CS 250: FOUNDATIONS OF COMPUTER SYSTEMS
[NETWORKING]

Computer Science
Colorado State University

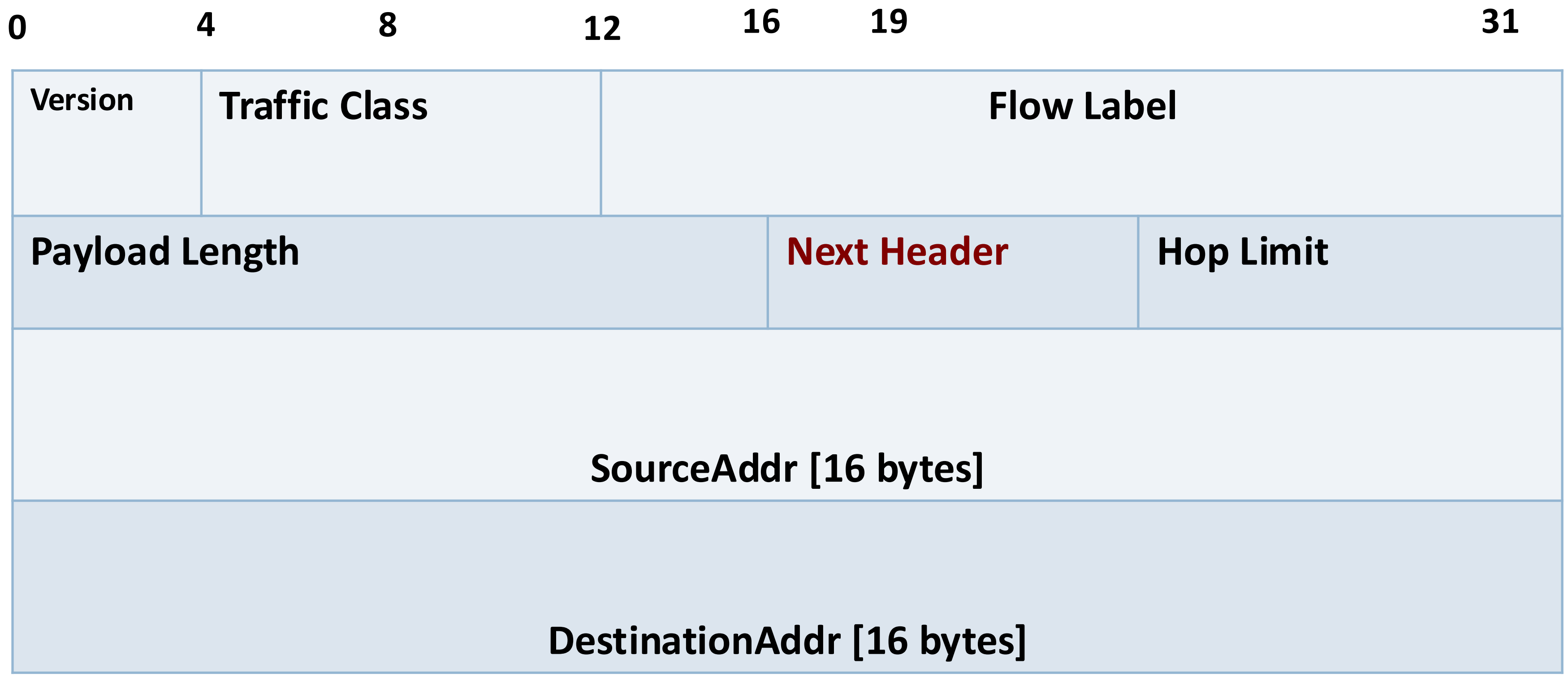
** Lecture slides created by: SHRIDEEP PALLICKARA

Topics covered in this lecture

- IPv6 (wrap-up)
- UDP
- TCP

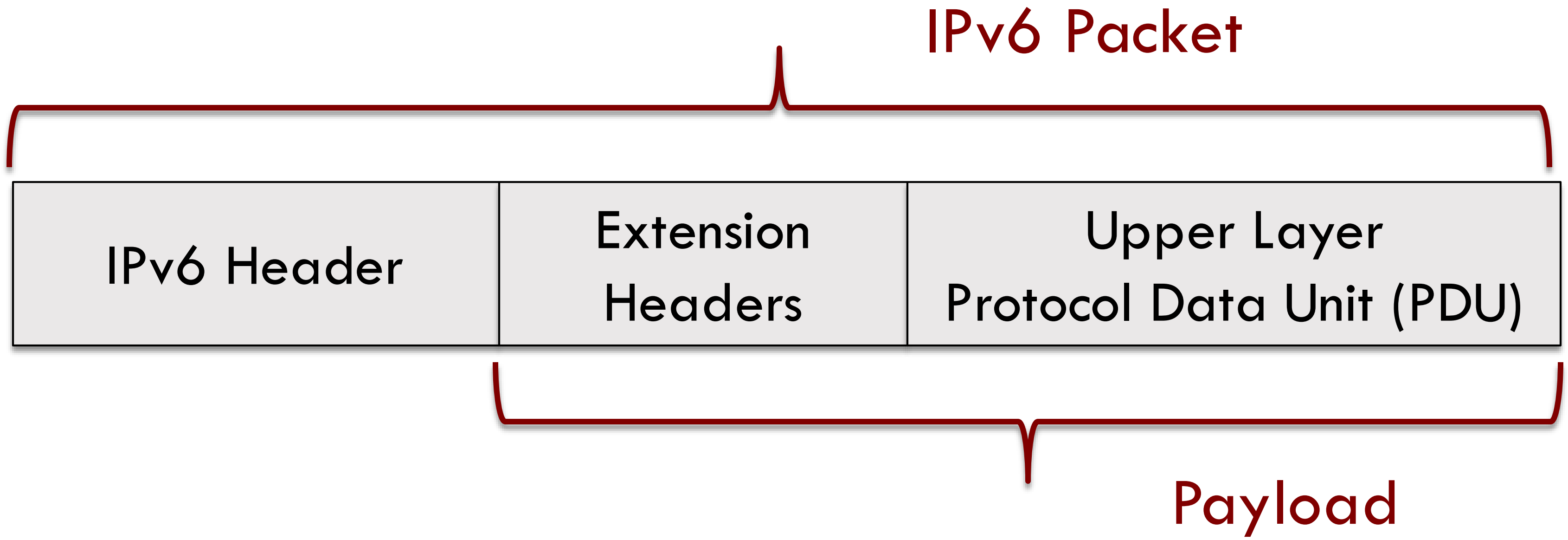
IPv6 (wrap-up)

IPv6 Packet Header



IPv6 Packet Header is fixed at 40 bytes … So there is no Header Length

Structure of the IPv6 Packet



PDU typically contains an upper layer protocol header and its payload.
For e.g.: a TCP segment, UDP Datagram, or an ICMPv6 message

Extension Header [1/2]

- If the `Next Header` field is non-zero
  - It defines an extension header
- Current extension header types
  - Information for routers, route definition, fragment handling, authentication, encryption, etc.
- Each extension header has a specific size and defined format

Extension Header [2/2]

- If an extension header is present?
  - **Follows** the basic header and **precedes** the payload AND
  - Includes a `Next Header`
- Every extension header starts off with the `Next Header`

IPv6 Extension Headers: The chain of pointers using the Next Header field

Each extension header must fall on a **64-bit (8-byte) boundary.** Use Padding to get there if less than that.

IPv6 Header Next Header=6 (TCP)	TCP Segment

IPv6 Header Next Header=43 (Routing)	Routing Header Next Header=6 (TCP)	TCP Segment

IPv6 Header Next Header=43 (Routing)	Routing Header Next Header=51 (AH)	Authentication Header Next Header=6 (TCP)	TCP Segment

Fragmentation Header: 44

UDP

Simple Demultiplexer

User Datagram Protocol

- **Simplest** possible transport protocol
  - Extends host-to-host into process-to-process communications
- No additional functionality to best-effort service provided by underlying network
- Adds **demultiplexing**
  - Allows applications on a host to **share** the service

UDP identification of processes

- Processes *indirectly* indentify each other
  - Abstract locator called **port**
- Source sends a message to a port
  - Destination receives messages from a port
- Process is identified by a port on a particular host

Format of a UDP header

0	16	31

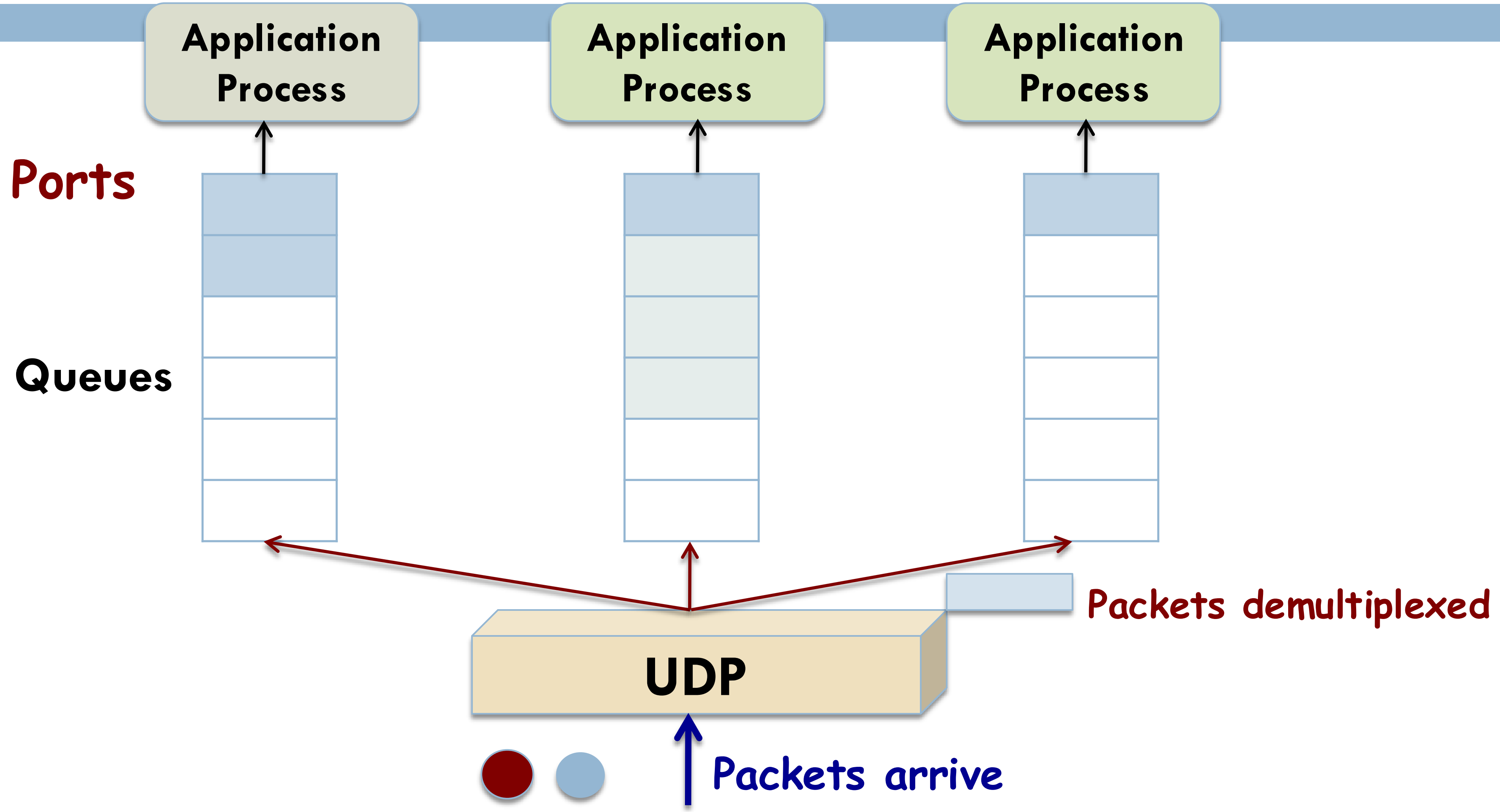
SrcPort	**DstPort**
Length	**Checksum**
Data	

A port is just an abstraction

- Typically implemented as a **message queue**
- When message arrives?
  - Protocol *appends* message to end of the queue

- **UDP**
  - If the queue is full, message is discarded
  - No flow-control mechanism

UDP message queue: The port abstraction



Some work that UDP does do besides demultiplexing: Checksumming

- UDP header
- Message body
- ***Psuedoheader***: From the IP header
  - Protocol number
  - Source IP address
  - Destination IP address

  } **Verify if message is delivered between the correct endpoints**
- UDP length
  - Used *twice*

Reliable Byte Stream
TCP (Transmission Control Protocol)

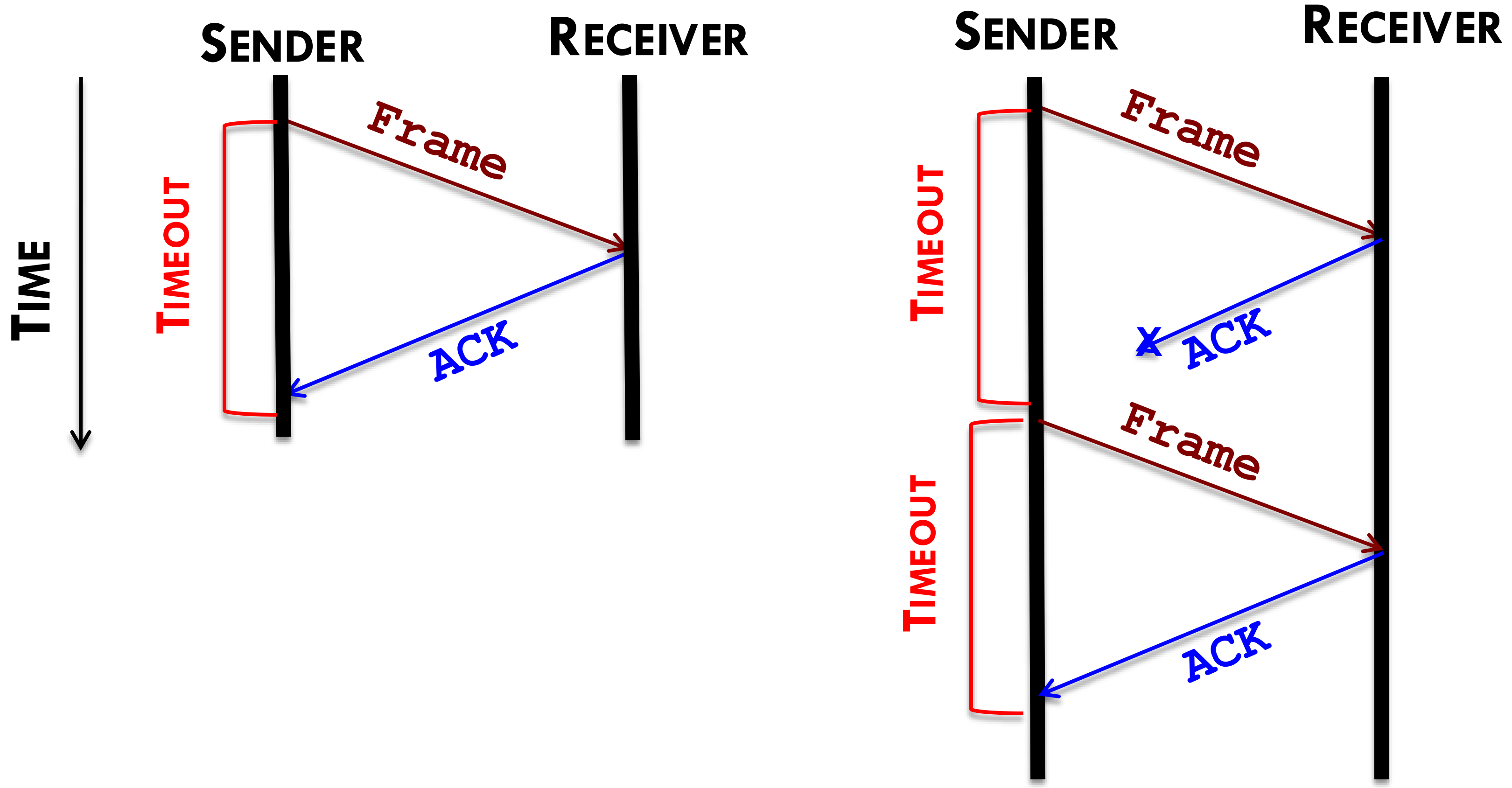
Components of Reliable delivery

- **Acknowledgements**
  - Confirm receipt of data [with an ACK]
- **Timeouts**
  - *Retransmit* if ACK not received within a specified time
- Use of ACKs and timeouts to implement reliable delivery
  - Sometime called ARQ (**a**utomatic **r**epeat re**q**uest)

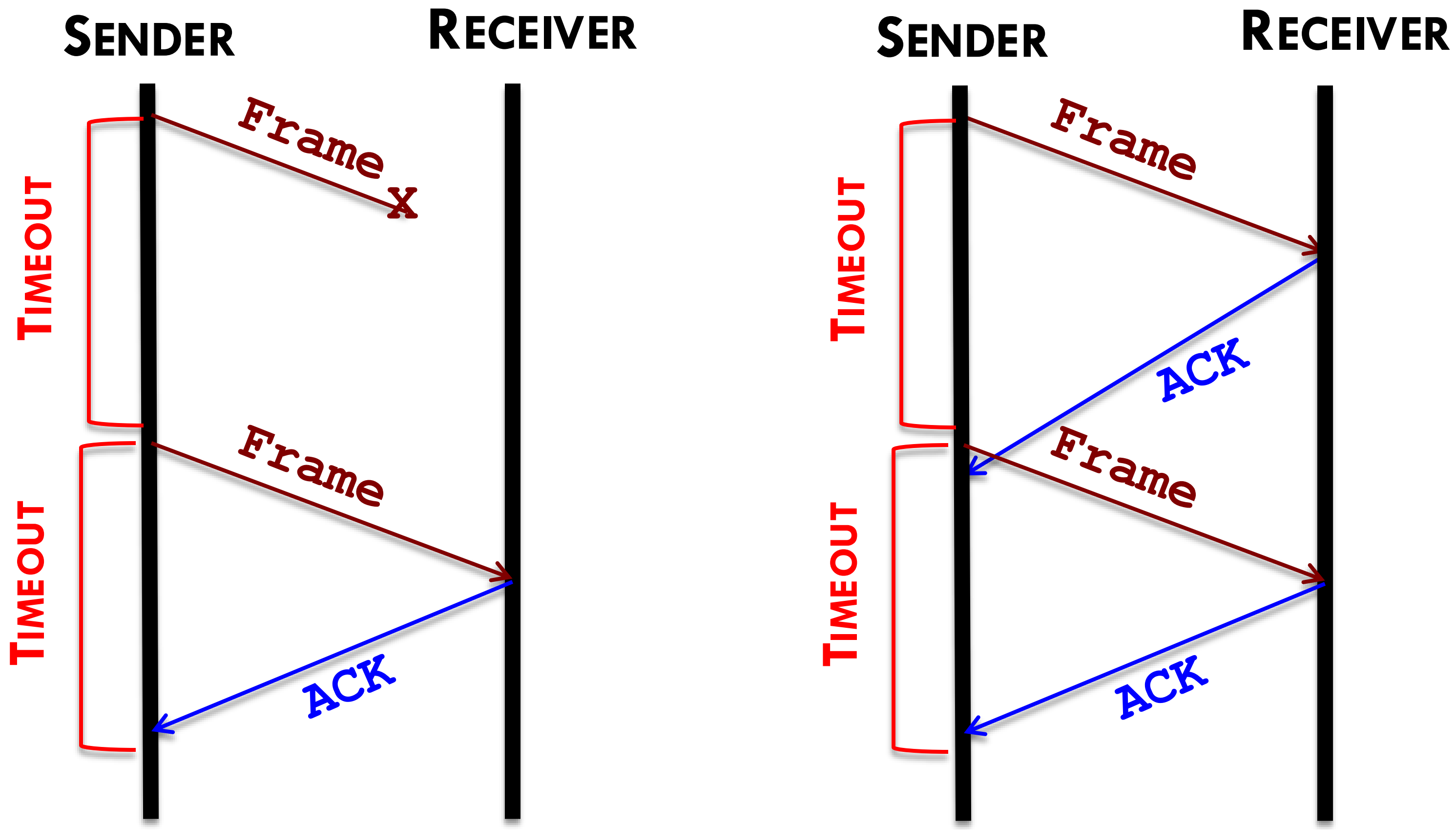
Simplest ARQ is the stop-and-wait algorithm

- After transmitting one frame
  - Sender **waits** for ACK before transmitting the next frame
- If the ACK does not arrive after a period of time
  - Sender **retransmits** the original frame

Stop-and-wait



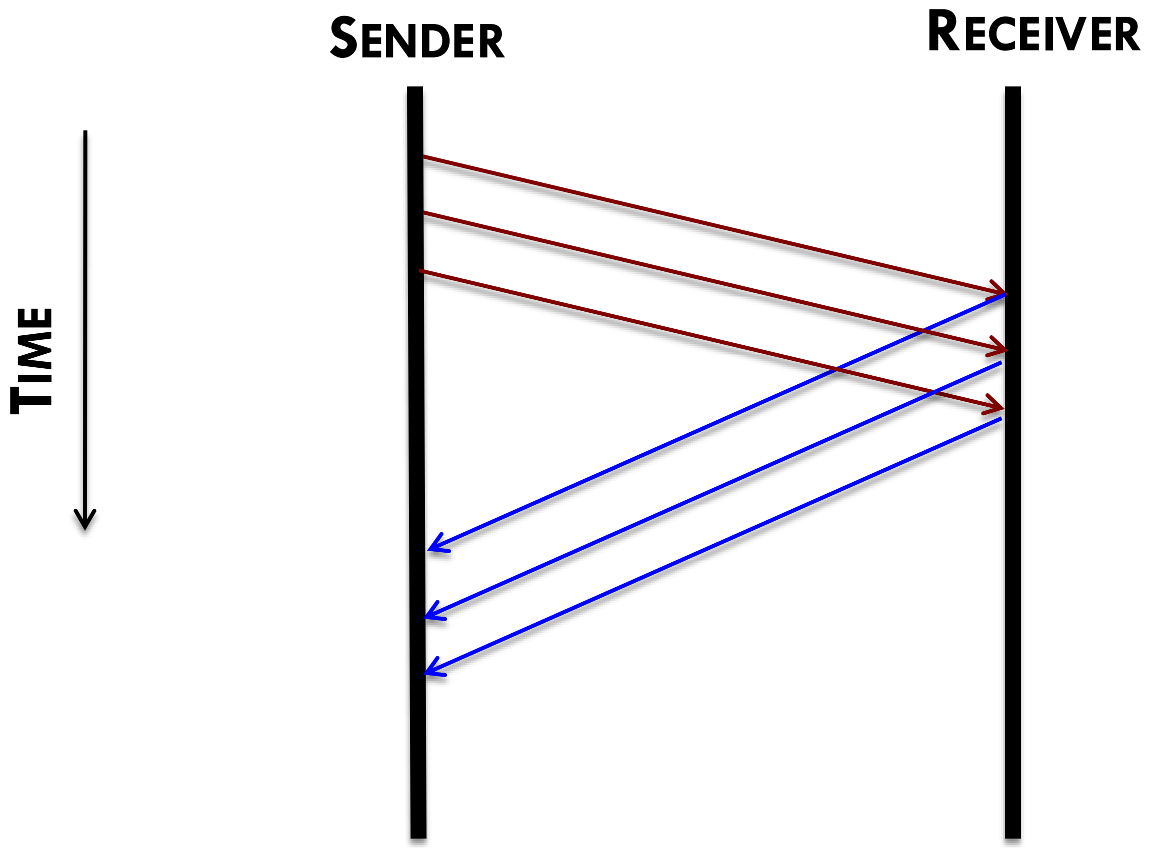
Stop-and-wait [2/2]



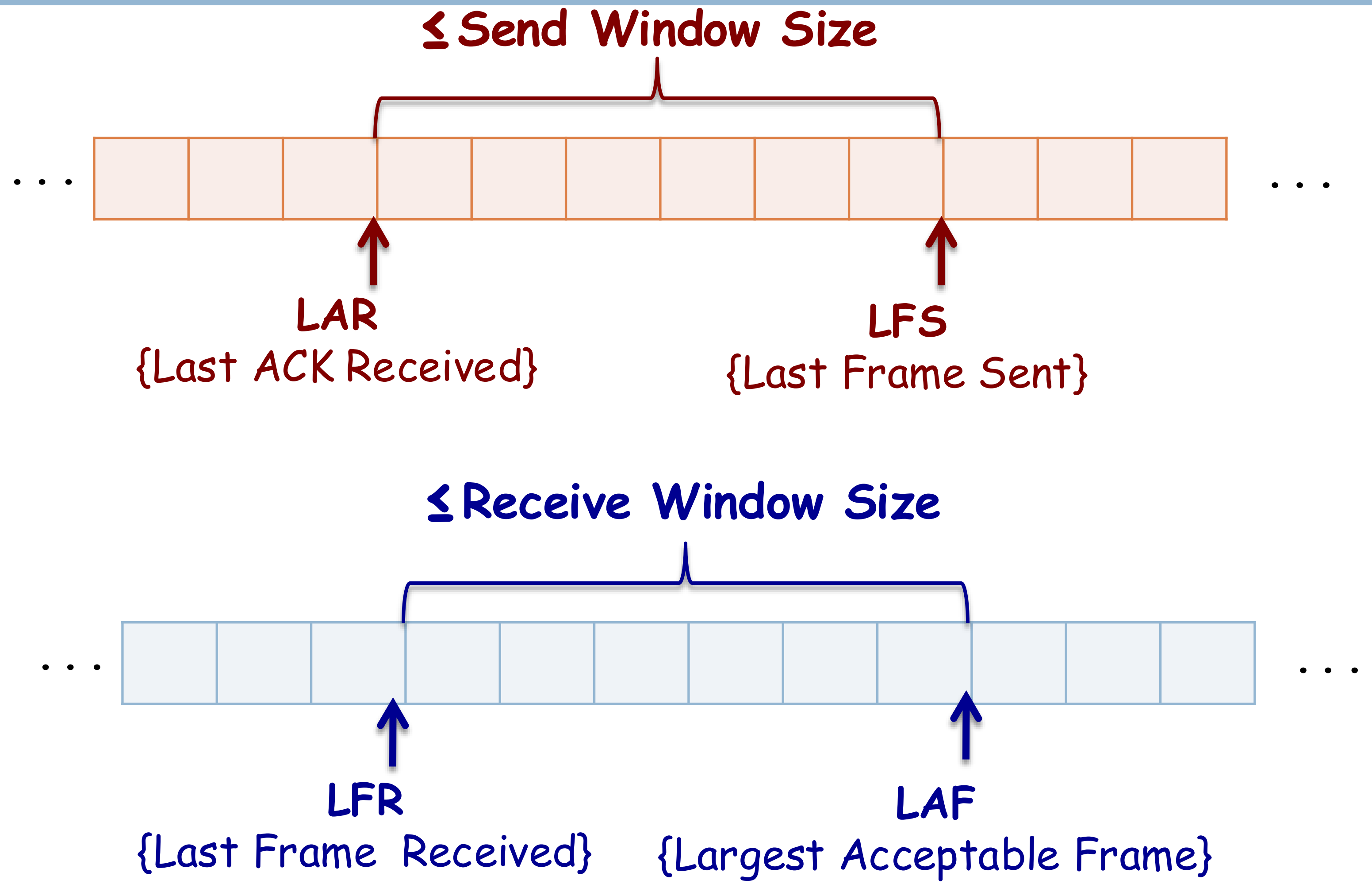
Sliding window: Try to fill the network pipe

- DELAY x BANDWIDTH product is 8 KB
- Data frames = 1KB
- Sender could transmit 9^{th} frame
  - When ACK for the 1^{st} frame arrives

Timeline for the sliding window



Sliding Window on Sender/Receiver



Transmission Control Protocol (TCP) [1/2]

- **Reliable, in-order** delivery of byte streams
- **Full duplex** protocol
  - Each connection supports a pair of byte streams
    - Flowing in different directions
- Includes **flow control** mechanism
  - Allows receiver to limit the data sender
    - Control how much data can be transmitted at a time

Transmission Control Protocol (TCP) [2/2]

- Includes **multiplexing** mechanism
  - Multiple applications on a given host

- Implements a **congestion-control** mechanism
  1. *Throttle* how fast TCP sends data
  2. Keep sender from *overloading* the network

Flow control and congestion control

- **Flow control** is an end-to-end issue
  - Don't overrun capacity of *receiver*
- **Congestion control** is about how hosts & networks interact
  - Don't cause *switches* and *links* to be overloaded

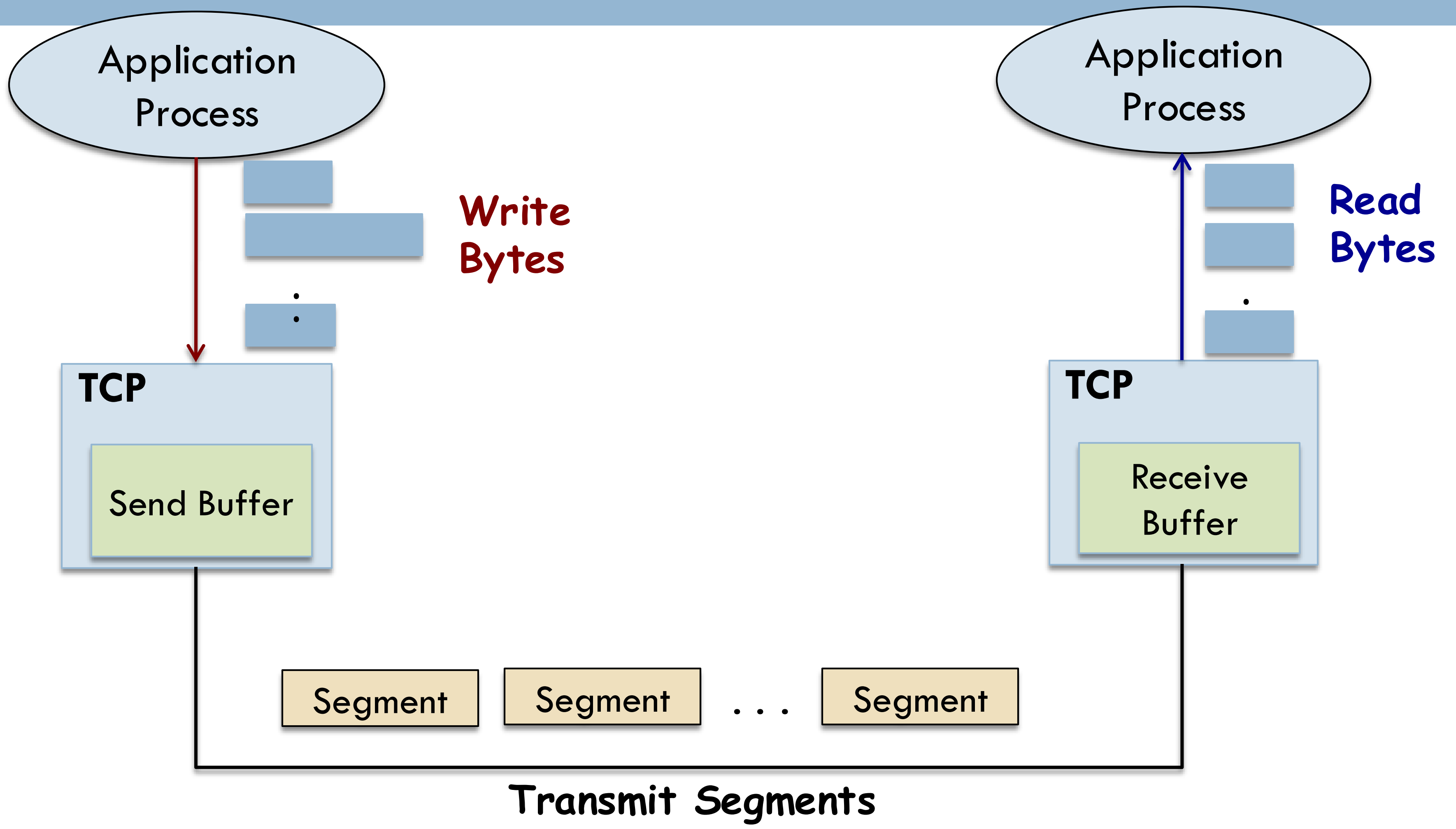
TCP: Setup and Teardown

- Two sides of the connection *agree* to exchange data
  - Establish **shared state**
  - 3 packets exchanged (SYN, SYN-ACK, ACK)
- Connection teardown
  - Let each host know it is OK to *free* the shared state
  - 4 packets exchanged (FIN, ACK, FIN, ACK)

TCP Segments & how they come about

- TCP
  - Accepts data from a data stream
  - Breaks it up into chunks
  - Adds a TCP header … creating a **TCP segment**
- Segment is then *encapsulated* in an IP datagram
- TCP packet is a term that you will often hear
  - Segment is more precise, packets are generally datagrams, frames are at the link layer

How TCP manages a byte stream



Maximum Segment Size (MSS)

- To avoid fragmentation in the IP layer, a host must specify the MSS as equal to the largest IP datagram that the host can handle **minus** (the IP and TCP header sizes)
- The **minimum** requirements (in bytes) at the hosts are as follows
  - IPv4: 576 – 20 – 20 = 536
  - IPv6: 1280 – 40 – 20 = 1220
- Each direction of the data flow can use a different MSS

TCP Header Format

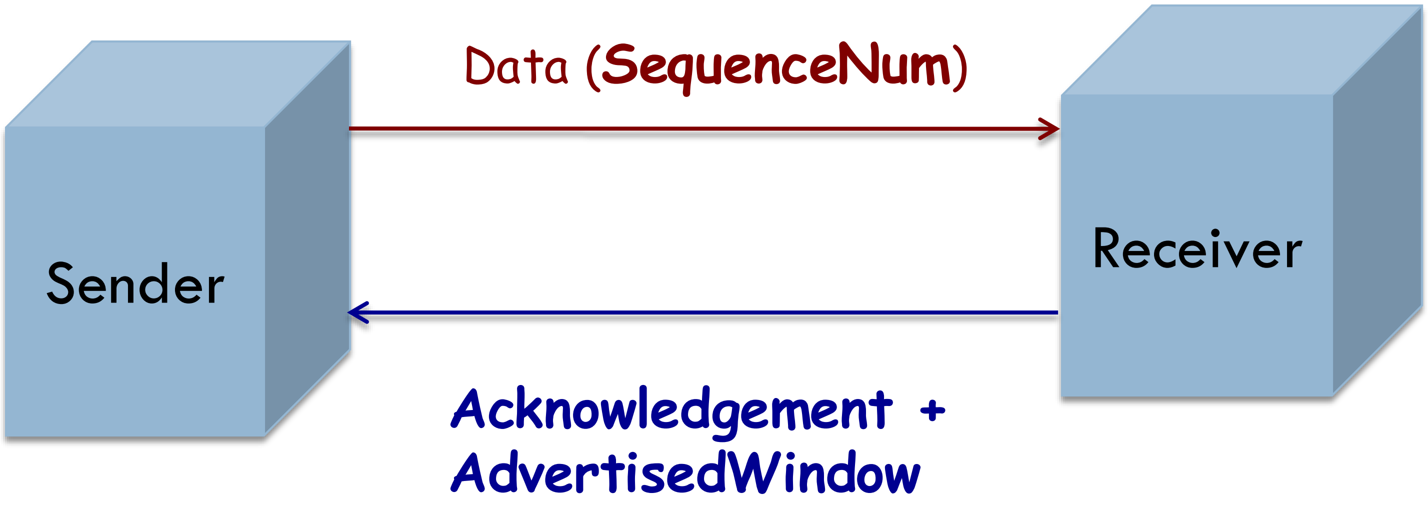
<table>
<tr><th>0</th><th>4</th><th>10</th><th>16 – 31</th></tr>
<tr><td colspan="3">SrcPort</td><td>DestPort</td></tr>
<tr><td colspan="4">SequenceNum</td></tr>
<tr><td colspan="4">Acknowledgement</td></tr>
<tr><td>HdrLen</td><td>0</td><td>Flags</td><td>AdvertisedWindow</td></tr>
<tr><td colspan="3">Checksum</td><td>UrgPtr</td></tr>
<tr><td colspan="4">Options (variable)</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

Sliding Window Protocol

Extensions

SourceAddr and **DestinationAddr** from IP

Relationship between SequenceNum, Acknowledgement and AdvertisedWindow



Each byte of data has a sequence number

SequenceNum contains sequence number for first byte of data in segment

TCP Sliding Window [1/2]

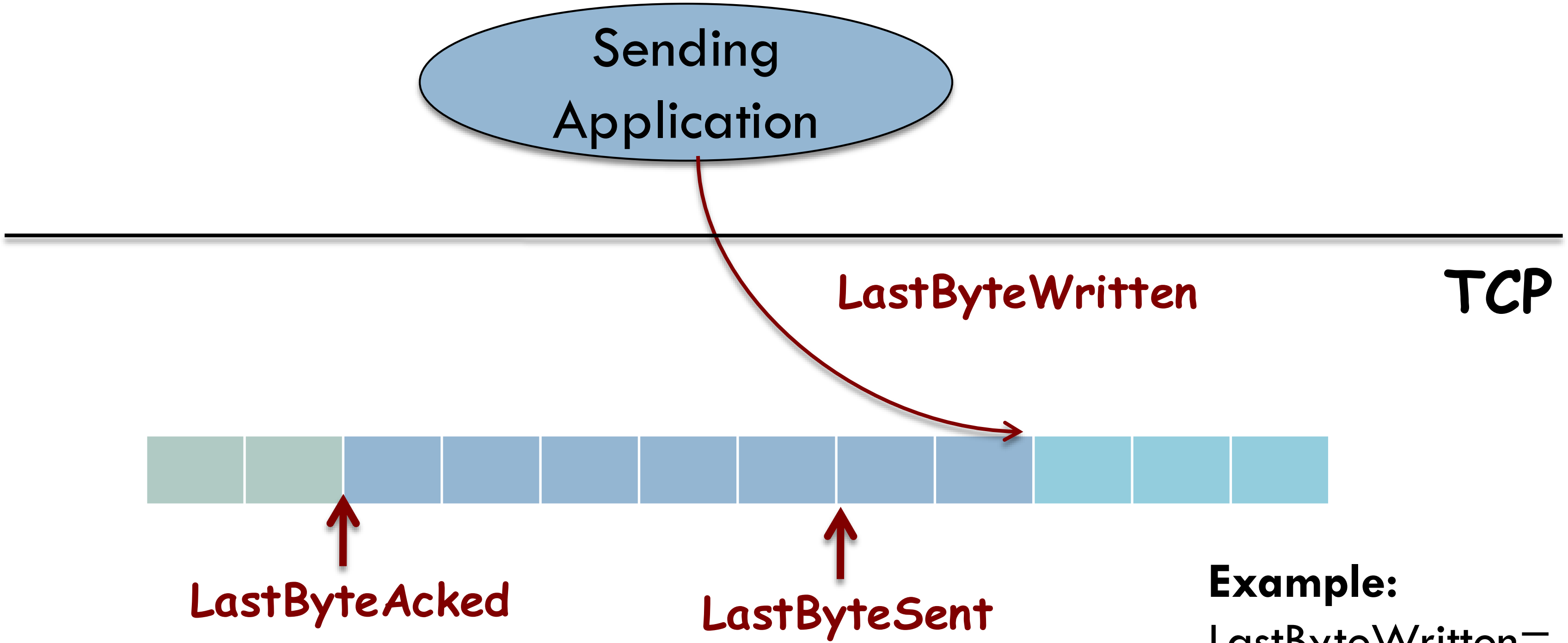
- Guarantees **reliable** delivery of data
- Data is delivered in **order**
- Enforces **flow control** between the sender and receiver

TCP Sliding Window [2/2]

- Sender has a **limit** on unacknowledged data
  - Limited to no more than **`AdvertisedWindow`** bytes of unacknowledged data

- Receiver **selects** **`AdvertisedWindow`**
  - Based on memory set aside for connection's buffer space

TCP Send Buffer



LastByteAcked ≤ LastByteSent

LastByteSent ≤ LastByteWritten

Example:

LastByteWritten= 3000

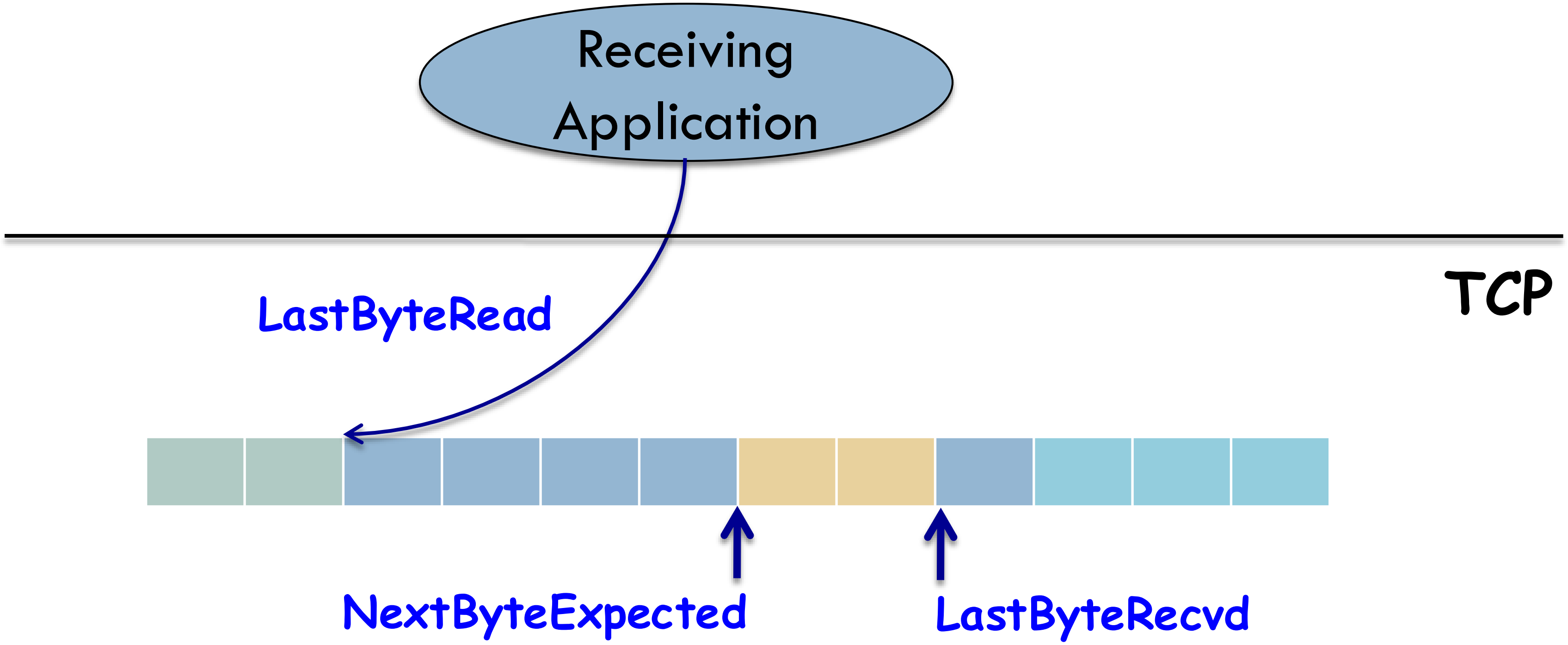
LastByteSent=2800

LastByteAcked:= 2400

How many unacknowledged bytes?

(3000-2400)= 600

TCP Receive Buffer



LastByteRead < NextByteExpected

NextByteExpected ≤ LastByteRecvd + 1

Flow Control: Buffers are of finite size

`MaxSendBuffer` and `MaxRcvBuffer`

- Receiver **throttles** sender
  - Advertises a window
  - No bigger than what it can buffer

LastByteRcvd – LastByteRead ≤ MaxRcvBuffer

AdvertisedWindow =

MaxRcvBuffer − **(** (NextByteExpected -1) − LastByteRead)**)**

Space Utilized in the receiver's buffer

The advertised window may potentially shrink

- If the process is reading data as fast as it arrives?
  - The advertised window *stays open*
    - i.e. `AdvertisedWindow = MaxRcvBuffer`
- If the receiving process falls behind?
  - Advertised window becomes *smaller* with every segment that arrives
  - Until it becomes 0

Flow Control: Buffers are of finite size MaxSendBuffer and MaxRcvBuffer

- On the sender size, TCP **adheres** to the advertised window from the receiver

LastByteSent – LastByteAcked ≤ AdvertisedWindow

EffectiveWindow =
AdvertisedWindow – (LastByteSent – LastByteAcked)

EffectiveWindow should be > 0 before source can send more data

Reliability is achieved by the sender detecting lost data and retransmitting it

- TCP uses two primary techniques to identify loss
  - Retransmission timeout (RTO)
  - Duplicate cumulative acknowledgements (DupAcks)
    - If the sender receives three duplicate acknowledgements, it retransmits the last unacknowledged packet

Selective Acknowledgements (SACK)

- Using SACK, a receiver informs the sender of **non-contiguous blocks** of data that have been received and queued successfully
- So, the sender need retransmit only the segments that have actually been lost

Issues with TCP

Protecting against wraparound: 32-bit sequence space

- TCP assumes each segment has a max lifetime
  - **Maximum segment lifetime** (MSL)
  - Currently this is 120 seconds
- Sequence number used on a connection might wrap-around
  - Within the MSL

Time until 32-bit sequence number wraps around

Bandwidth	Time until wraparound
T1 (1.5 Mbps)	6.4 hours
Ethernet (10 Mbps)	57 minutes
T3 (45 Mbps)	13 minutes
FDDI (100 mbps)	6 minutes
STS-3 (155 Mbps)	4 minutes
STS-12 (622 Mbps)	55 seconds
STS-24 (1.2 Gbps)	28 seconds

STS : Synchronous Transport Signal

FDDI: Fiber Distributed Data Interface

Keeping the pipe full

- **`AdvertisedWindow`** field (16-bits) must be big enough
  - To allow sender to keep the pipe full
  - 16 bit allows us a max window size of 64 KB (2^{16})

- If receiver has unlimited buffer space?
  - **`AdvertisedWindow`** dictated by DELAY X BANDWIDTH product

Required Window Size for 100 ms delay

Bandwidth	Delay x Bandwidth Product
T1 (1.5 Mbps)	18 KB
Ethernet (10 Mbps)	122 KB
T3 (45 Mbps)	549 KB
FDDI (100 mbps)	1.2 MB
STS-3 (155 Mbps)	1.8 MB
STS-12 (622 Mbps)	7.4 MB
STS-24 (1.2 Gbps)	14.8 MB

STS : Synchronous Transport Signal

FDDI: Fiber Distributed Data Interface

TCP extensions: Use 32-bit timestamp to extend sequence number space

- **Distinguish** between different incarnations of the same sequence number
- Timestamp not treated as part of sequence number
  - For ordering etc.
  - Just protects against wraparound

TCP Extension: Allow TCP to advertise larger window

- Fill larger DELAY x BANDWIDTH pipes
- Include option defining **scaling** factor
- Option allows TCP endpoints to agree that **`AdvertisedWindow`** counts **larger chunks**

A caveat regarding Options

- You cannot solve all problems with Options

- TCP Header has room for only **44 bytes of options**
  - `HdrLen` is 4 bits long, so header length cannot exceed 16 x 32-bit = 64 bytes
  - Adding a TCP option that extends the space available for options?

OSI Network Architecture

OSI network architecture

- Model is a product of the Open Systems Interconnection (OSI) project
  - At the International Organization for Standardization (ISO)
- Partitions network functionality into **7 layers**
- Physical Layer
  - Handles transmission of **raw bits**
  - Standardizes electrical, mechanical, and signaling interfaces
    - **0** bit should be received as **0** not **1**

OSI network architecture: Data link Layer

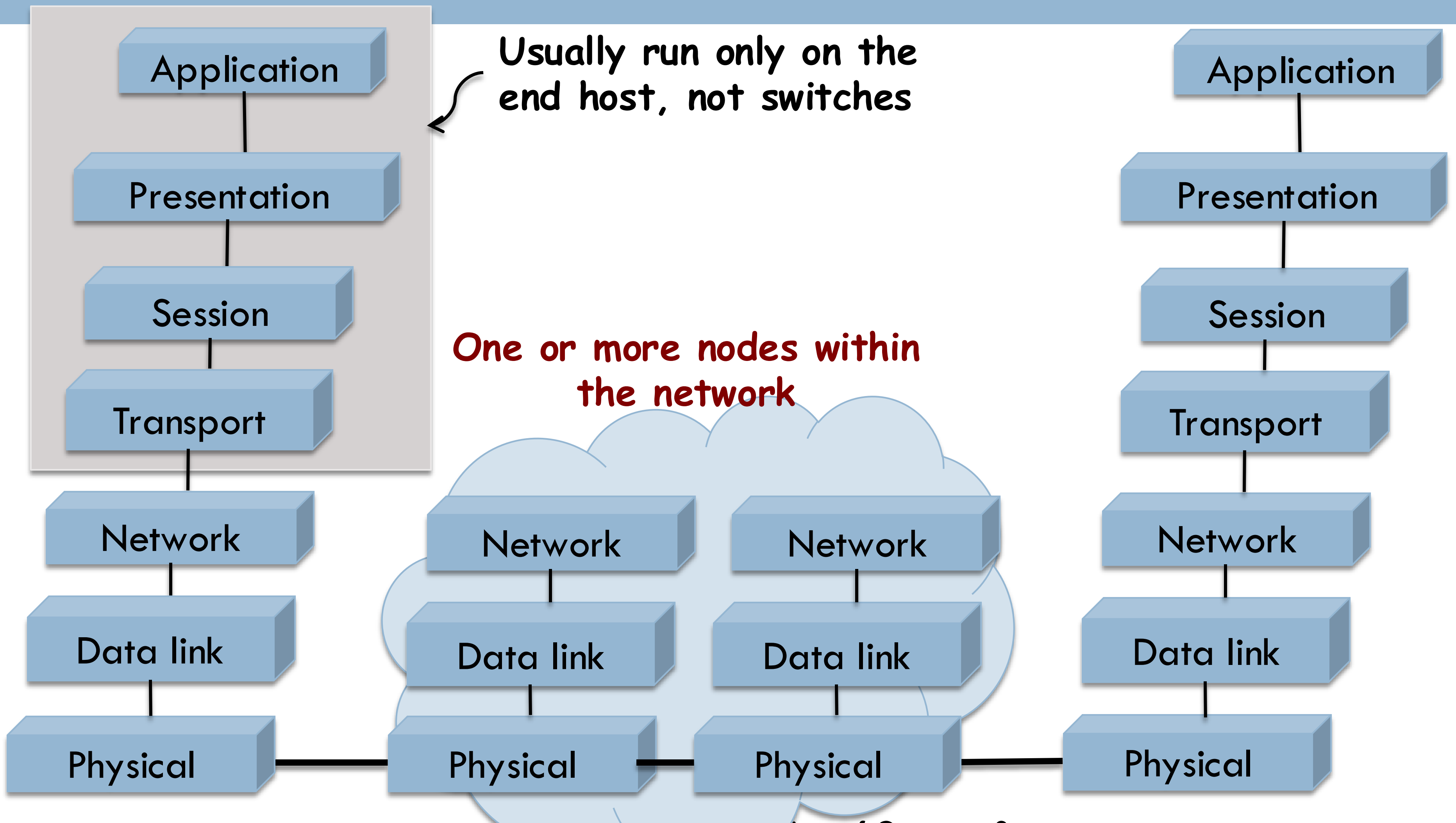
- Collects stream of bits into a **frame**
  - Puts special *bit pattern* at the start/end of each frame
  - Frames, not raw bits, are delivered to host
- Compute **checksum** for frame
  - Check for correctness and request retransmission
- Network adaptors & device drivers implement this

OSI network architecture

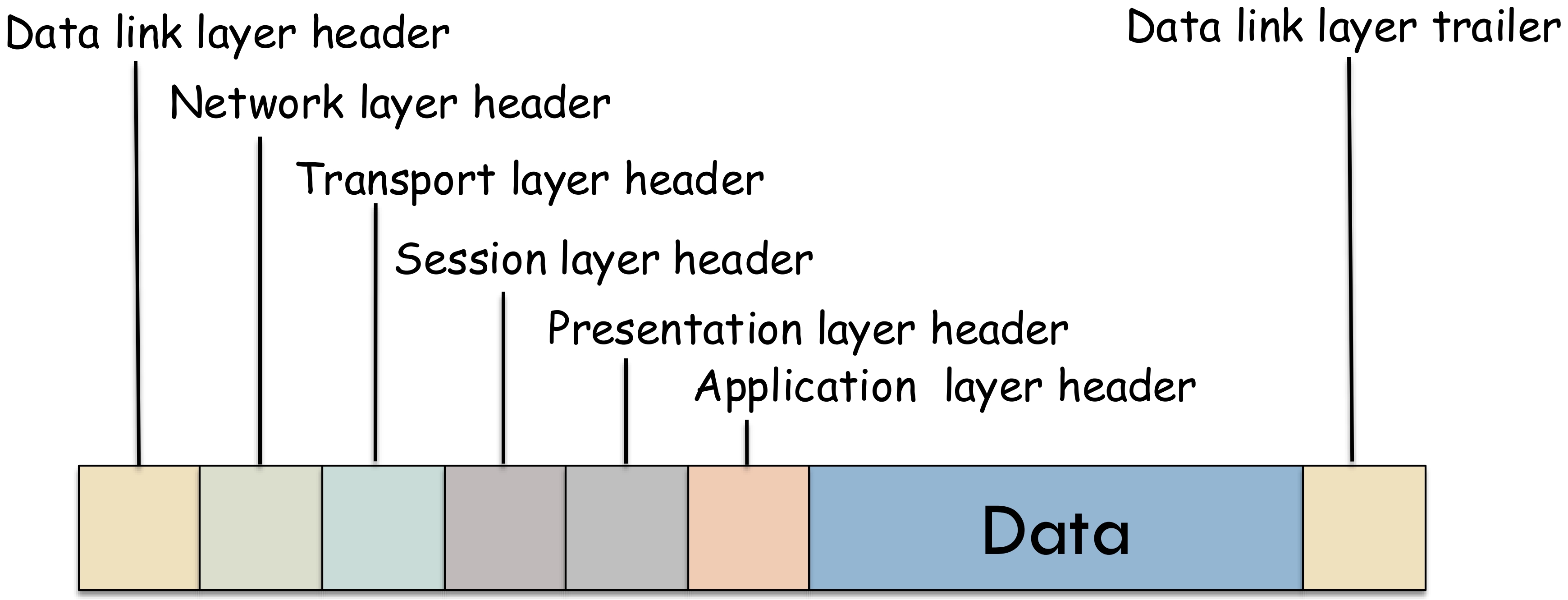
- Network layer
  - Handles routing among nodes in a **packet-switched** network
  - Unit of data exchanged is **packet** not frames

- Layers implemented on all network nodes?
  - Physical, data and network

OSI Architecture



How messages flowing through the OSI stack will appear on the network



OSI network architecture

- Transport
  - Implements process-process **channel**
  - **Messages** {not packet or frame}
- Presentation
  - **Format** of data exchanged between peers
- Session
  - **Namespace** to tie different transport-streams that are part of the application

The contents of this slide-set are based on the following references

- *Computer Networks: A Systems Approach. Larry Peterson and Bruce Davie. 4th edition. Morgan Kaufmann. ISBN: 978-0-12-370548-8. [Chapter 5, 6]*