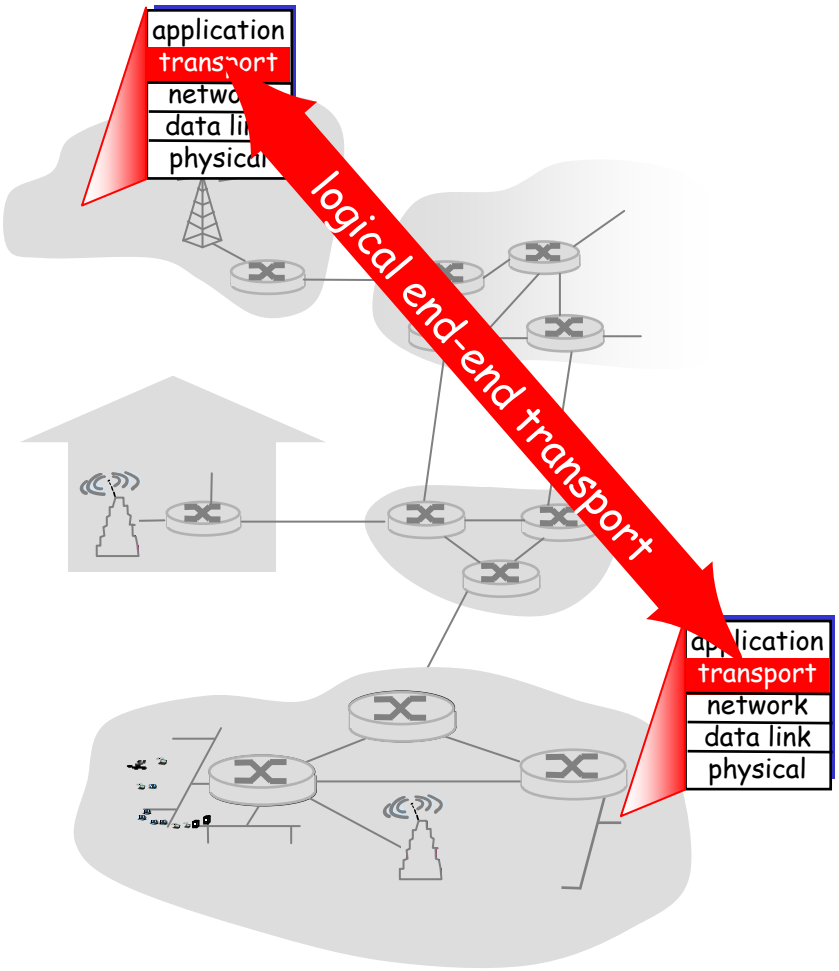
Transport services and protocols

- provide *logical communication* between app processes running on different hosts
- transport protocols run in end systems
  - send side: breaks app messages into segments, passes to network layer
  - rcv side: reassembles segments into messages, passes to app layer
- more than one transport protocol available to apps
  - Internet: TCP and UDP



Transport vs. network layer

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

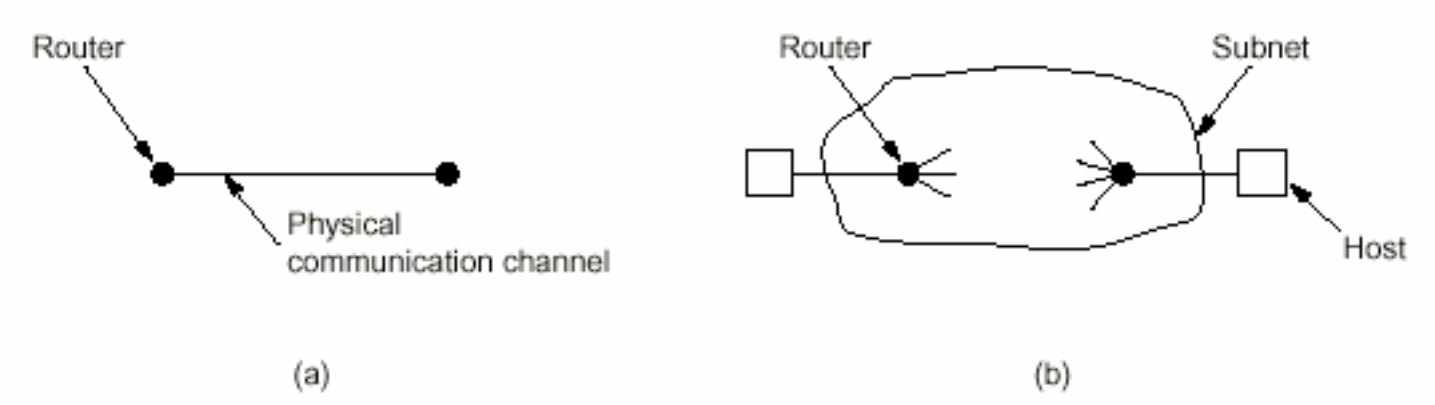
research office analogy:

12 researchers sending and receiving letters

- processes = researchers
- app messages = letters (in envelopes)
- host = institute
- transport protocol = executed by secretary
- network-layer protocol = executed by postal service

Transport Layer(4)

Different environments for data link and transport protocols, resp.



Implications for transport protocols:

- explicit addressing of destinations is required
- potential existence of storage capacity in the subnet
- connection establishment and release is more complicated
- larger and dynamically varying number of connections leading to more complex buffer maagement

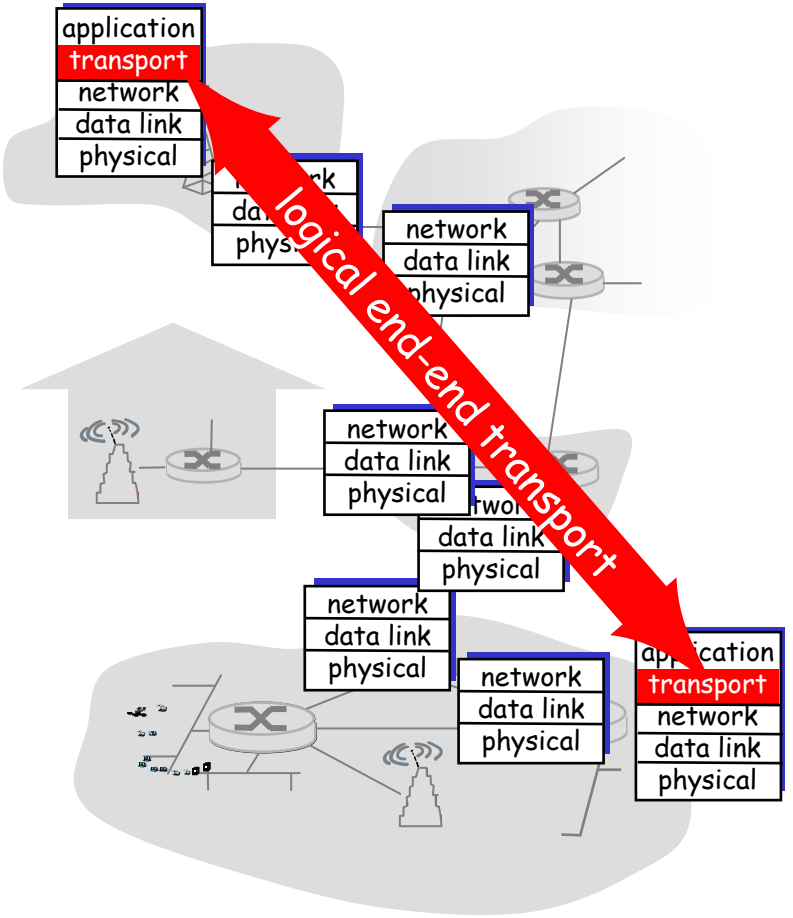
Transport Layer

Topics:

- understand principles behind transport layer services:
  - addressing
  - connection setup and release
  - flow control
  - congestion control
- learn about transport layer protocols in the Internet:
  - UDP: connectionless transport
  - TCP: connection-oriented transport
  - TCP congestion control

Internet transport-layer protocols

- reliable, in-order delivery (TCP)
  - congestion control
  - flow control
  - connection setup and release
- unreliable, unordered delivery: UDP
  - no-frills extension of “best-effort” IP
- services not available:
  - delay guarantees
  - bandwidth guarantees



Transport Layer(3)

Two principal alternatives:

- protocol is connection-oriented (setup required) and reliable, like the telephone system (TCP)
- protocol is connectionless (no setup required) and unreliable, like the postal system (UDP)

Two basic philosophies for organizing the layer:

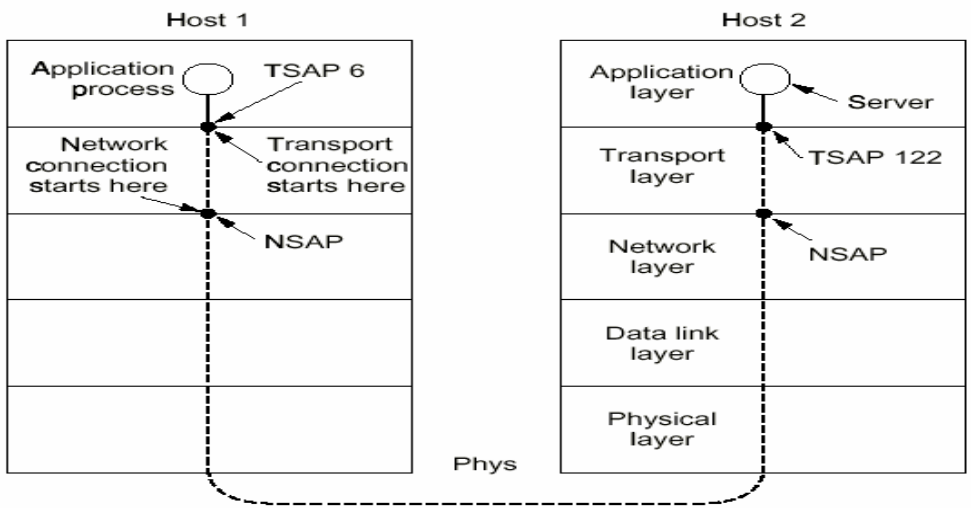
- virtual circuits (identity of the connection)
- datagram (identity of the independent packet)

Comparison of datagram and virtual circuit protocols

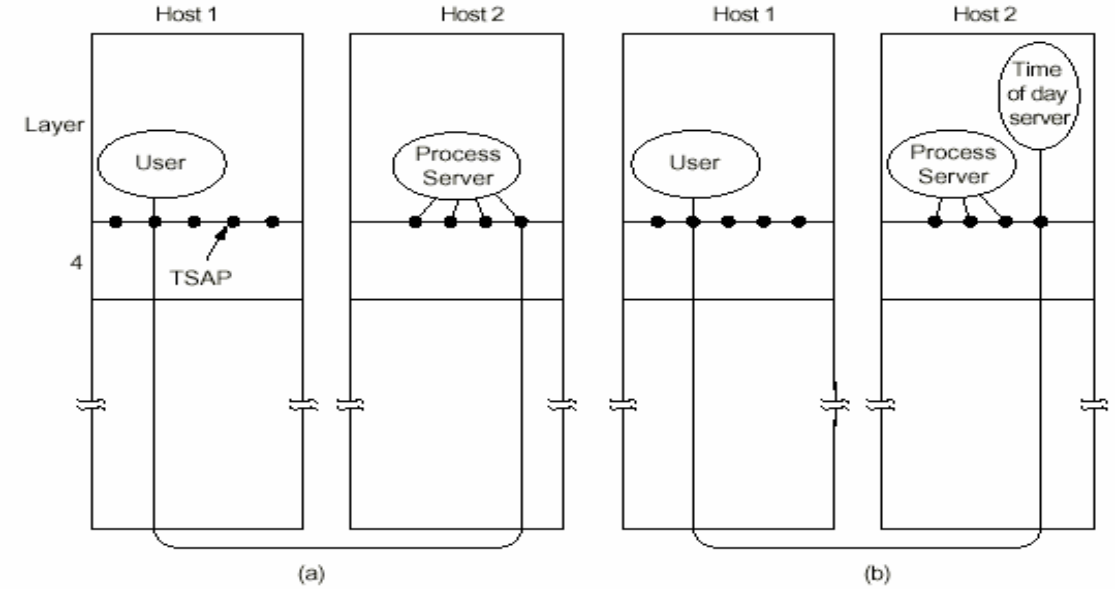
Issue	Datagram subnet	VC subnet
Circuit setup	Not needed	Required
Addressing	Each packet contains the full source and destination address	Each packet contains a short VC number
State information	Subnet does not hold state information	Each VC requires subnet table space
Routing	Each packet is routed independently	Route chosen when VC is set up; all packets follow this route
Effect of router failures	None, except for packets lost during the crash	All VCs that passed through the failed router are terminated
Congestion control	Difficult	Easy if enough buffers can be allocated in advance for each VC

Transport Layer(5)

Addressing (first approach)



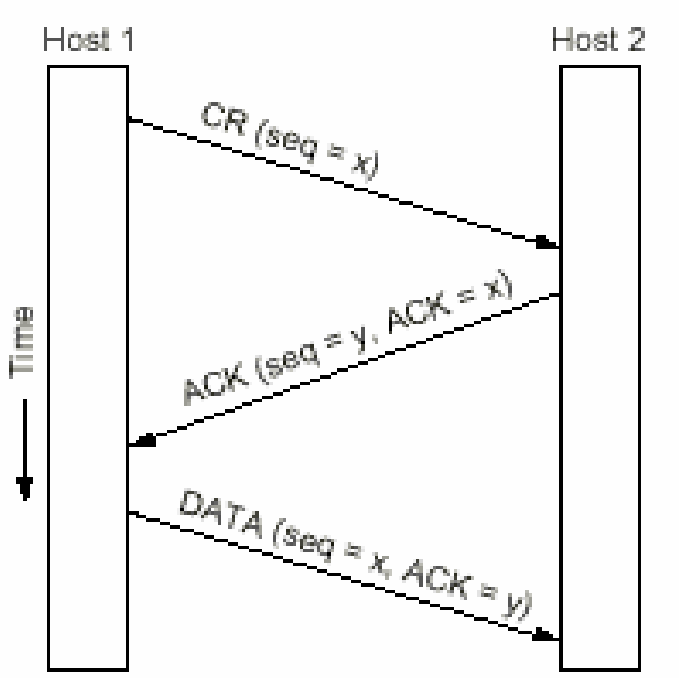
Addressing (second approach)



Transport Layer(6)

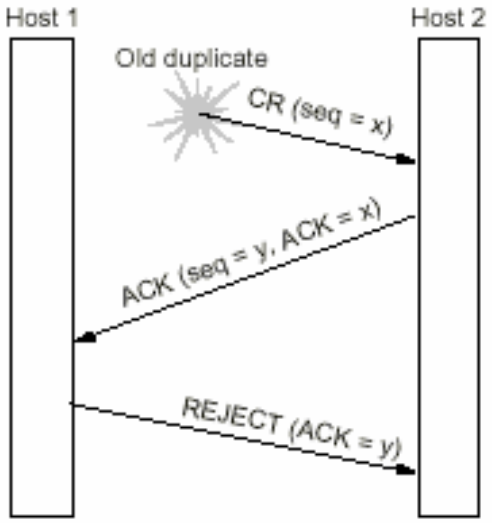
Establishing a Connection (connection request and ack)
Problem: How to deal with delayed duplicates?
Solution: Getting both sides (sender and receiver) to agree on the initial sequence number of the first data packet which is unique for all (sufficient many) connections established by the sender
Example: The three-way handshake (also adopted in TCP)

Normal operation

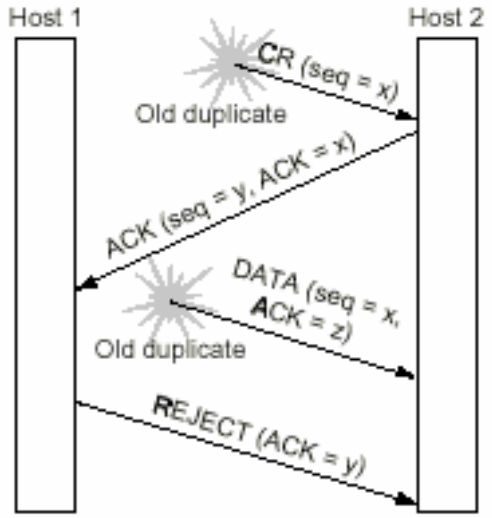


Transport Layer(8)

Duplicate CR (Connection request)



Duplicate CR and ACK to Connection accepted

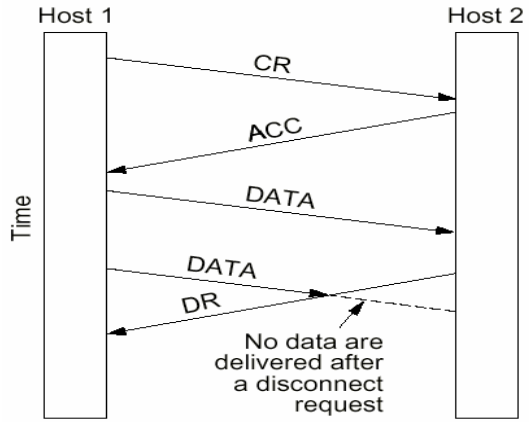


Transport Layer(9)

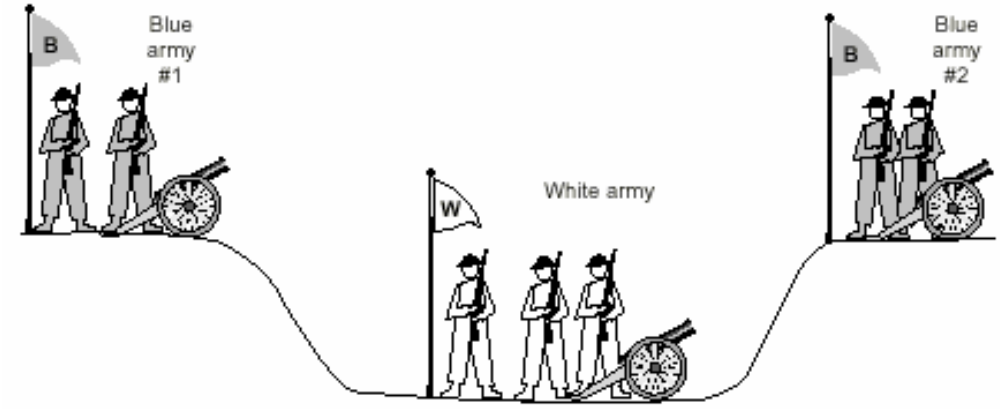
Releasing a Connection

Easier than establishing one, but still with pitfalls

Abrupt disconnection with loss of data



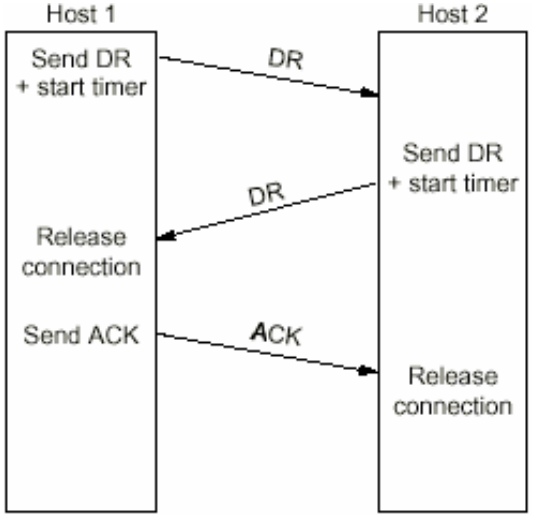
The two-army problem



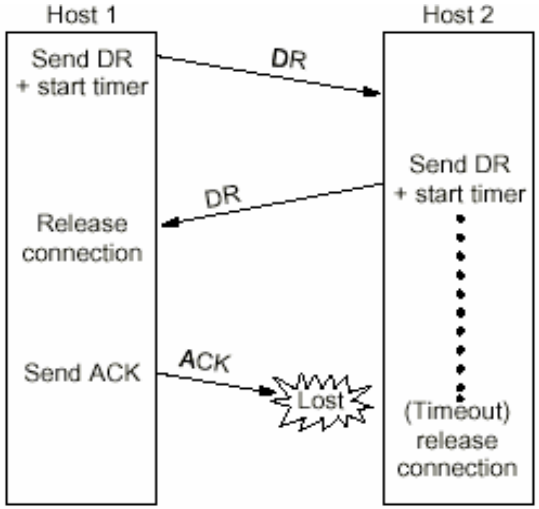
Transport Layer(10)

Four scenarios of releasing using a three-way handshake

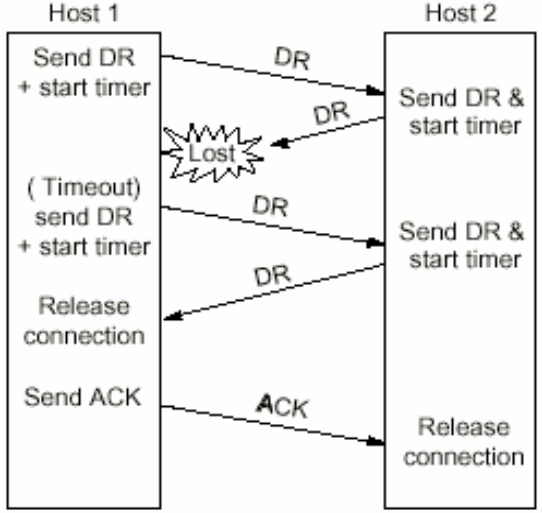
a) normal use



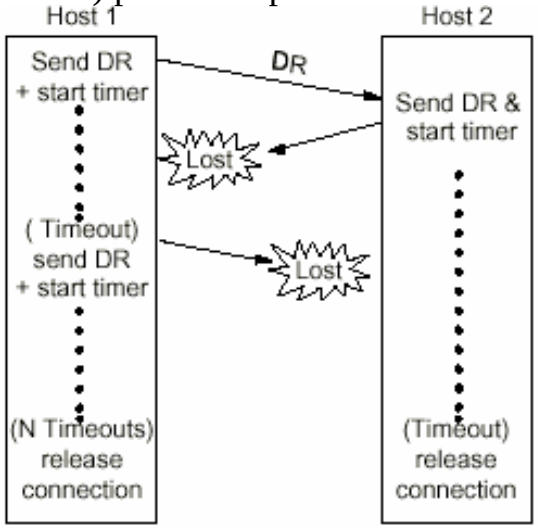
b) final *ACK* is lost



c) second DR lost



d) case c) plus attempts to retransmit also fail



Transport Layer(11)

Flow control and buffering

There are similarities and differences w.r.t. to the same topic in the data link layer.

Basic Similarity:

sliding window or similar scheme is needed on each connection to keep a fast transmitter from overrunning a slow receiver

Main difference:

a router usually has relatively few lines whereas a host may have numerous connections with possibly different packet sizes ---> having one buffer for all may not be adequate ---> more elaborate buffer management

Another bottleneck: the carrying capacity of the subnet

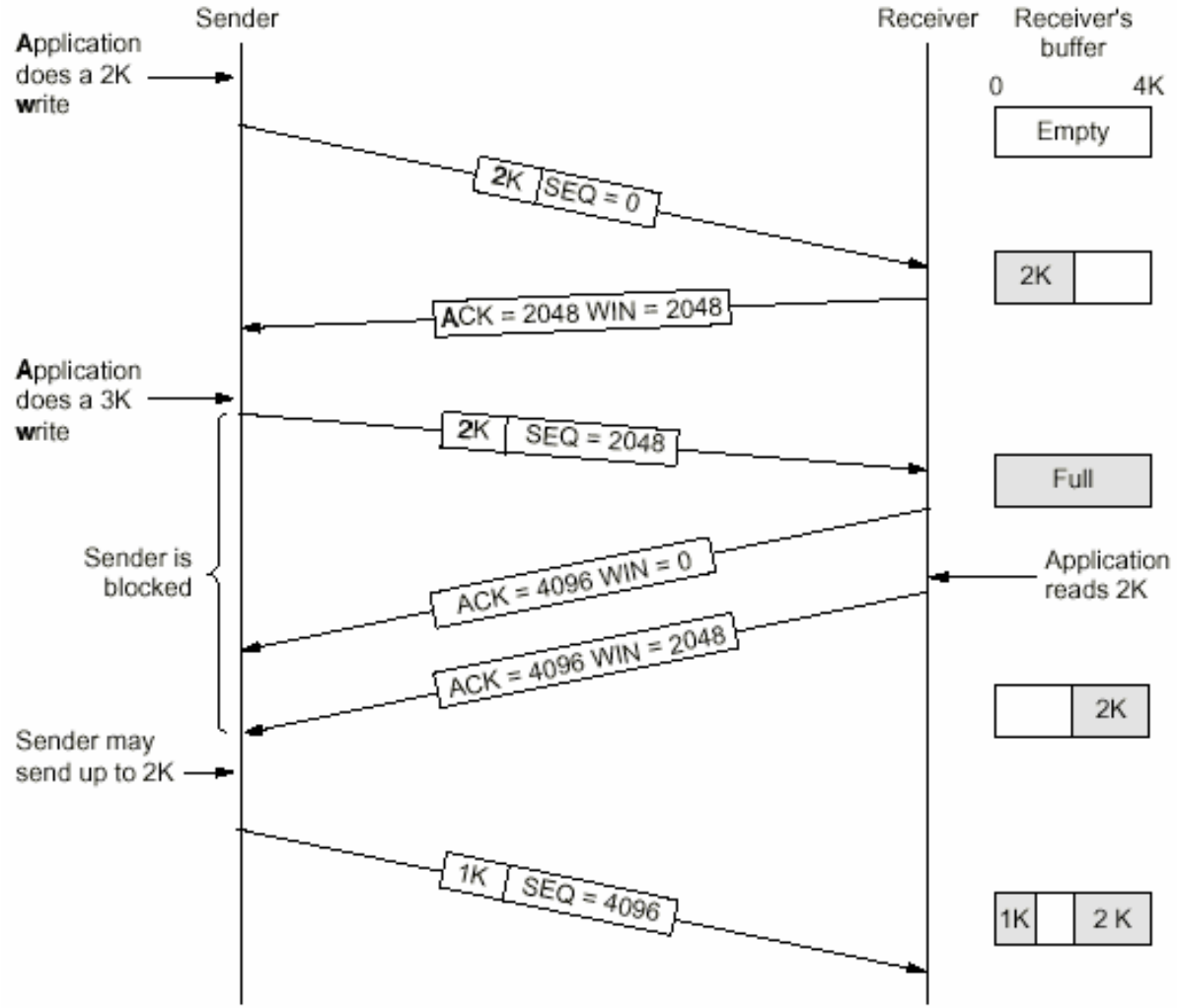
---> congestion problem and no longer a flow control problem

Solution approach (also adopted in TCP):

A sliding window flow control scheme in which the sender adjusts dynamically the window size to match the network's carrying capacity and not the buffer size of the receiver as it is done on the data link layer.

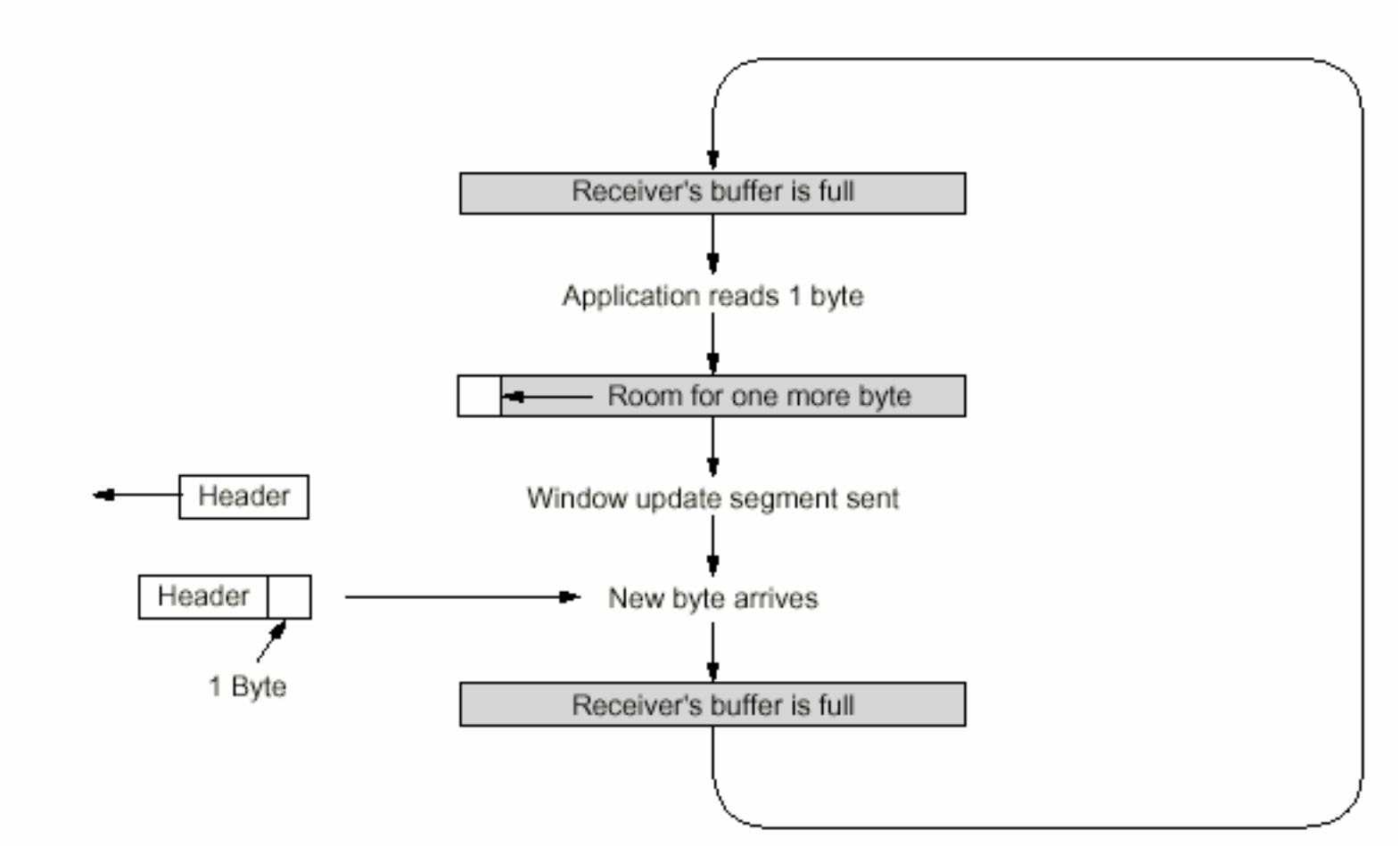
Transport Layer(12)

The window management (transmission policy) in TCP



Transport Layer(13)

The silly window syndrome



Transport Layer(14)

The Internet TCP (Transmission Control Protocol)

Goal:

Provide a reliable (connection-oriented) end-to-end byte stream over an unreliable internetwork

Service to provide:

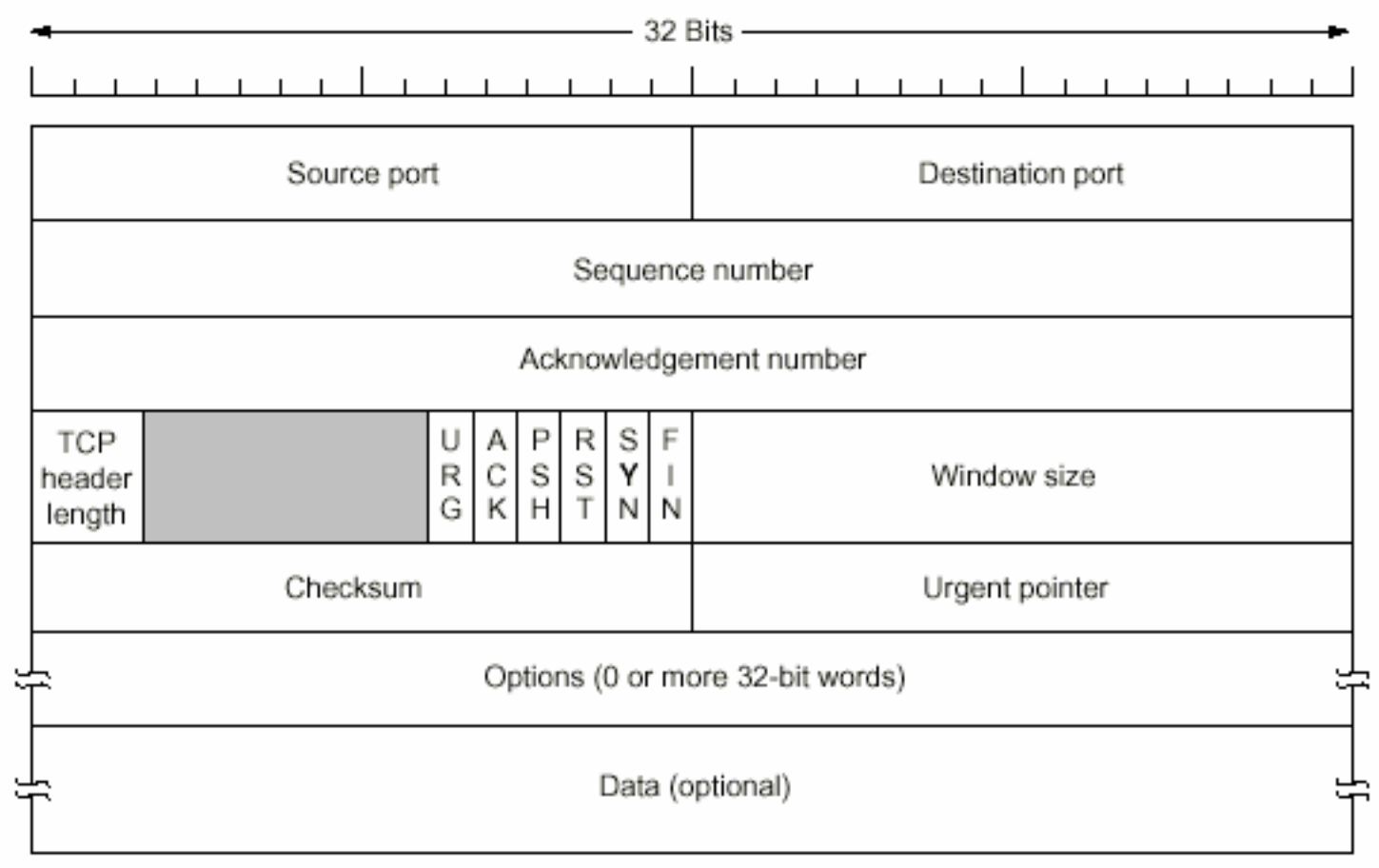
- Accepted user data streams from local processes are broken into pieces called segments not exceeding 64K bytes (in order to fit in the IP payload field)
- Reconstruction of the original byte stream by reassembling the pieces in the proper sequence
- Time out and retransmission in order to guarantee proper delivery by executing a sliding window protocol

The TCP service model

- both the sender and receiver create end points called sockets
- Each socket has an identifier (address):= (IP address, 16-bit number local to that host called port)
- To obtain TCP service, a connection must be established between sockets of the sender and receiver
- A socket may be used for multiple connections at the same time
- Connections are identified by the socket identifiers at both ends: (socket1, socket2)
- Port numbers below 1024 are reserved for standard services (e.g. #21 for file transfer)
- TCP does **not** support multicasting or broadcasting
- data is exchanged in the form of segments beginning with a 20 byte header (+ optional part)

Transport Layer(16)

The TCP Segment header



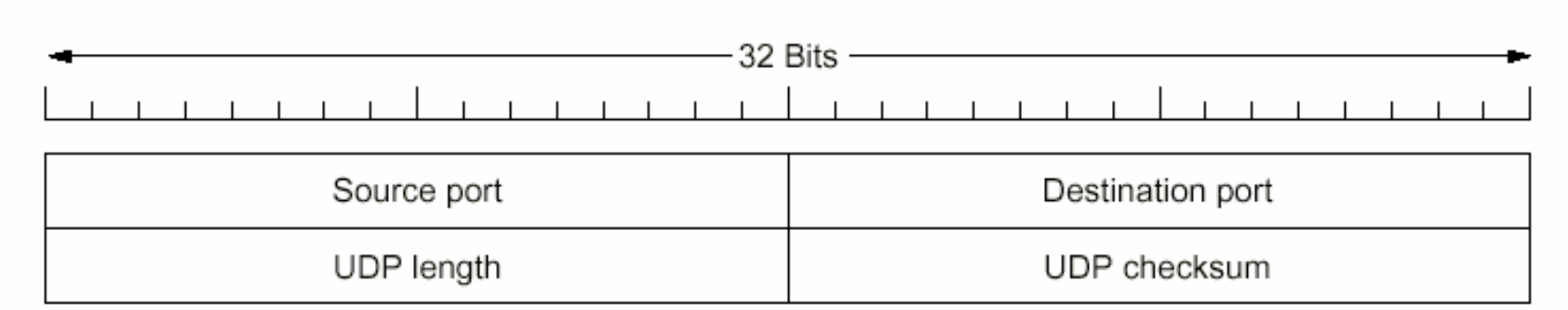
Transport Layer(19)

UDP (User Data Protocol)

Goal:

Provide a connectionless (unreliable) way for applications to send encapsulated raw IP datagrams --->
UDP is nothing else than IP + transport layer header

The UDP header



Typical use:

Client-server applications that have one request and one response

Transport Layer(20)

Summary

The transport layer is the key to understanding layered protocols. Its most important service is to provide an end-to-end, reliable, connection-oriented byte stream from sender to receiver. To do so, it must

- establish connections over unreliable networks
  ---> it must cope with delayed duplicate packets
  ---> it is done by means of a three-way-handshake
- release connections which is easier but still has to face the two-army problem
- handle the service primitives that permit establishing, using, and releasing of connections
- manage connections(transmission policy) and timer

The main Internet transport protocol is TCP

- data is exchanged in the form of segments
- it uses a fixed 20-byte header + optional part + zero or more data bytes
- the basic transmission protocol used is the sliding window protocol
- a great deal of work has gone into optimizing TCP performance using various algorithms