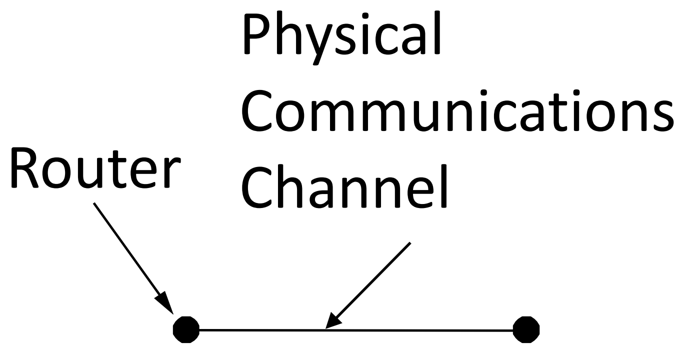
# CSC8004 Computer Networks

# The Transport Layer

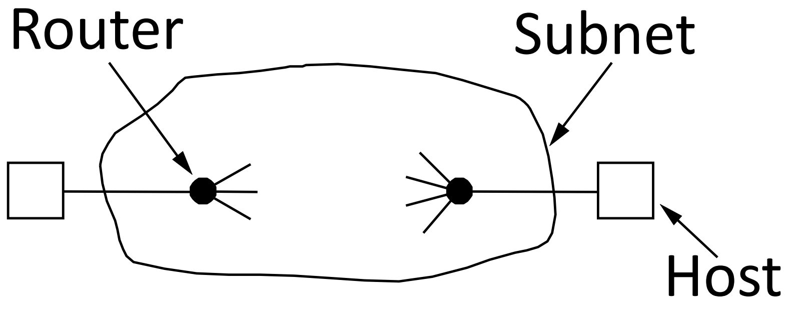
Ellis Solaiman

## Comparison with datalink layer

- Many of the functions of the Transport layer are in principle similar to those in the Data Link layer, but are subtly different, because a complex network may be sitting in between the communicating hosts



**Data Link Layer**



**Transport Layer**

## Features

Basic features of the Transport Layer:

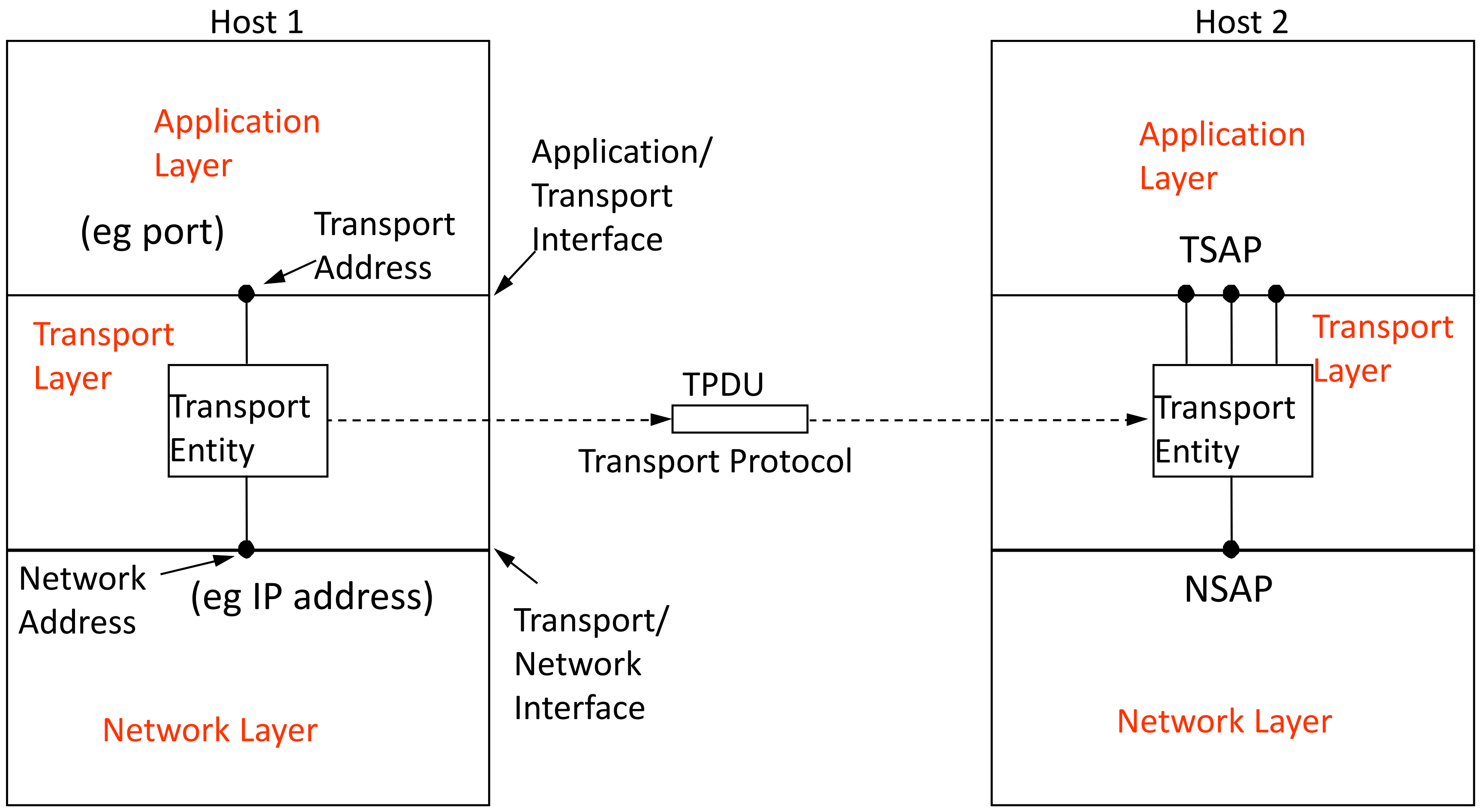
- The transport layer is responsible for providing a **reliable end-to-end connection** between two application processes anywhere on the network
- The physical subnet has been completely abstracted away
- This is first layer that does not involve intermediate nodes

## Features (2)

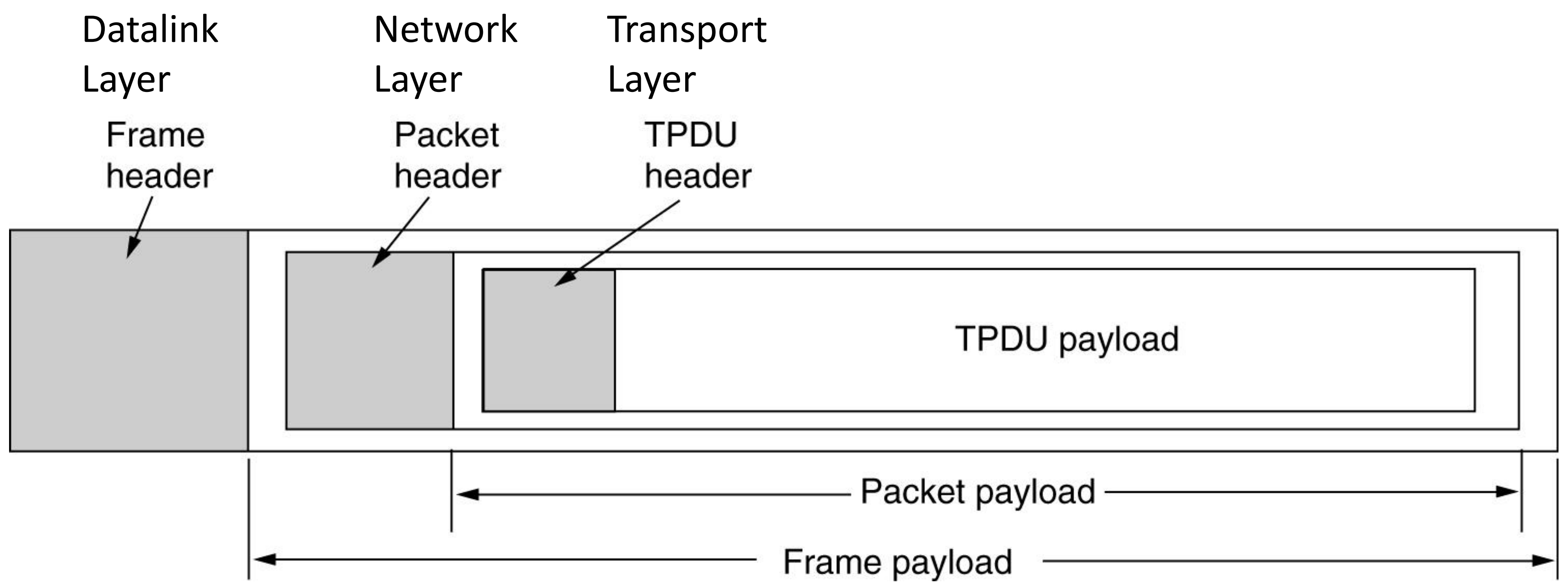
- This is similar to Datalink Layer so why do we need a separate Transport Layer?
- The Network layer is part of the communication subnet and is run by the communications providers. The transport layer runs only on the communicating hosts

    - The Network layer may offer a service which is unreliable. The users have no control over the subnet, so the only possibility to improve the quality of service is to put another layer on top of the Network layer

# Relationship to other layers



# Transport Protocol Data Unit (TPDU)



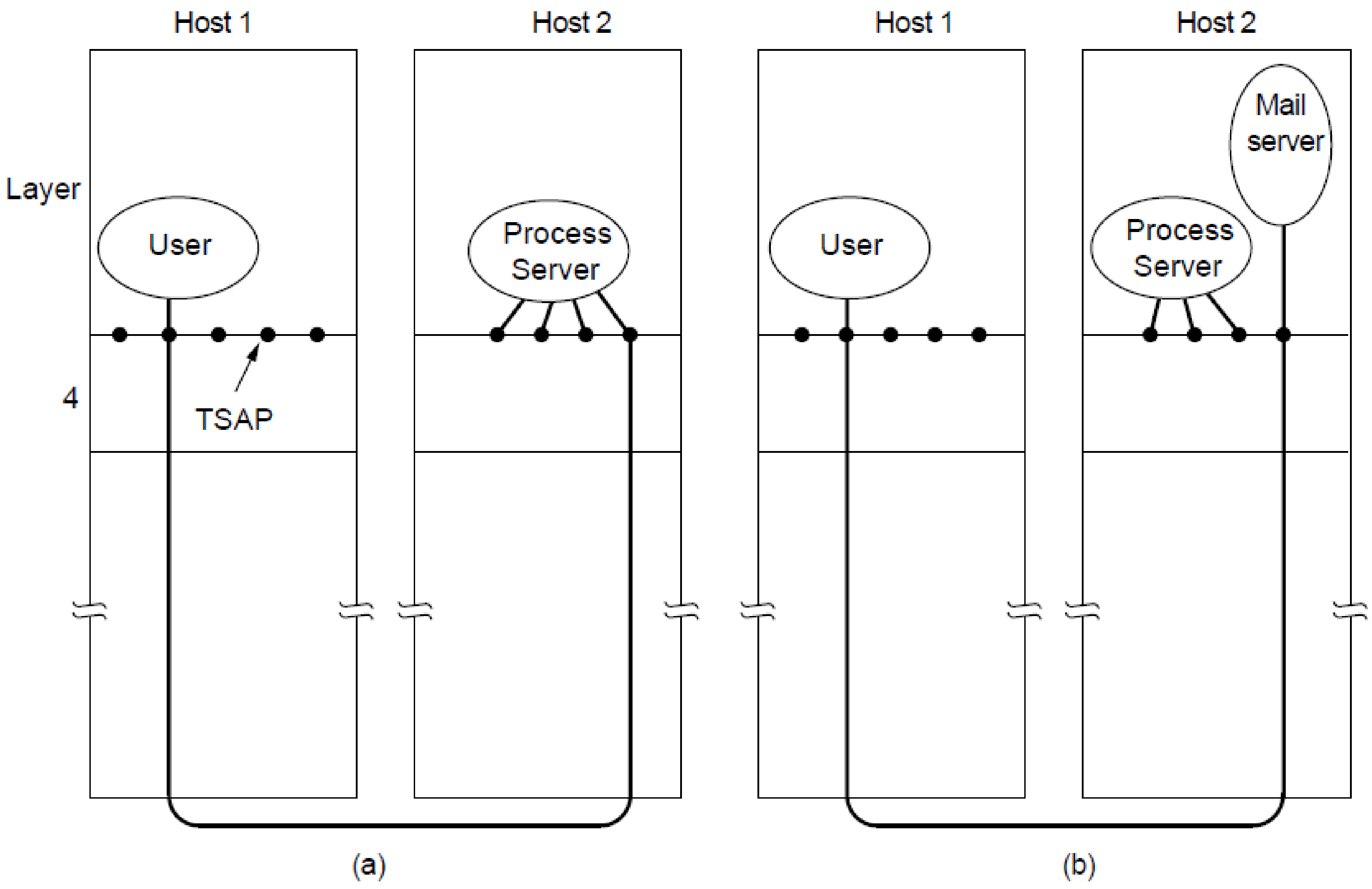
# Connection

- 3-way handshake
  - Host A transmits a new connection request which is accompanied by its initial sequence number x
  - When host B receives the request, it sends host A a connection acceptance acknowledging A’s sequence number x and initiates its own new initial sequence number y
  - Host A acknowledges the acceptance with B’s sequence number y
  - x becomes the starting sequence number for data sent by Host A
  - y becomes the starting sequence number for data sent by Host B

## Connection (2)

- Some service processes will always be listening for connection requests on known addresses
- Others may not actually exist when a connection request is made, and will not be listening
- This problem is solved by the use of a process server which exists all the time on every host and is listening to a well known TSAP (Transport Service Access Point)

# Connection (3)



How a user process in host 1 establishes a connection with a mail server in host 2 via a process server.

## Connection (4)

The *Initial Connection Protocol* proceeds as follows:

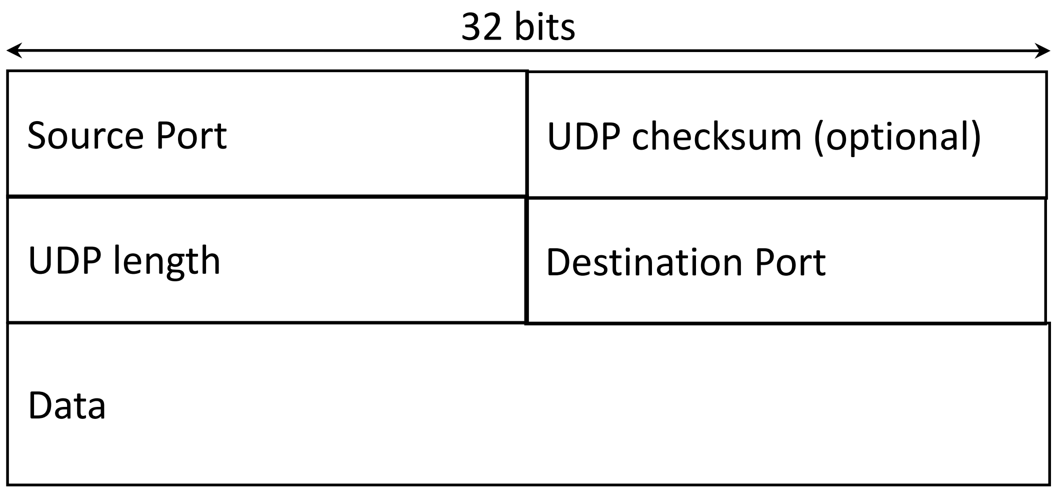
- the caller establishes a connection to the destination process server using its well known TSAP address and asks for the required service by name or TSAP address
- the destination process server creates the required service process, and attaches it to a new TSAP
- the process server returns the new TSAP address to the caller
- the caller now releases the connection to the process server, and establishes a connection to the new service process using the new TSAP address

## Protocols

- The Internet supports two transport layer protocols:
  - The Transmission Control Protocol (TCP) for reliable service (connection oriented)
  - The User Datagram Protocol (UDP) (connectionless)
- **UDP** was designed for “one request, one response” applications (in which case setting up a connection would be too much work)
- **TCP** was specifically designed to be dynamically adjustable to the properties of the Internet (with its many different protocols/technologies) and to be robust in the face of many possible kinds of failures

# UDP

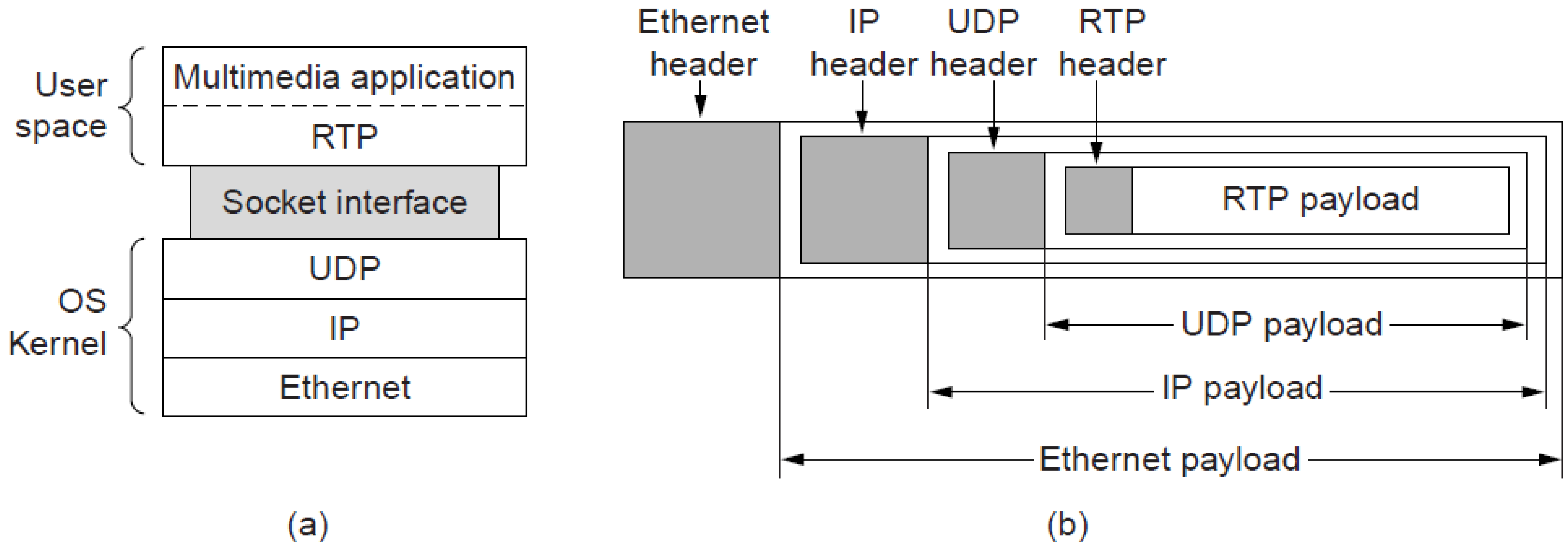
- An unreliable transport protocol
- Almost a “null” transport layer - “IP with extra header”
- Does not provide flow control, error control, connection management, guaranteed in-order packet delivery



## UDP (2)

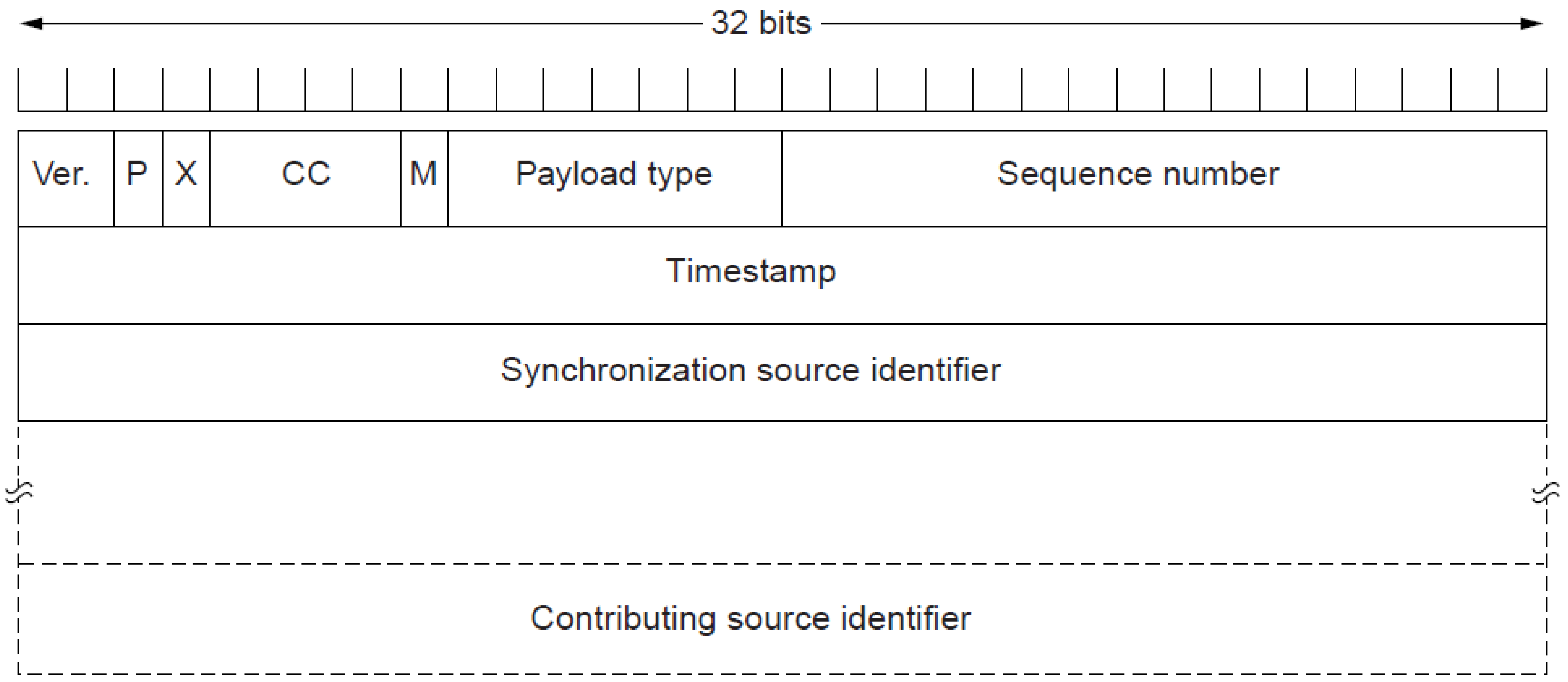
- No connection needs to be set up
- Throughput may be higher because UDP packets are easier to process, especially at the source
- The user doesn’t care if the data is transmitted reliably or not
- The users wants to implement their own transport protocol
- UDP is not a very “popular” protocol
- Many machines are configured to reject UDP traffic that seeks out non-standard ports (to prevent hacking)

# Real-Time Transport (1)



(a) The position of RTP in the protocol stack. (b) Packet nesting.

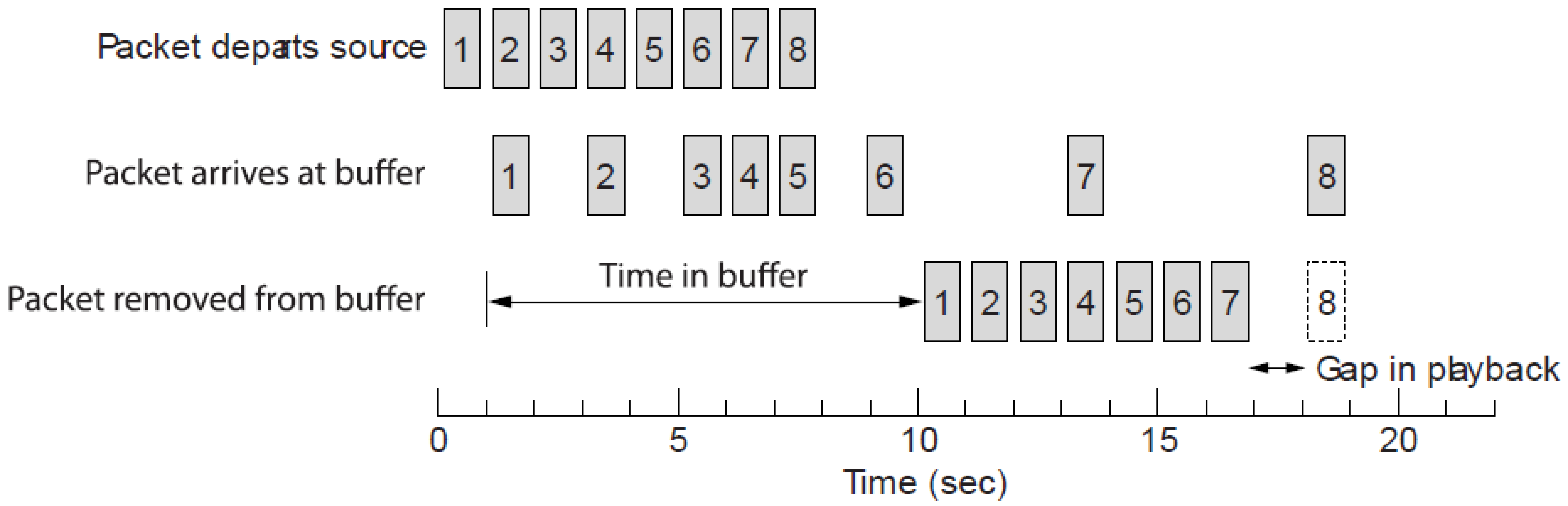
# Real-Time Transport (2)


32 bits
Ver.
P
X
CC
M
Payload type
Sequence number
Timestamp
Synchronization source identifier
Contributing source identifier

The RTP header

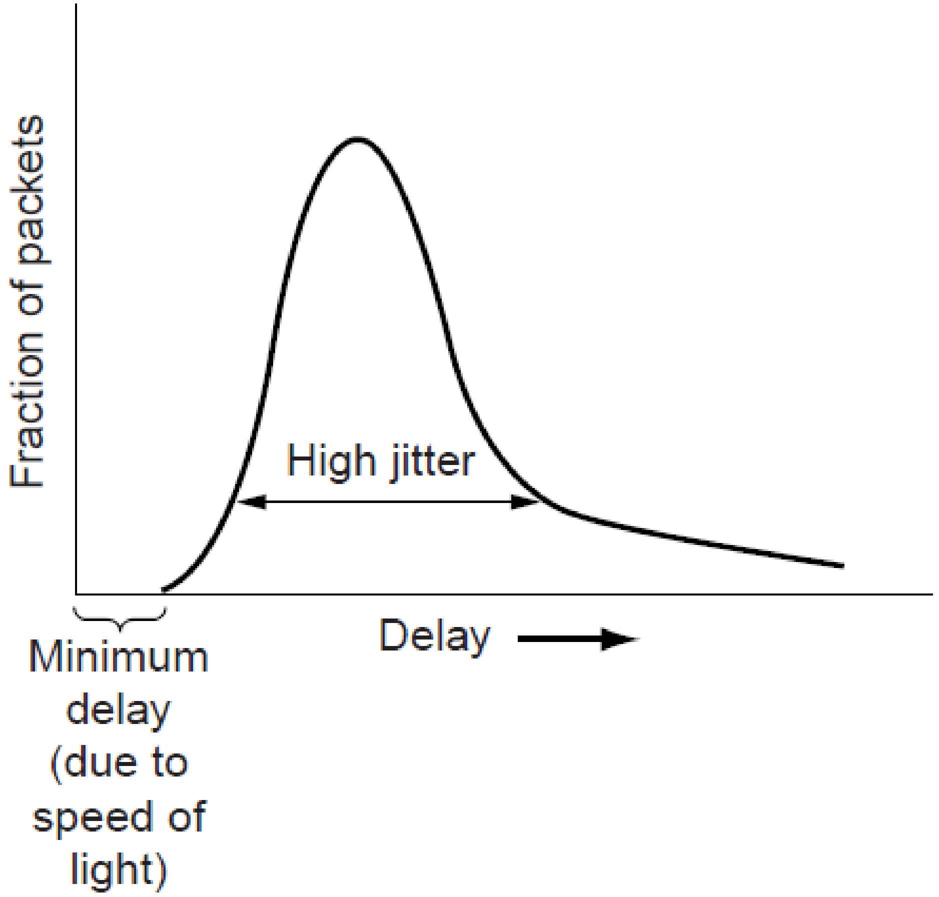
(Realtime Transport Protocol)

# Real-Time Transport (3)



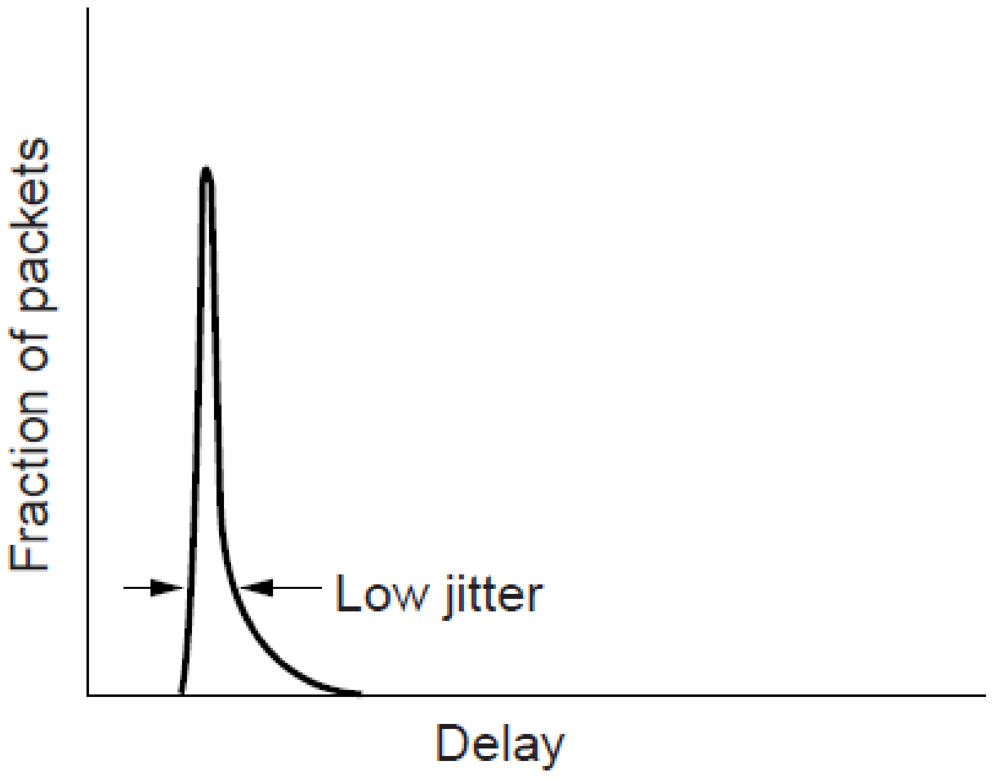
Smoothing the output stream by buffering packets

# Real-Time Transport (4)



High jitter

## Real-Time Transport (5)



Low jitter

## TCP

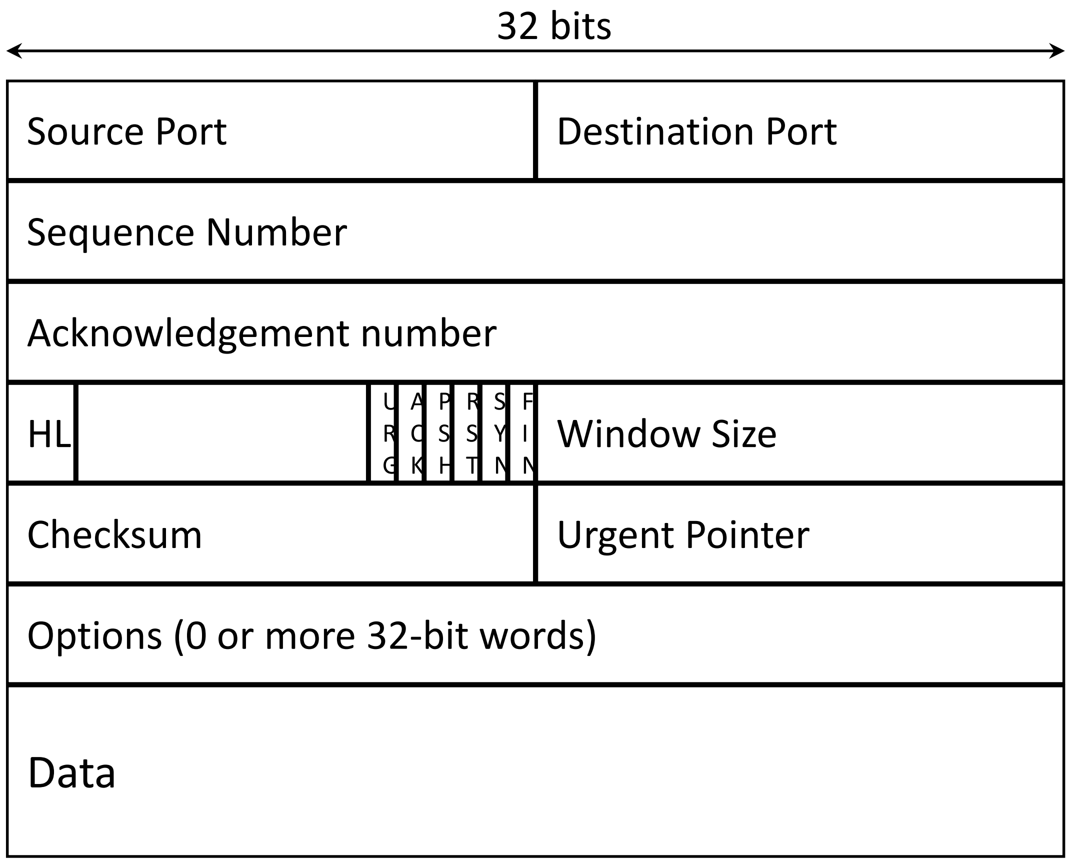
- Provides an end-to-end reliable connection
- The TCP TSAP is known as the socket
- A socket is an IP address and a port number – together they identify a process to be connected to
  - IP address used by the network layer to identify the host
  - Port number used by the transport layer to identify the process on that host
- Port numbers less than 1024 reserved for standard services
  - eg email (port 25) [http://www.iana.org/assignments/port-numbers](http://www.iana.org/assignments/port-numbers)
- A socket to socket connection may be viewed as a bi-directional full-duplex point-to-point connection between two hosts (TCP does not support broadcasting)
- Two or more connections may terminate at the same socket
- TCP uses 3-way handshake techniques for connection/disconnection, in conjunction with a sliding window protocol for flow control

## The TCP Service Model (1)

| Port | Protocol | Use |
|---|---|---|
| 20, 21 | FTP | File transfer |
| 22 | SSH | Remote login, replacement for Telnet |
| 25 | SMTP | Email |
| 80 | HTTP | World Wide Web |
| 110 | POP-3 | Remote email access |
| 143 | IMAP | Remote email access |
| 443 | HTTPS | Secure Web (HTTP over SSL/TLS) |
| 543 | RTSP | Media player control |
| 631 | IPP | Printer sharing |

Some assigned ports

# TCP segment format



- A TCP data unit is called a segment
- Each network imposes its own maximum segment size
- Segments may be fragmented

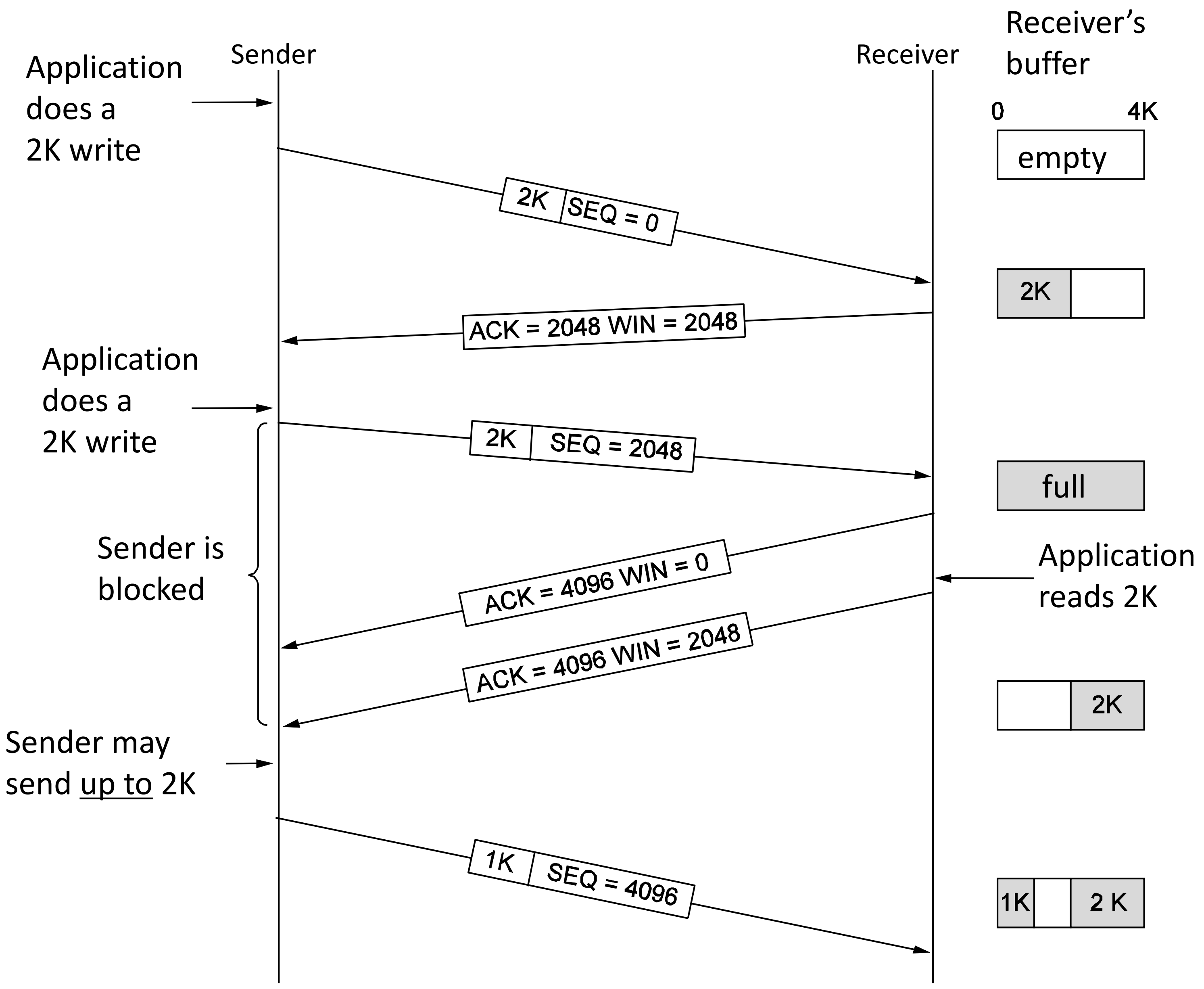
## TCP segment format (2)

- Source & Destination Ports: 16 bit port identifiers for each packet
- Sequence number: The packet’s unique sequence ID
- Acknowledgement number: The sequence number of the next byte expected by the receiver
- Window size: Specifies the size of free space in the receiver’s sliding window
- Checksum: Checksums the TCP header and IP address fields.
- Urgent Pointer: Points to urgent data in the TCP data field (for example, when user has pressed ^C)

## TCP flow control

- TCP uses a modified version of sliding window - window sizes may change dynamically
- In acknowledgements
  - Window size tells the sender how many bytes it may transmit
  - Ack field tells the sender which byte to transmit next
- Retransmissions may have different sizes from the original transmission, because more or less buffer space may be available

# TCP flow control (2)



## Flow control

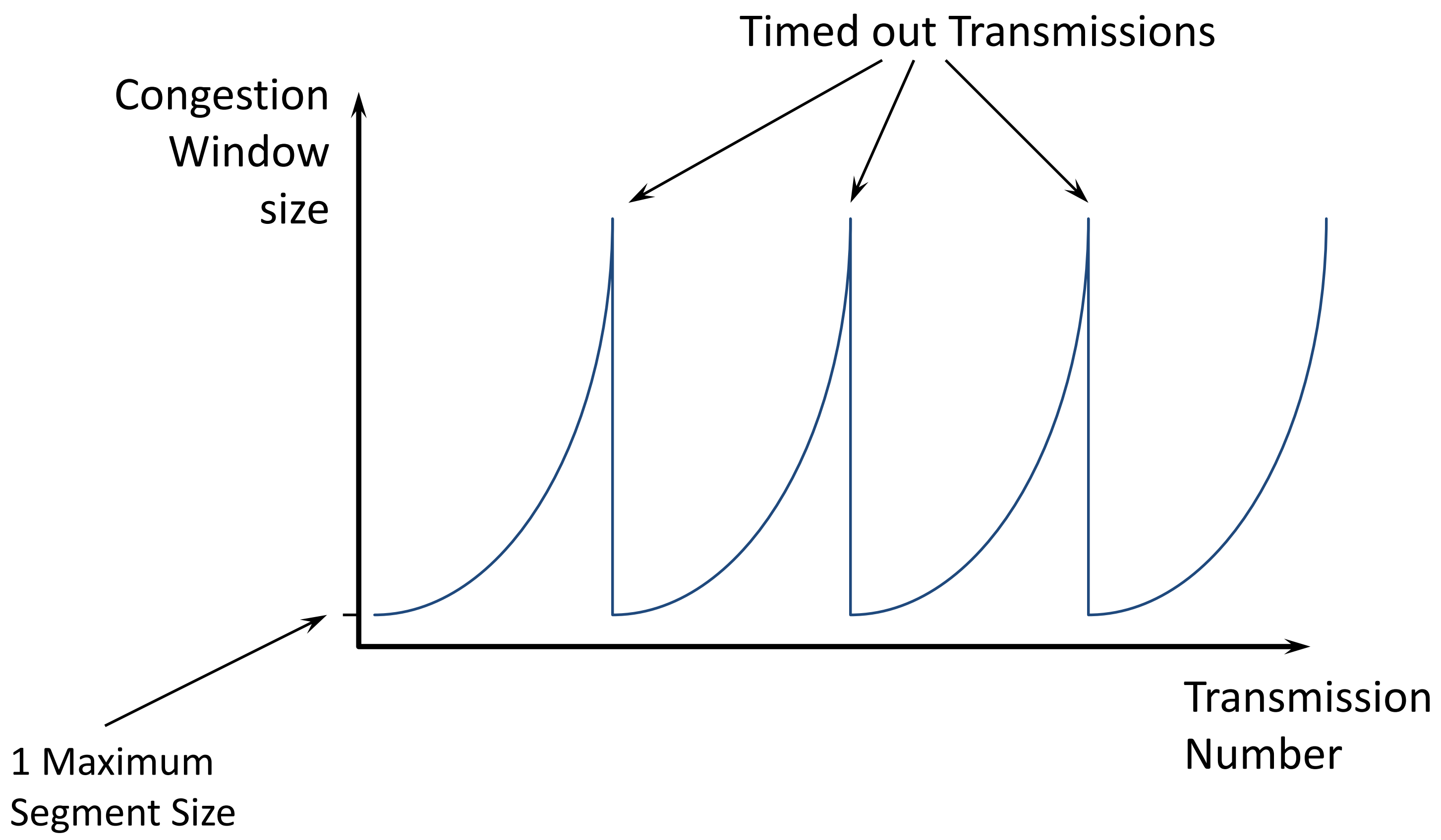
- TCP operates a ***sliding window*** mechanism where each acknowledge includes the amount of data the receiver is now willing to accept.
- A number of measures are commonly employed to ensure that the window size is roughly a multiple of the senders desired segment size.
- Different flow control schemes are one of the main differences between different versions of TCP used today (there are several).

## TCP congestion control - slow start

Start small and gradually build up the traffic

- Congestion window starts at 1 max segment (TPDU) size for network
- The congestion window size is increased exponentially
    - increased by 1 max segment size each time a segment ack is received
    - effectively doubling each time a complete window is transmitted
- Eventually packets will be lost, indicating congestion
- The congestion window is then reduced back to max segment size and everything starts all over again
- TCP Slow Start by itself is inefficient
- Although the congestion window builds exponentially, it drops suddenly every time a packet times out
- This leads to low throughput.

## TCP congestion control - slow start (2)

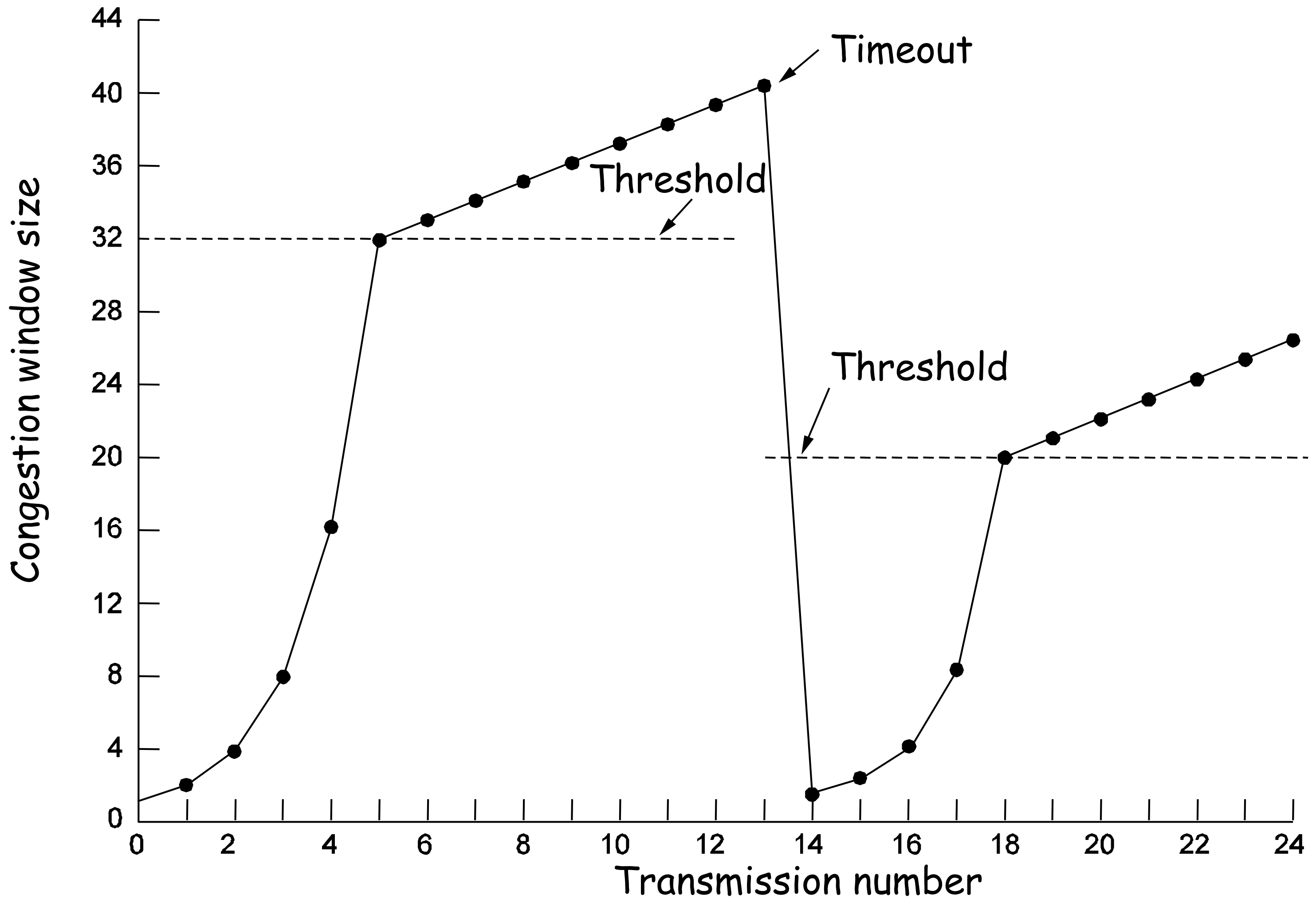


# TCP congestion control - threshold

Establish a threshold at which the rate increase is linear instead of exponential, to improve efficiency

- Start the congestion window size at 1 max segment size
- Start the threshold at (eg) 32K
- Increase the congestion window size exponentially (using slow start) until the threshold is reached
- Once the threshold is passed, only increase the congestion window size linearly
  - Increased by 1 max segment size each time a complete window is acked
- If a timeout occurs, reset threshold to 1/2 of the current congestion window size then reset the congestion window size to 1 segment

# TCP congestion control – threshold (2)


Timeout
Threshold
Threshold
Congestion window size
Transmission number