Transmission Control Protocol Architecture

The Transmission Control Protocol

- A programmer assumes
  - the data will arrive correctly
    - or the OS will inform the application that an error has occurred
  - OS guarantees that data will be delivered reliably
- Protocol software must provide the following:
  - Guarantee prompt and reliable communication
  - Data be delivered in exactly the same order that it was sent
  - There be no loss or duplication
- In the TCP/IP suite, the Transmission Control Protocol (TCP) provides reliable transport service

The Service TCP Provides to Applications

- The service offered by TCP has the following features:
- Connection Orientation
  - an application must first request a connection to a destination
- Point-to-Point Communication
  - each TCP connection has exactly two endpoints (session)
- Complete Reliability
  - TCP guarantees that the data sent across a connection will be delivered completely and in order
- Full Duplex Communication
  - allows data to flow in either direction

The Service TCP Provides to Applications

- Stream Interface
  - an application sends a continuous sequence of octets
  - it does not group data into records or messages
- Reliable Connection Startup
  - TCP allows two applications to reliably start communication
- Graceful Connection Shutdown
  - TCP insures that both sides have agreed to shut down the connection

End-to-End Service and Virtual Connections

- TCP is classified as an end-to-end protocol
  - it provides communication between an application on one computer to an application on another computer
- The connections in TCP are called virtual connections
  - because connections are achieved in software
- TCP software modules on two machines exchange messages to achieve the illusion of a connection
- TCP uses IP to carry messages
  - IP treats each TCP message as data to be transferred
- End System Reboot
  - either of the two end systems might crash and reboot
- Heterogeneous End Systems
  - A sender can generate data so fast that it overruns a slow receiver

Techniques That Transport Protocols Uses

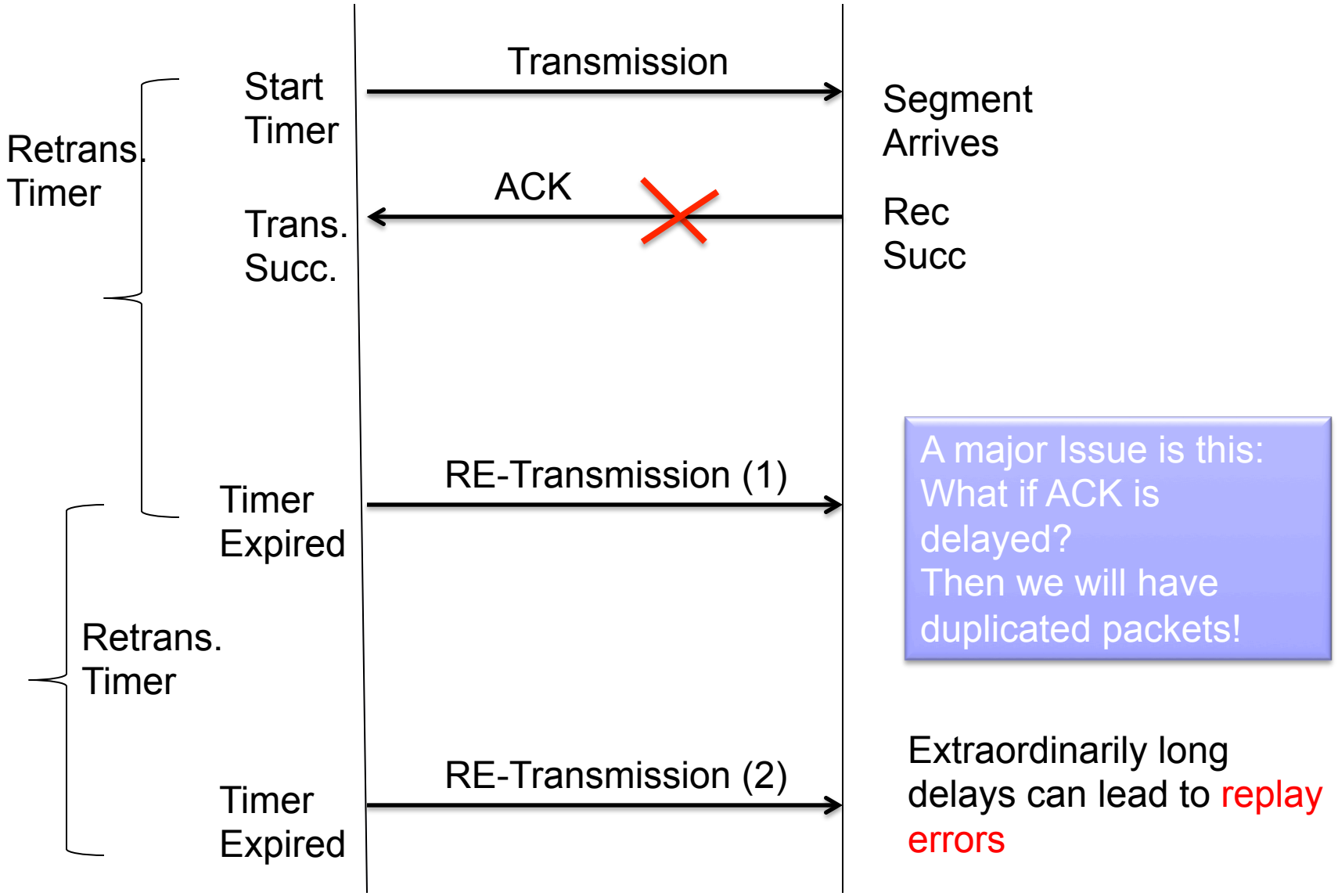
- There are techniques that communication systems use to overcome some of the problems
  - For example, to compensate for bits that are changed during transmission
    - a protocol might include parity bits
    - a checksum
    - or a cyclic redundancy check (CRC)
- Sequencing handles duplicates and out-of-order delivery
- Retransmissions Handle Lost Packets
- Techniques Avoid Replay
- Flow Control Prevents Data Overrun
- Avoid Congestion
- Handle Packet Loss

Sequencing Handle Duplicates and Out-of-Order Delivery

- To handle duplicate packets and out-of-order deliveries transport protocols use sequencing
  - The sender attaches a sequence number to each packet
  - The receiver stores both the sequence number of the last packet received in order
  - as well as a list of additional packets that arrived out of order
- The receiver examines the sequence number
  - to determine how the packet should be handled
- If the packet is the next one expected (i.e., has arrived in order)
  - the protocol software delivers the packet to the next highest layer
  - the protocol checks its list
    - to see whether additional packets can also be delivered
- If the packet has arrived out of order
  - the protocol software adds the packet to the list
- If the packet has already been delivered or the sequence number matches one of the packets waiting on the list
  - the software discards the new copy

Retransmissions Handle Lost Packets

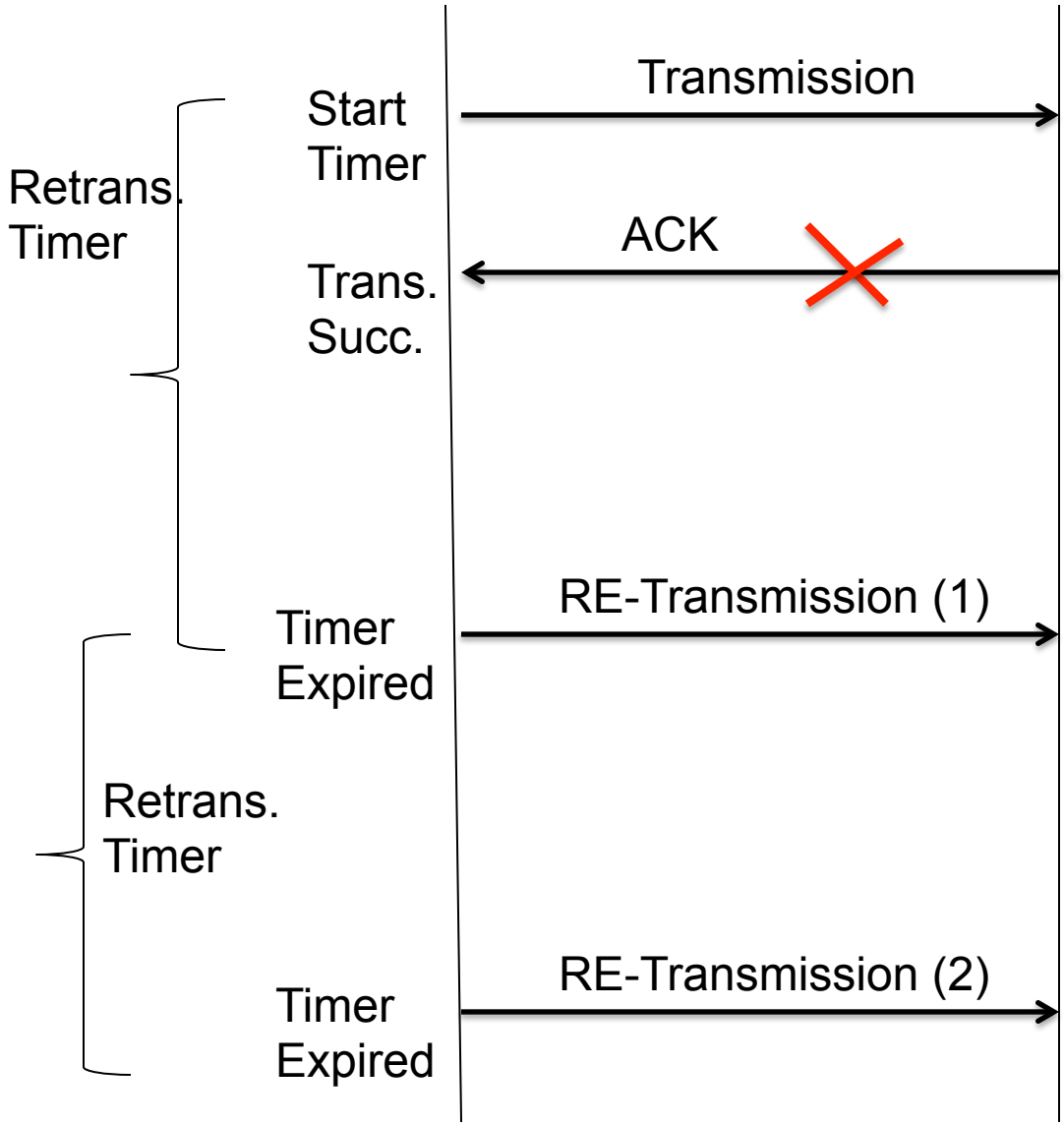
- To handle packet loss transport protocols use positive acknowledgement (ACK) with retransmission
  - the receiver sends a small ACK message that reports successful reception
- Whenever it sends a packet
  - the sender starts a timer
- If an acknowledgement arrives before the timer expires
  - the software cancels the timer
- If the timer expires before an acknowledgement arrives
  - the protocol sends another copy of the packet and starts the timer again
- Sending a second copy is known as retransmitting
  - retransmission cannot succeed if a hardware failure has permanently disconnected the network or if the receiving computer has crashed
  - there is a bound for the maximum number of retransmissions
    - if bound exceeded, the destination will be declared unreachable



A major Issue is this:
What if ACK is delayed?
Then we will have duplicated packets!

Extraordinarily long delays can lead to replay errors

RTO= Avg(n*RTT)
RTO=Retransmission Timeout
RTT= Round Trip Time
When Loss occurs:
Retransmission Time=RTO*2^n

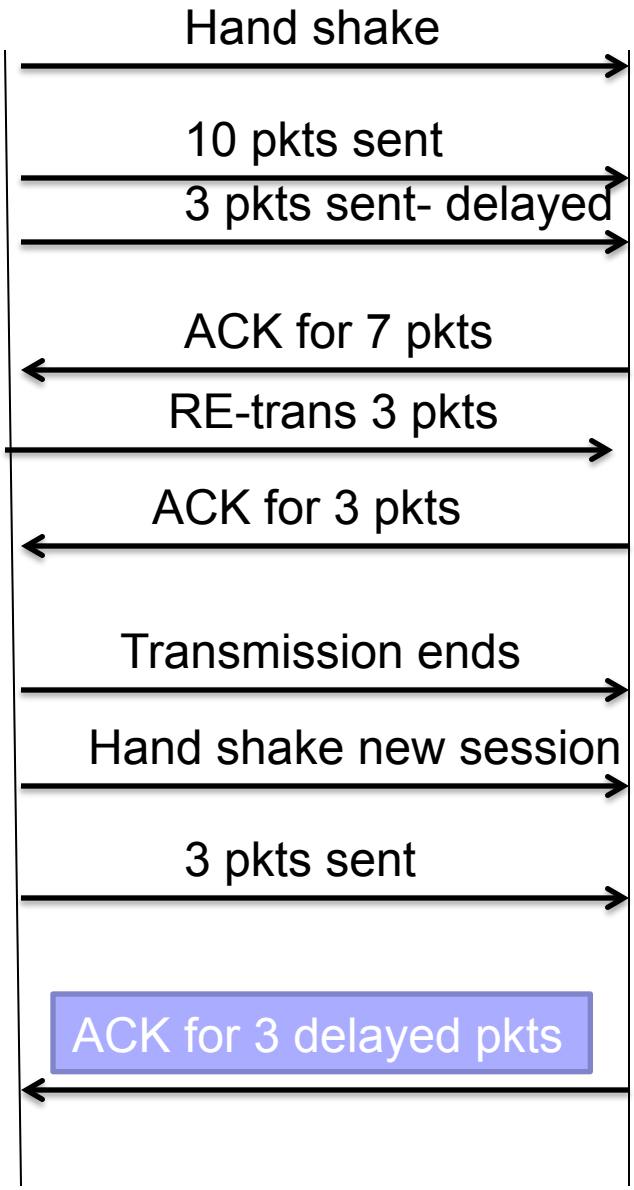


A major Issue is this:
What if ACK is delayed?
Then we will have duplicated packets!

Extraordinarily long delays can lead to replay errors

Techniques Avoid Replay

- Extraordinarily long delays can lead to replay errors
- A packet from an earlier conversation might be accepted and the correct packet discarded as a duplicate
- Replay errors can also occur with control packets
- A protocol should be designed so that the duplicate message will not cause the second connection to be closed
- To prevent replays, protocols mark each session with a unique ID
- The protocol discards any arriving packet that contains an incorrect ID
- An ID must not be reused until a reasonable time has passed



The protocol discards any arriving packet that contains an incorrect ID

An ID must not be reused until a reasonable time has passed

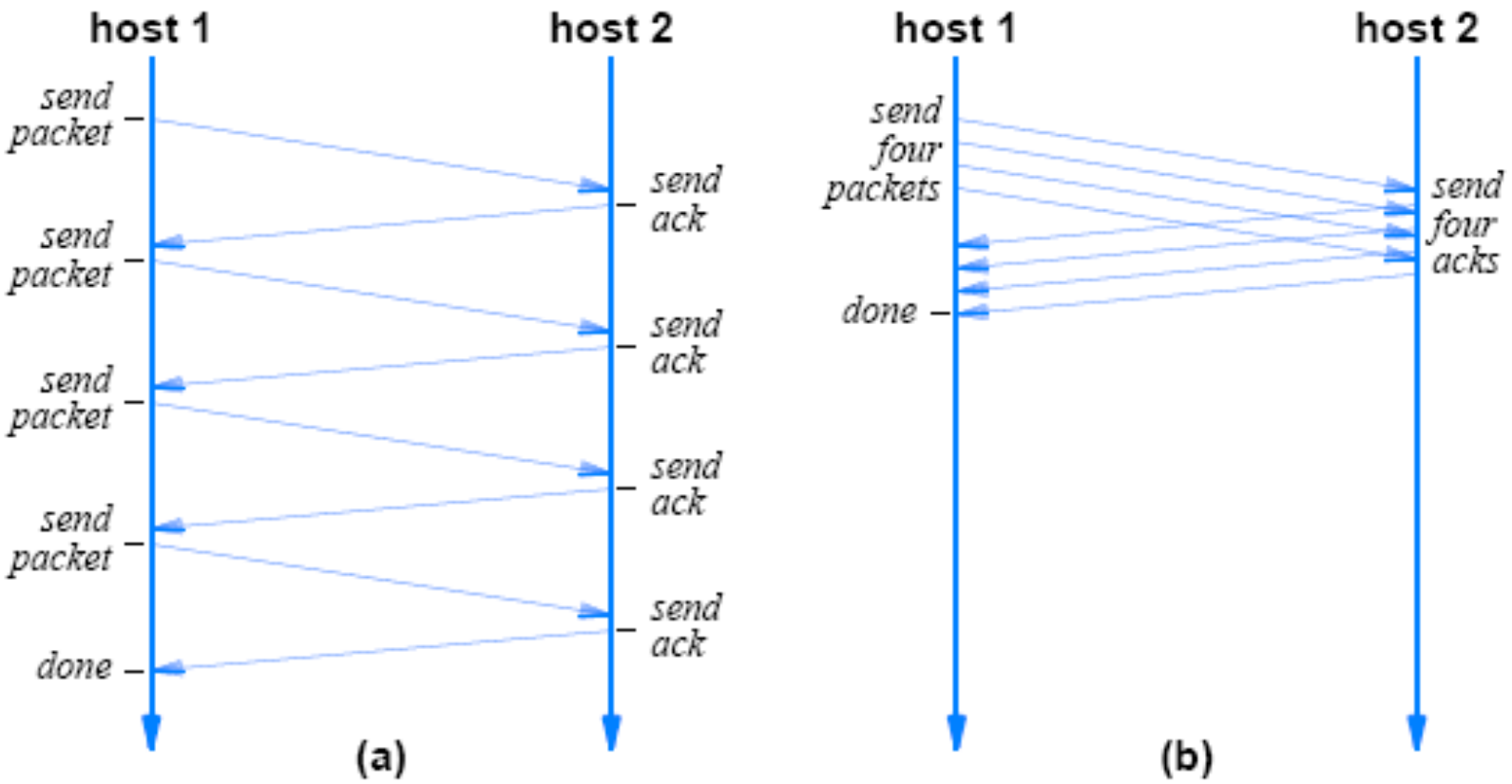
Which is which?

Extraordinarily long delays can lead to replay errors

Flow Control Prevents Data Overrun

- Techniques are available to prevent a fast computer from sending so much data to overrun a slower receiver
  - Flow control techniques are employed to handle the problem
- The simplest form of flow control is a stop-and-go
- Another flow control technique known as sliding window
  - The sender and receiver use a fixed window size
    - which is the maximum amount of data that can be sent before an acknowledgement arrives
  - The sender retains a copy in case retransmission is needed
  - The receiver must have pre-allocated buffer space

Flow Control Prevents Data Overrun



The receiver and transmitter must agree on the window size.
The receiver dedicates a buffer size consistent with window size to the connection

Flow Control Prevents Data Overrun

- Comparing delay between stop-and-go and sliding window

  where
  - T_w is the throughput that can be achieved with a sliding window protocol
  - T_g is the throughput that can be achieved with a stop-and-go protocol
  - W is the window size

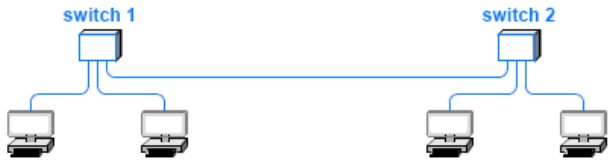
$$T_w = T_g \times W$$

- Throughput cannot be increased arbitrarily, by just increasing the window size
  - The bandwidth of the underlying network imposes an upper bound; bits cannot be sent faster than the hardware can carry them
  - The equation can be rewritten (B is the underlying bandwidth):

$$T_w = min(B, T_g \times W)$$

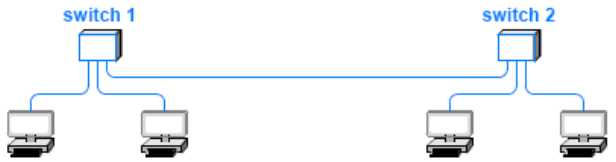
Example

Throughput Example



- Assume transmitting at 2Mbps. The packet length is 1000 bytes. The delay between transmitter and receiver is 50 msec. What is the throughput for stop-and-go
  - RTT = 100 msec.
  - Maximum data rate = [1000 x 8 bits]/100 msec = 80,000 bits/sec
  - Only 4 percent of the transmission rate (2Mbps)

Congestion Control



- Consider what happens if both computers attached to switch1 attempt to send data to a computer attached to switch2
  - Switch1 receives data at an aggregate rate of 2 Gbps, but can only forward 1 Gbps to switch2
  - This situation is known as congestion
- Congestion results in delay
- If congestion persists
  - the switch will run out of memory and begin discarding packets
- Retransmission can be used to recover lost packets
  - But retransmission sends more packets into the network
- If the situation persists, network can become unusable
  - this condition is known as congestion collapse

Techniques to Avoid Congestion

- In the Internet, congestion usually occurs in routers
- Transport protocols attempt to avoid congestion collapse
  - by monitoring the network and reacting quickly once congestion starts
- There are two basic approaches:

1- Arrange for intermediate systems (i.e., routers) to inform a sender when congestion occurs

  - implemented either by having routers send a special message to the source of packets when congestion occurs
  - or by having routers set a bit in the header of each packet that experiences delay caused by congestion

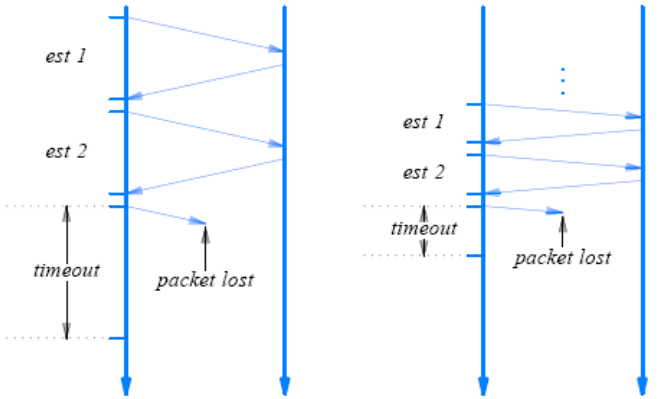
2- Use increased delay or packet loss as an estimate of congestion

  - Implemented by the computer that receives the packet including information in the ACK to inform the original sender

Which one is easier?

Techniques to Avoid Congestion

- Using delay and loss to estimate congestion is reasonable in the Internet because:
  - Modern network hardware works well
  - Most delay and loss results from congestion, not hardware failures
- The appropriate response to congestion
  - Reducing the rate at which packets are being transmitted
  - Sliding window protocols can achieve the effect of reducing the rate by temporarily reducing the window size



Retransmission time can impact the throughput!

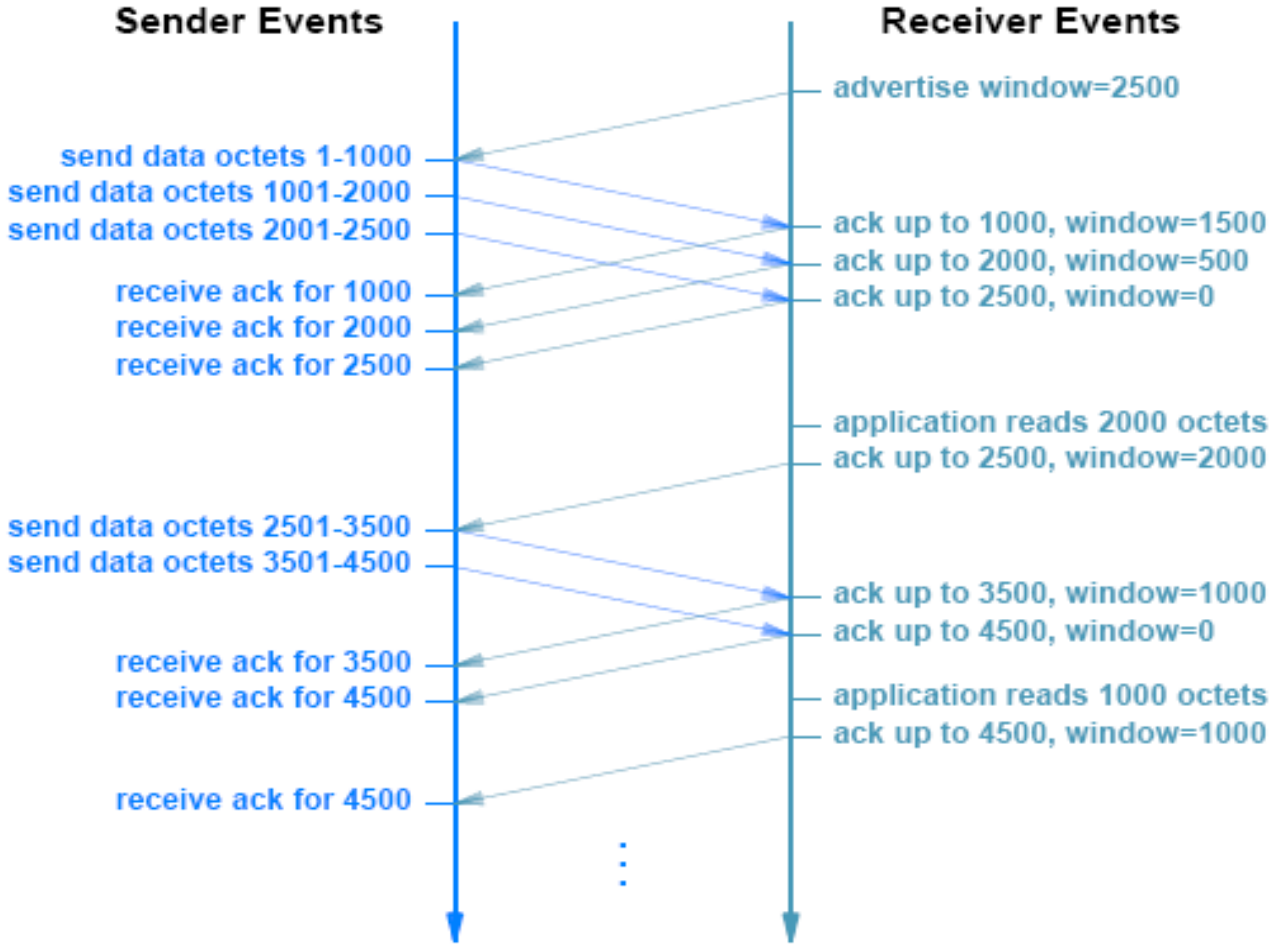
Techniques Used in TCP to Handle Packet Loss

- TCP uses retransmission to compensate for packet loss
- TCP provides data flow in both directions
  - both sides of a communication participate in retransmission
  - when TCP receives data, it sends an ACK back to the sender
- Whenever it sends data
  - TCP starts a timer, and retransmits the data if the timer expires
- How long should TCP wait before retransmitting?
  - ACKs from a computer on a LAN are expected to arrive within a few *ms*
  - but a satellite connection requires hundreds of *ms*
- On one hand
  - waiting too long for such an ACK leaves the network idle and does not maximize throughput
- On the other hand
  - retransmitting quickly does not work well because the unnecessary traffic consumes network bandwidth and lowers throughput

Adaptive Retransmission

- TCP designers realized that a fixed timeout would not operate well for the Internet
  - Thus, they chose to make TCP's retransmission adaptive
  - TCP monitors current delay on each connection
    - It adapts (changes) the retransmission timer accordingly
- TCP cannot know the exact delays
  - TCP estimates round-trip delay for each active connection by measuring the time needed to receive a response
  - TCP records the time at which the message was sent
- As it sends data packets and receives ACKs
  - TCP generates a sequence of round-trip estimates
  - It uses a statistical function to produce a weighted average
  - TCP keeps an estimate of the variance
  - It uses a linear combination of the estimated mean and variance to compute estimated time

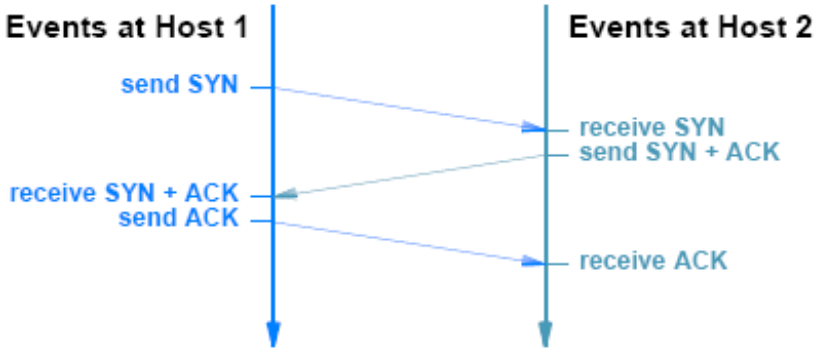
Buffers, Flow Control, and Windows

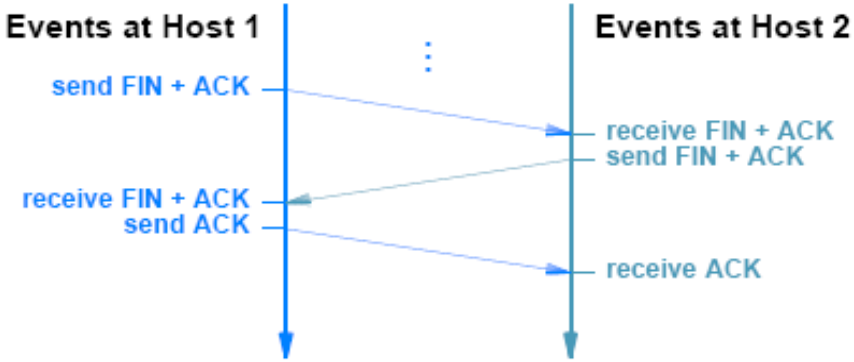


It is possible to stop data flow by advertising ZERO window size by the receiver!
It is also possible to increase transmission by advertising larger window size by the transmitter!

TCP's Three-Way Handshake

- Term synchronization segment (SYN segment) to describe the control messages used in a 3-way handshake to create a connection
- Term FIN segment (short for finish segment) to describe control messages used in a 3-way handshake to close a connection
- TCP requires each end to generate a random 32-bit sequence number that becomes the initial sequence



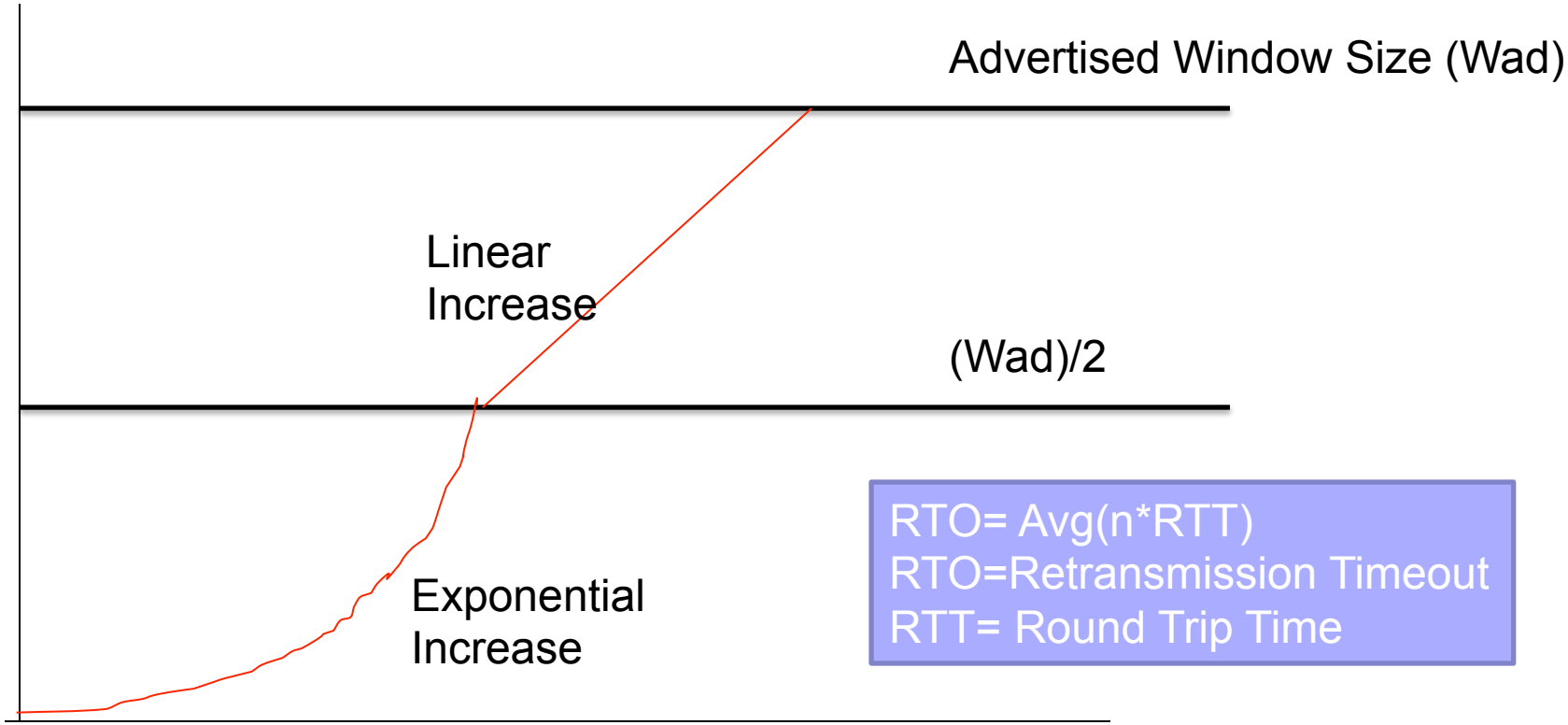


Starting the connection- agree on window size!

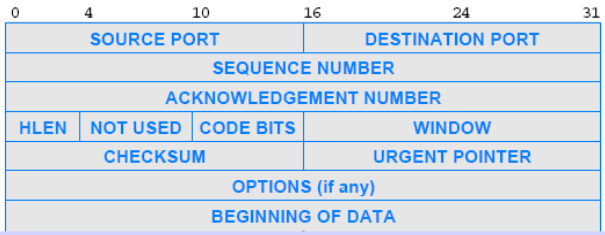
Terminating the connection gracefully!

TCP Congestion Control (slow Start)

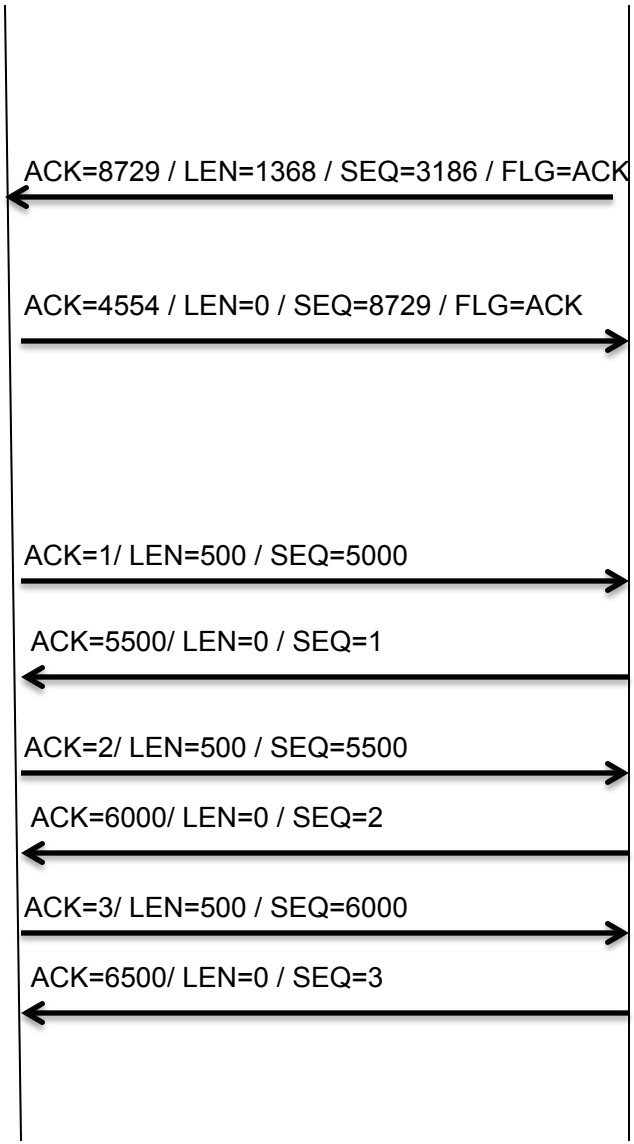
- One of the most interesting aspects of TCP is a mechanism for congestion control
- It is based on the available buffer size



TCP Segment Format

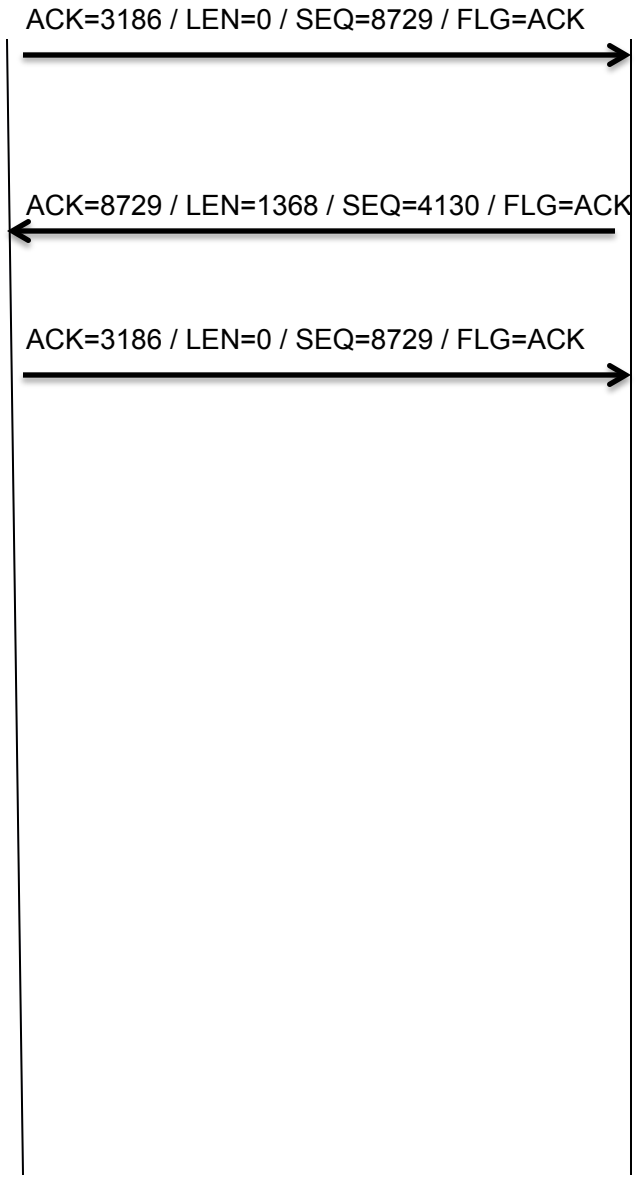


- TCP uses the term segment to refer to a message
- ACKNOWLEDGEMENT NUMBER
  - specifies the sequence number of the data that is expected next
- WINDOW
  - specifies how much additional buffer space is available beyond the ACKed data
    - The ACK always refers to the first position for which data is missing
  - If segments arrive out of order, a receiving TCP generates the same ACK multiple times until the missing data arrives
- SEQUENCE NUMBER
  - Refers to outgoing data, it gives the sequence number of the first byte of data being carried in the segment
  - A receiver uses the sequence number to reorder segments that arrive out of order and to compute an acknowledgement number



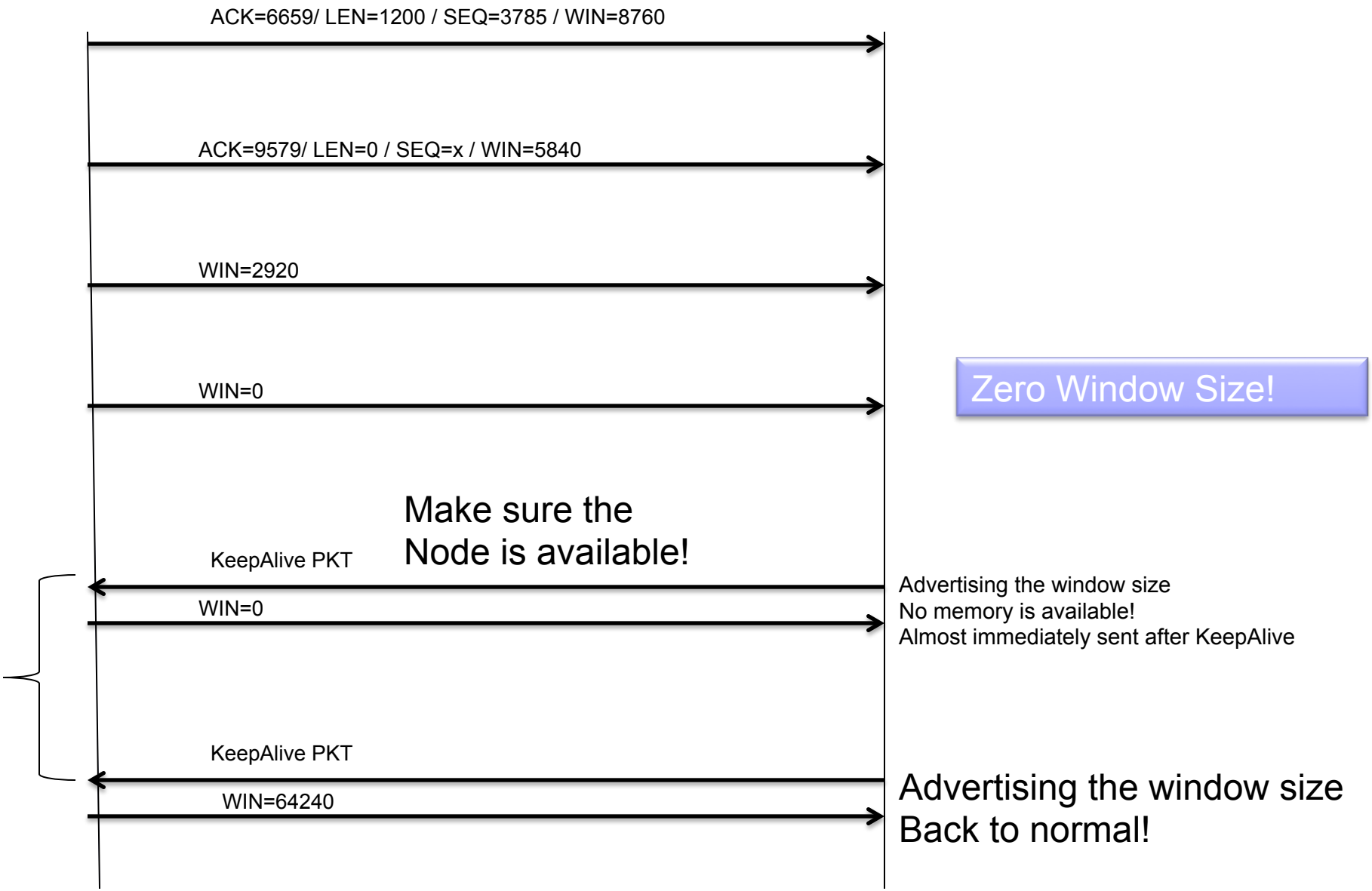
In this case ACK(i) = LEN(i-1) + SEQ(i-1)

Another Example



This is not the right Packet! ACK is incorrect!

Incorrect ACK number!


ACK=6659/ LEN=1200 / SEQ=3785 / WIN=8760
ACK=9579/ LEN=0 / SEQ=x / WIN=5840
WIN=2920
WIN=0
Zero Window Size!
Make sure the
Node is available!
KeepAlive PKT
WIN=0
Advertising the window size
No memory is available!
Almost immediately sent after KeepAlive
KeepAlive PKT
WIN=64240
Advertising the window size
Back to normal!

Understanding a Slow Network

- Slow server
  - Due to many packet losses
  - Long latency between transmitted packets
- Slow intermediate nodes (slow wire)
  - Slow acknowledgment packets
  - Typically ACK packets are almost immediate
- Slow client
  - The client keep sending too many duplicated packets

Analyzing TCP packets can greatly assist us regarding the health of the network!