Introduction to transport layer

The transport layer is the layer in the open system interconnection   model responsible for end-to-end communication over a network.

It provides logical communication between application processes running on different host.

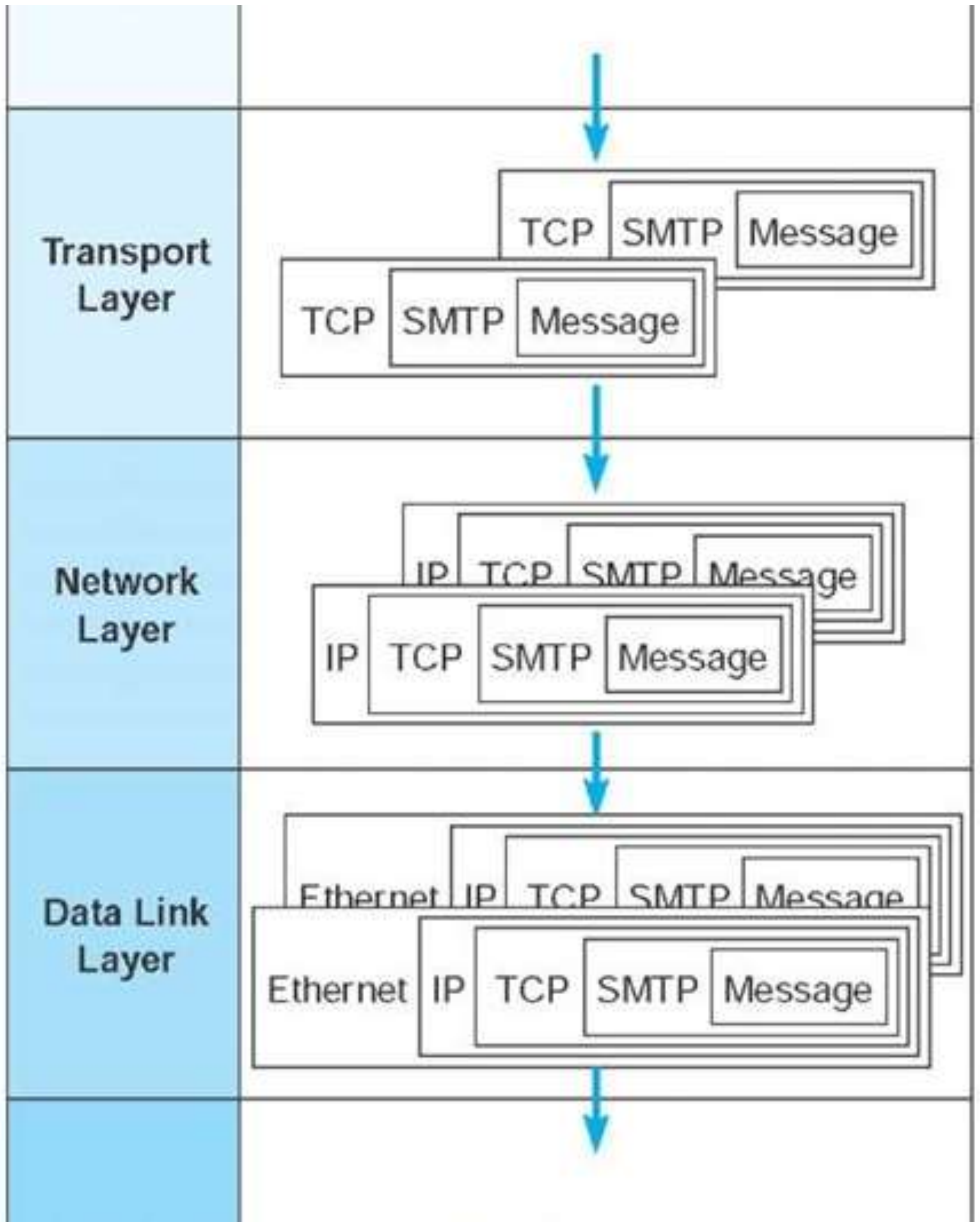
It provides services   such as connection oriented communication, reliability, flow, control and multiplexing.

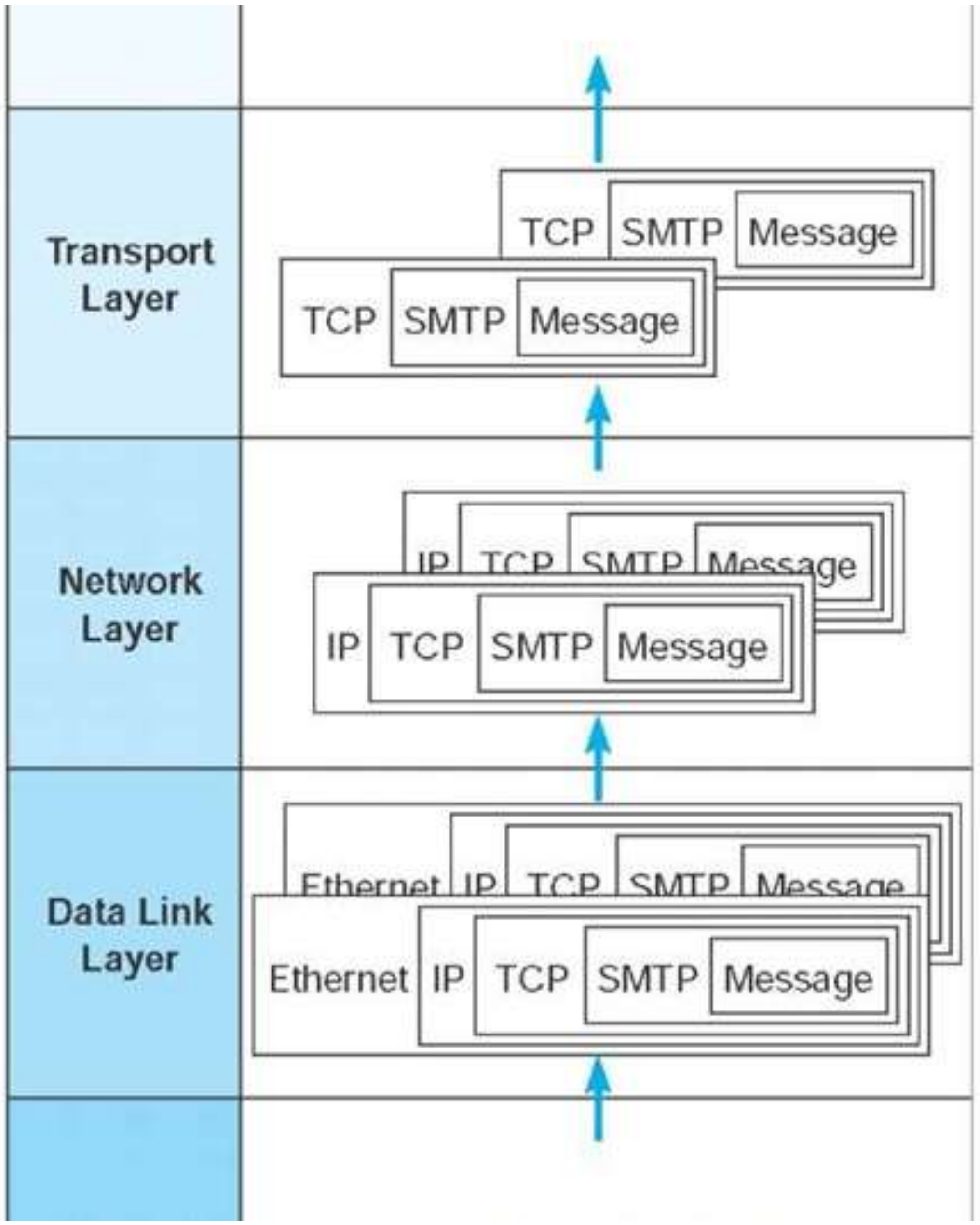
Services provided by transport protocol

- Connection oriented communication
- Same order delivery
- Reliability
- Flow control
- Congestion avoidance
- Multiplexing port

Relation between transport layer and network layer

- The Network layer and transport layer are responsible for moving messages from end to end in a network.
- They are so closely tied together that they are usually discussed together.
- The transport layer performs three functions: linking the application layer to the network, segmenting (breaking long messages into smaller packets for transmission), and session management (establishing an end-to-end connection between the sender and receiver).
- The network layer performs two functions: routing (determining the next computer to which the message should be sent to reach the final destination) and addressing (finding the address of that next computer).
- There are several standard transport and network layer protocols that specify how packets are to be organized, in the same way that there are standards for data link layer packets, However, only one protocol is in widespread use today


Transport Layer
TCP | SMTP | Message
TCP | SMTP | Message
Network Layer
IP | TCP | SMTP | Message
IP | TCP | SMTP | Message
Data Link Layer
Ethernet | IP | TCP | SMTP | Message
Ethernet | IP | TCP | SMTP | Message


Transport Layer
TCP | SMTP | Message
TCP | SMTP | Message
Network Layer
IP | TCP | SMTP | Message
IP | TCP | SMTP | Message
Data Link Layer
Ethernet | IP | TCP | SMTP | Message
Ethernet | IP | TCP | SMTP | Message

Transport Layer in internet

Connection oriented

Requires a session connection be established before any data can be sent.

This method is often called a "reliable" network service. It can guarantee that data will arrive in the same order.

Connection-oriented services set up virtual links between end systems through a network.

TCP is used in connection oriented.

Connection less

Does not require a session connection between sender and receiver.

The sender simply starts sending packets (called datagrams) to the destination.

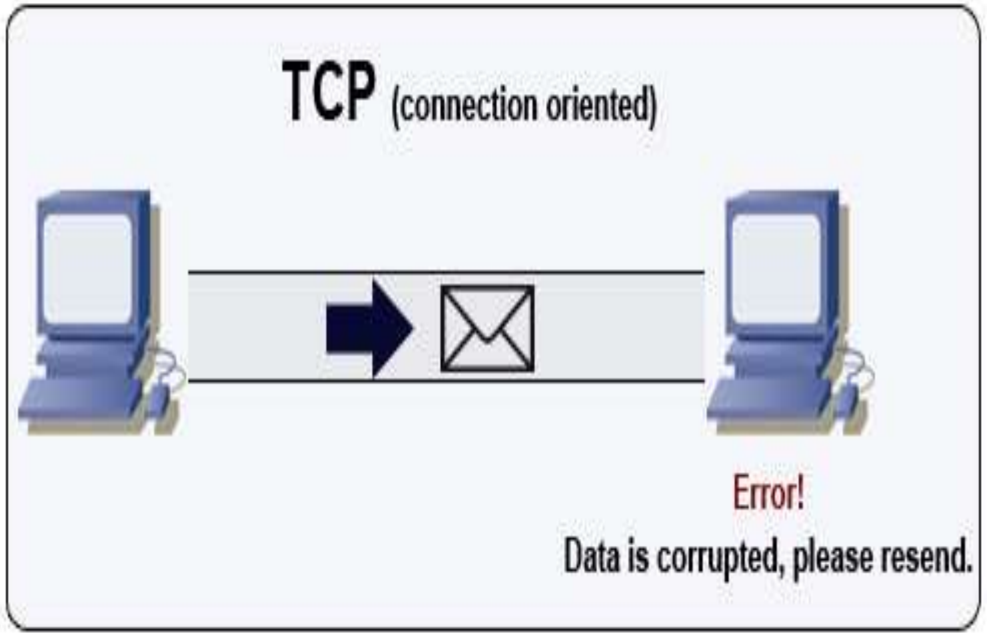
This service does not have the reliability of the connection-oriented method, but it is useful for periodic burst transfers.

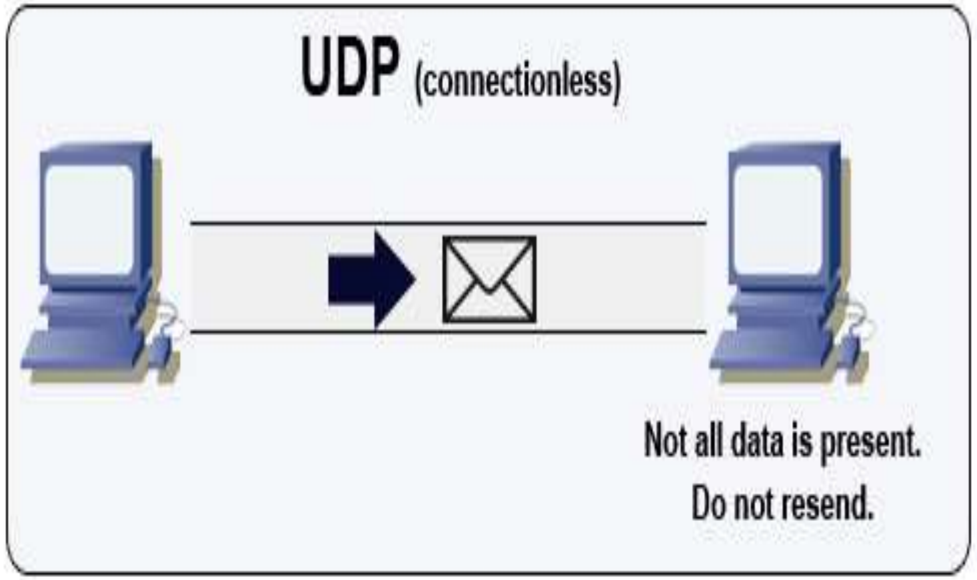
Neither system must maintain state information for the systems that they send transmission to or receive transmission from.

A connectionless network provides minimal services.

Less reliable than connection oriented

UDP is used in connection less services


TCP (connection oriented)
Error!
Data is corrupted, please resend.

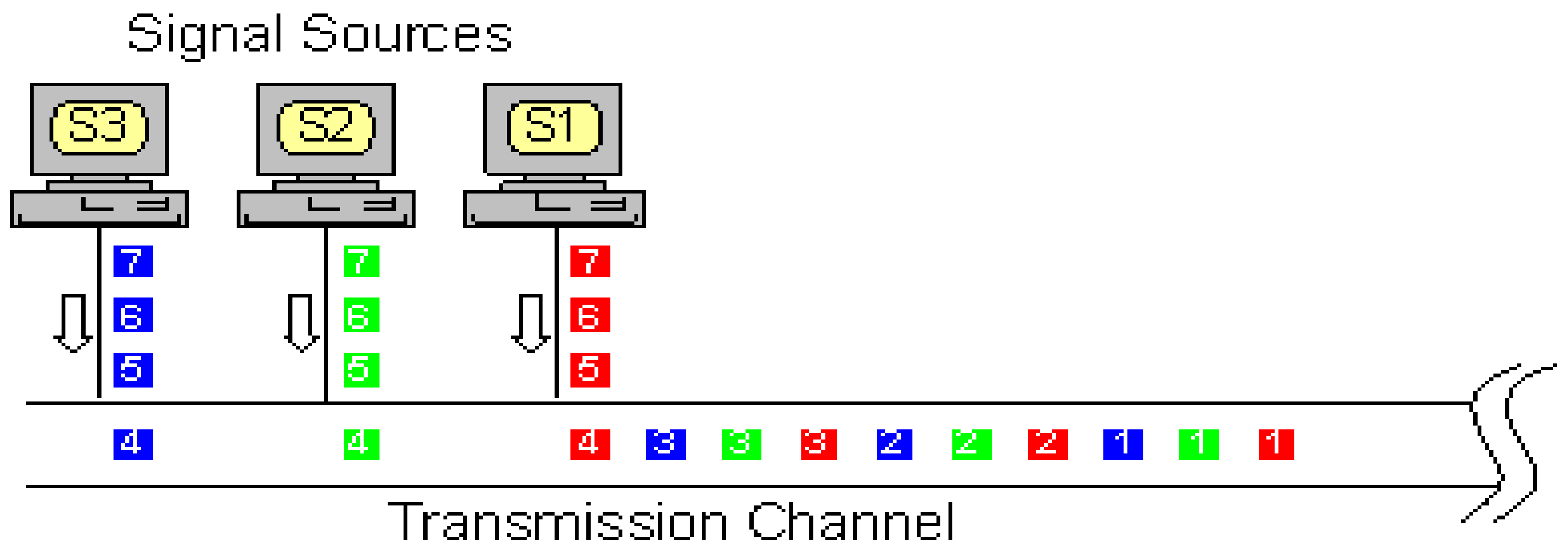

UDP (connectionless)
Not all data is present.
Do not resend.

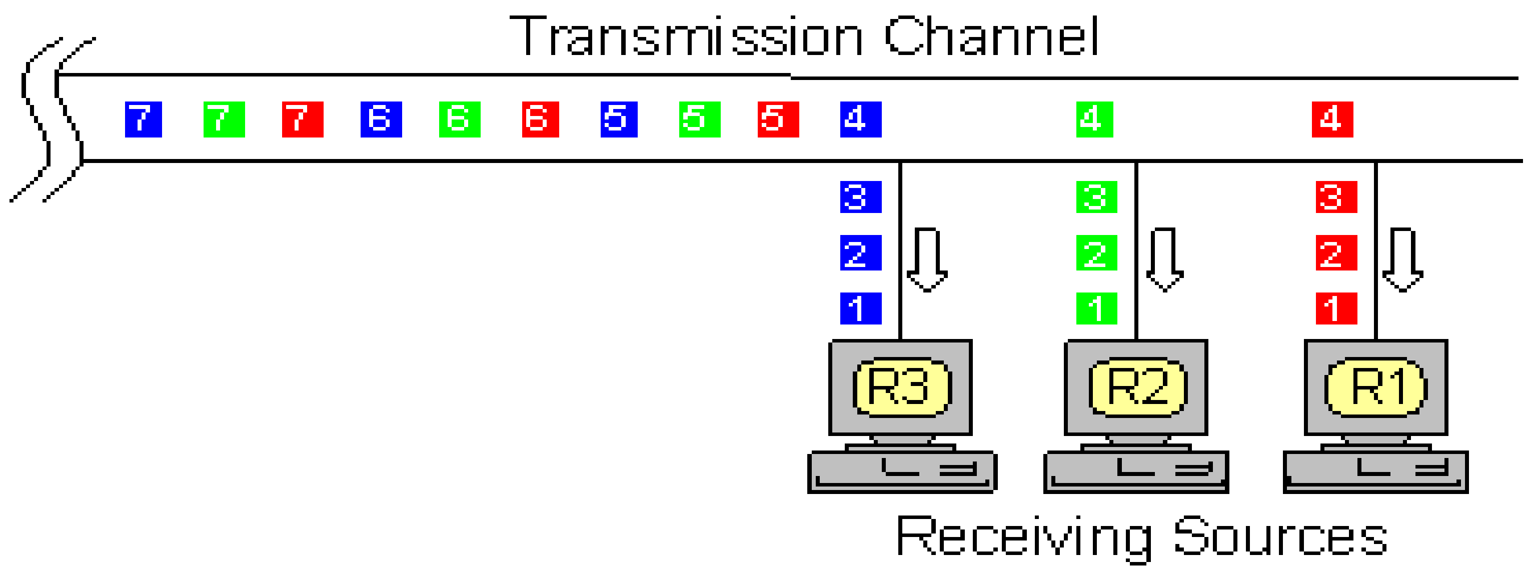
Multiplexing

- Multiplexing is the process in which multiple Data Streams, coming from different Sources, are combined and Transmitted over a Single Data Channel or Data Stream
- In Electronic Communications, the two basic forms of Multiplexing are Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM).
- Multiplexing is done by an equipment called Multiplexer (MUX). It is placed at the Transmitting End of the communication link.

Demultiplexing

- At the Receiving End, the Composite Signal is separated by an equipment called Demultiplexer.
- Demultiplexer performs the reverse process of Multiplexing and routes the separated signals to their corresponding Receivers or Destinations.

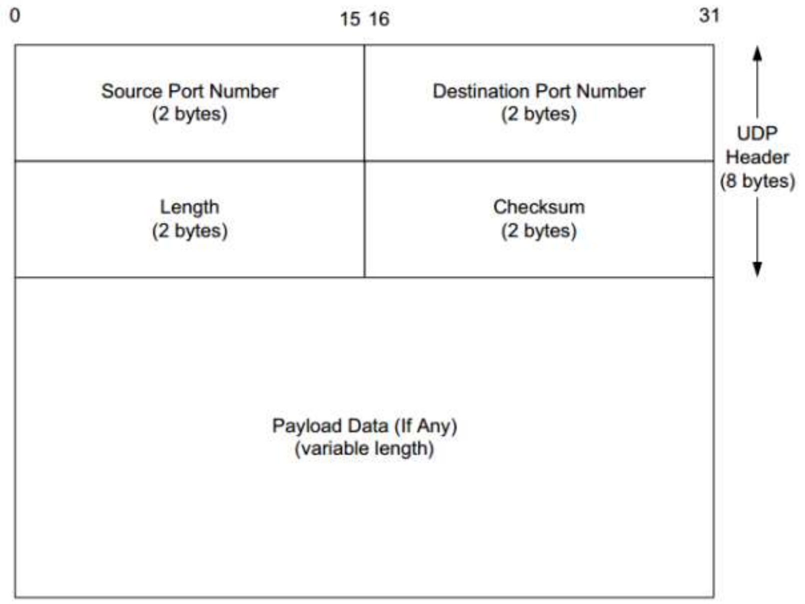

Signal Sources
S3
S2
S1
7
6
5
4
3
2
1
Transmission Channel


Transmission Channel
7
6
5
4
3
2
1
R3
R2
R1
Receiving Sources

UDP

- It is connectionless, unreliable and Process-to-process communication .
- UDP is used to send message in the form of datagrams, which comprise one message units.
- No handshaking dialogs for reliability, ordering and data integrity between receiver and sender.
- Protocol assumes that error-checking and correction is not required, thus avoiding processing at the network interface level.
- UDP send packets and those packets are received in a different order than that in which they were sent, allowing for better performance.
- UDP segments may be lost, or delivered out of order
- Eg: video conferencing and real-time computer games, multimedia, online phone

UDP Segment Structure


0
15 16
31
Source Port Number (2 bytes)
Destination Port Number (2 bytes)
Length (2 bytes)
Checksum (2 bytes)
Payload Data (If Any) (variable length)
UDP Header (8 bytes)

The first 8 bytes header information, while the remaining bytes contain message data. UDP datagram header contains 4 fields of two bytes each:

- Source Port - This 16 bits information is used to identify the source port of the packet.
- Destination Port - This 16 bits information, is used identify application level service on destination machine.
- Length - Length of UDP packet (including header). It is 16-bits field and minimum value is 8- byte.
- Checksum - This field stores the checksum value generated by the sender before sending.

ADVANTAGES

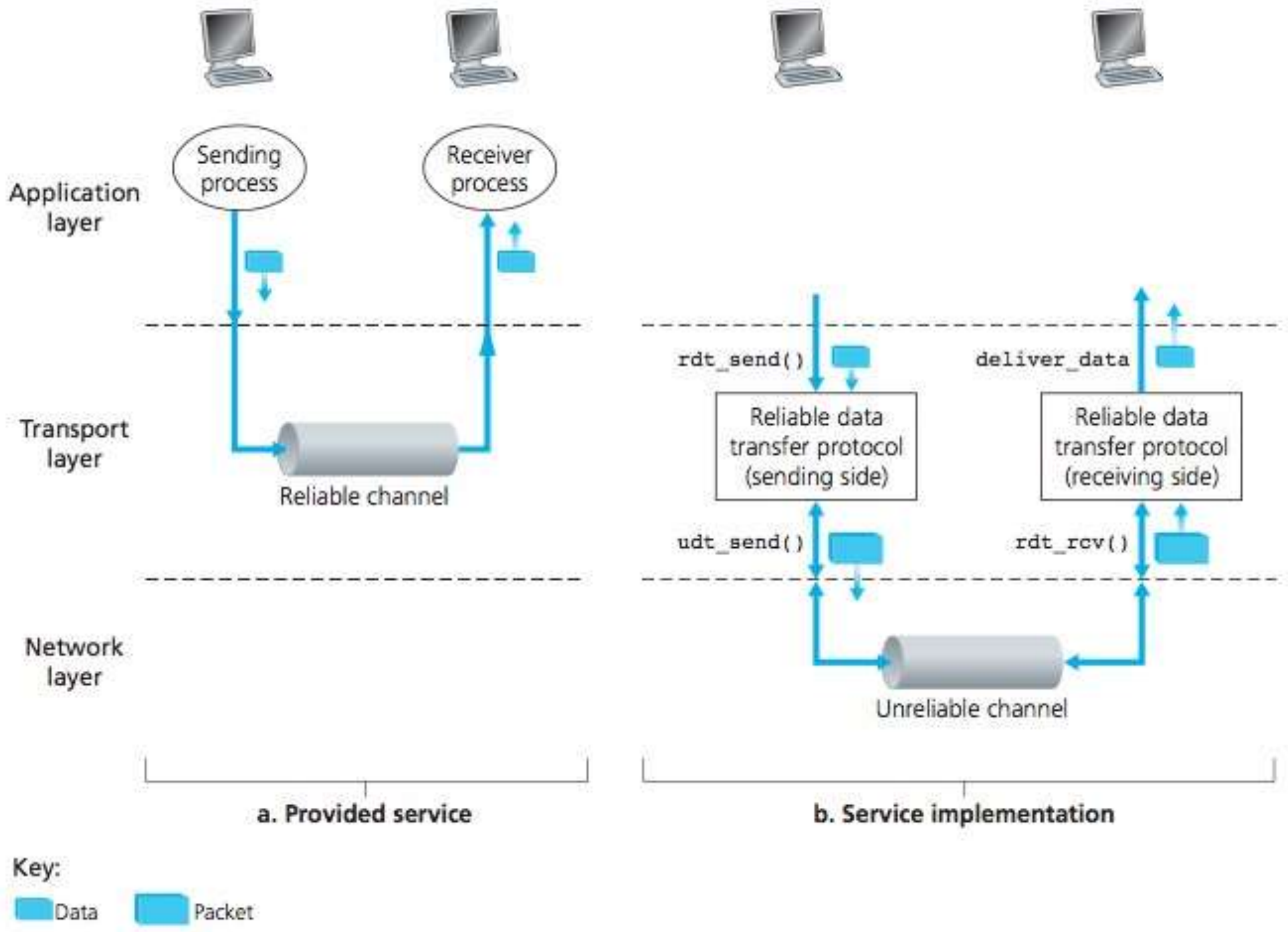
- **No connection establishment:** UDP does not use any handshaking signals, the delay in making connections can be avoided.
- **Speed:** UDP is fast
- **Topology support:** UDP supports both one-to-one and one-to-many connections
- **Header size:** UDP has only 8 byte headers for every segment, making UDP less consuming of network bandwidth.

DISADVANTAGE

- **Lack of handshaking signals: C**annot guarantee that the data will actually be delivered at the destination.
- **Use of sessions.** : UDP doesn't have any support for sessions due to its connection-less nature.
- **Reliability.** UDP does not guarantee that these segments will be delivered to the destination in the same order as they were created at the source
- **Security.** firewalls and routers do not allow UDP packets because hackers can use UDP ports.
- **Flow control.** No flow control poor designed UDP application can tie up a big chunk of network bandwidth.

RELIABLE DATA TRANSFER

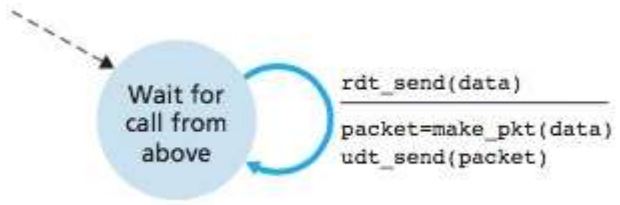
- Located on the Transport Layer next to protocols like TCP or UDP.
- RDT protocol ensure delivery of all packets and enable the receiver to deliver the packets in order to its application layer.
- RDT protocol can be designed using some basic tools. Known as a stop-and-wait protocol.
- Stop-and-wait has poor performance in a long-distance connection.
- To improve transmission rate RDT protocol must use pipelining.
- TCP ensures all parts of a message to reach the destination undamaged. But sometimes parts of a message fails to reach the destination or damaged message deliver.
- Thus, TCP has to detect and recover from lost or damaged message , Recovering from errors is called reliable data transfer.



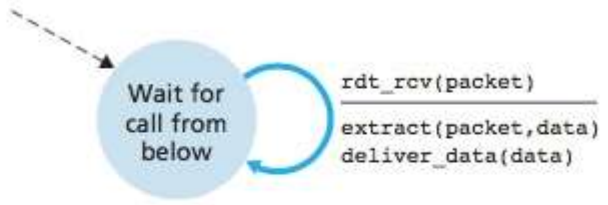
Application layer
Transport layer
Network layer
Sending process
Receiver process
Reliable channel
rdt_send()
deliver_data
Reliable data transfer protocol (sending side)
Reliable data transfer protocol (receiving side)
udt_send()
rdt_rcv()
Unreliable channel
a. Provided service
b. Service implementation
Key:
Data
Packet

Building a Reliable Data Transfer Protocol

1. Reliable Data Transfer over a Perfectly Reliable Channel: rdt1.0



a. rdt1.0: sending side



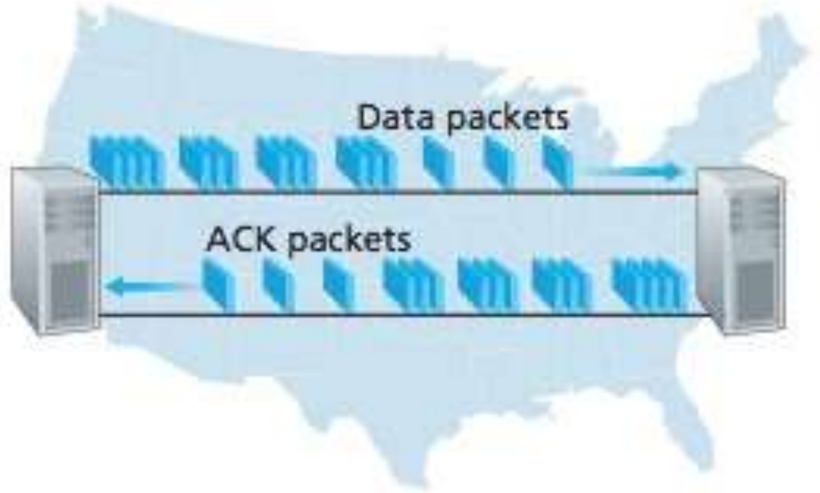
b. rdt1.0: receiving side

A perfect reliable channel doesnot need for the receiver side to provide any feedback to the sender since nothing can go wrong. No need forth receiver to ask the sender to slow down.

- No bit error
- No loss of packets

- This protocol allows for multiple data packets to be sent while waiting for ACK. Results better network utilization
- sender and receiver need buffers to hold multiple packets
- packets need sequence numbers in order to identify them
- an acknowledgement needs to refer to corresponding sequence number
- retransmission can give rise to duplicate packets
- sequence numbers in packets allow receiver to detect duplicates

Pipelined Reliable Data Transfer Protocols



ADVANTAGE

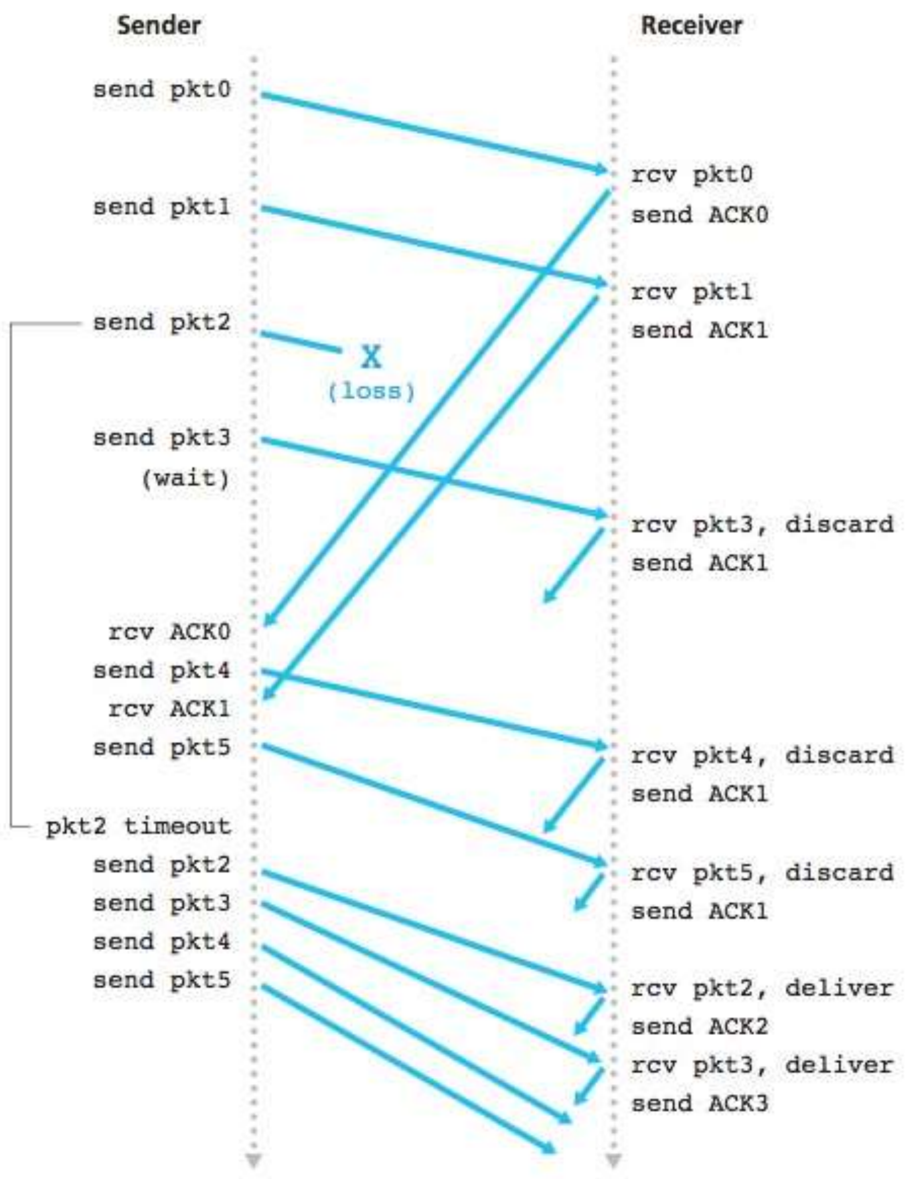
- much better than stop-and-wait

DISADVANTAGE

- More complicated to deal with reliability issues, e.g., corrupted, lost, out of order data.
- Two generic approaches to solving this
    - Go-Back-N protocols
    - Selective repeat protocols

Go-Back-N(GBN)

- The GBN sender must respond to three types of events:
- **Invocation from above:** rdt_send() is called, the sender first checks to see if the window is not full, a packet is created and sent, and variables are appropriately updated;
- **Receipt of an ACK**: ACK for a packet with sequence number n ensures correctly received at the receiver;
- **A timeout event**: Sender resends all packets and wait for ACK If an ACK is received but there are still additional transmitted but not yet ACK packets, the timer is restarted.


Sender
Receiver
send pkt0
rcv pkt0
send ACK0
send pkt1
rcv pkt1
send ACK1
send pkt2
X
(loss)
send pkt3
(wait)
rcv pkt3, discard
send ACK1
rcv ACK0
send pkt4
rcv ACK1
send pkt5
rcv pkt4, discard
send ACK1
pkt2 timeout
send pkt2
send pkt3
send pkt4
send pkt5
rcv pkt5, discard
send ACK1
rcv pkt2, deliver
send ACK2
rcv pkt3, deliver
send ACK3

Advantage over Go-Back-N:

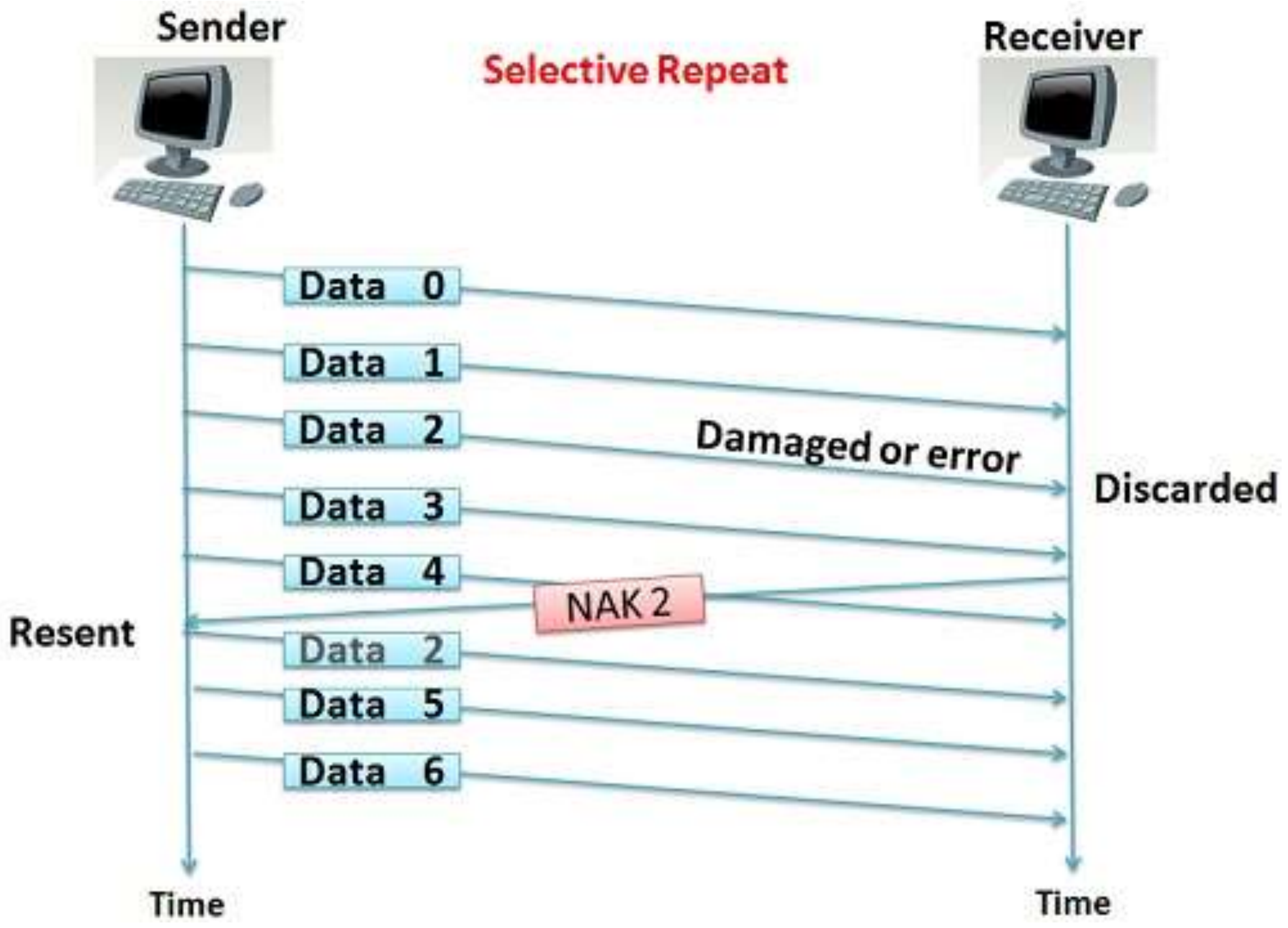
- Fewer Retransmissions.

Disadvantages:

- More complexity at sender and receiver
- Each frame must be acknowledged individually
- Receiver may receive frames out of sequence

Selective Repeat

- SR receiver ACK a correctly received packet whether or not it is in order.
- Out-of-order packets are buffered until any missing packets are received.
- Receiver re ACK already received packets with certain sequence numbers below the current window base.
- If no ACK for packet send_base propagating from the receiver to the sender, the sender will eventually retransmit packet send_base.
- Both sender and receiver maintain a window of outstanding and acceptable sequence numbers, respectively.
- The sender’s window size starts out at 0 and grows to some predefined maximum.
- The receiver has a buffer reserved for each sequence number within its fixed window.


Sender
Selective Repeat
Receiver
Data 0
Data 1
Data 2
Damaged or error
Discarded
Data 3
Data 4
NAK 2
Resent
Data 2
Data 5
Data 6
Time
Time

ADVANTAGE

- The sender only retransmits frames, for which a NAK is received.
- This will increase the efficiency of the protocol.

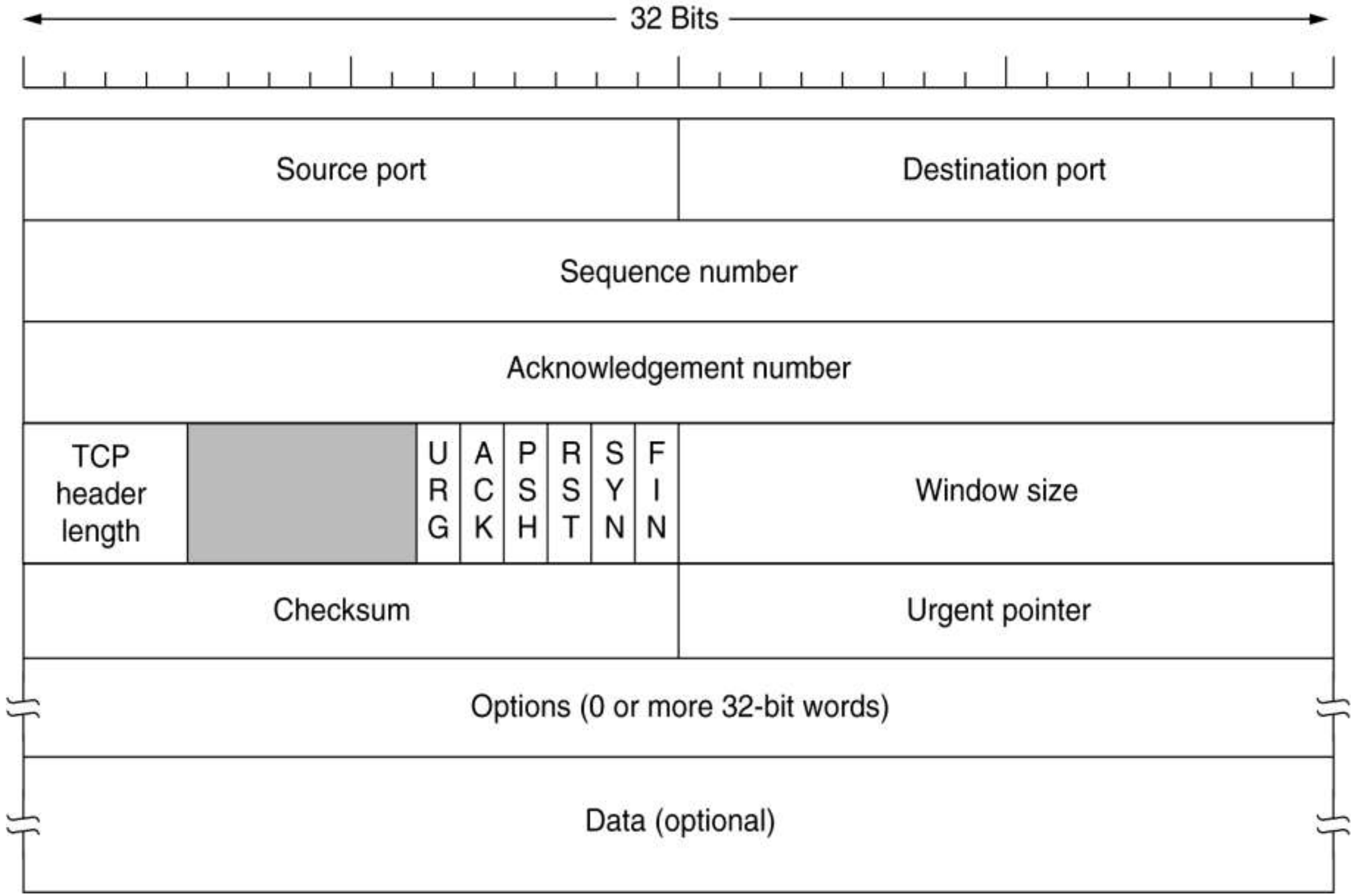
DISADVANTAGE

- More complexity at sender and receiver
- Each packet must be acknowledged individual
- Receiver may receive packets out of sequence

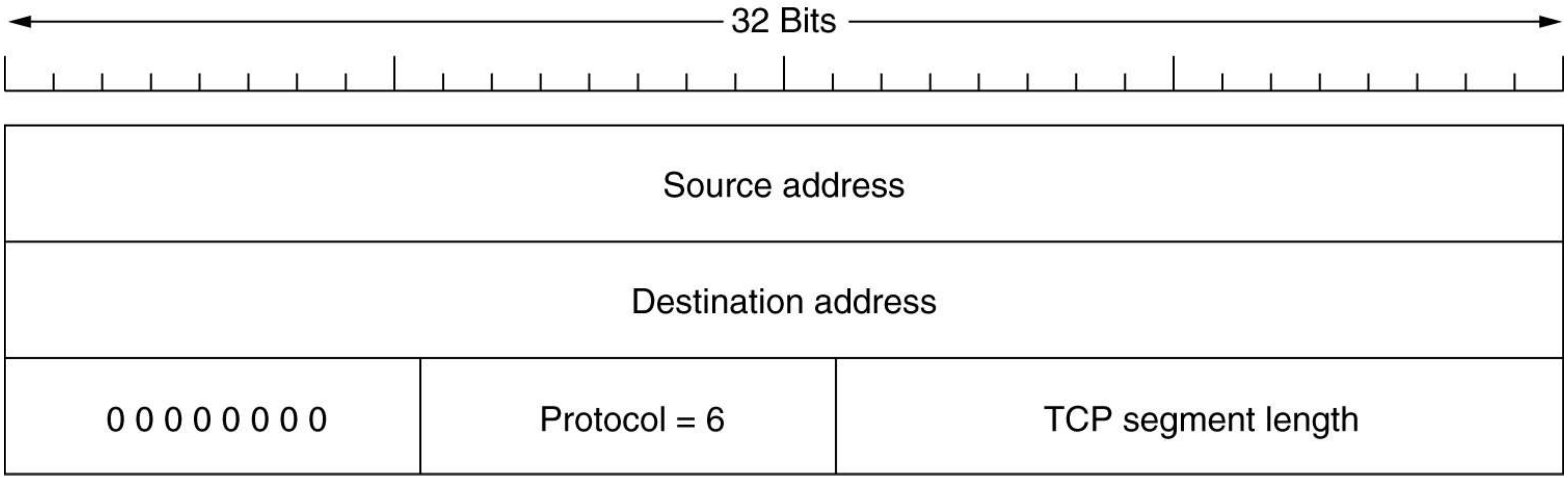
Transmission Control Protocol (TCP)

- End to end communication
- Reliable, Connection oriented
- Flow control
- Multiplexing
- Full duplex

TCP Header Segment



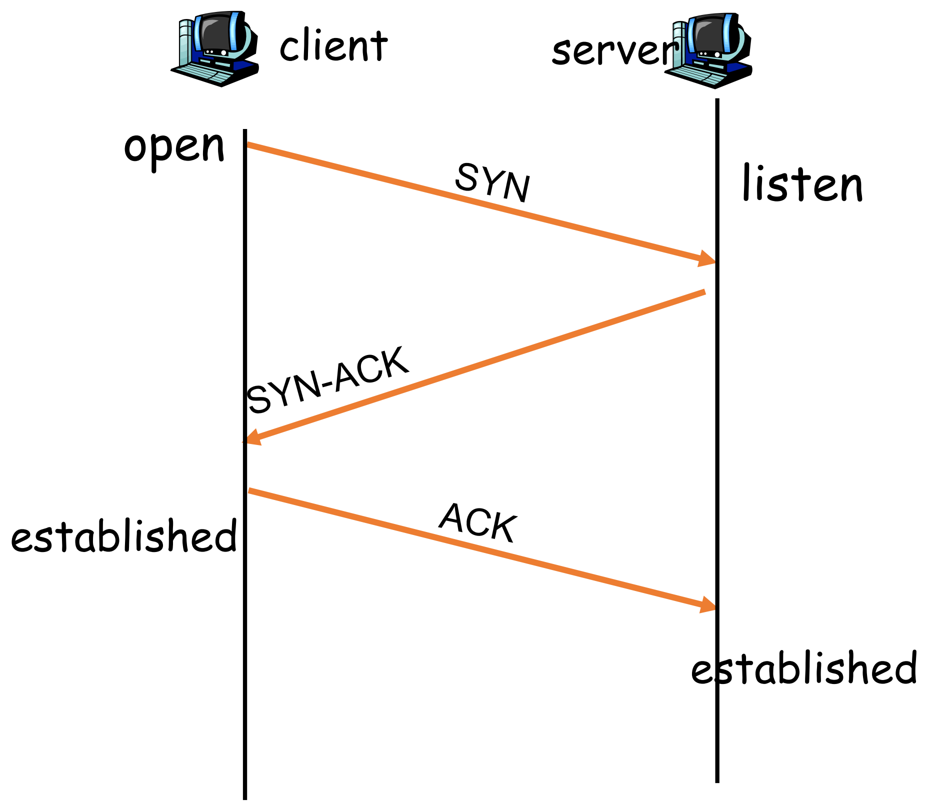
TCP Checksum header



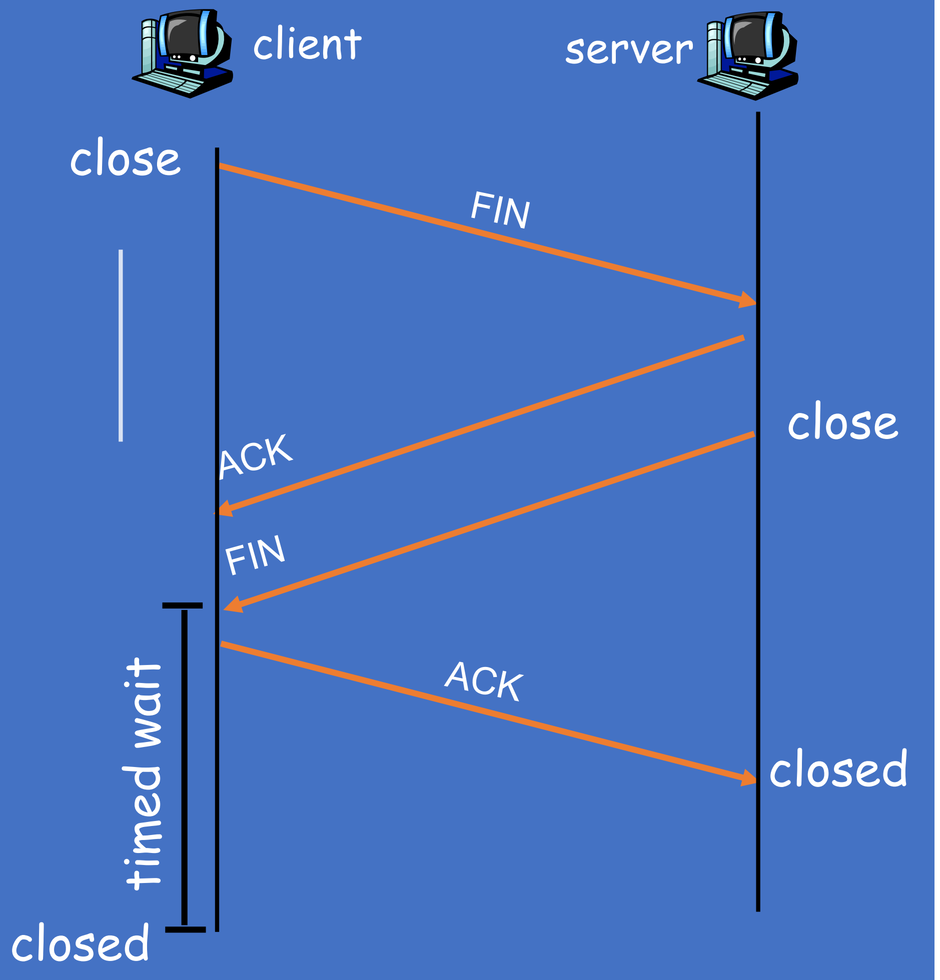
Connection establishment

- 3 way handshaking

1. Client end system sends TCP SYN control segment to server
2. Server end system receives SYN, replies with SYN-ACK and allocates buffer
3. Client receives SYN-ACK

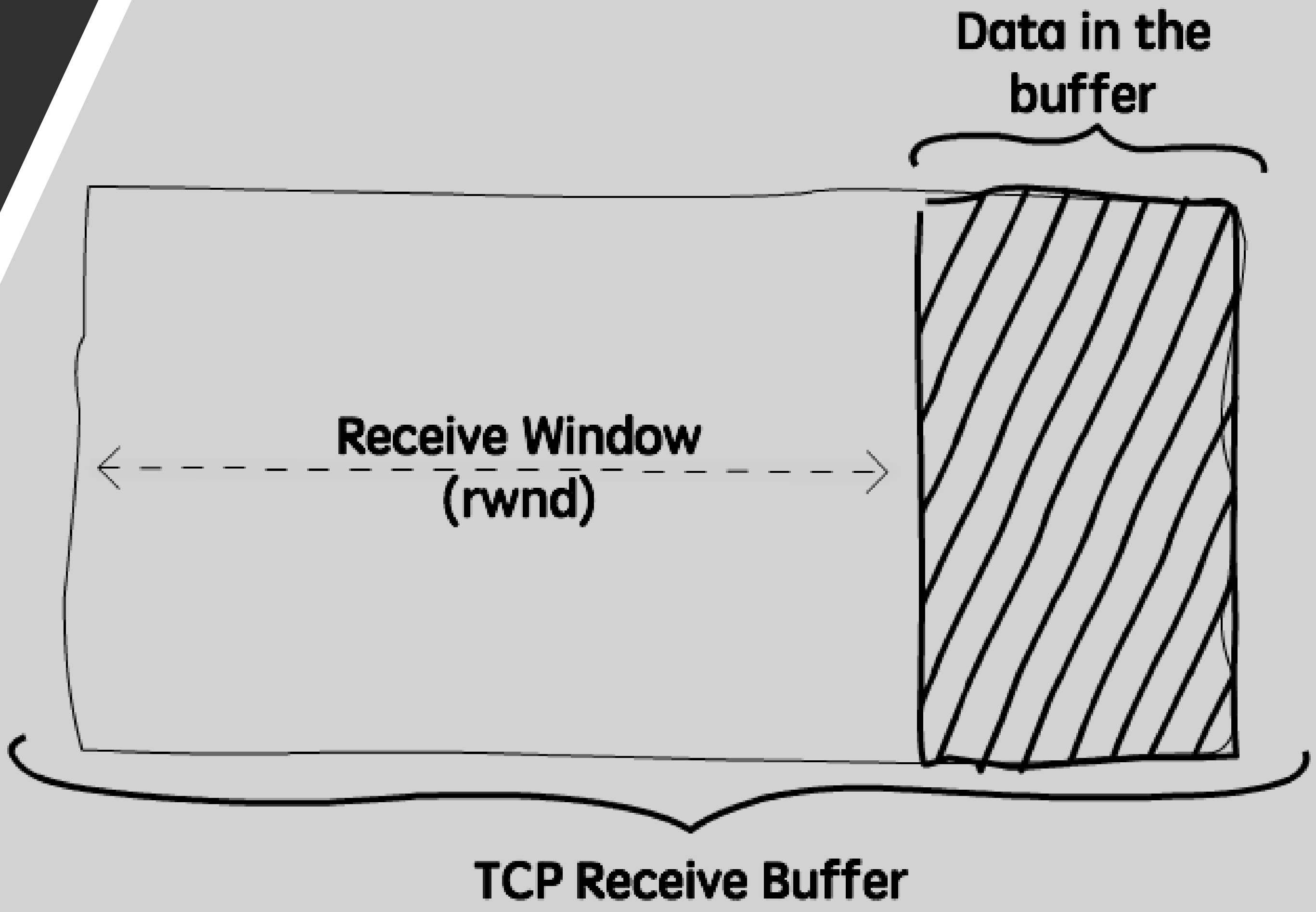


Connection release



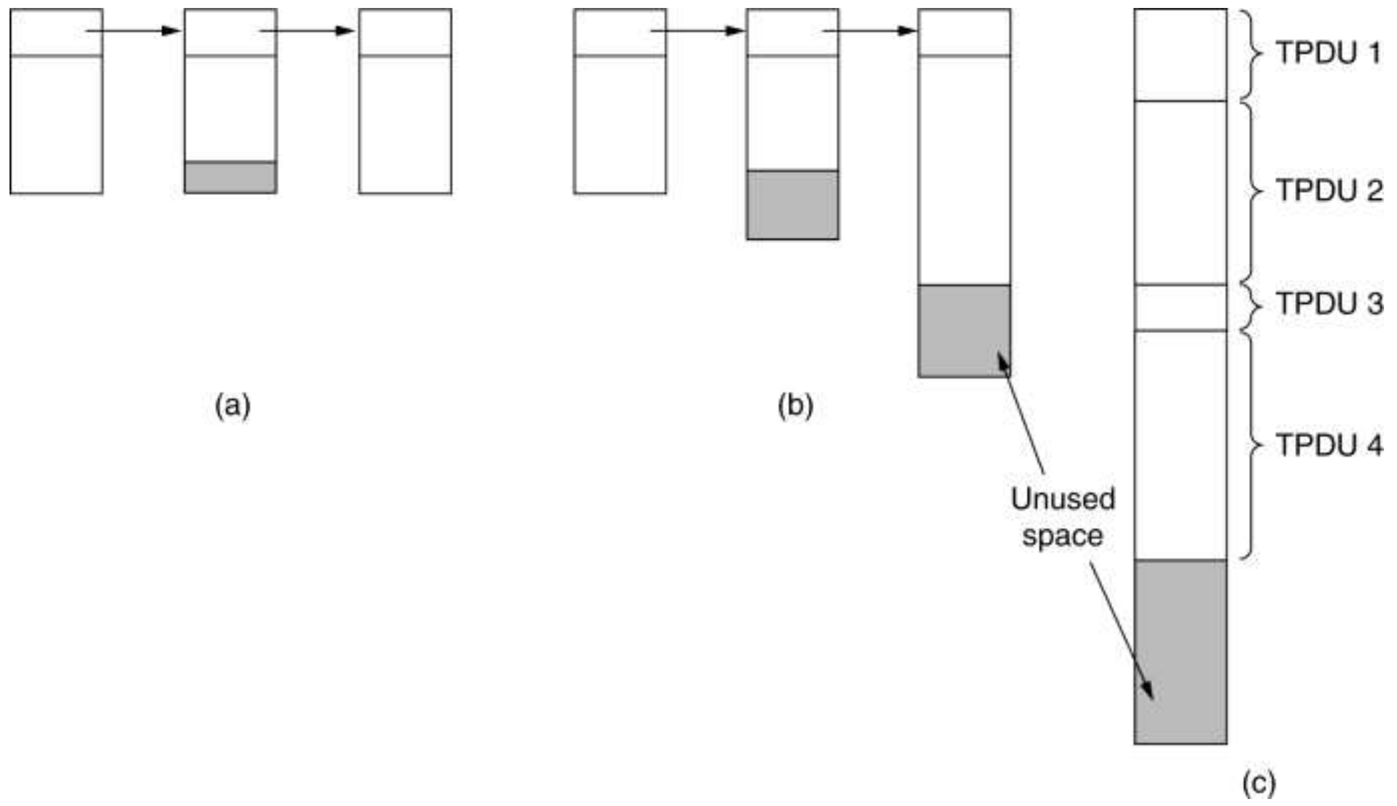
Flow control

- Controlling the flow of packets
- Buffer are used to store the packets



Approaches for buffer organization

a) Chained fixed size
b) Chained variable size
c) One larger circular buffer per connection



Congestion

Overloading routers with packets

Nodes receiving faster than it can deliver

Factors causing congestion

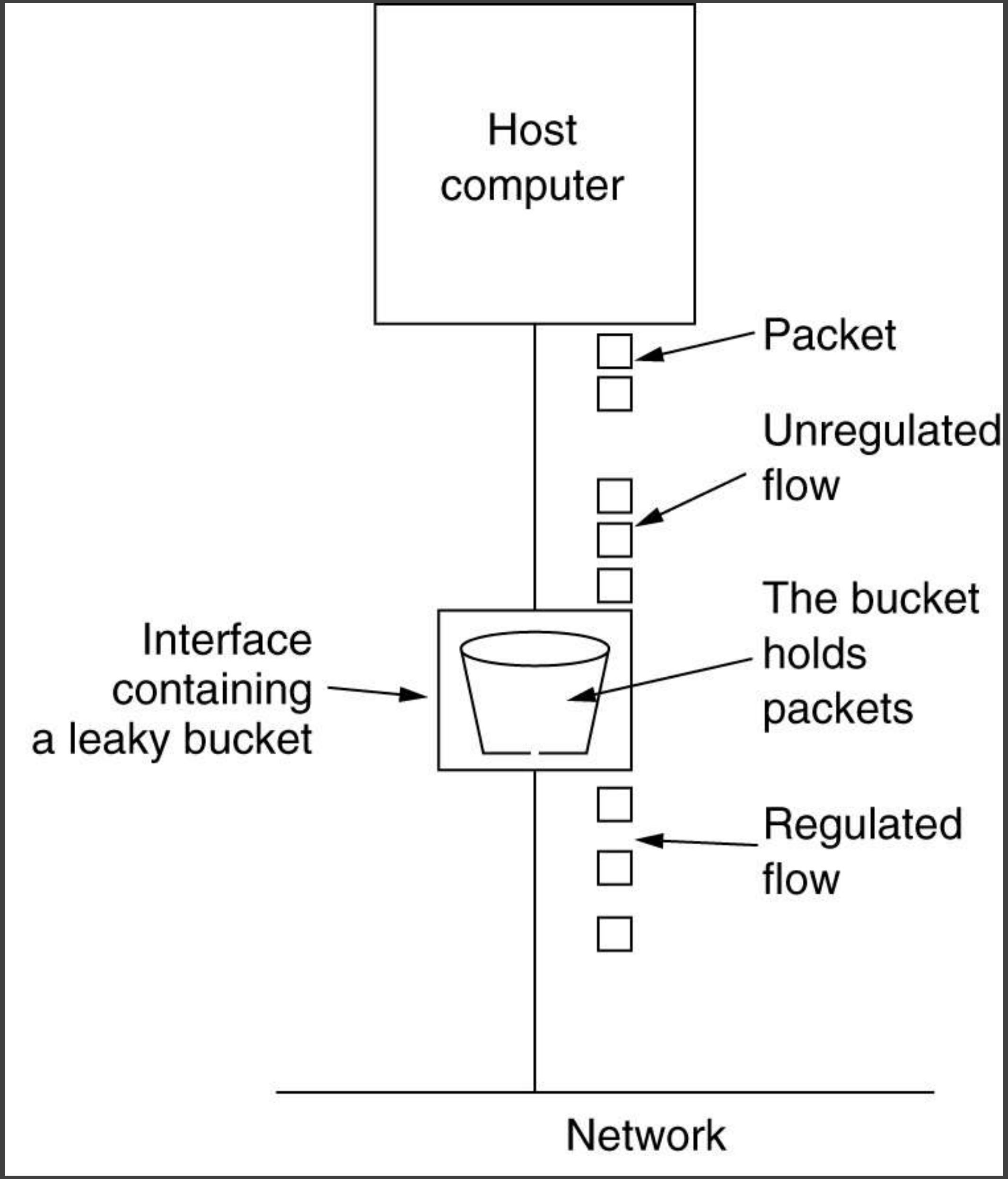
- Slow processor
- Insufficient memory
- Bursty traffic
- Packet arrival rate higher

Congestion control

- Warning bit
- Choke packets
- Load shedding
- Random early discard
- Traffic shaping

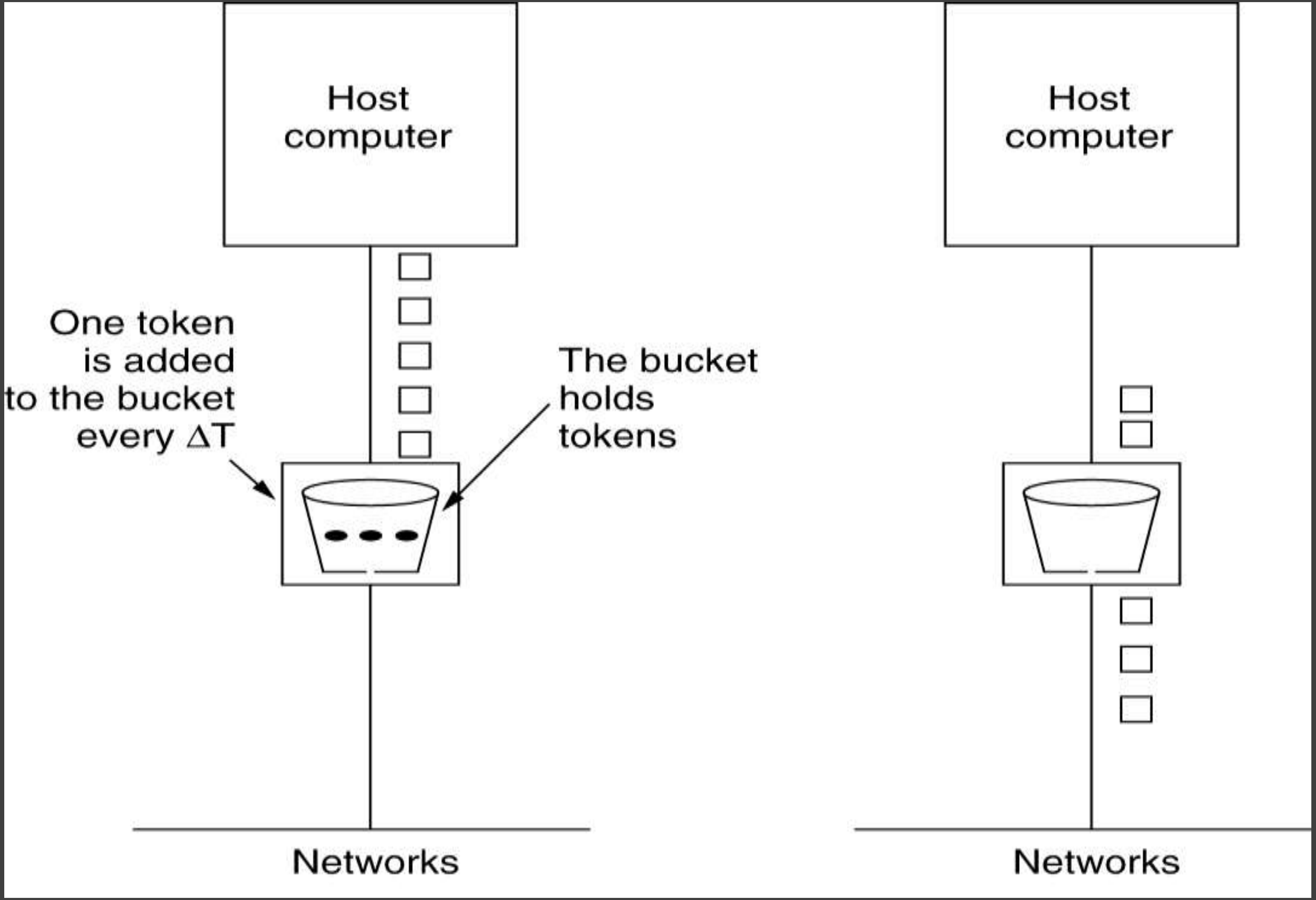
Traffic shaping

- Leaky bucket algorithm
  - Enforces a constant output rate
  - Packets are discarded after buffer is full



Traffic shaping

- Token Bucket algorithm
  - Output varies
  - Bucket hold tokens
  - Host capture and destroy token for transfer


Host computer
One token is added to the bucket every ΔT
The bucket holds tokens
Networks
Host computer
Networks