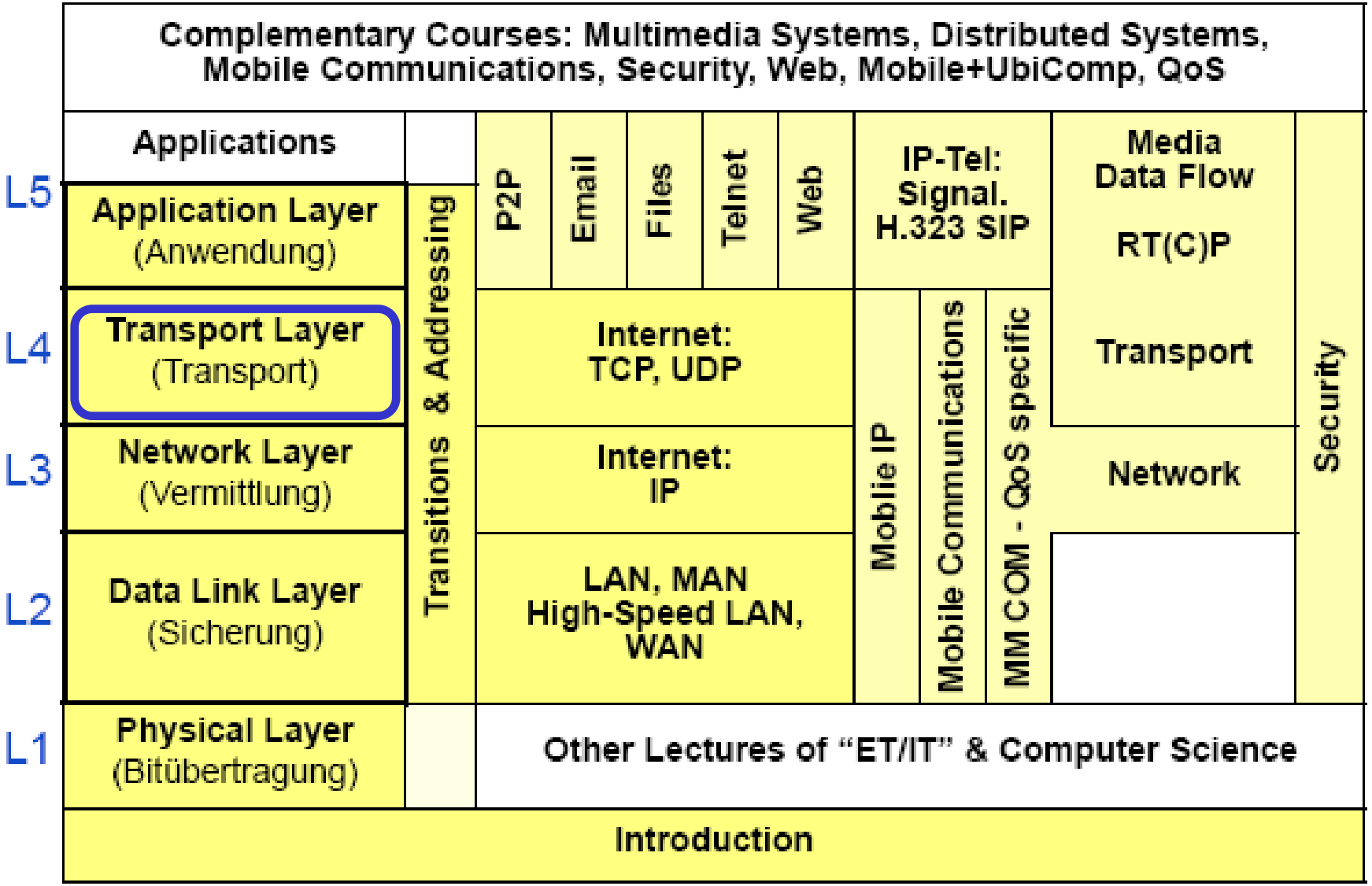
Computer Networks I

Transport Layer

Prof. Dr.-Ing. **Lars Wolf**

IBR, TU Braunschweig
Mühlenpfordtstr. 23, D-38106 Braunschweig, Germany,
Email: wolf@ibr.cs.tu-bs.de

Scope


Complementary Courses: Multimedia Systems, Distributed Systems, Mobile Communications, Security, Web, Mobile+UbiComp, QoS
Applications
L5
Application Layer (Anwendung)
L4
Transport Layer (Transport)
L3
Network Layer (Vermittlung)
L2
Data Link Layer (Sicherung)
L1
Physical Layer (Bitübertragung)
Transitions & Addressing
P2P
Email
Files
Telnet
Web
IP-Tel: Signal. H.323 SIP
Media Data Flow
RT(C)P
Transport
Internet: TCP, UDP
Internet: IP
Network
LAN, MAN High-Speed LAN, WAN
Moblie IP
Mobile Communications
MM COM - QoS specific
Security
Other Lectures of "ET/IT" & Computer Science
Introduction

Overview

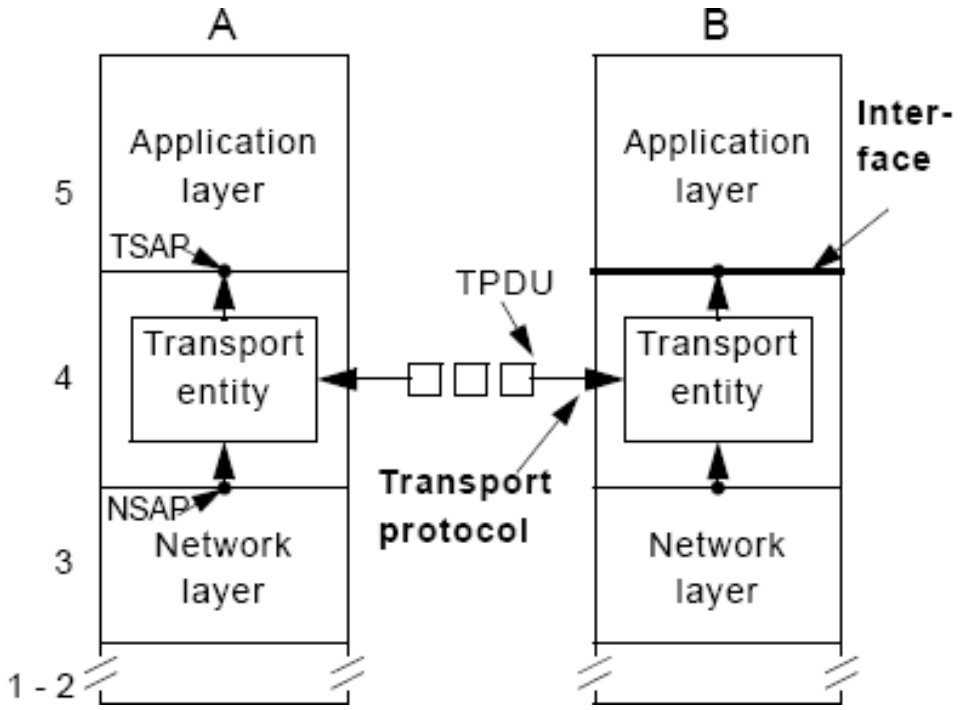
1 Transport Layer Function

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 (lower layers) used

1.1 Transport Service

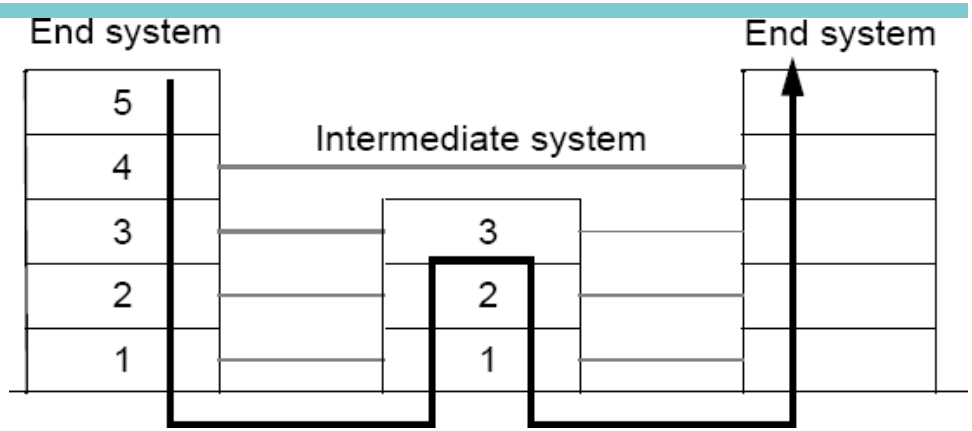


Connection oriented service
- 3 phases: connection set-up, data transfer, disconnect

Connectionless service
- transfer of isolated units

Realization: transport entity
- software and/or hardware?
- software part usually contained within the kernel (process, library)

Transport Service



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

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

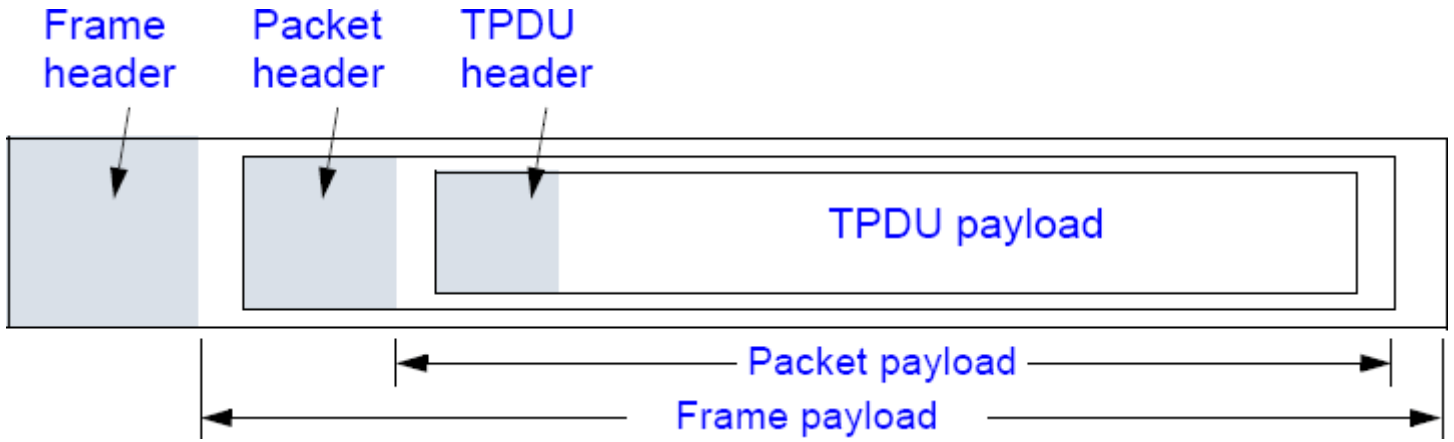
Transport Service: Terminology

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`

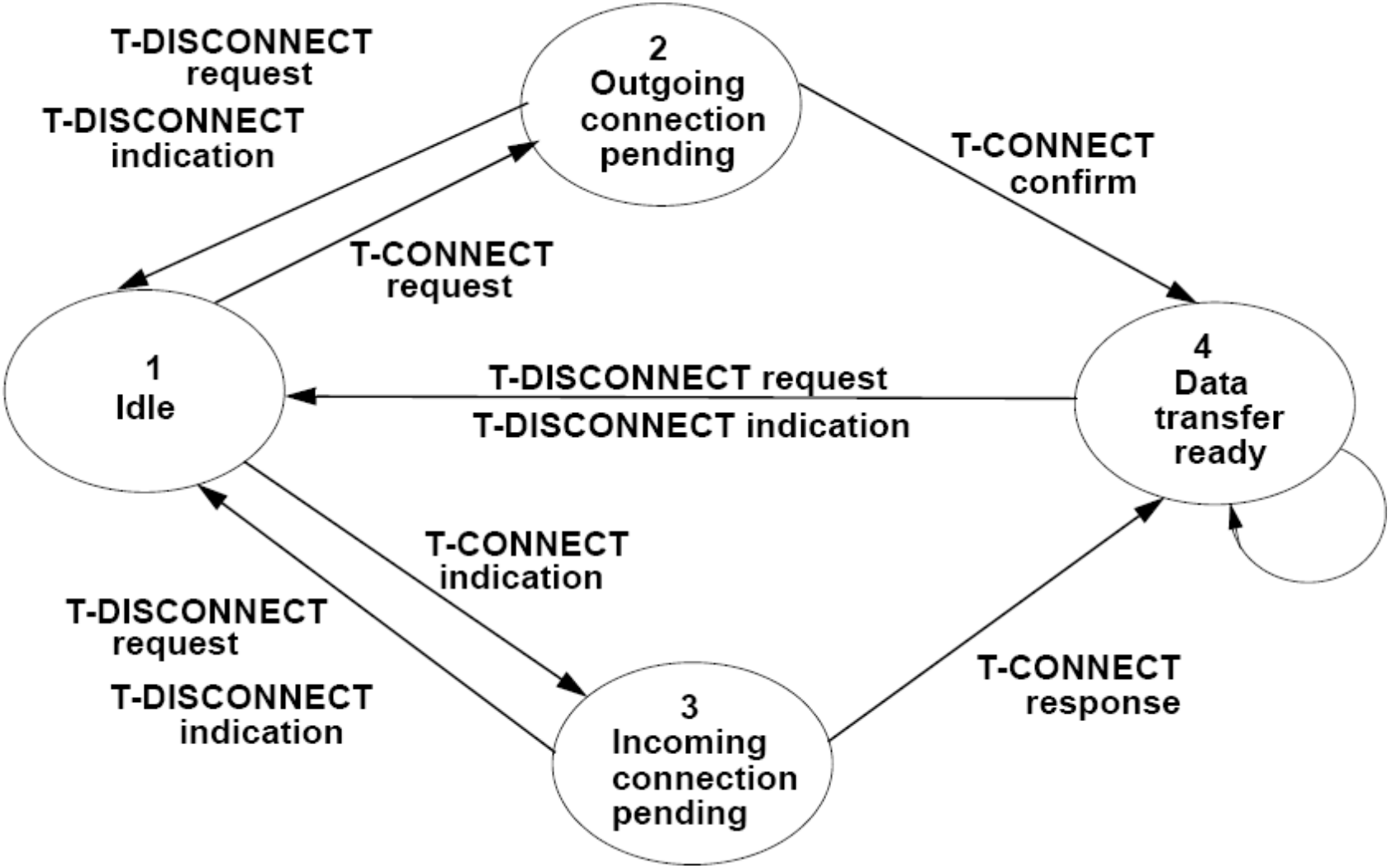
Nesting of TDPUs, packets, and frames:



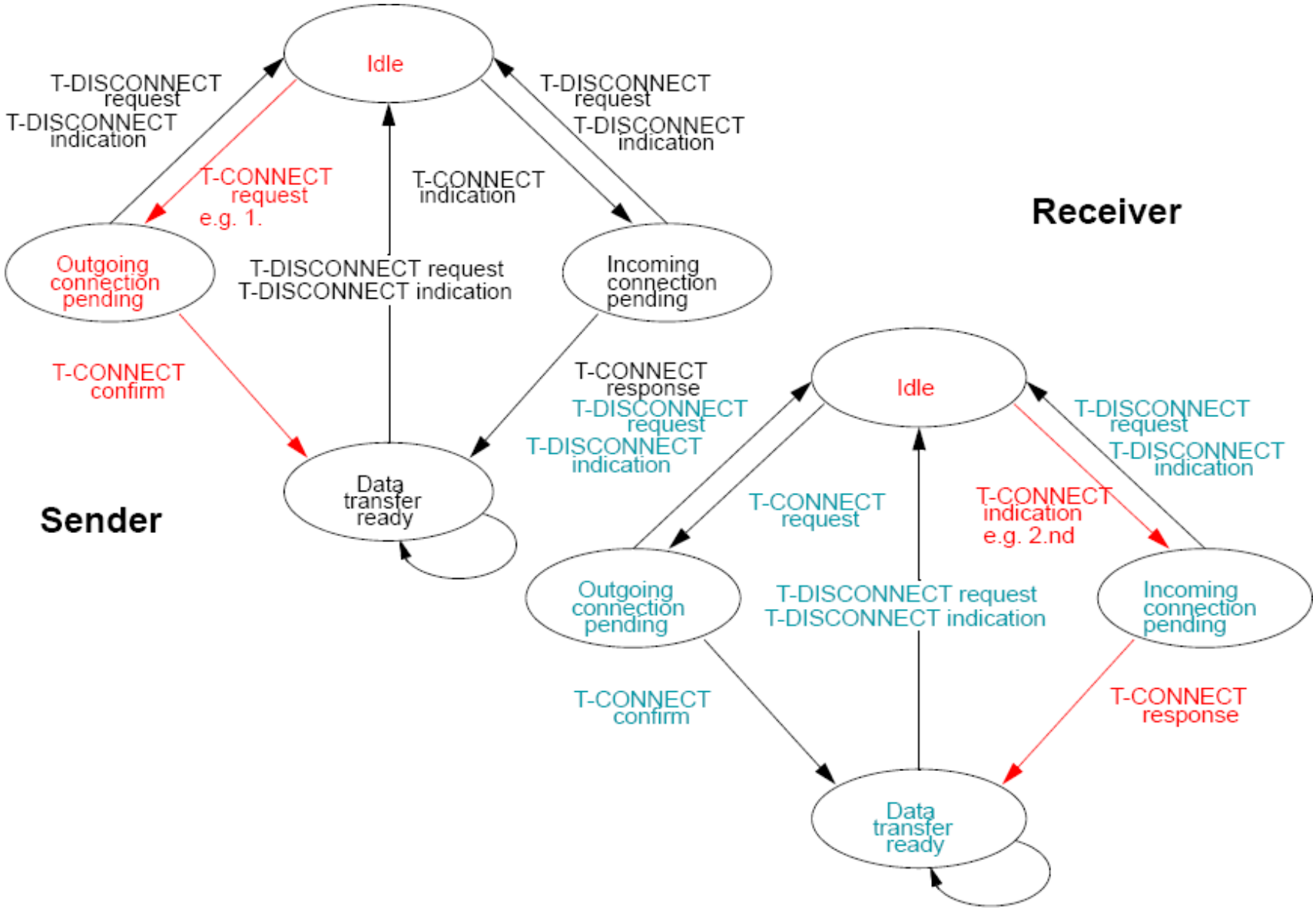
1.2 Connection-Oriented Service: State Transition Diagr.

Example: ISO-OSI nomenclature

- state transition diagram



Connection-Oriented Service: State Transition Diagr.



1.3 Transport Services 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

Transport Service Primitives: More Practical

Berkeley Socket Primitives:

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

2 Elements of Transport Protocols

Transport service is implemented

- by transport protocol used between the two transport entities

Several issues have to be addressed such as:

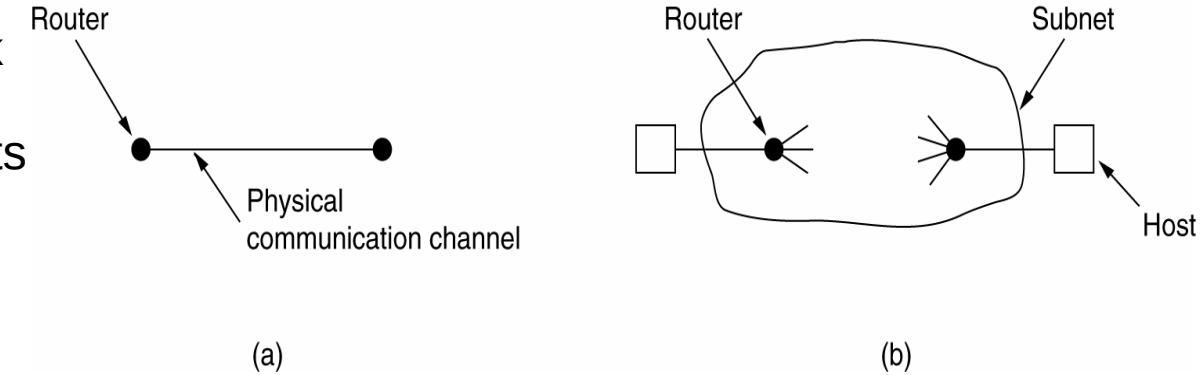
- Addressing
- Connection Establishment
- Connection Release
- Flow Control and Buffering
- Multiplexing
- Crash Recovery

Elements of Transport Protocols

Transport protocols resemble somehow data link protocols:

- but significant differences exist

Transport and data link protocols operate in different environments



Addressing:

- DL: outgoing line of a router specifies particular router
- TL: explicit addressing of destinations is required

Connection establishment:

- DL: peer is always there
- TL: more different situations

Storage capacity in the subnet:

- DL: neglectable
- TL: serious problem, packet may be stored somewhere and delivered later

Buffering and flow control:

- DL: few outgoing lines allow simple buffer allocation schemes
- TL: potentially large, dynamically #conn. requires flexible schemes

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)

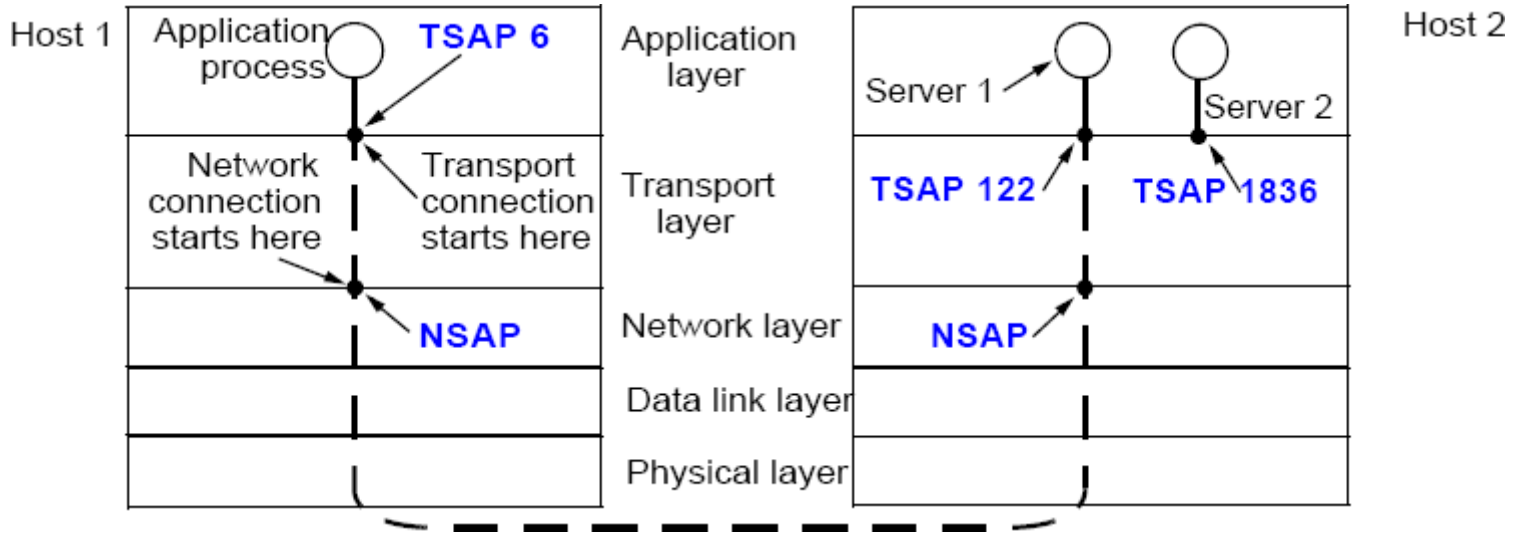
Define transport addresses:

- 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:

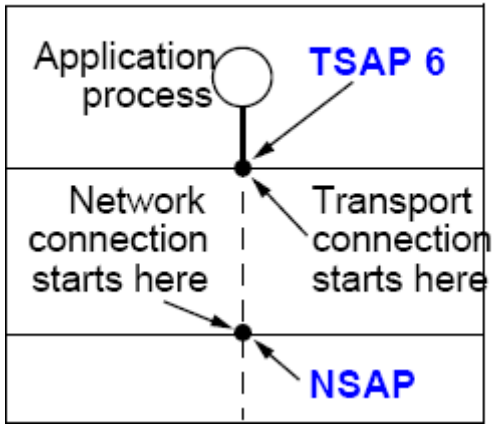


Steps

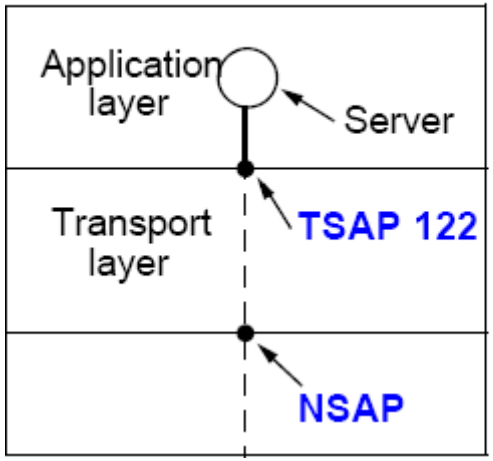
In general

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

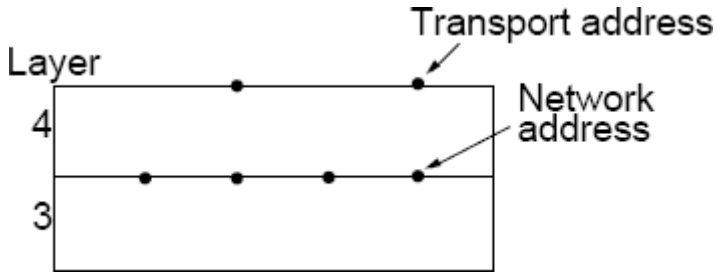


Host 2: Receiver - Server



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

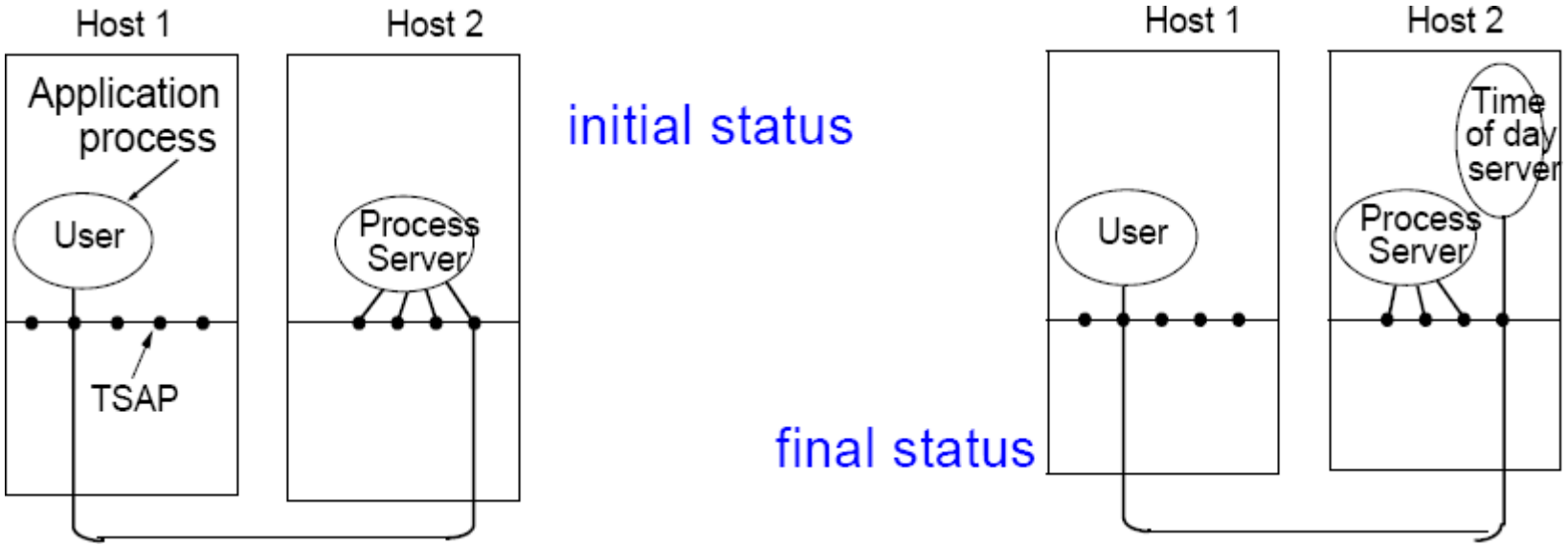
example: service ’time of day’ (port 13)

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 to have seldomly used servers active and listening

Determination of Appropriate Service Provider TSAP

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. 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 (layer 3!!) ?

- 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

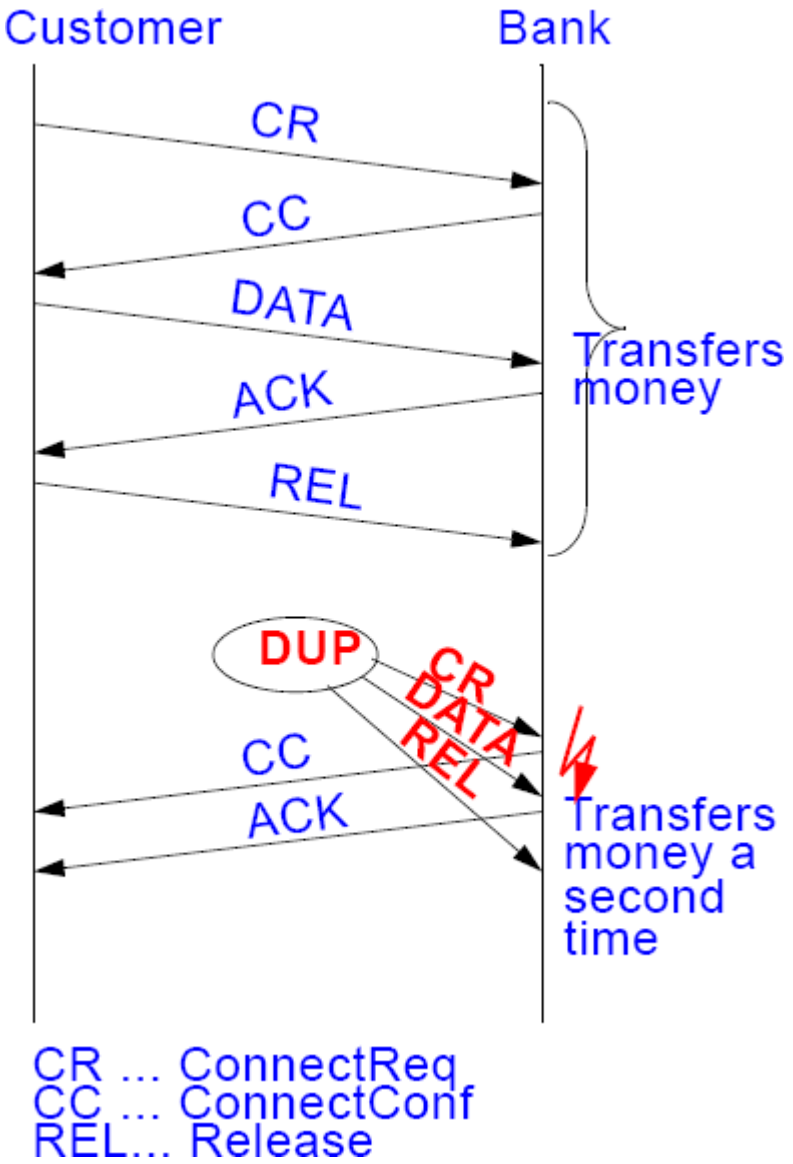
Example: Duplicates

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



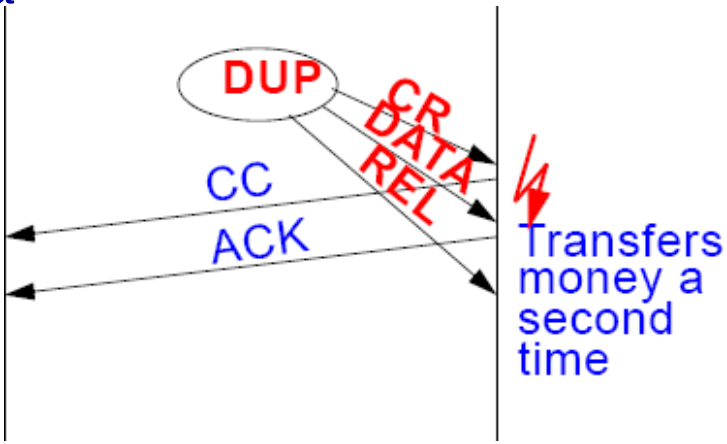
Duplicates – Description of Problematic Issues

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
     - 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

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

Duplicates: Approach to Limit Packet Lifetime

Enabling the above method

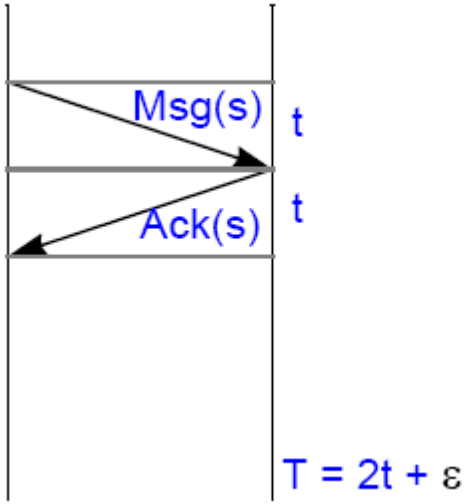
'3. to identify PDUs individually: individual sequential numbers for each PDU'

- 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).

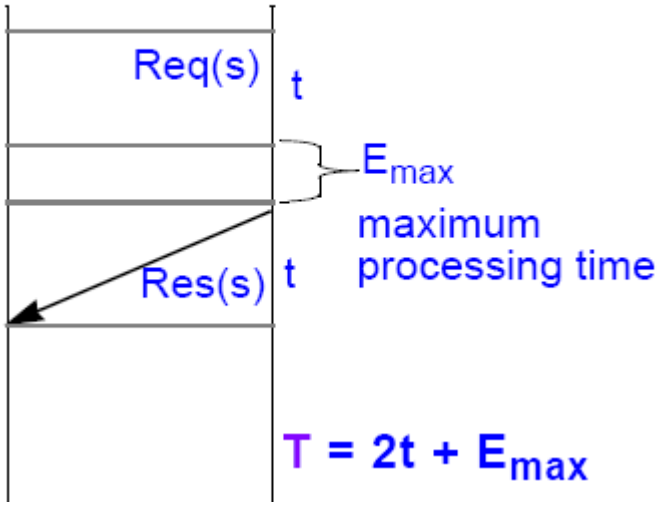
Mandatory prerequisite for this solution

- 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

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. Timestamp in each packet
   - packet exceeds maximum predefined / configured lifetime
     - => packet is discarded
   - *notice*: requires “consistent” network time (synchronized clocks)

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

TCP/IP term: Maximum Segment Lifetime (MSL)

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

4.2 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

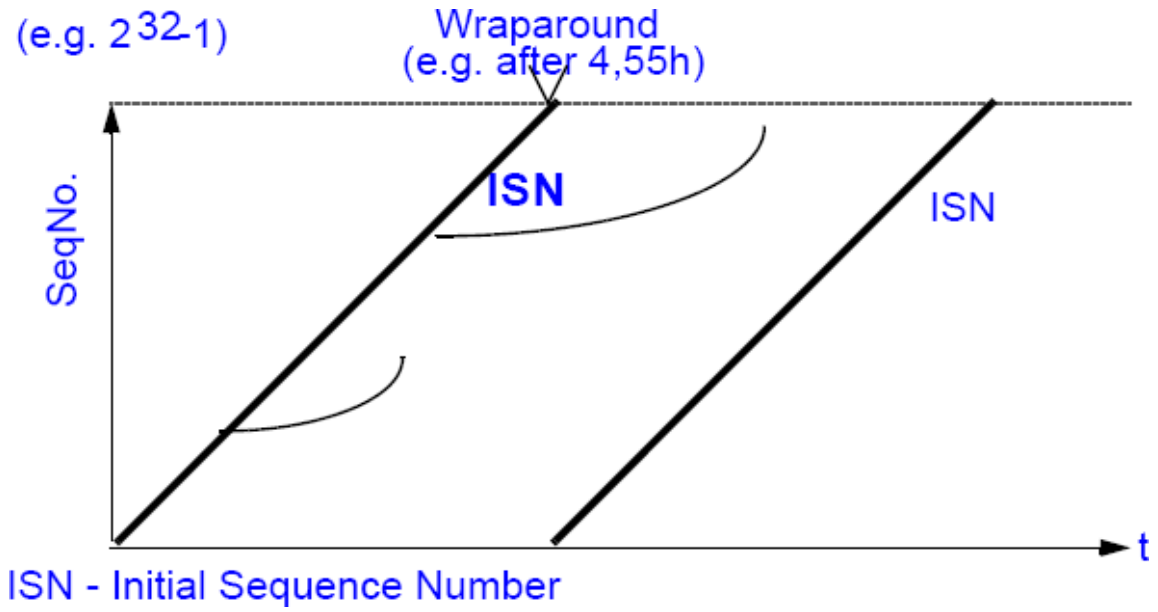
➔ resolves problems with duplicates within a single connection

- 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

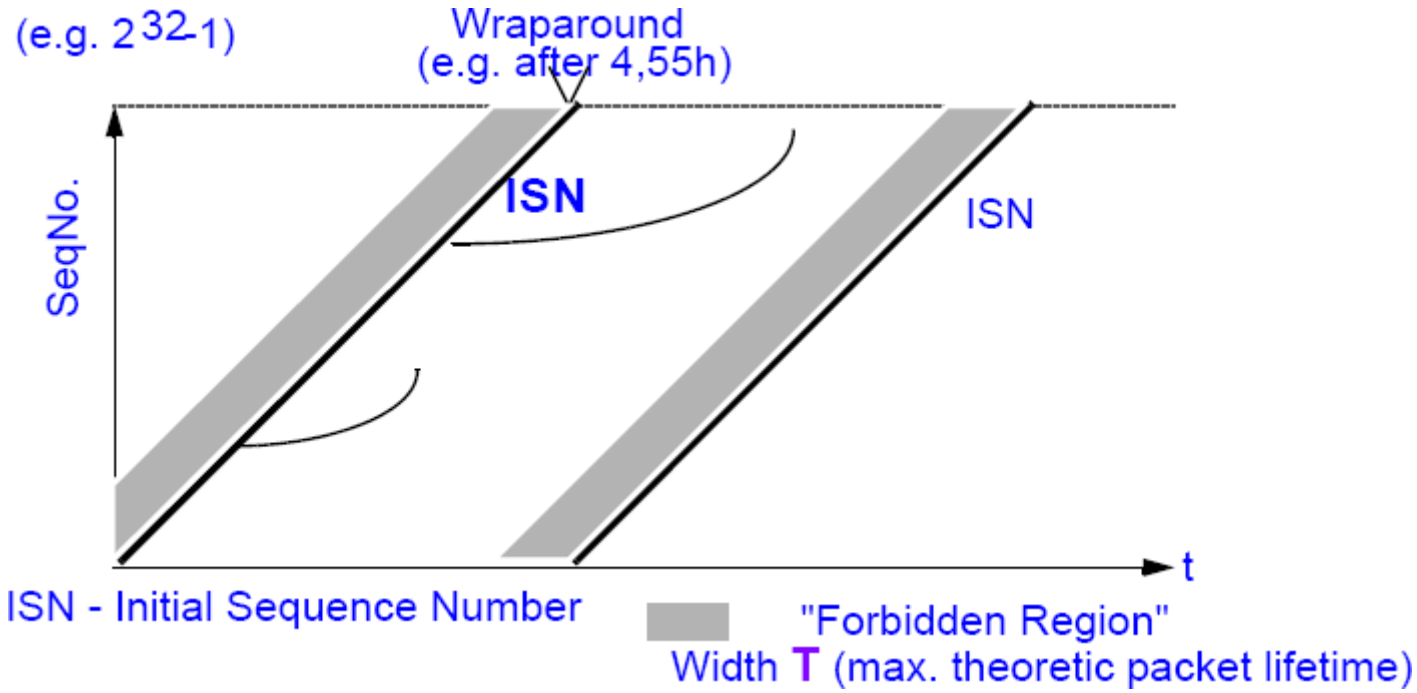
Initial Sequence Number Allocation and Handling of Consecutive Connections



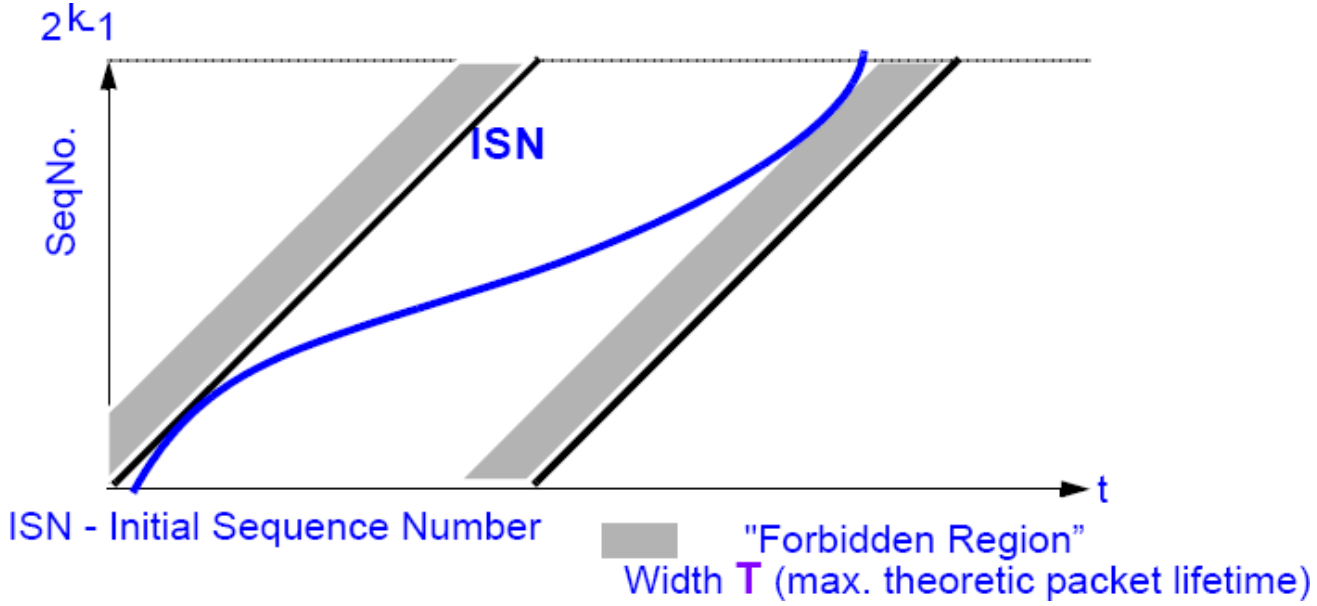
Method

- endsystems
  - timer continues to run at switch-off / system crash
- allocation of initial SeqNo (ISN) depends on timestamp
  - linear curve in theory,
    staircase curve in reality because of discrete clock ticks)
- 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



Initial Sequence Number Allocation and Handling of Consecutive Connections



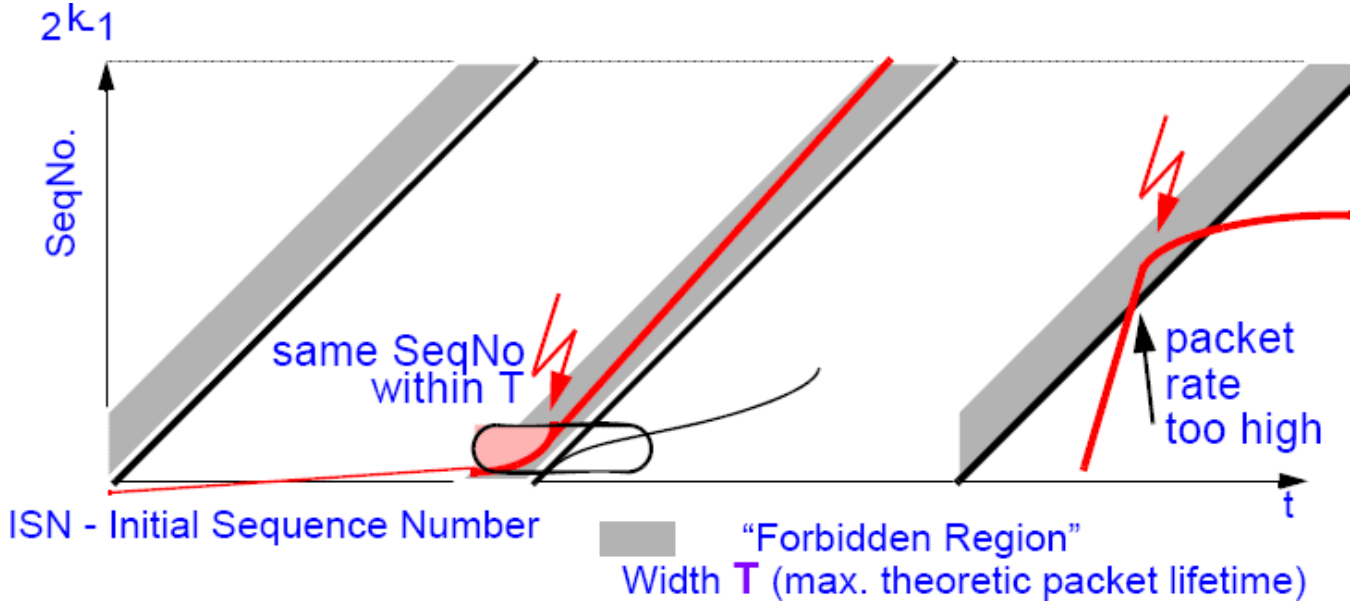
No problem, if

- "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

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

Note:

- 32 bit sequence numbers considered as sufficient when designing TCP/IP (consider technology available at that time)
- sequence number range exploitation
  - 10 Mbit/sec in ca. 57 min
  - 1 Gbit/sec in ca. 17 sec

➔ Using timestamps in

- "TCP extensions for high speed paths"
  - PAWS "Protect Against Wrapped Sequence Numbers"

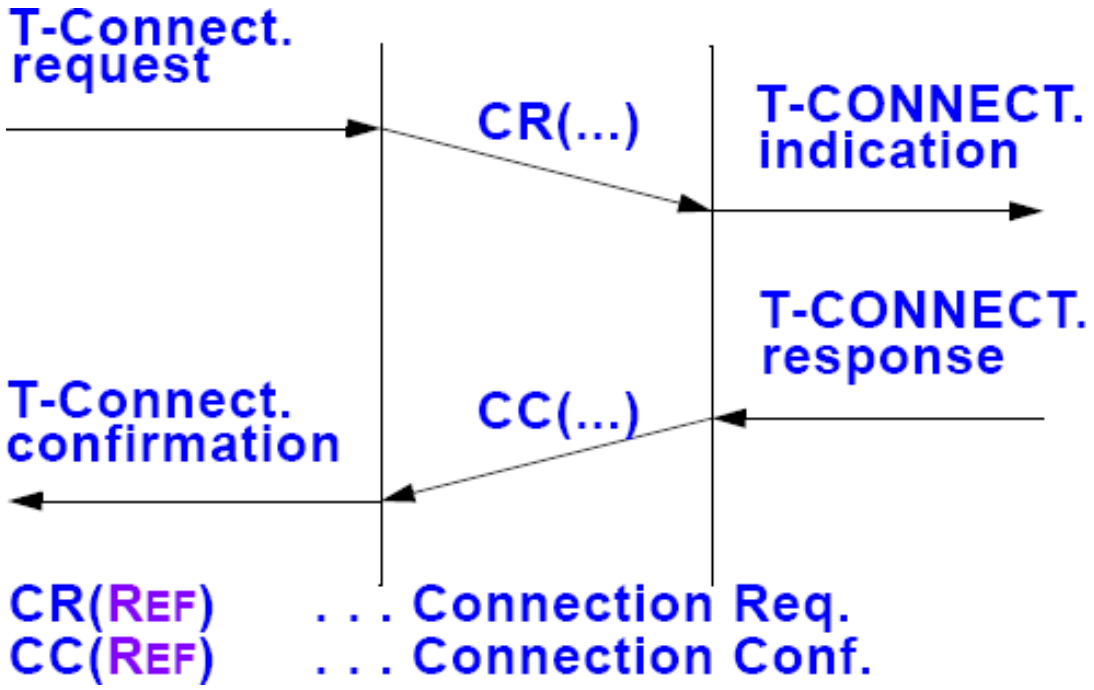
Further literature in addition to Tanenbaum

- 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

5 Reliable Connection Establishment

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

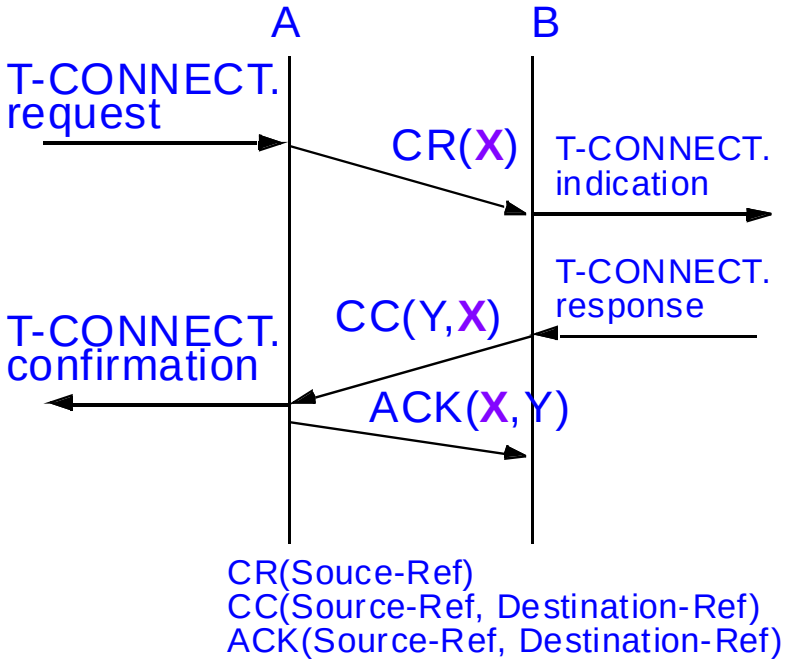
- by simple protocol
  - with approach using 2 messages (2 phases)
    - problems may occur due to delayed duplicates
    - compare with previous example (bank transaction)



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

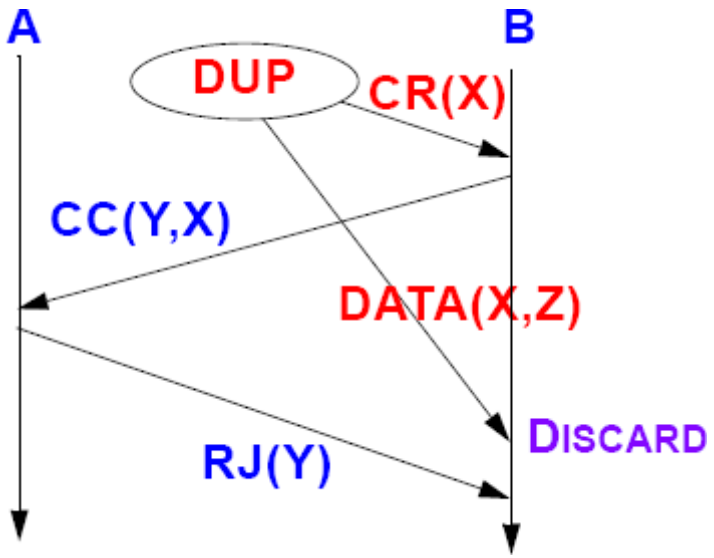


3. Acknowledgment
   - initiator (A) acknowledges
     - sequence numbers **X**, **Y** (as received before)
   - only after receiving a valid ACK, receiver (B) accepts data

Note:

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

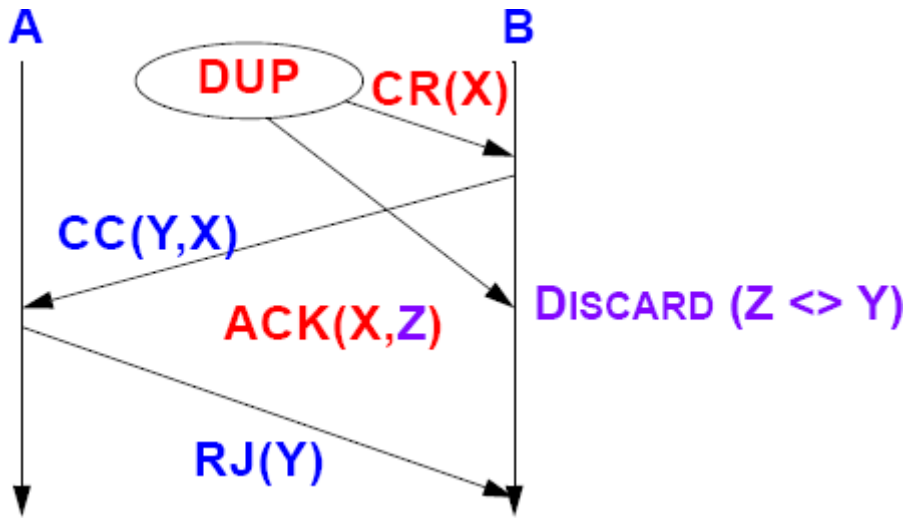
Three Way Handshake Protocol: Results



CR and data duplicate

- duplicated data is discarded

Three Way Handshake Protocol: Results



Connect Request (CR) Duplicate
and Acknowledgment (ACK) Duplicate

- ACK (X,Z) discarded because
  - ACK (X, Y) expected
  - ACK (X, Z) received,
    - Y != Z will be ensured by B under the 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

Variant: symmetric disconnect

- disconnect takes place separately for each direction

Variant: asymmetric disconnect

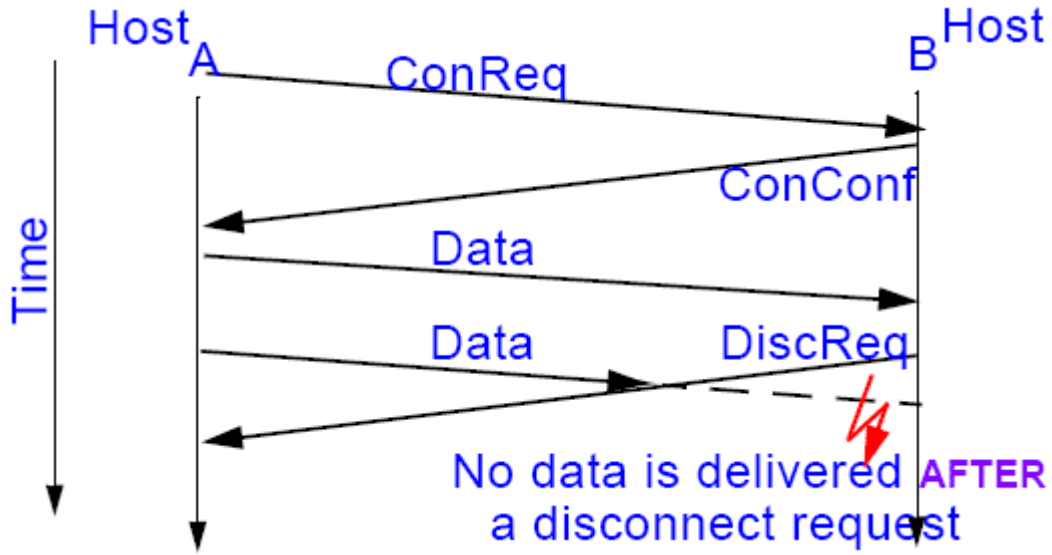
- disconnect in one direction implies disconnect for both “directions”

6.1 Asymmetric Disconnect

Approach

- disconnect in one direction implies disconnect for both "directions"
  - analog to telephone
- may result in data losses

Example



➔ approach for a solution:

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

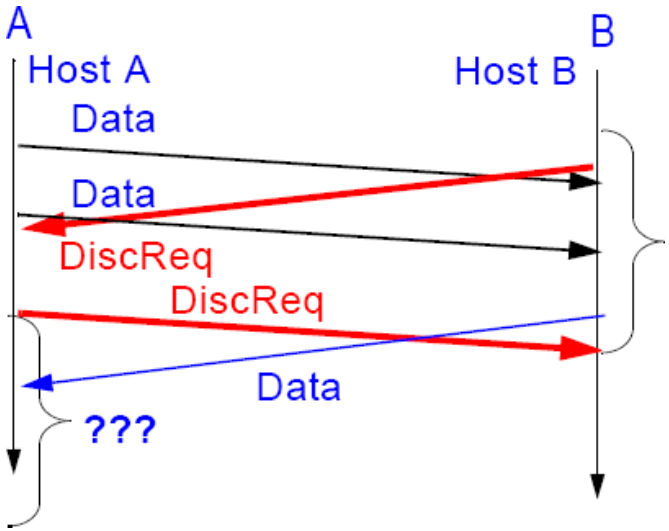
6.2 Symmetric Disconnect

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

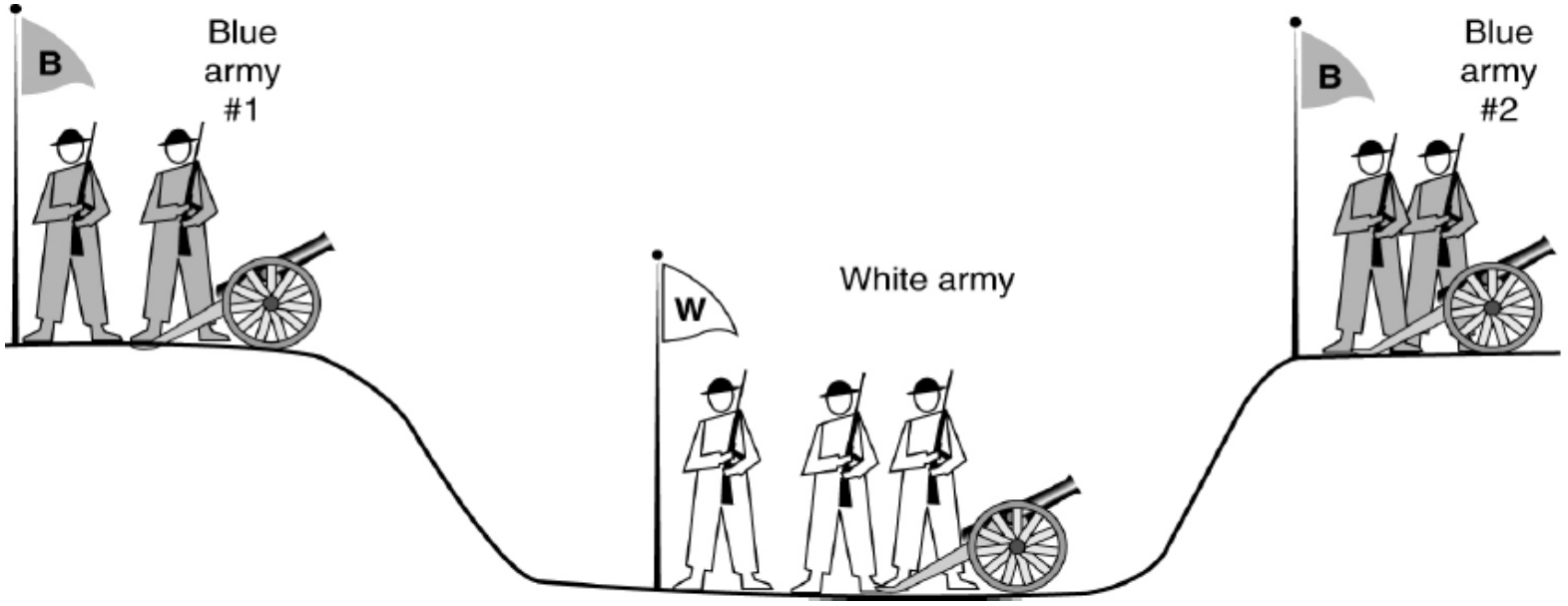
- host can continue to receive data even after it has sent DISCONNECT TPDU
  - both sides have to issue a disconnect
  - host received DISC. → stops to send data
  - host sent DISC. → may continue to receive data

Properties

- works when each process has fixed amount of data to send and knows when it has sent it (and how much data will arrive)
- not as obvious as considered if something can go wrong
  - if host does not know about data to be received after having sent a disconnect

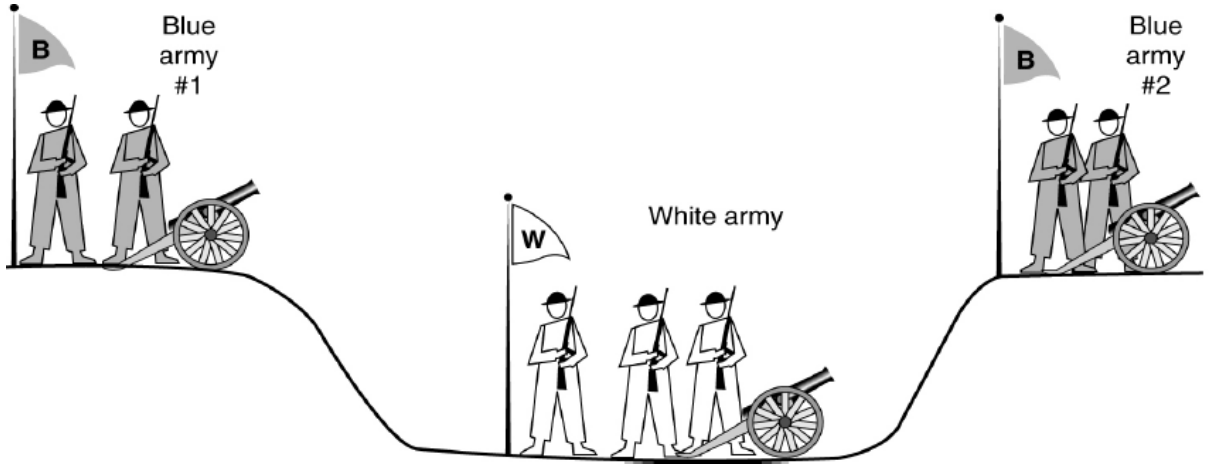


Two-Army Problem



- 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

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

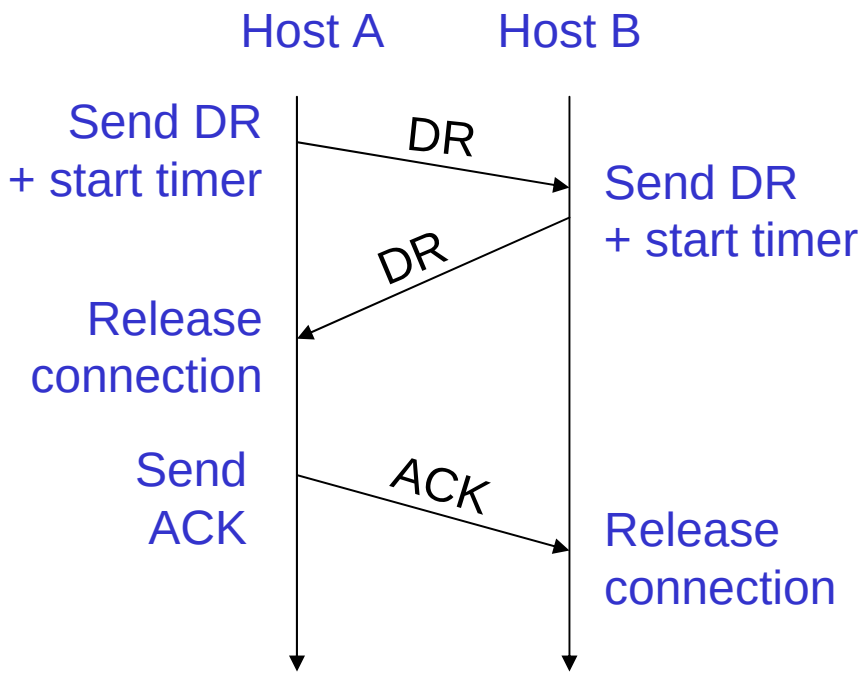
- blue1 ---> blue2: let us attack at 11:11
- blue1 <--- blue2: OK
- blue1 ---> blue2: OK
- blue1 <--- blue2: OK

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

Disconnect with Three-Way Handshake

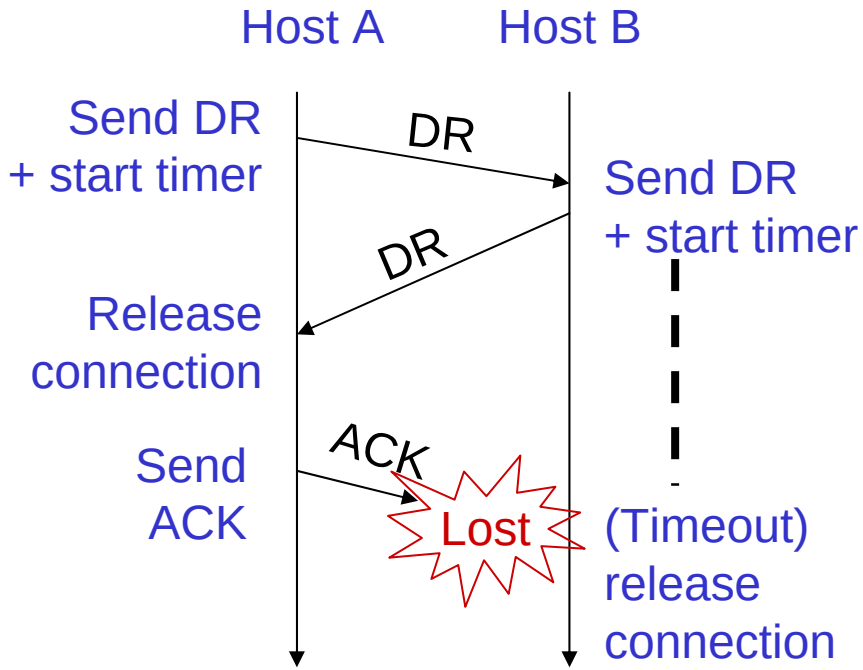
Regular disconnect with Three-Way handshake



Disconnect with Three-Way Handshake

Last acknowledgment lost

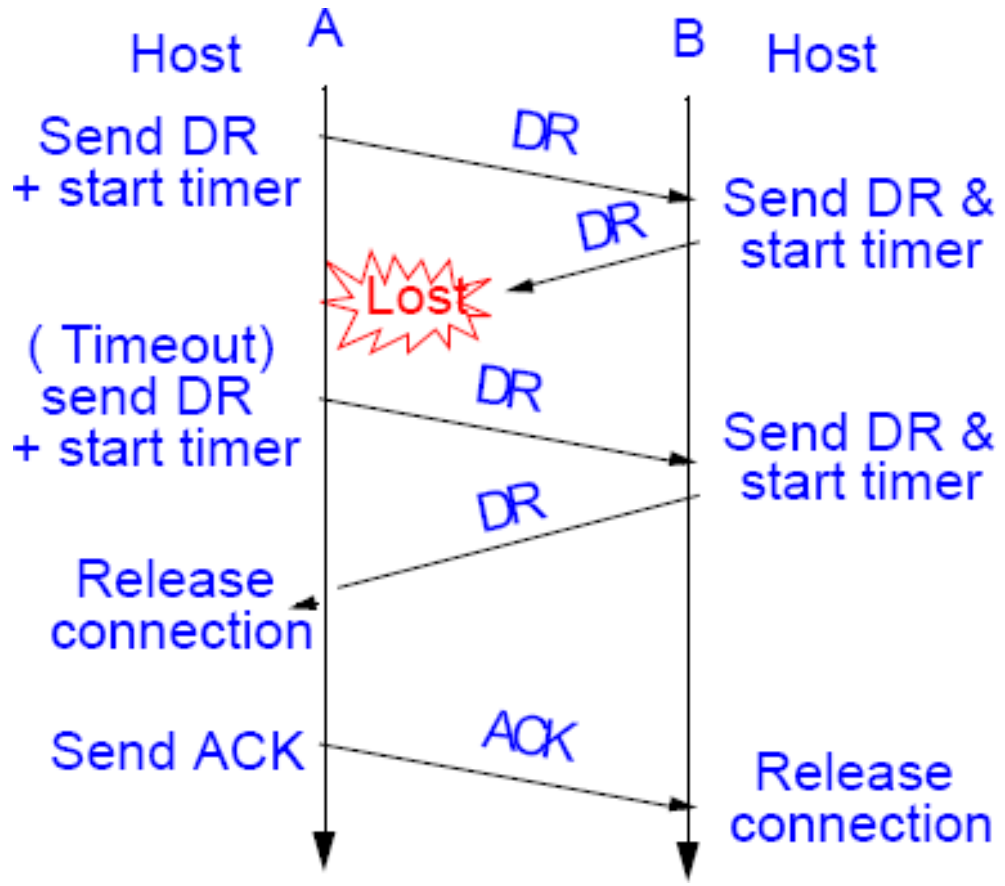
- timer disconnects from host 2
- therefore no further problems



Disconnect with Three-Way Handshake

Second “disconnect request” lost

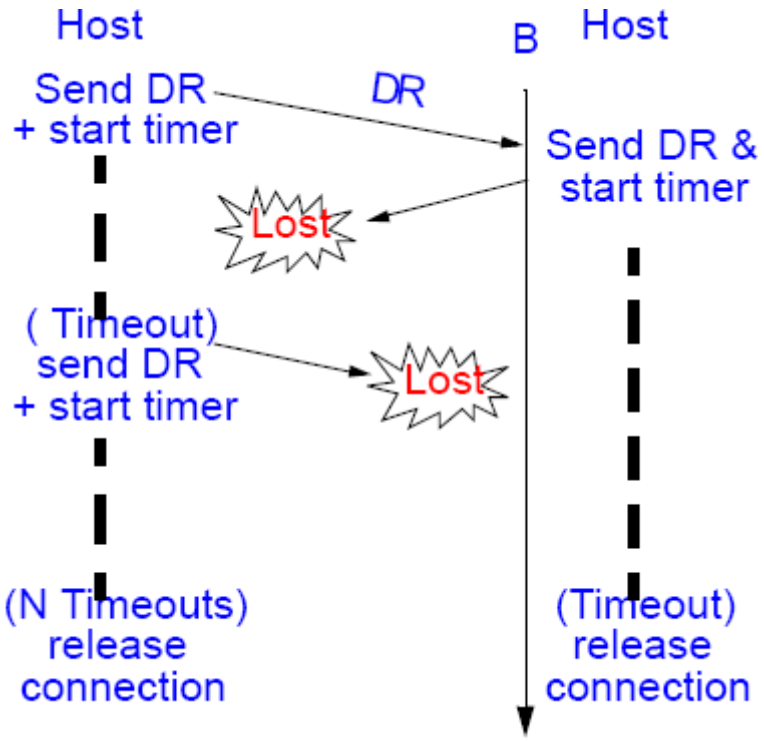
- repeat to send “disconnect release”
  - because response was an unexpected DR and ACK
- loss is repaired
  - otherwise procedure as described using timer



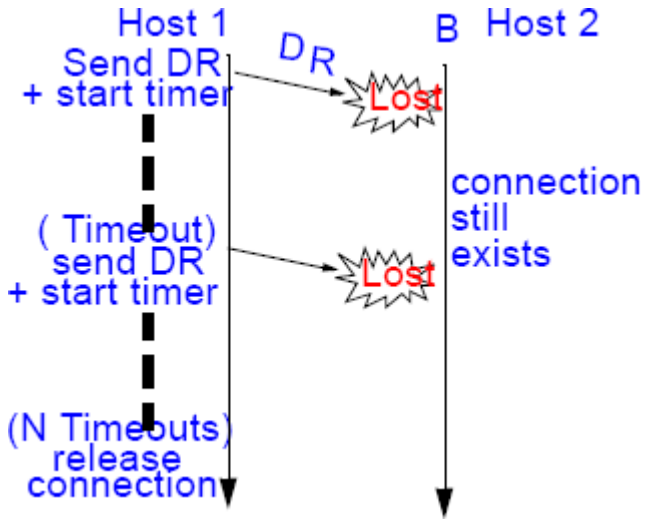
Disconnect with Three-Way Handshake

Second and all further “disconnect releases” lost

- disconnects by timeout



Disconnect with Three-Way Handshake



All “disconnect releases” lost

- resulting problem:
  - Host1 disconnects,
    but Host2 retains inconsistent information: “half-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)
      (but these might be lost as well ...)

7 Flow Control on Transport Layer

Joint characteristics (flow control on data link layer)

- fast sender shall not flood slow receiver
- sender has to store all unacknowledged packets

Differences (flow control on data link layer)

- L2-DLL: router serves few “bitpipes”
- 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 #transport-connections
- poor buffer utilization for low throughput connections

i.e.

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

➔ poor for intermittent (and potentially 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 allocation 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 memory utilization

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

i.e.

➔ good for intemittent (and bursty) traffic with low throughput

➔ poor for traffic with high throughput

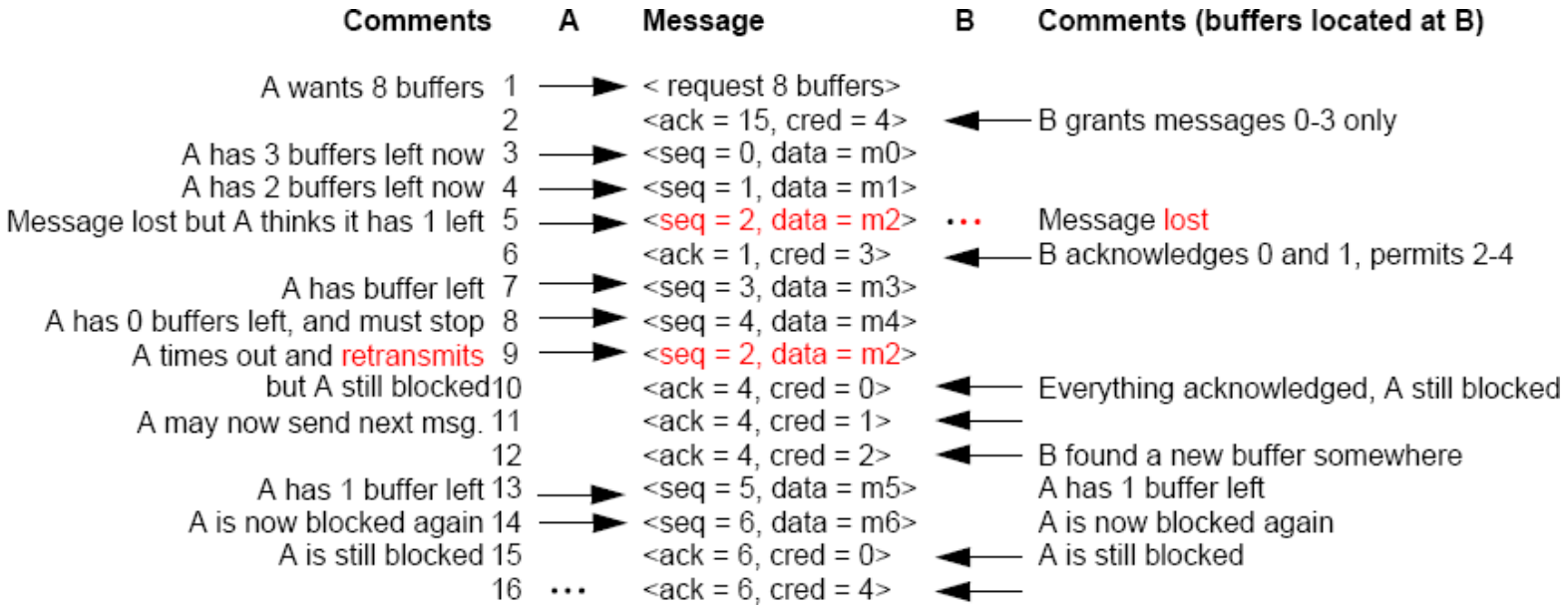
7.3 Credit Mechanism

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



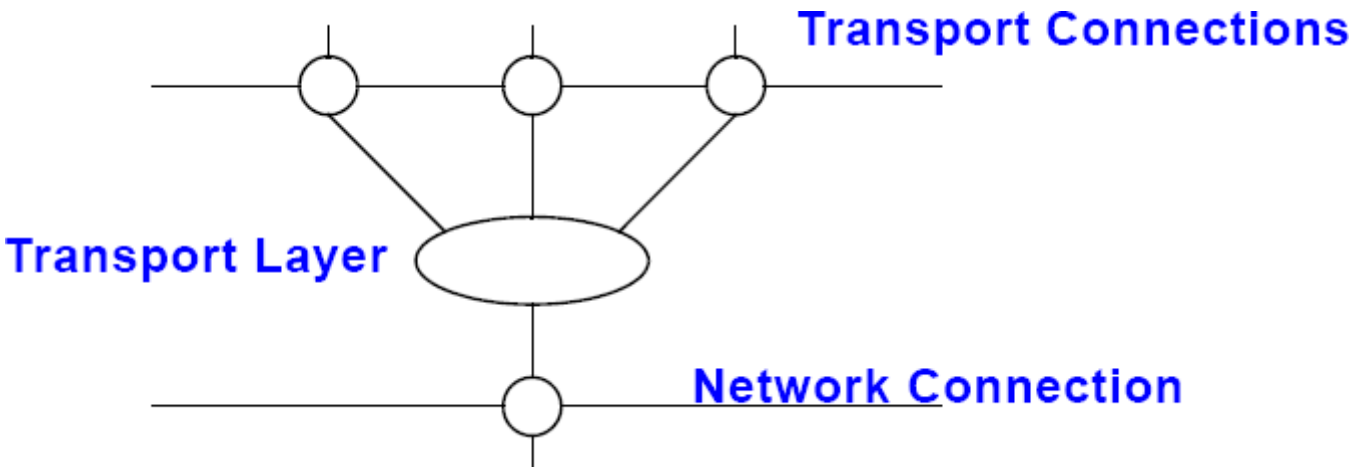
Example: with dynamic buffer allocation

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

Dynamic adjustment to

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

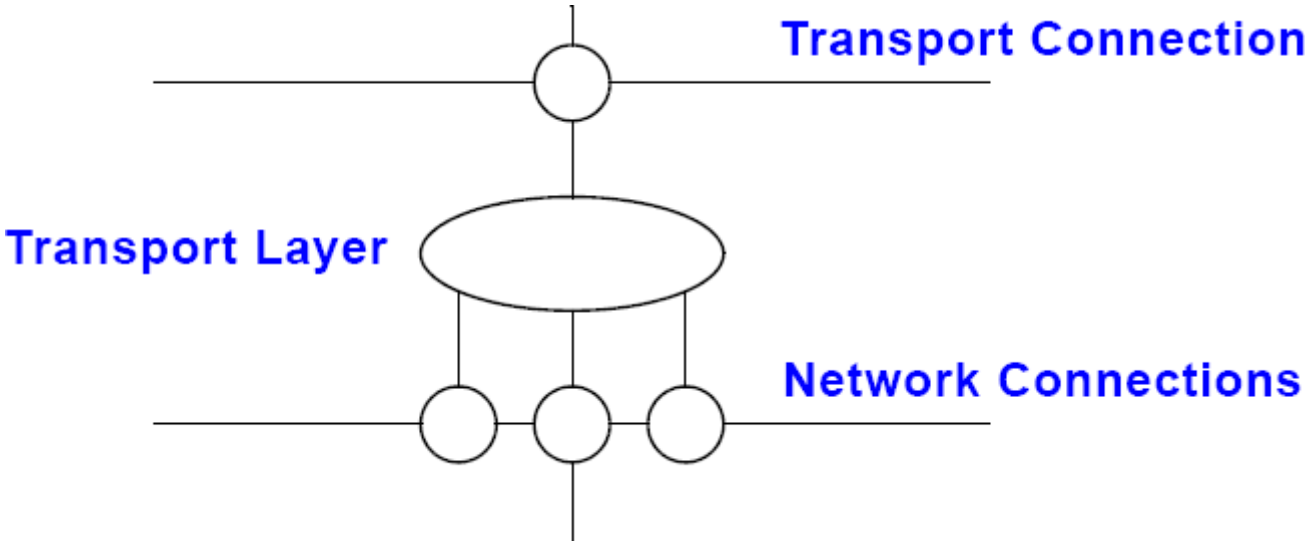
8 Multiplexing / Demupltiplexing



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

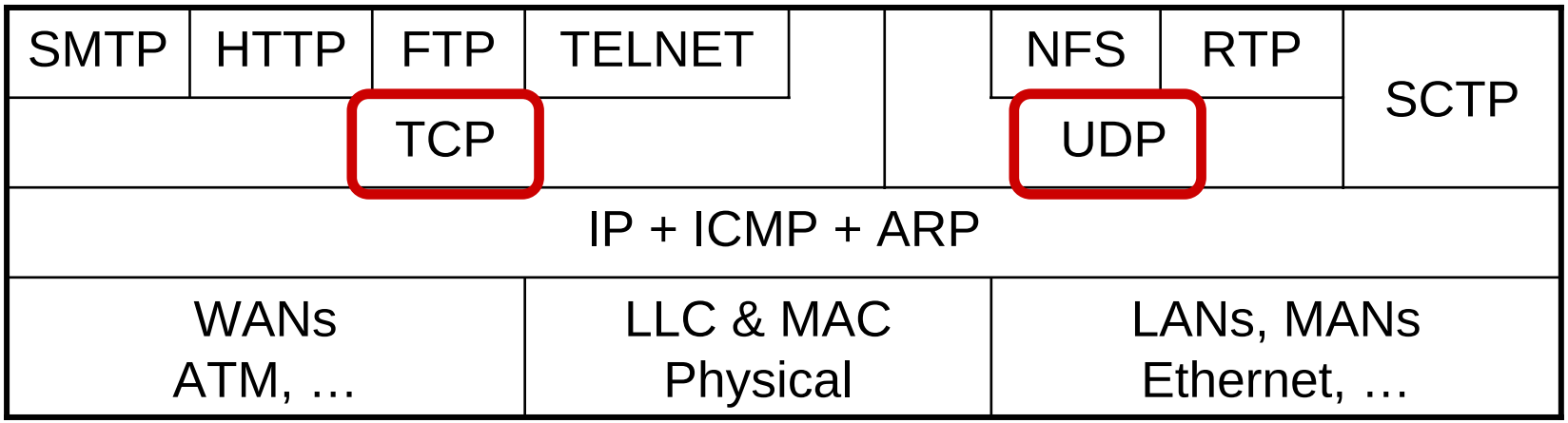
Splitting function

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

Comment

- also known as “upward” multiplexing

9 Some Familiar Internet Protocols



Here only a VERY short introduction to UDP and TCP can be given.

Further (many) details about UDP, TCP, and also SCTP are given in Computer Networks II.

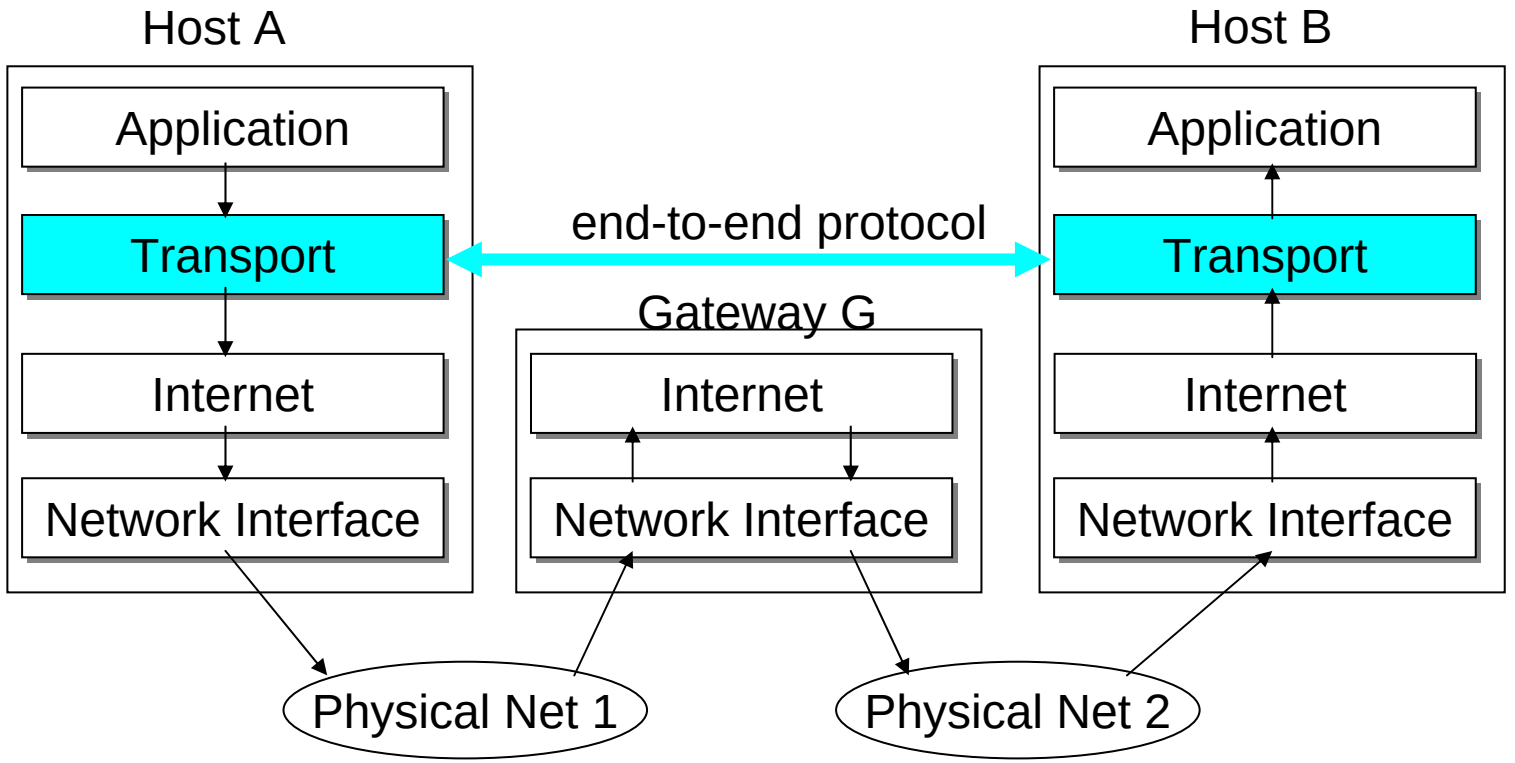
Internet Transport Layer (in General & Addressing)

Lowest level end-to-end protocol

- header generated by sender is interpreted only by destination
- routers / gateways view transport header as part of the payload

Adds extra functionality to the best effort packet delivery service provided by IP

- makes up for shortcomings of core network



9.1 Some Functions of Transport Protocols

Multiplexing/demultiplexing data for multiple applications
- uses “port’’ abstraction

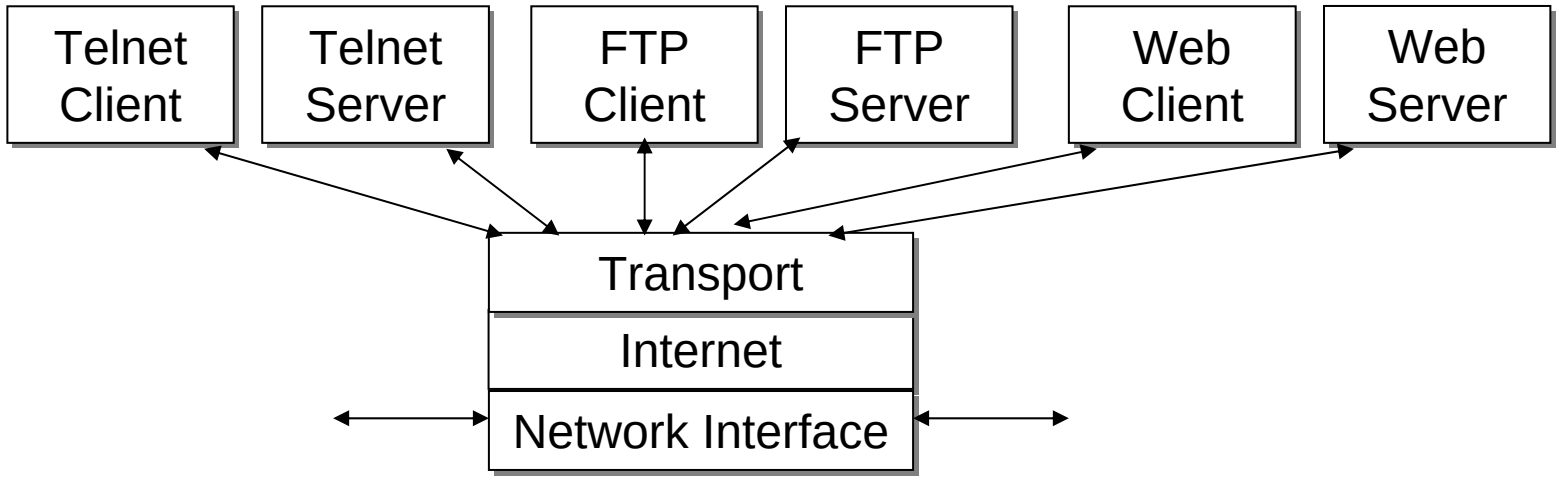
Connection establishment
- logical end-to-end connection

Error control
- hides unreliability of network layer from applications
- some types of errors:
  - corruption, loss, duplication, reordering

End-to-end flow control
- to avoid flooding the receiver

Congestion control
- to avoid flooding the network

Transport Layer: End-to-End Communications



- communication between applications required
- applications communicate
  - locally by interprocess communication
  - between systems via TRANSPORT SERVICES

Transport layer

- interprocess end-to-end communication via communication networks

Internet protocol IP

- enables only end system - to - end system communication

9.2 UDP – User Datagram Protocol

Specification:
- RFC 768

UDP is a simple transport protocol
- unreliable
- connectionless
- message-oriented

UDP is mostly IP with a short transport header
- source and destination port
- ports allow for dispatching of messages to receiver process

Characteristics
- no flow control
  - application may transmit
    - as fast as it can/wants to and
    - as fast as the network permits
- no error control or retransmission
  - no guarantee about packet sequencing
  - packet delivery to receiver not ensured
  - possibility of duplicated packets
- may be used with broadcast / multicast and streaming

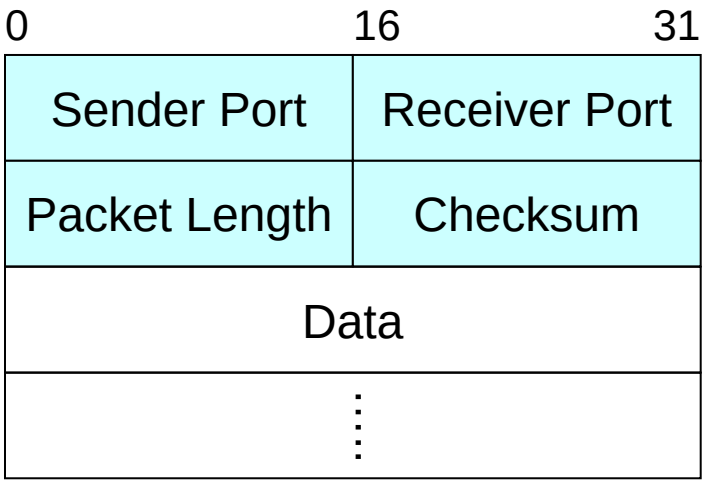
UDP: Message Format

Sender port
- 16 bit sender identification
- is optional
- use: response may be sent there

Receiver port
- receiver identification

Packet length
- in bytes (including UDP header)
- minimum: 8 (byte) i.e. header without data

Checksum
- of header and data for error detection
- use of checksum optional



UDP Header

9.3 TCP – Transmission Control Protocol

Motivation: network layer provides unreliable connectionless service

- packets and messages may be
  - duplicated, delivered in wrong order, faulty
- given such an unreliable service
  - each application would have to implement error detection and correction separately
- network or service can
  - impose packet length
  - define additional requirements to optimize data transmission
  - i.e. each application would have to be adapted separately
- ➔ do not reinvent the wheel for every application

➔TCP is the Internet transport protocol providing

- reliable end-to end byte stream over an unreliable internetwork

Specification:

- RFC 793: originally
- RFC 1122 and RFC 1323: errors corrected, enhancements implemented

TCP Features

Reliable, bidirectional, unstructured byte stream
- fully ordered, fully reliable

Point-to-point

Connections established & torn down

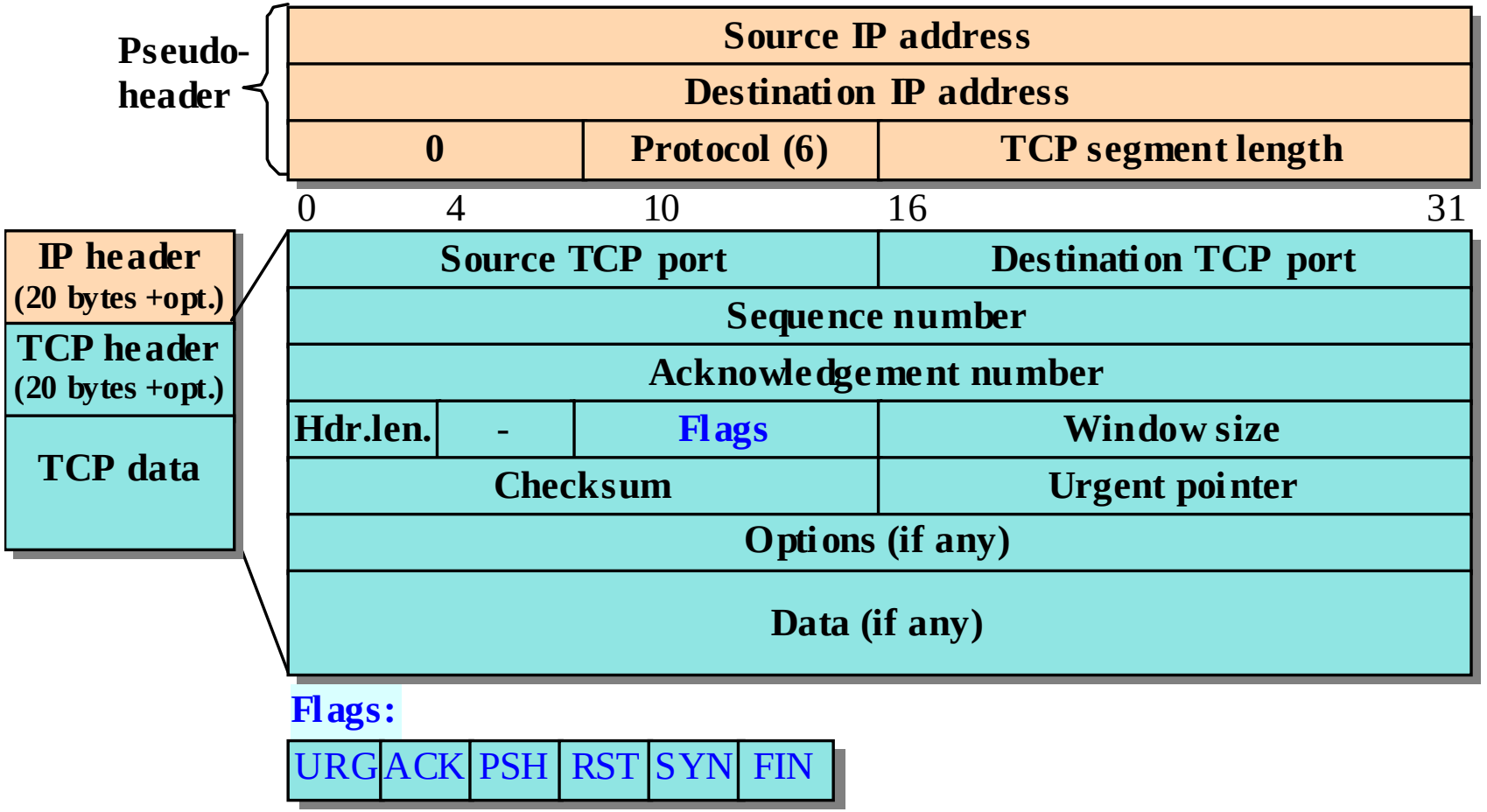
Multiplexing/ demultiplexing
- ports at both ends

Error control
- users see correct, ordered byte sequences

End-end flow control
- avoid to overwhelm any machine at either end

Congestion avoidance
- avoid to create traffic jam within network

TCP Segments



TCP Message Exchange