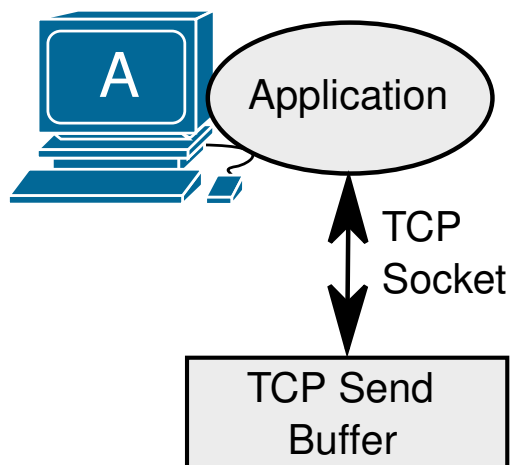


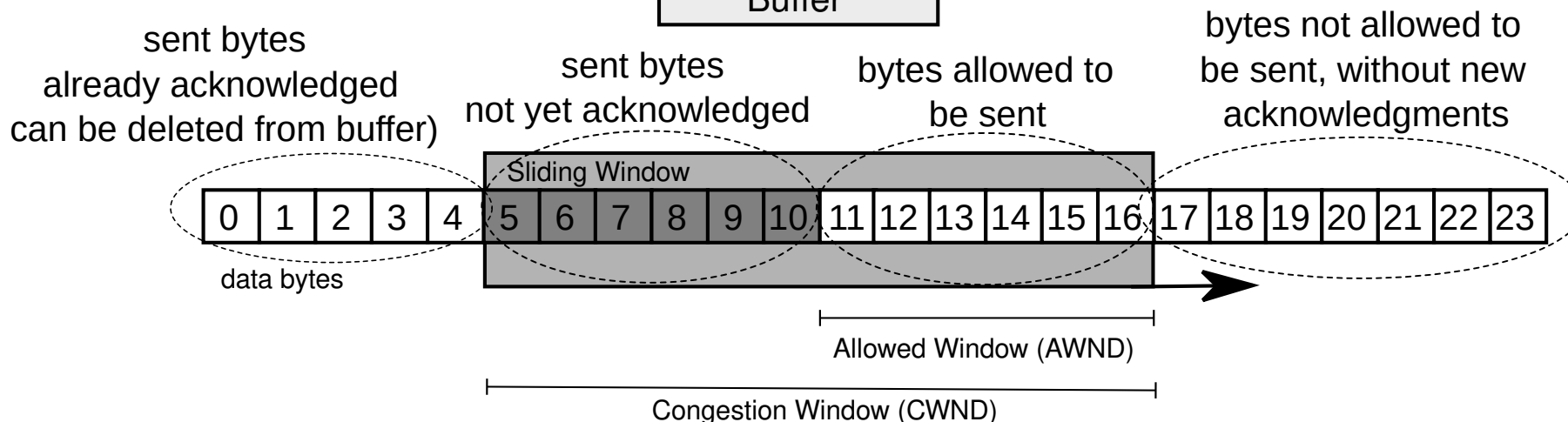
# TCP Extra Information

# TCP Congestion Control

- Uses a sliding window to determine the number of packets/bytes the sender is allowed to transmit.



- Congestion Window (CWND)
  - ◆ Set by sender to avoid overloading network.
  - ◆ The maximum value of CWND is RWND.
- Allowed Window (AWND)
  - ◆ Number of bytes allowed to be sent.
  - ◆  $AWND = \min(CWND - NACK, RWND)$ 
    - $RWND = W - NACK$
    - $W$ : Last receiver reported window.
    - $NACK$ : Not Acknowledged bytes.



# Other TCP Algorithms

- NewReno (1996)

- ◆ Allows for partial ACK.
- ◆ When a loss occurs, CWND is defined as  $\beta \cdot \text{CWND}$ , with  $\beta=0.5$ . When a ACK arrives, CWND is updated as  $\text{CWND}=\text{CWND}+\alpha$ , with  $\alpha=1 \text{ MSS}$ .
- ◆ Used by default in Windows and supported by Mac OS X.
  - Used in Windows XP and earlier.
  - After Windows Vista, Compound TCP can also be enabled.

- CUBIC (2005)

- ◆ Uses a cubic function to control the CWND.
- ◆ Used by Linux (kernel 2.6.19 and later) and supported by Mac OS X.

- Compound TCP (2006)

- ◆ Adapts its behavior by use of a scalable delay-based component. T
  - Increases throughput more quickly in the congestion avoidance phase.
- ◆ The AWND depend on the RTT measurements from successfully acknowledged packets.
- ◆ Windows OS supports it as an option.

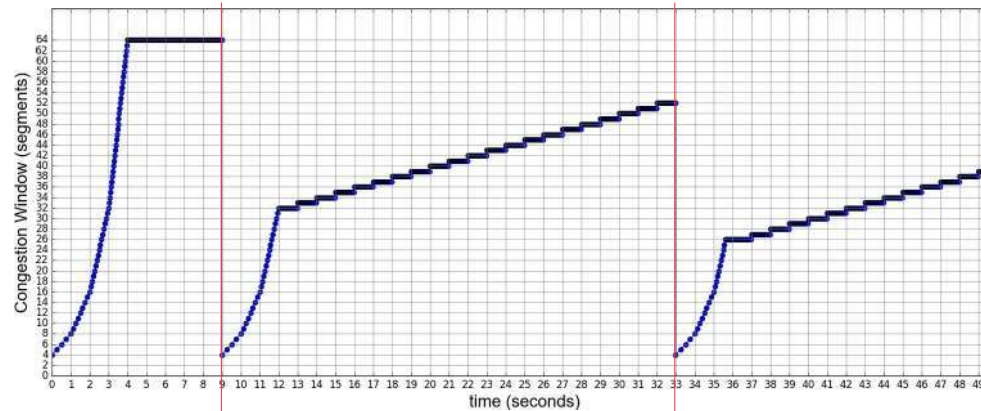
- Low Extra Delay Background Transport (LEDBAT)

- ◆ Delay-based congestion control algorithm that uses all the available bandwidth while limiting the increase in delay. Measures one-way delay.
- ◆ Supported by Windows 10 and latest versions of Mac OS X.



# TCP Algorithms Comparison

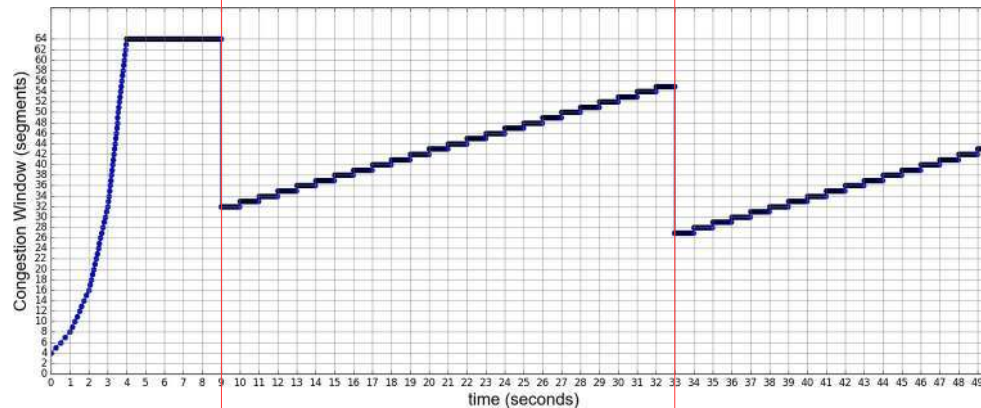
- Tahoe



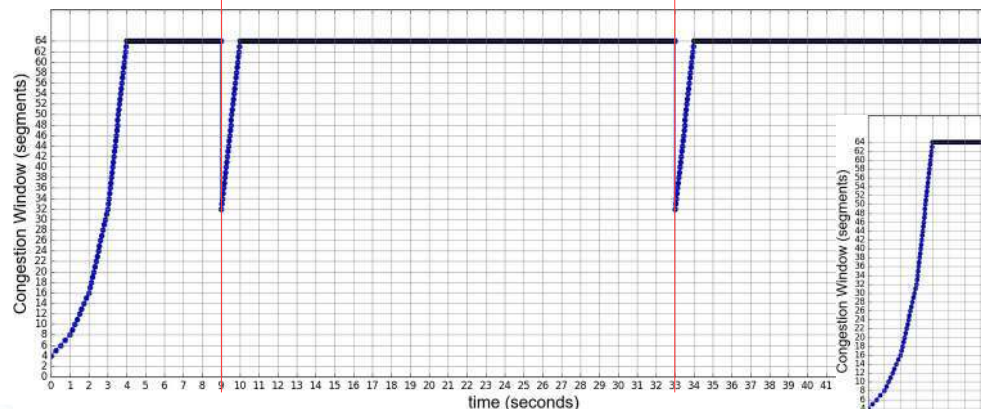
- Behavior and performance depends greatly on traffic and losses patterns, and end-points' behaviors
- In these examples:

- Using a traffic pattern that sends a number of packets equal to the CWND, in one second interval, equally spaced over the interval.
- Last packet of interval 8s-9s and 32s-33s are lost.
- ACWD is 64, and initial CWND is 4 MSS (segments).
- Receiver sends one ACK per packet.
- Each dot represents a received ACK.
- Tahoe sends 1585 packets, Reno 2017 packets and NewReno 2938 packets.

- Reno



- NewReno



- Cubic





# TCP Window Scaling

- Window field has only 16 bits, allows only 65536 bytes as maximum windows.
- Nowadays, devices have much more available memory and networks much more throughput.
- A small Window limits performance.
- The **TCP Window Scale** option was introduced by RFC 7323 in 2014.
  - Commonly used in current OS.
- The TCP Window Scale option defines the (power of 2) exponent that will be multiplied by the standard Window value.
  - Scaled Window Size = Window \*  $2^{\text{(Window Scale exponent)}}$ .
  - Sent/defined by each host, independently of the other, in the TCP session establishing packets (with SYN flag).
    - ➔ If a host receives a <SYN> packet containing a Window Scale option, it SHOULD send its own Window Scale option in the <SYN,ACK> packet.
  - The maximum scale exponent is limited to 14 for a maximum permissible window size of 1 GiB ( $2^{(14+16)}$ ).
    - ➔ Common exponent values are 6 ( $2^6=64$ ), 7 ( $2^7=128$ ) and 8 ( $2^8=256$ ).

```
Internet Protocol Version 4, Src: 192.168.17.157, Dst: 31.220.43.112
Transmission Control Protocol, Src Port: 42290, Dst Port: 443, Seq: 0, Len: 0
  Source Port: 42290
  Destination Port: 443
  [Stream index: 15]
  [Conversation completeness: Incomplete, DATA (15)]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 2087853377
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 0
  Acknowledgment number (raw): 0
  1010 .... = Header Length: 40 bytes (10)
  Flags: 0x002 (SYN)
  Window: 64240
  [Calculated window size: 64240]
  Checksum: 0x1dc0 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation
    TCP Option - Maximum segment size: 1460 bytes
    TCP Option - SACK permitted
    TCP Option - Timestamps
    TCP Option - No-Operation (NOP)
    TCP Option - Window scale: 7 (multiply by 128)
```

```
Transmission Control Protocol, Src Port: 443, Dst Port: 42064, Seq: 0, Ack: 1, Len: 0
  Source Port: 443
  Destination Port: 42064
  [Stream index: 0]
  [Conversation completeness: Complete, WITH_DATA (31)]
  [TCP Segment Len: 0]
  Sequence Number: 0 (relative sequence number)
  Sequence Number (raw): 1619231038
  [Next Sequence Number: 1 (relative sequence number)]
  Acknowledgment Number: 1 (relative ack number)
  Acknowledgment number (raw): 2954385044
  1010 .... = Header Length: 40 bytes (10)
  Flags: 0x012 (SYN, ACK)
  Window: 28960
  [Calculated window size: 28960]
  Checksum: 0xf3d7 [unverified]
  [Checksum Status: Unverified]
  Urgent Pointer: 0
  Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation
    TCP Option - Maximum segment size: 1460 bytes
    TCP Option - SACK permitted
    TCP Option - Timestamps
    TCP Option - No-Operation (NOP)
    TCP Option - Window scale: 6 (multiply by 64)
```



# QUIC

- QUIC was developed and deployed by Google in 2013, but was presented as a standard in 2021 by RFC9000.
- QUIC packets are carried in UDP datagrams to better facilitate deployment in existing systems and networks.
- QUIC handshake combines negotiation of cryptographic (TLS) and transport parameters.
  - ◆ Is structured to permit the exchange of application data as soon as possible.
- Provides the necessary feedback to implement reliable delivery and congestion control.
- Application protocols exchange information over a QUIC connection via streams which are ordered sequences of bytes. Two types of streams can be created:
  - ◆ Bidirectional streams, which allow both endpoints to send data.
  - ◆ Unidirectional streams, which allow a single endpoint to send data.
- Avoids head-of-line blocking across multiple streams.
  - ◆ When a packet loss occurs, only streams with data in that packet are blocked waiting for a retransmission to be received, while other streams can continue making progress.
- Two levels of data flow control in QUIC:
  - ◆ Stream flow control, which prevents a single stream from consuming the entire receive buffer for a connection by limiting the amount of data that can be sent on each stream.
  - ◆ Connection flow control, which prevents senders from exceeding a receiver's buffer capacity for the connection by limiting the total bytes of stream data sent in STREAM frames on all streams.

