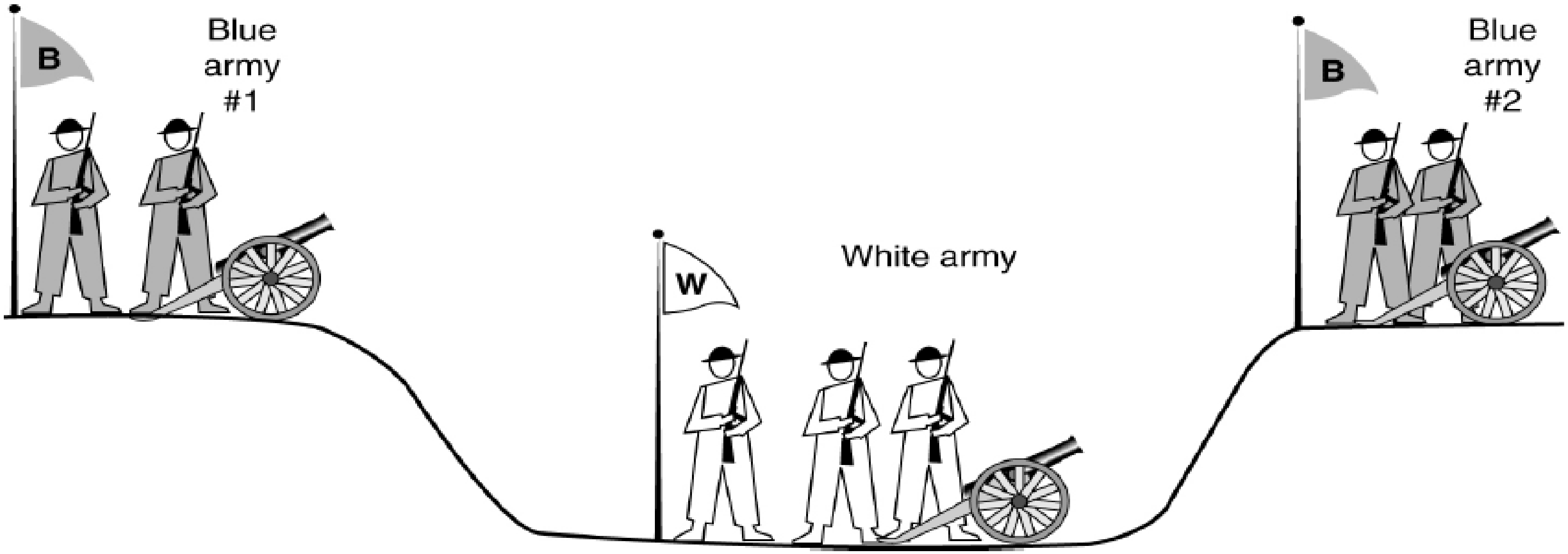


5.3 Two-Army Problem



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

Two-Army Problem



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

Notices

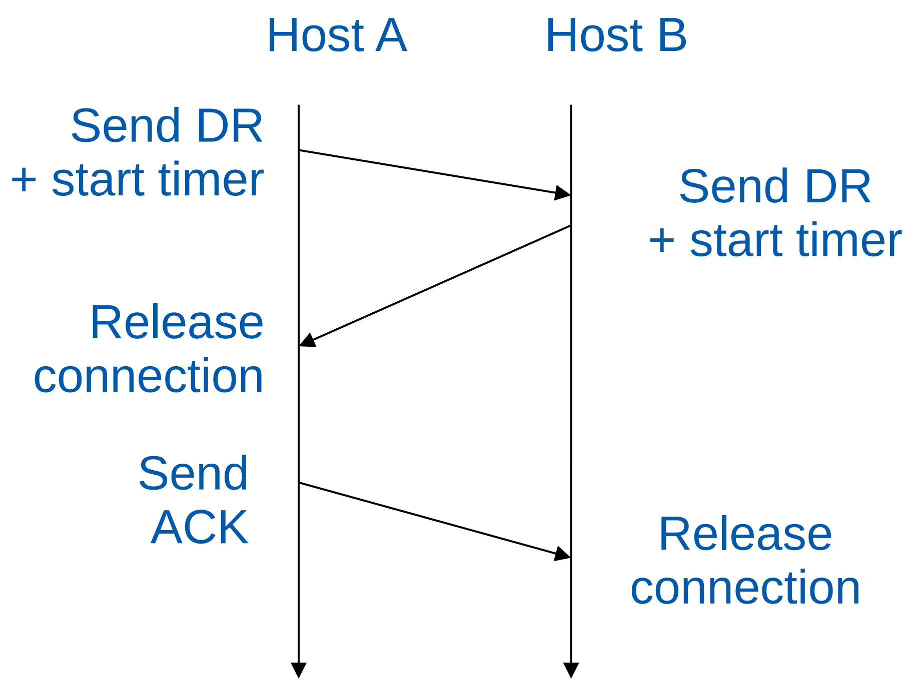
- blue1 → blue2: let us attack at 11:11
- blue1 ← blue2: OK
- blue1 → blue2: OK
- blue1 ← blue2: OK

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

TECHNISCHE UNIVERSITÄT DARMSTADT

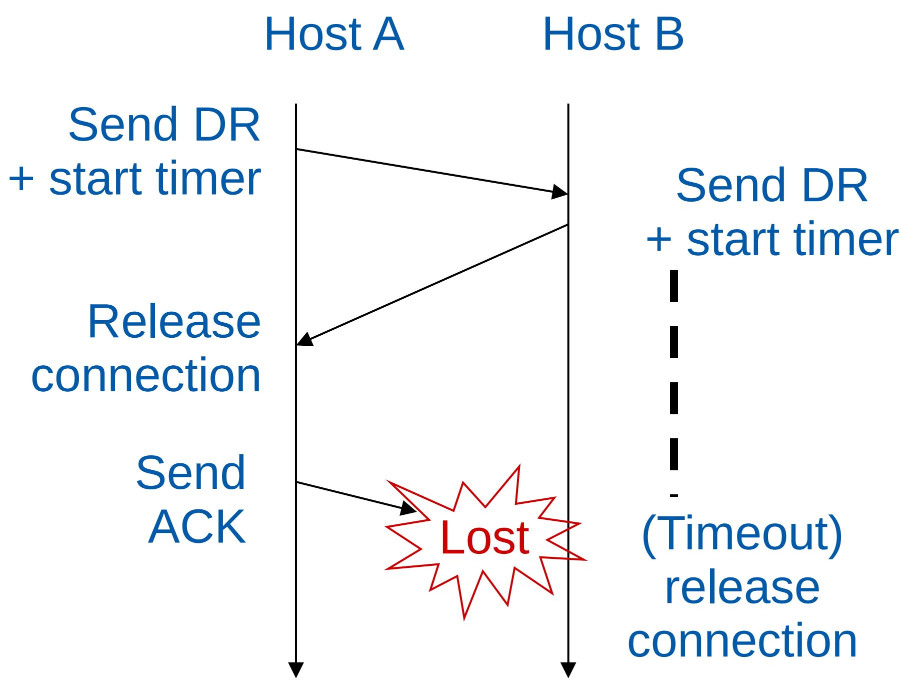
Regular disconnect with Three-Way handshake



Disconnect with Three-Way Handshake

Last acknowledgment lost

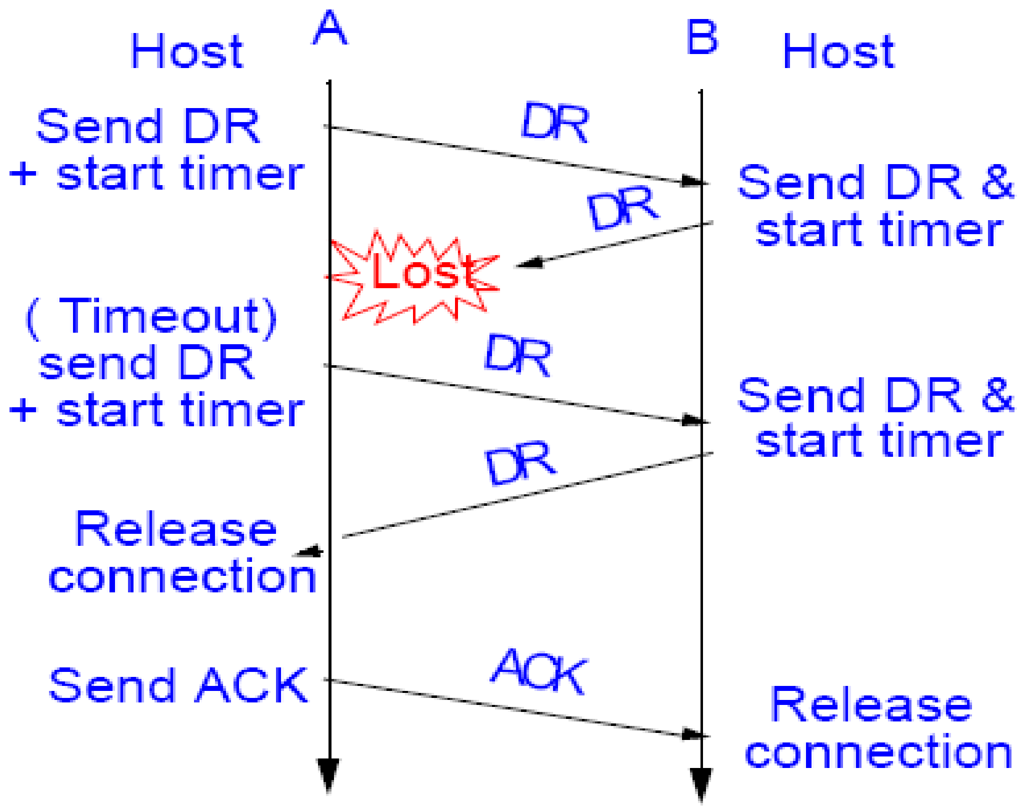
- timer disconnects from host 2
- therefore no further problems



Disconnect with Three-Way Handshake

Second “disconnect request” lost

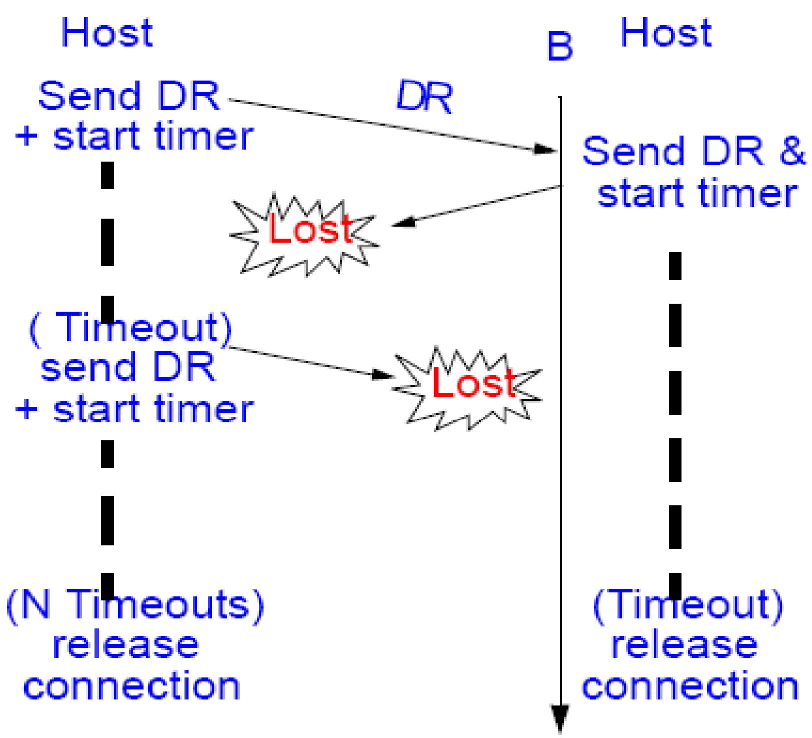
- Host A repeats to send “disconnect request”
  - because response was an unexpected DR and ACK
- loss is repaired
  - otherwise procedure as described using timer



TECHNISCHE UNIVERSITÄT DARMSTADT

Second and all further “disconnect request” lost

- disconnects by timeout

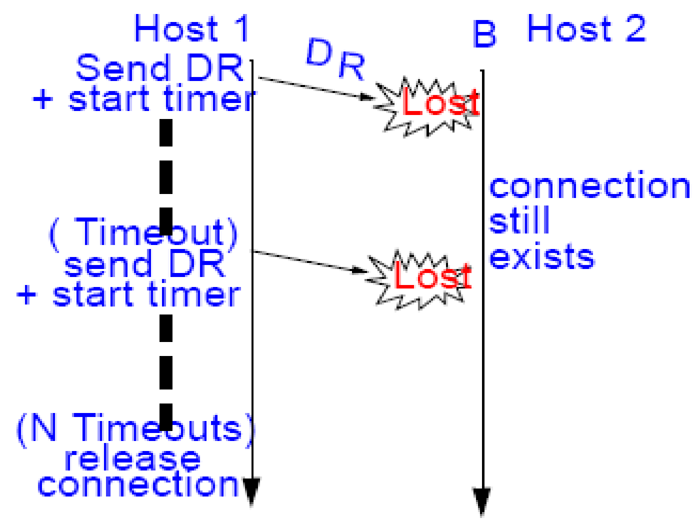


Disconnect with Three-Way Handshake

All “disconnect request” lost

- resulting problem:
  - Host 1 disconnects,
  - but Host 2 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)



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 connections to other routers
- L4-TL: endsystem contains a multitude of
  - connections
  - data transfer sequences
- L4-TL: receiver may (but does not always have to) store packets

Strategies

- sliding window / static buffer allocation
- sliding window / no buffer allocation
- credit mechanism / dynamic buffer allocation

TECHNISCHE UNIVERSITÄT DARMSTADT

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

6.2 Sliding Window / No Buffer Allocation

Flow control

- sliding window (or no flow control)

Buffer reservation

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

Characteristics

+ optimized memory utilization

- possibly high rate of ignored TPDUs during high traffic load

i.e.

➔ good for bursty traffic with low throughput

➔ poor for traffic with high throughput

6.3 Credit Mechanism

Flow control

- credit mechanism

Buffer reservation

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

Principle

- sender requests required buffer amount
- receiver reserves as many buffers as the current 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

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>	←	

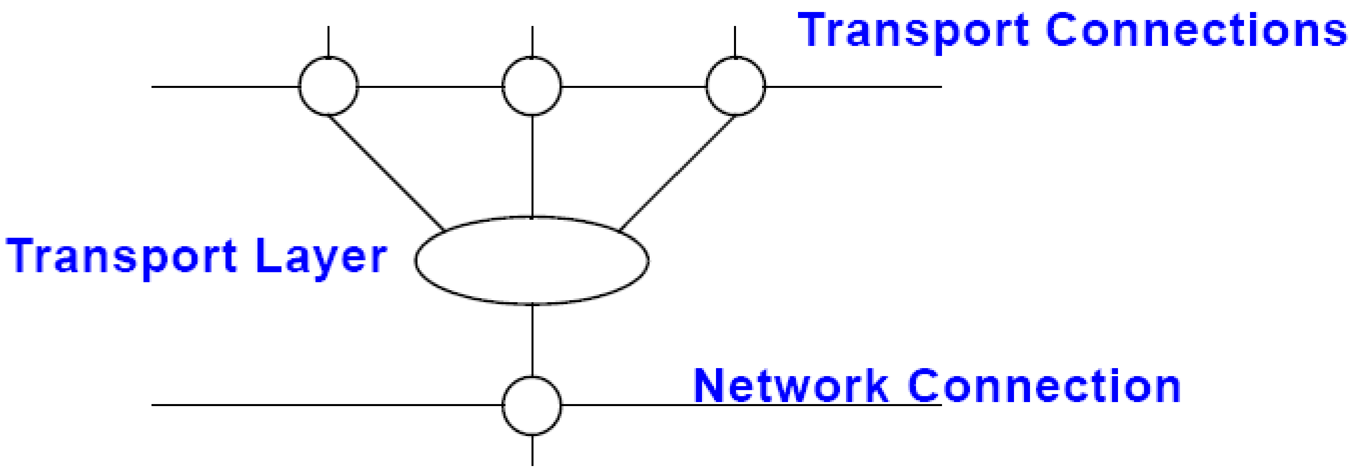
Example: with dynamic buffer allocation

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

Dynamic adjustment to

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

7 Multiplexing / Demultiplexing



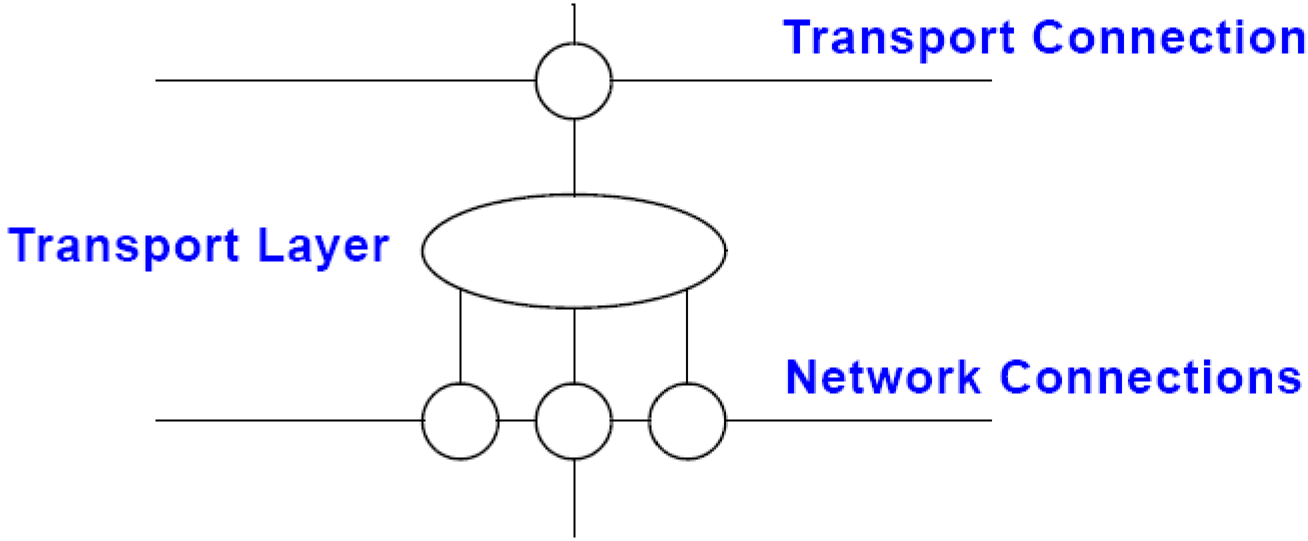
Application

- 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-N connection
    - ➔ possibly poor throughput
  - too few T connections per N 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