Transport Layer Issues

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

Xuetao Wei

weixt@sustech.edu.cn

Contents

- Basics of Transport Layer
- Reliable Transportation
- Flow Control
- Congestion Control
- Multipath TCP

quiz考前四个

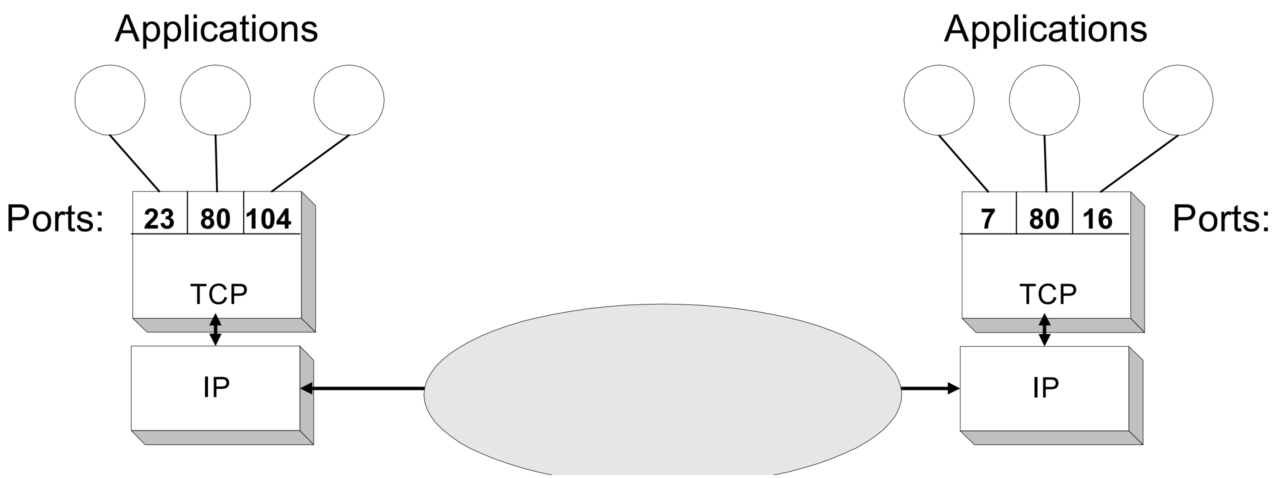
Network Layer vs Transport Layer

- **Network layer**
  - Host-to-host communication
  - Host addressing
  - Routing

- **Transport layer**
  - App-to-app communication
  - Reliable communication
  - Flow & Congestion control

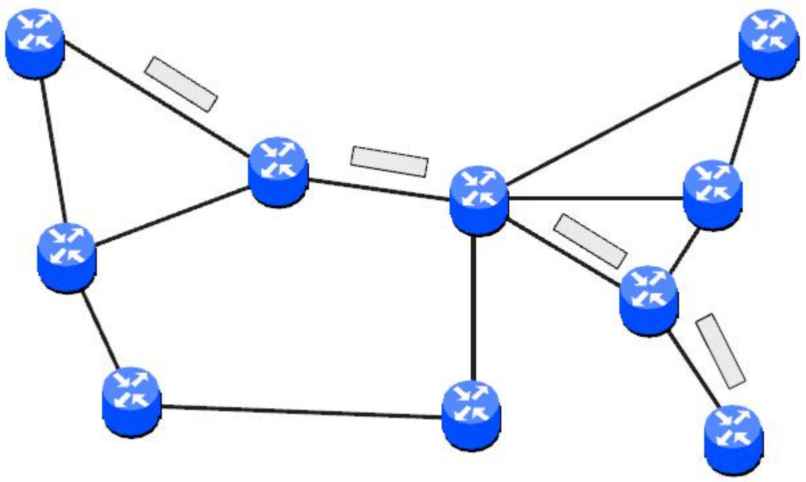
Logical Communication

- Socket = <IP address, port number>
- A pair of sockets **<client IP, server port>** and **<server IP, server port>** identify a transport layer connection (app to app)

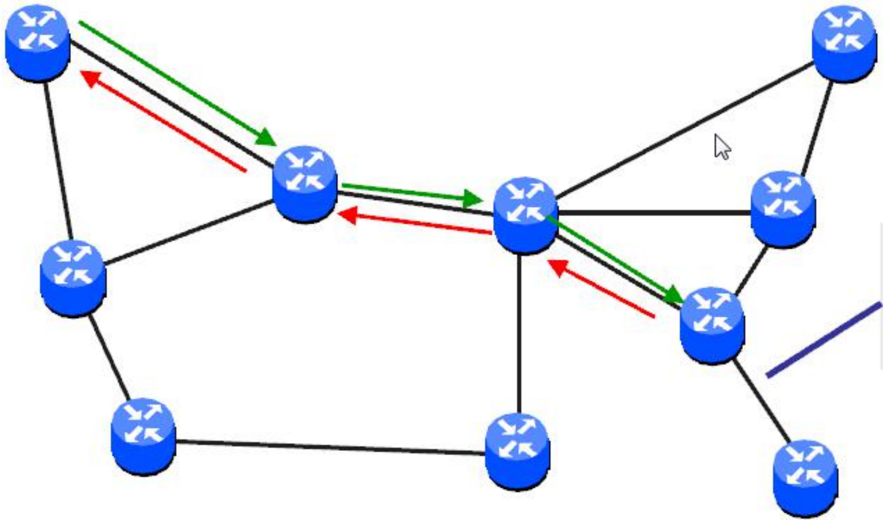


Packet vs Circuit Switching

Mechanism at Transport Layer for Reliable Transmission!



Unreliable Transmission



Reliable Transmission

TCP vs UDP

- TCP: Transport Control Protocol
  - Reliable bidirectional stream of bytes

重点考tcp

TCP Segment Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Sequence Number			
64	Acknowledgment Number			
96	Data Offset / Res / Flags		Window Size	
128	Header and Data Checksum		Urgent Pointer	
160...	Options			

- UDP: User Datagram Protocol
  - Simple (unreliable) message delivery

UDP Datagram Header Format

Bit #	0 – 7	8 – 15	16 – 23	24 – 31
0	Source Port		Destination Port	
32	Length		Header and Data Checksum	

UDP Application

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Features
  - Simple: no connection establishment (which can add delay)
  - Small packet header overhead (eight bytes long)
  - No congestion control (packets are sent as soon as possible)

- Applications
  - Multimedia streaming
  - Simple query protocols like DNS

But we ofen need **reliable & disciplined** data transmission => **TCP**

TCP at Glance

TCP的特点，这四个要记住

- **Connection oriented**
  - Explicit set-up and tear-down of TCP session
- **Reliable, in-order delivery**
  - Checksums to detect corrupted data
  - Sequence numbers to detect losses and reorder data
  - Acknowledgments & retransmissions for reliable delivery
- **Flow control** 重点
  - Prevent overflow of the receiver's buffer space
- **Congestion control**
  - Adapt to network congestion for the greater good

Contents

In-order Delivery of Packets

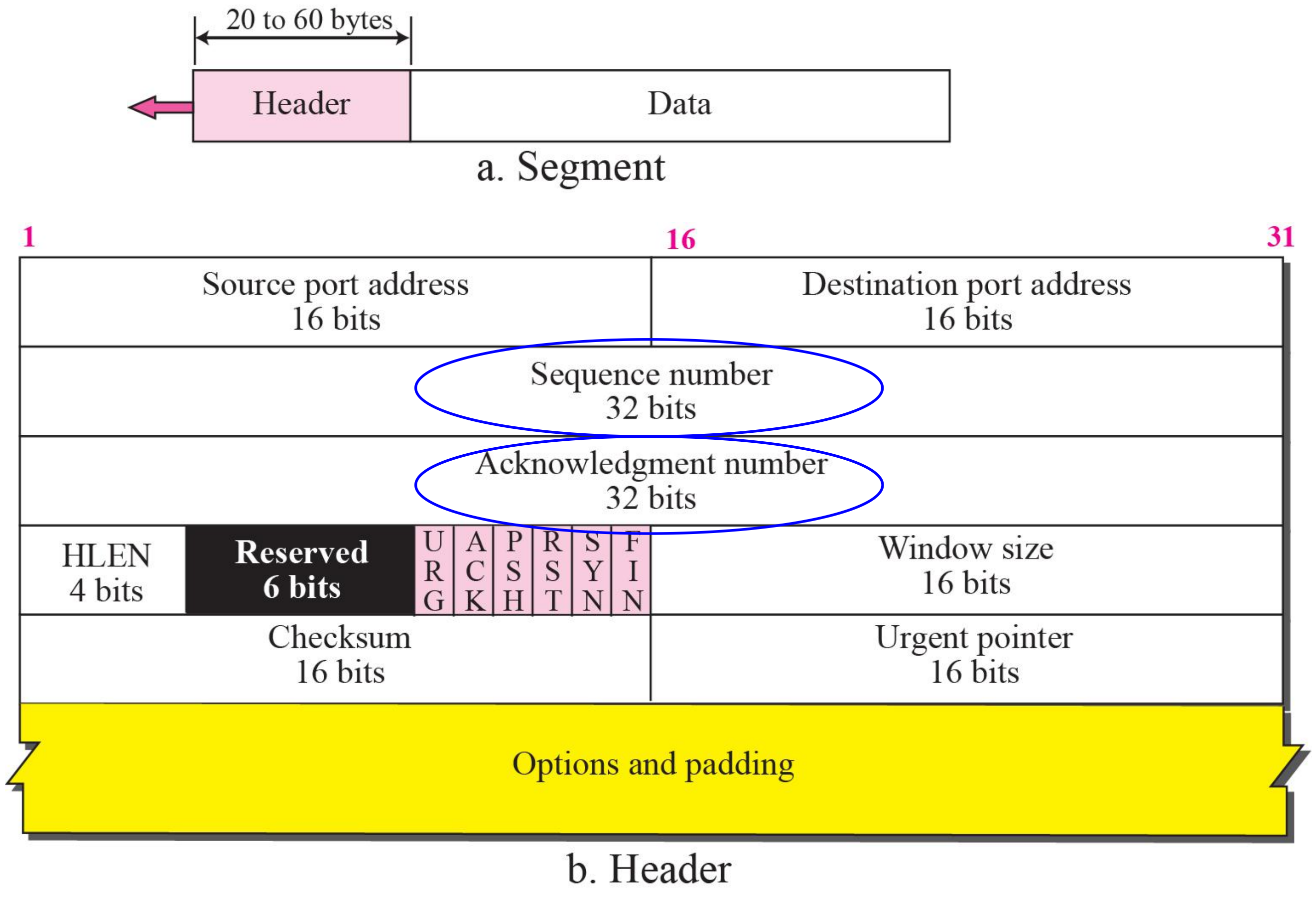
- **Option 1**
  - Packets are required to arrive at the receiver in-order
  - Almost impossible to achieve (for packet switching network)

- **Option 2**
  - Packets may arrive out-of-order, but get identified by the receiver in-order
  - More relaxed requirement and is possible to achieve, but HOW?

Answer => **sequence number** for packets

TCP Segment

- IP Packet = IP Header + IP Data (TCP Segment)



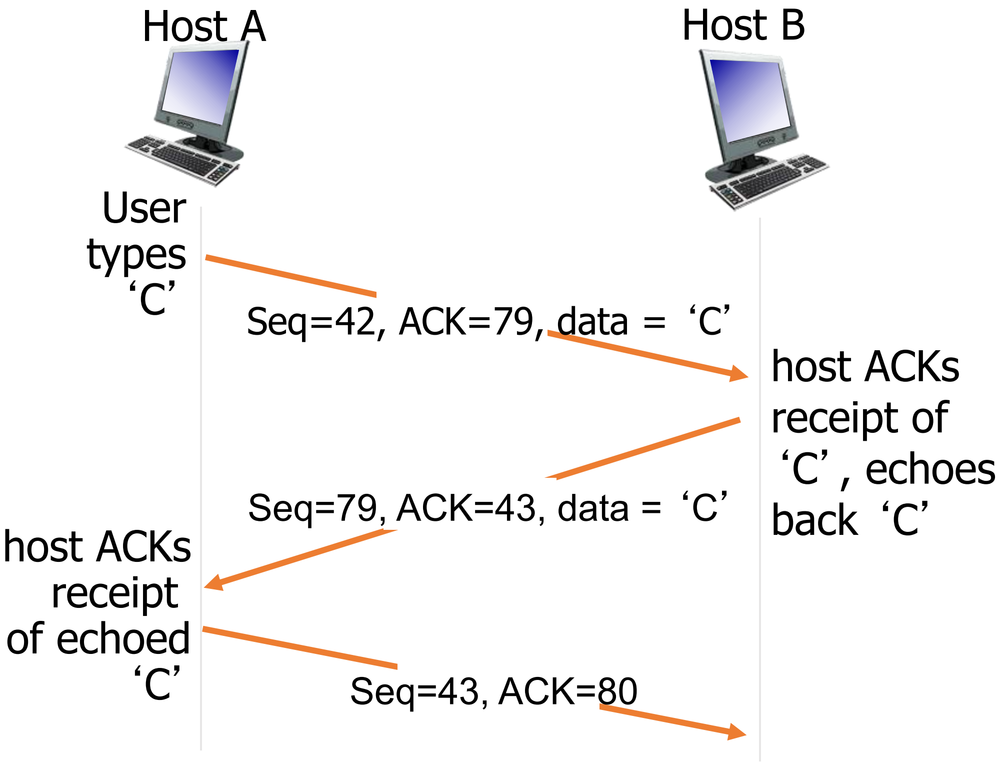
Sequence Number

- **Sequence number (SeqNo)**
  - 32 bits long ($0 <= SeqNo <= 2^{32} -1$)
  - Each SeqNo identifies a byte (not a segment) in the byte stream

- **Acknowledgement Number (AckNo)**
  - An A -> B segment can contain an acknowledgement for a B -> A segment earlier
  - The AckNo contains the next SeqNo that a host wants to receive

Example: The acknowledgement for byte stream 0-1500 is AckNo=1501

TCP: Sequence Numbers and Acks



simple telnet scenario

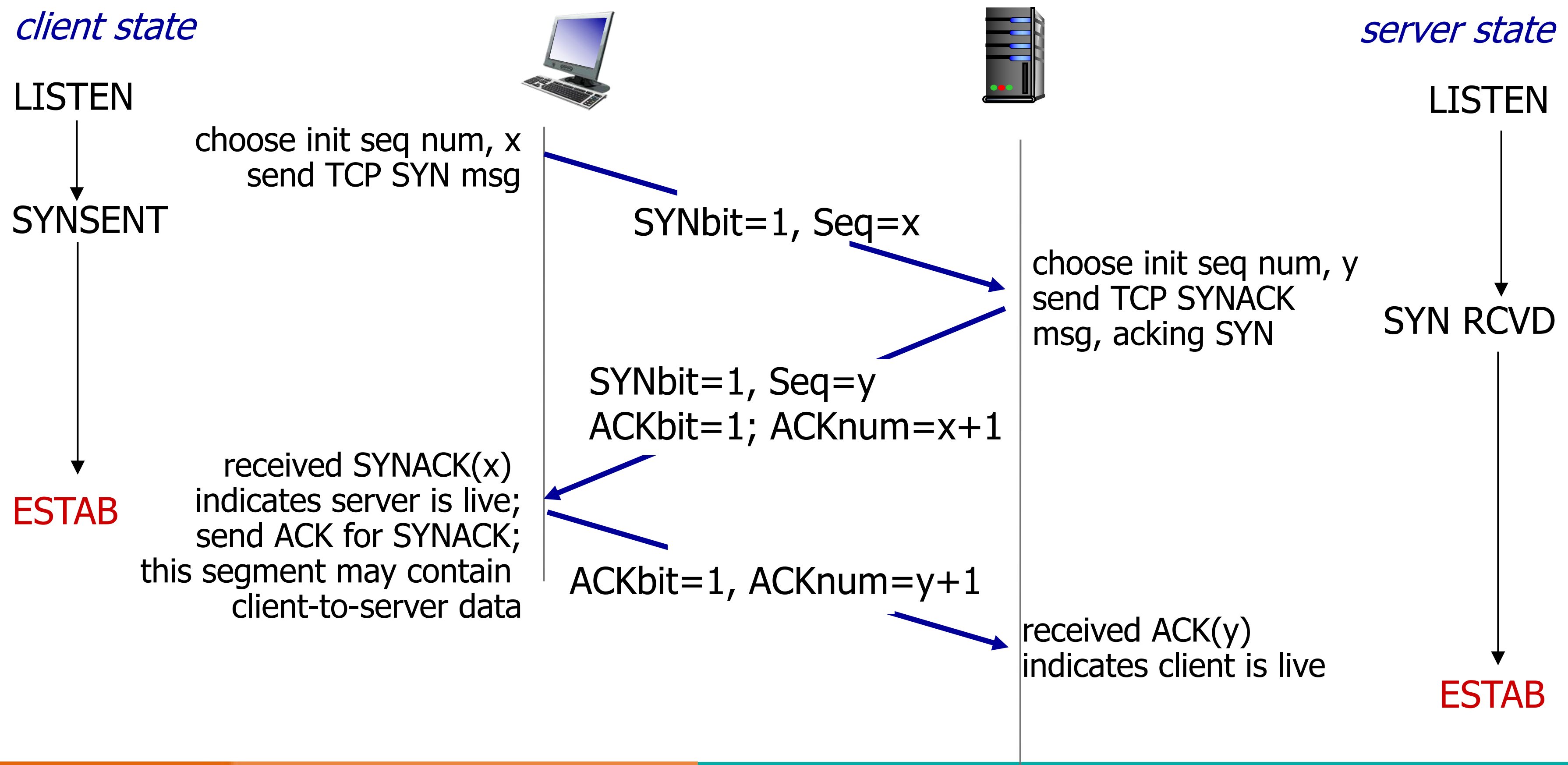
Connection Management

- **Connection establishment**
- **Connection termination**
- **State diagram**

Connection Establishment

重点

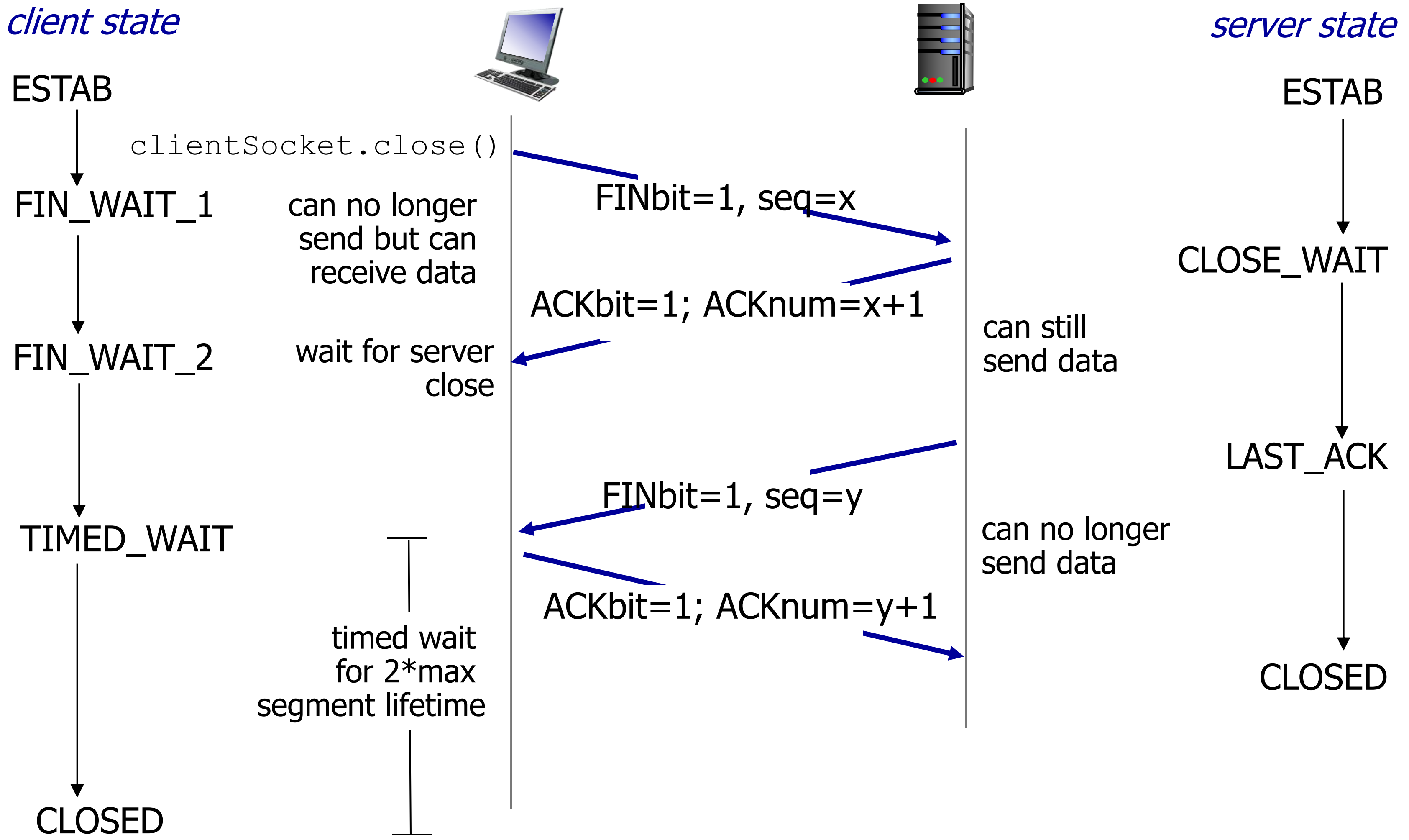
Three-way handshake



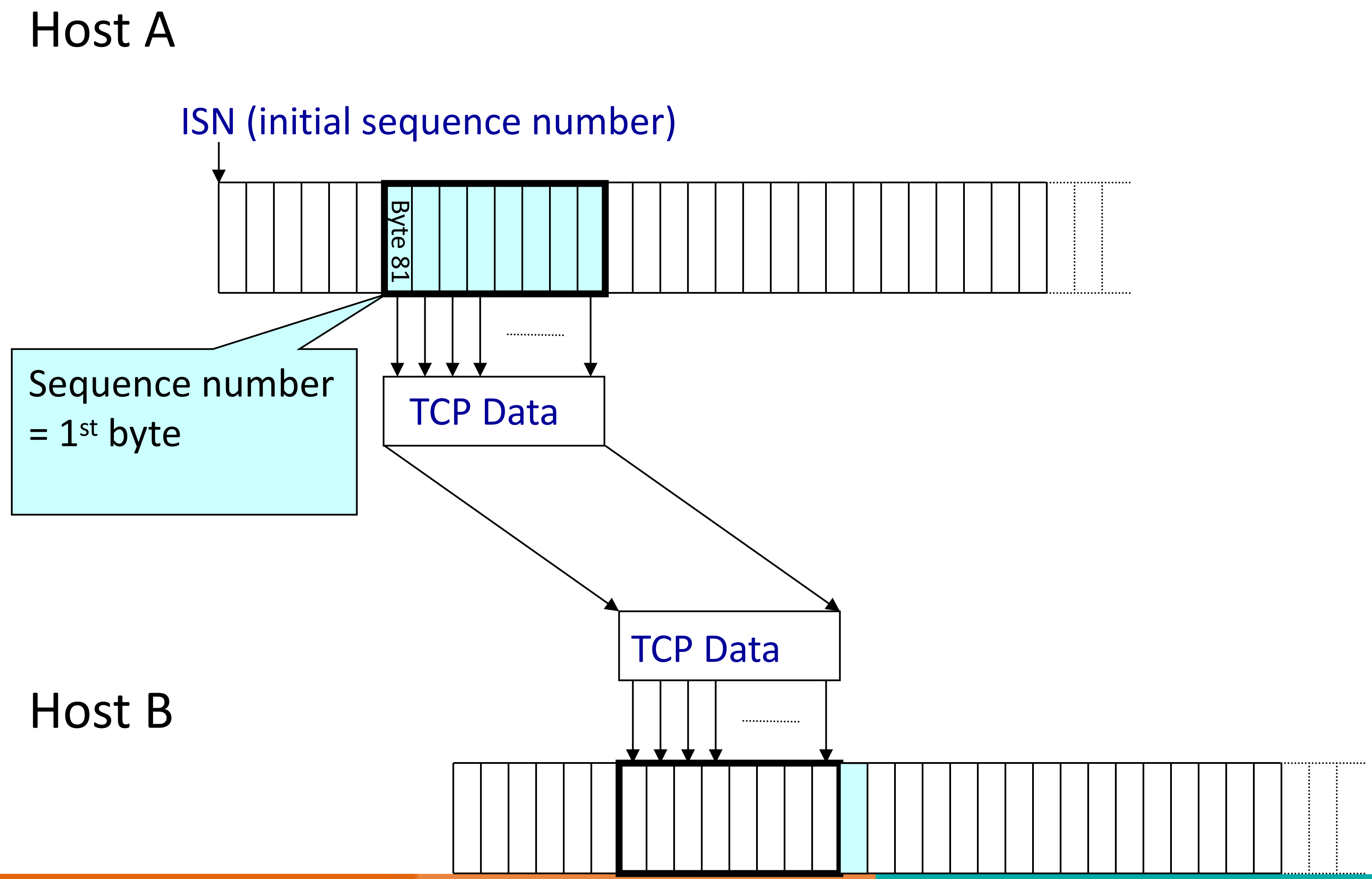
Connection Termination

Four-way handshake

重点



Initial Sequence Number (ISN)



Initial Sequence Number (ISN)

- Sequence number for the very first byte: why not 0 for ISN?

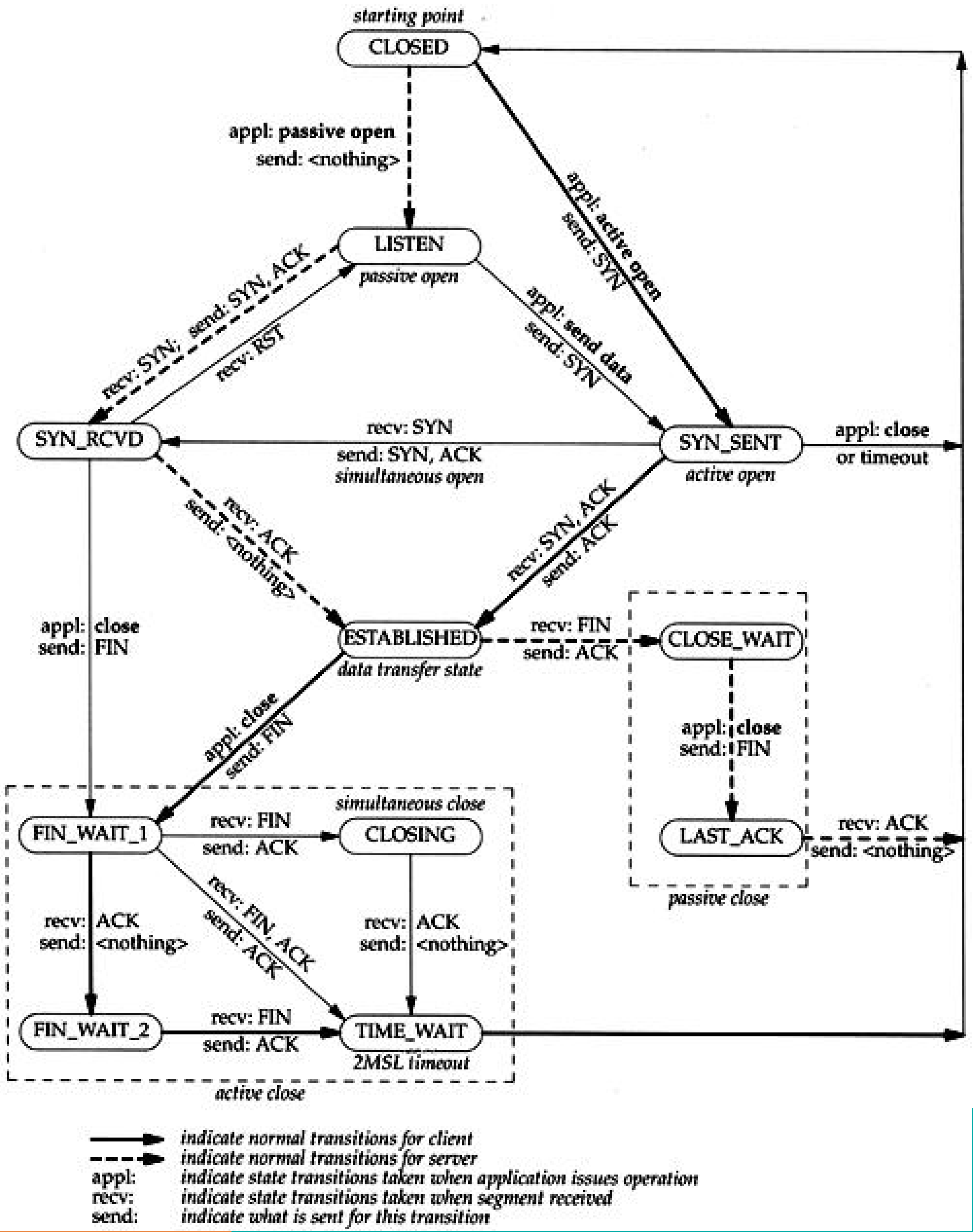
- Practical issue
  - IP addresses and ports uniquely identify a connection, but ports may get reused
  - … and there is a chance an old packet is still in flight
  - … and might be associated with the new connection

- So, TCP requires changing the ISN over time
  - Set from a 32-bit clock that ticks every 4 microseconds
  - … which only wraps around once every 4.55 hours!

- But, this means the hosts need to exchange ISNs

TCP State Diagram

要理解这个



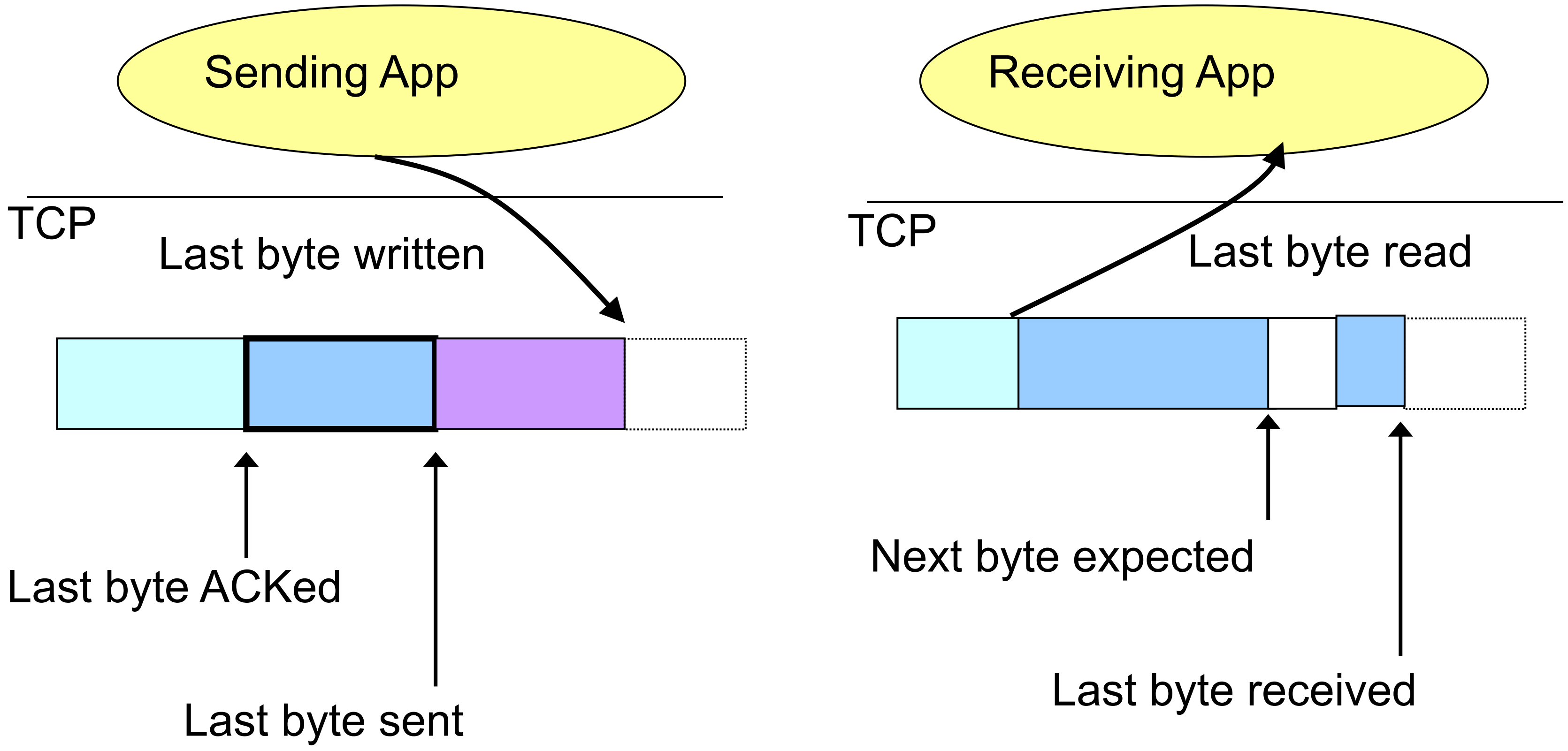
Reliable Transmission

重要

这三种实现可靠传输的方式要记住

- **Detect bit errors:** checksum
  - Used to detect corrupted data at the receiver
  - …leading the receiver to drop the packet
- **Detect missing data:** sequence number
  - Used to detect a gap in the stream of bytes
  - ... and for putting the data back in order
- **Recover from lost data:** retransmission
  - Sender retransmits lost or corrupted data
  - Sender retransmits lost or corrupted data

Sender and Receiver Buffering



Timeout for Data Loss Detection

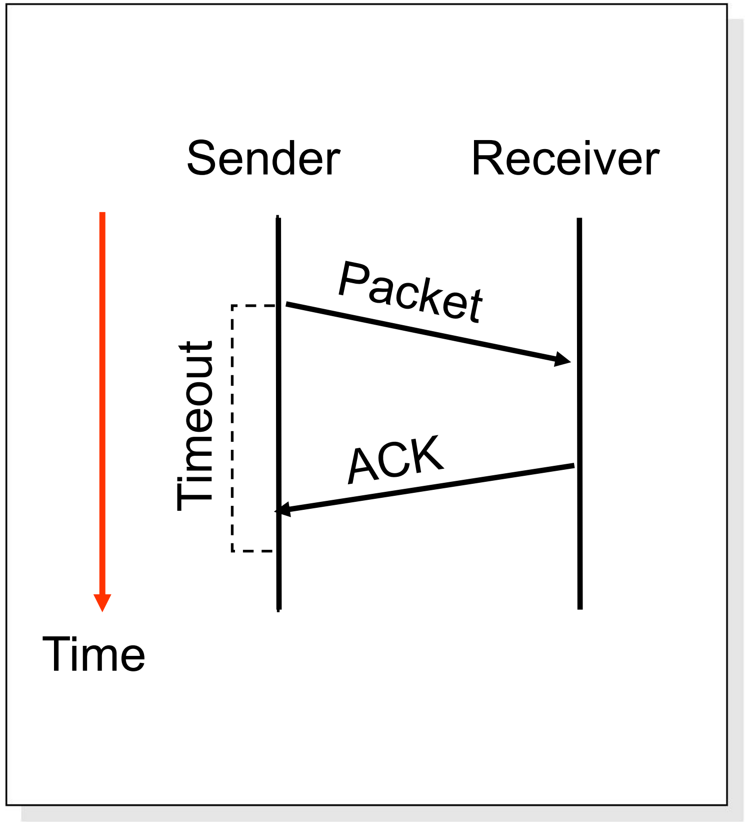
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Automatic Repeat reQuest (ARQ)

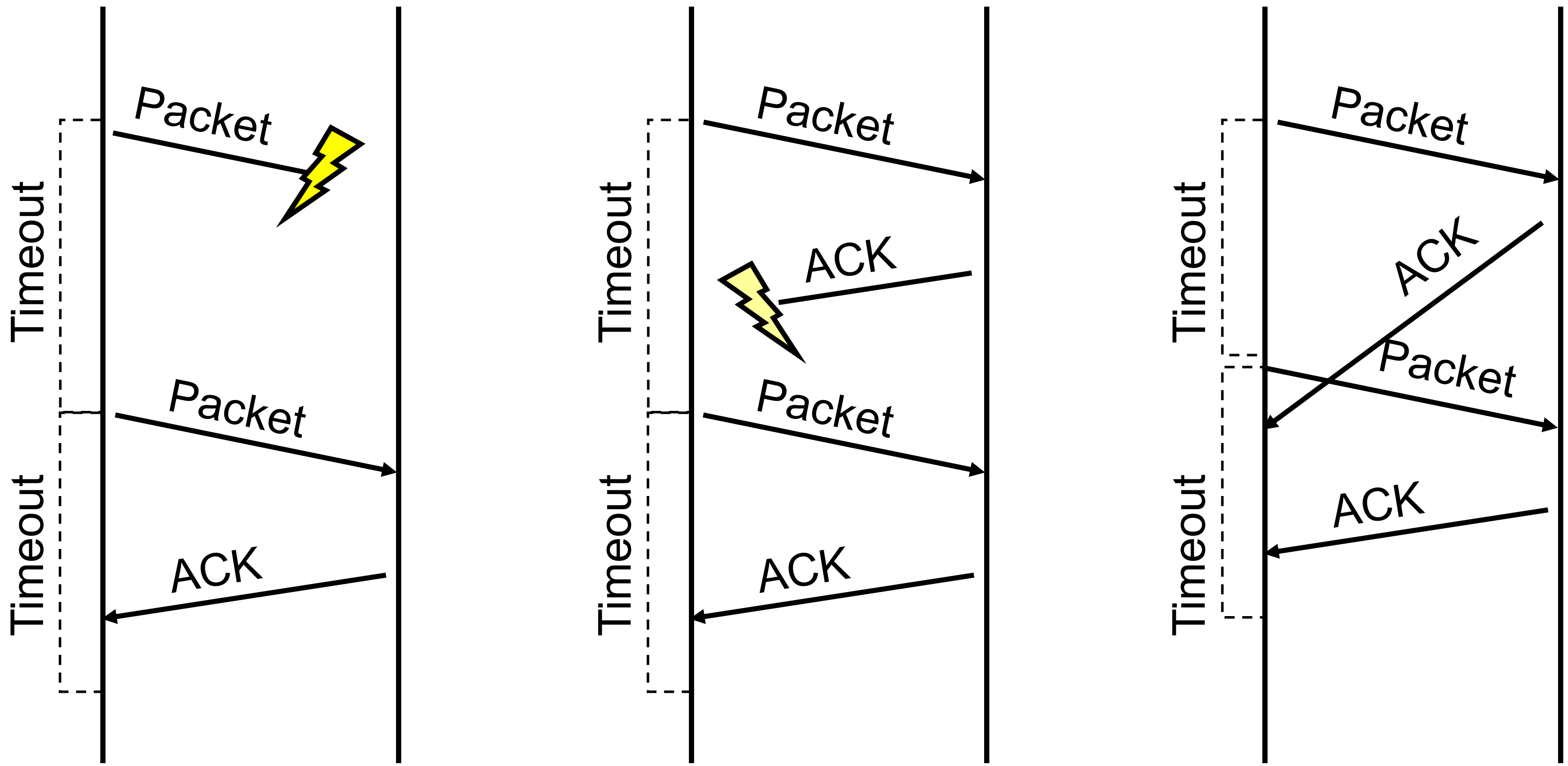
  ARQ重要

  - Receiver sends acknowledgment (ACK) when it receives packet
  - Sender waits for ACK and timeouts if it does not arrive within some time period

- Simplest ARQ protocol
  - Stop and wait: send a packet, stop and wait until ACK arrives



Reasons for Retransmission



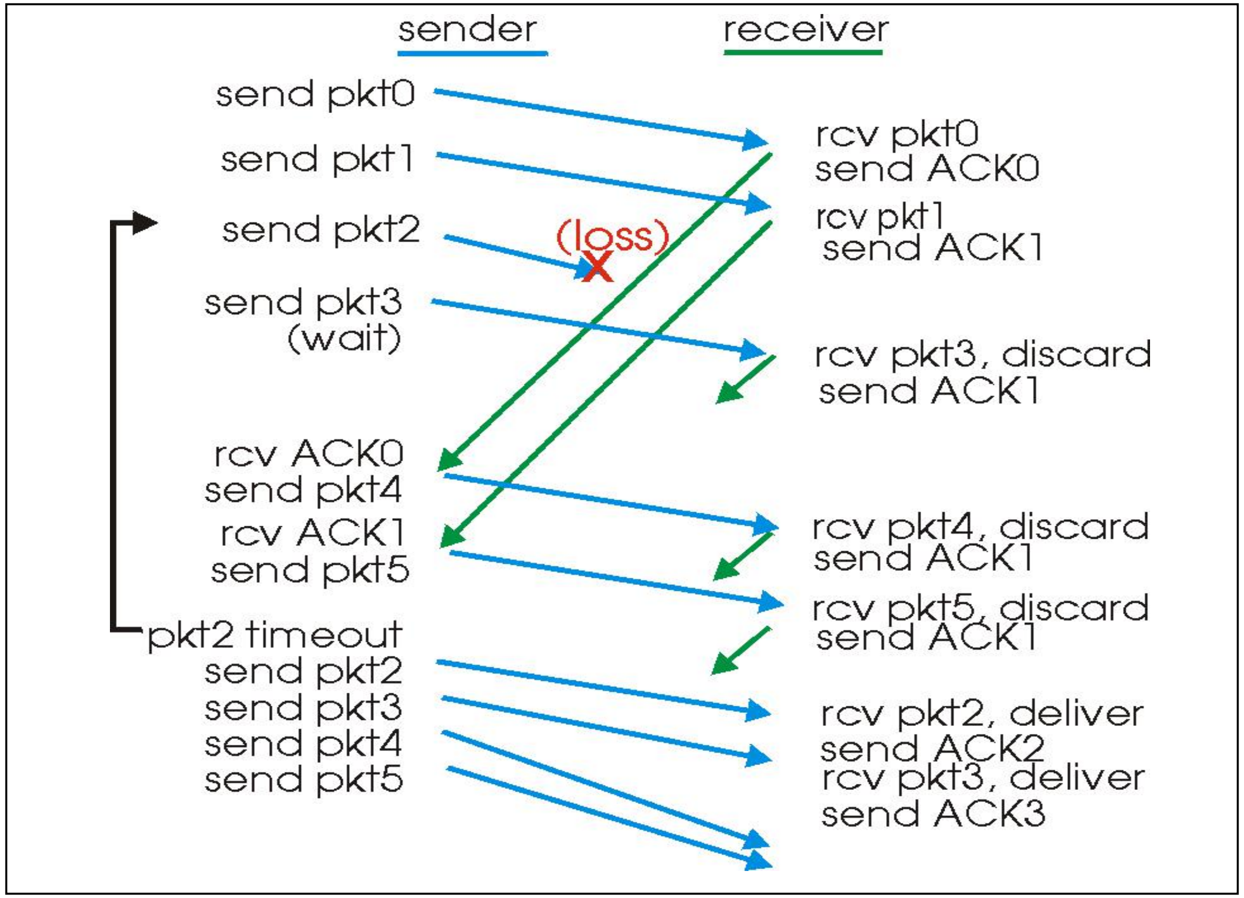
Packet lost

ACK lost
DUPLICATE PACKET

Early timeout
DUPLICATE PACKETS

Timeout-based Retransmission

- Sender transmits a packet and waits until timer expires
- … and then retransmits from **the lost packet onward**



Timeout Setting

- Sender sets a timeout to wait for an ACK
  - Too short: wasted retransmissions
  - Too long: excessive delays when packet lost

- TCP sets timeout as a function of the Round-Trip-Time (RTT)
  - Expect ACK to arrive after an “round-trip time”
  - … plus additional time spent on queuing

- But, how does the sender know the RTT?
  - Can estimate the RTT by watching the ACKs
  - Smooth estimate: keep a running average of the RTT
    EstimatedRTT = a * EstimatedRTT + (1 –a) * SampleRTT
  - Compute timeout: TimeOut = 2 * EstimatedRTT

Fast Retransmission

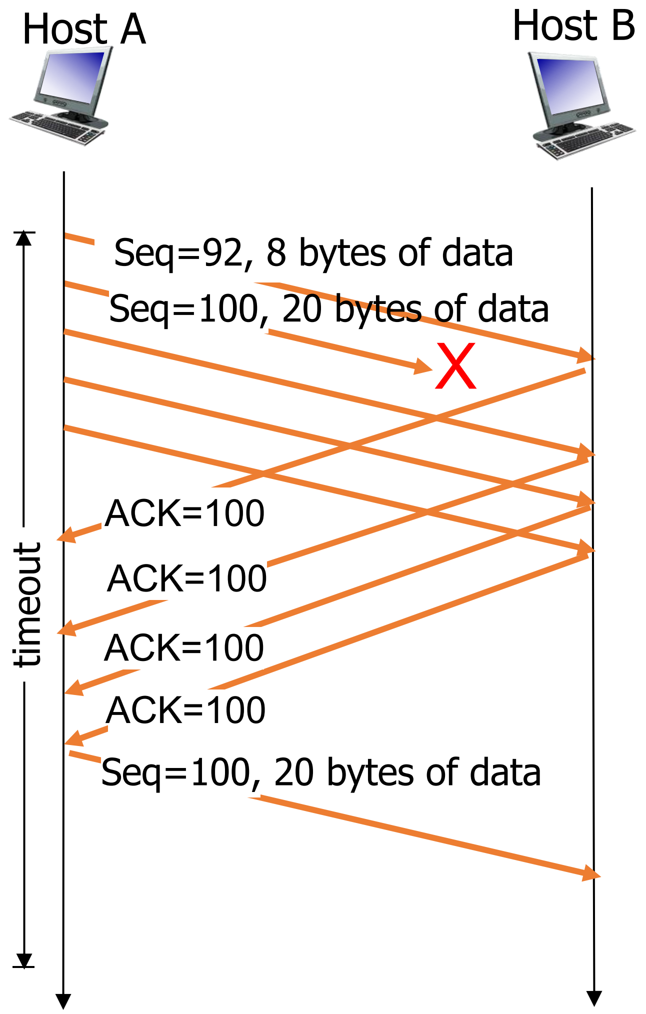
- ARQ/Timeout is slow and inefficient

- Better solution possible under sliding window
  - Although packet n might have been lost
  - … packets n+1, n+2, and so on might get through

- Idea: have the receiver send ACK packets
  - ACK says that receiver is still awaiting n^{th} packet
  - And *repeated* ACKs suggest later packets have arrived
  - Sender can view the "duplicate ACKs" as an early hint for lost of the nth packet

- Fast retransmission
  - Sender retransmits data after the triple duplicate ACK

Tripple Duplicate ACK



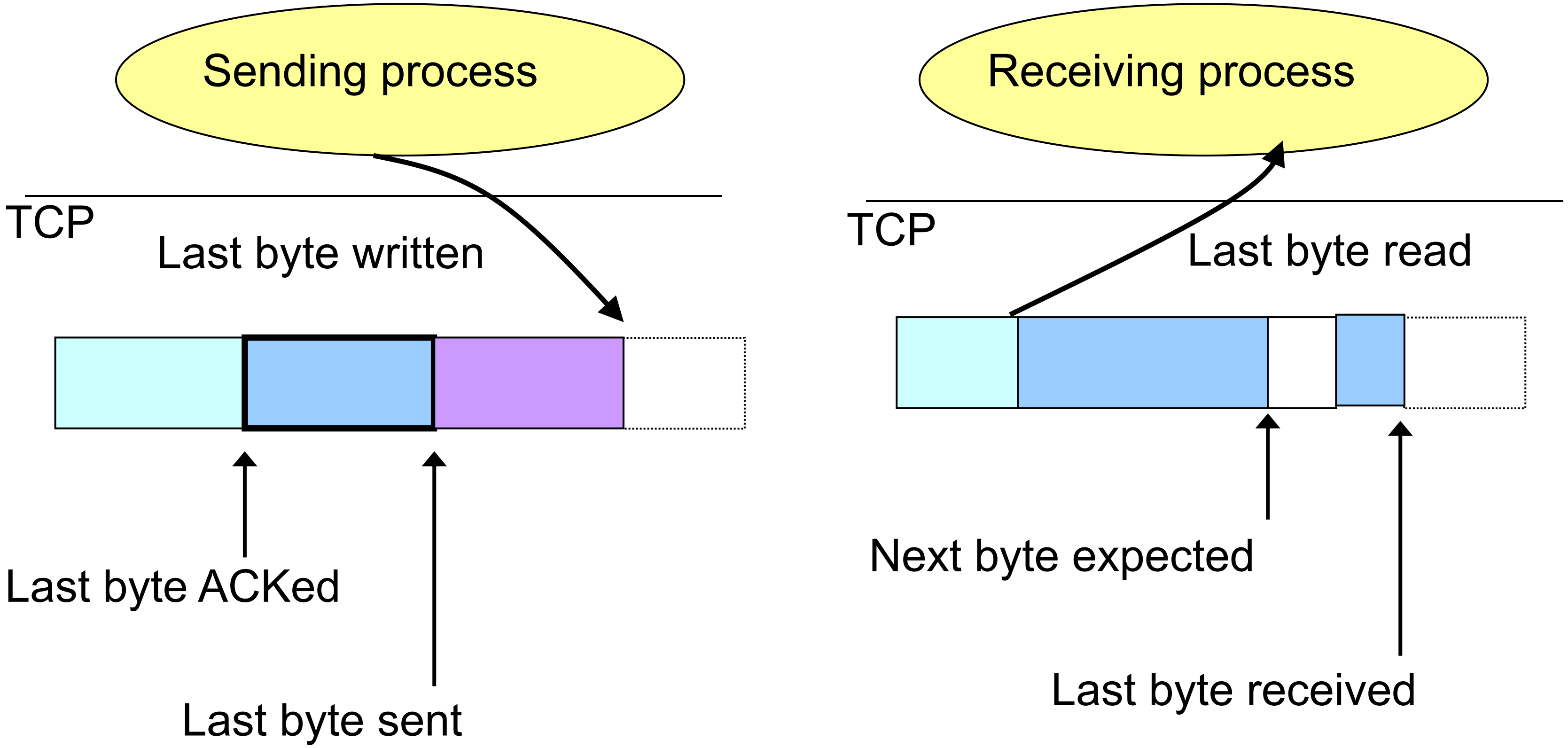
fast retransmit after sender receipt of triple duplicate ACK

Contents

- Basics of Transport Layer
- Reliable Transportation
- Flow Control
- Congestion Control
- Multipath TCP

Flow Control

- The receiver controls the sender, so sender won't overflow receiver's buffer by transmitting too much or too fast



Sliding Window

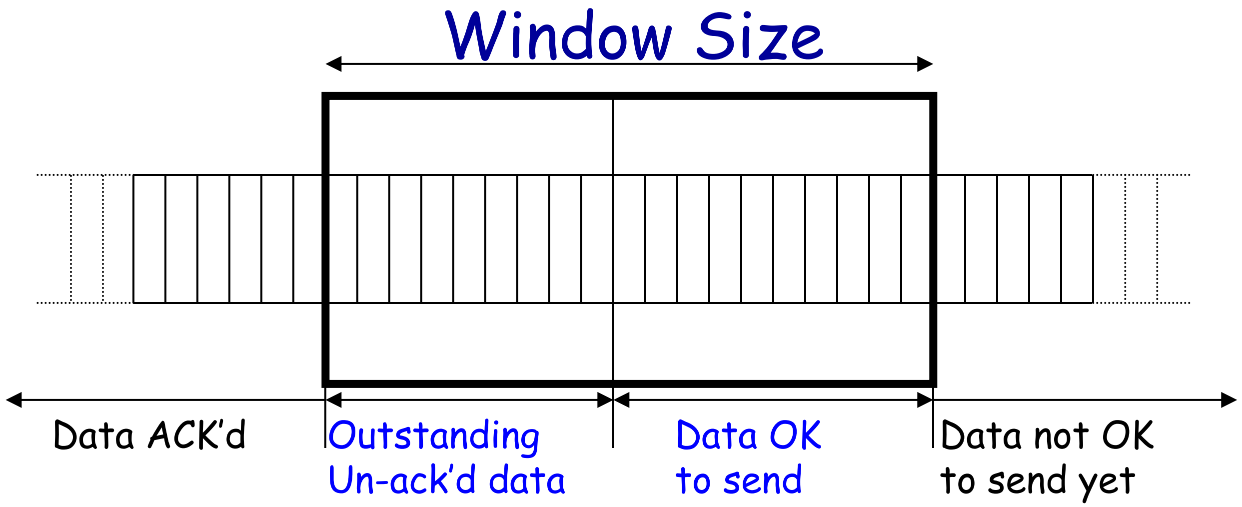
- Stop-and-wait is inefficient
  - Only one TCP segment is “in flight” at a time
  - Especially bad when delay-bandwidth product is high

- Sliding window
  - Allow a larger amount of data “in flight”
  - Allow sender to get ahead of the receiver

Receiver Buffering

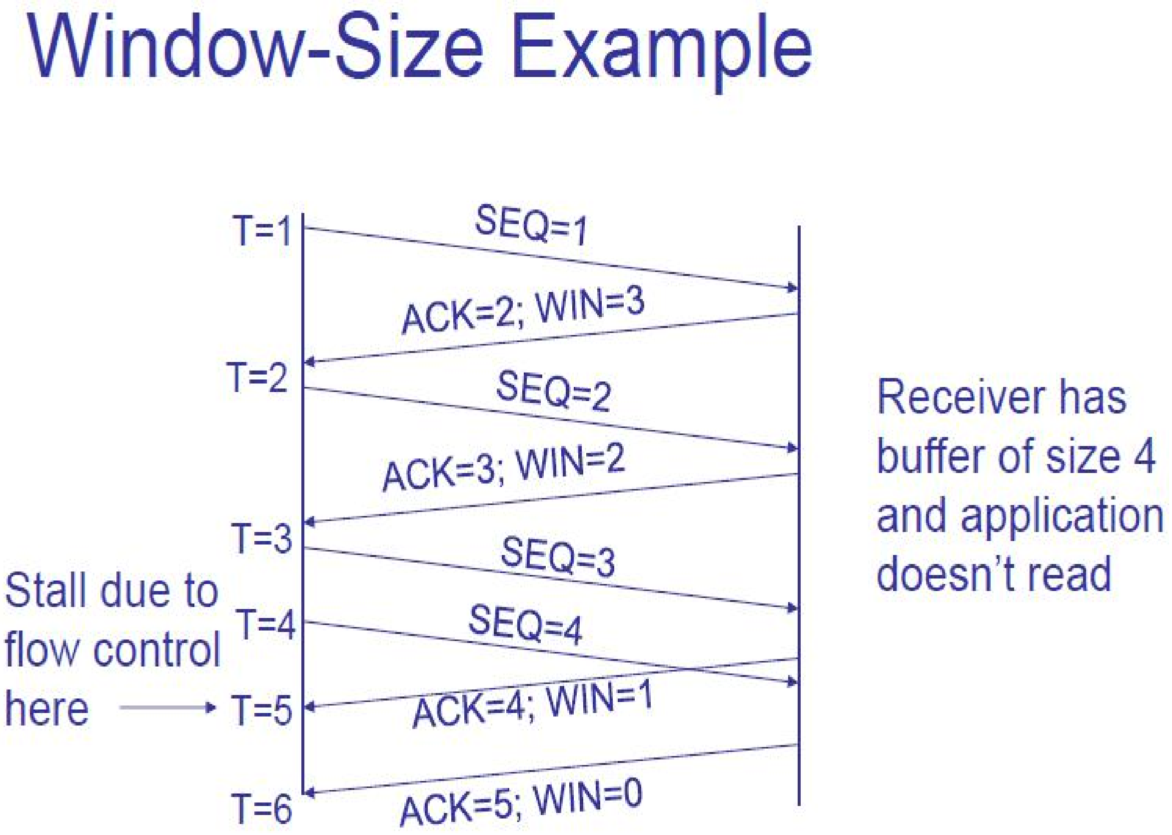
- Window size
  - Amount that can be sent without acknowledgment
  - Receiver needs to be able to store this amount of data

- Receiver advertises the window to the sender



TCP Header Field for Receiver Buffering

Source port		Destination port	
Sequence number			
Acknowledgment			
HdrLen	0	Flags	Advertised window
Checksum		Urgent pointer	
Options (variable)			
Data			

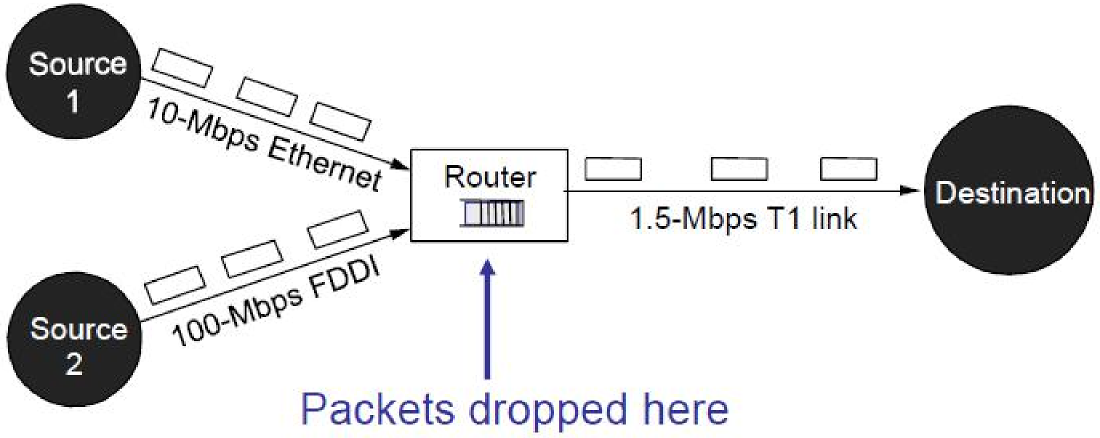


Contents

- Basics of Transport Layer
- Reliable Transportation
- Flow Control
- Congestion Control
- Multipath TCP

Problem of Congestion

- Problem
  - Too many sources sending too much data too fast for *network* to handle
- Manifestations
  - Lost packets (buffer overflow at routers)
  - Long delays (queueing in router buffers)



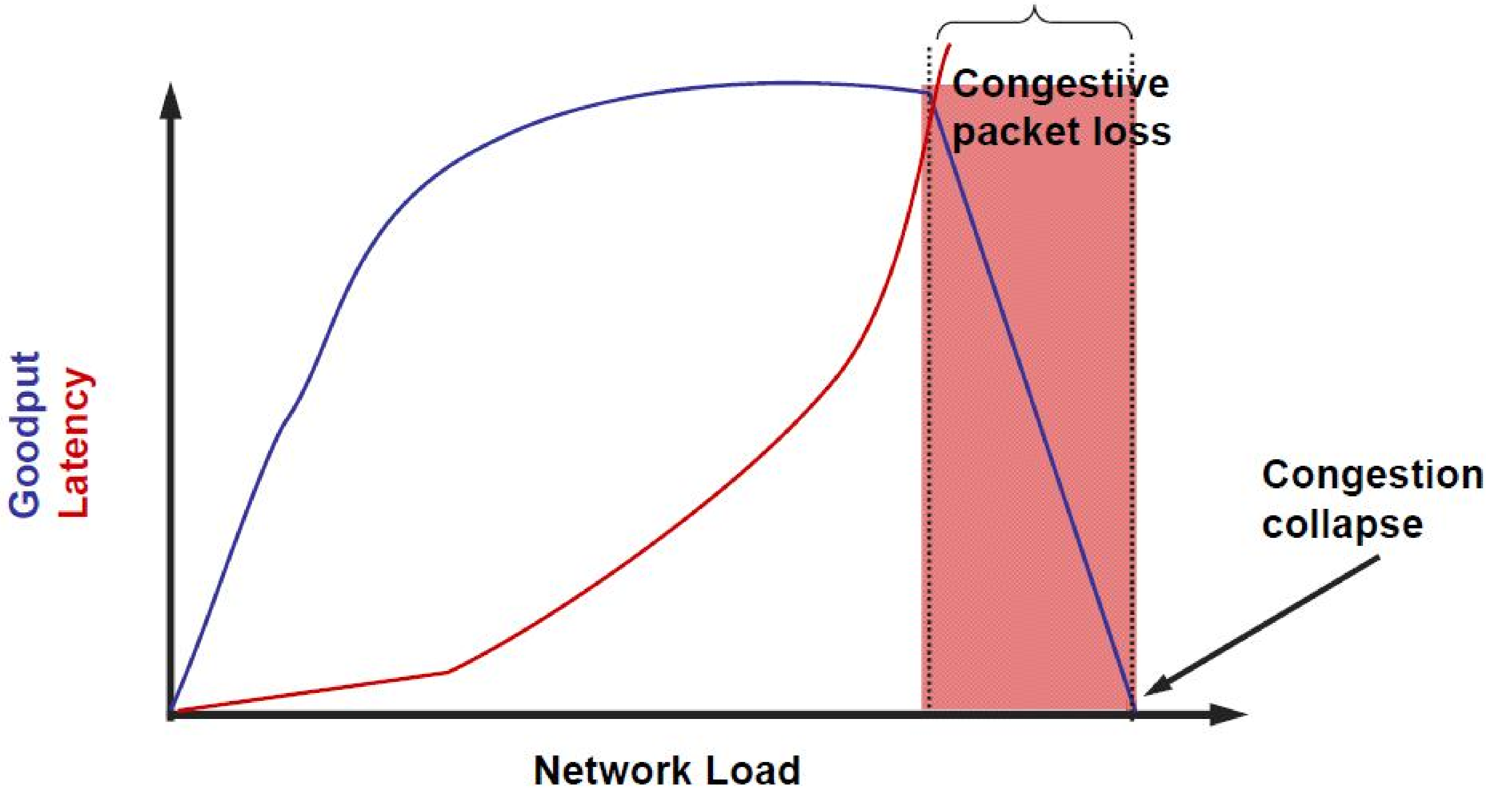
How to Learn Congestion?

- **Delay**
  - Round-trip time (RTT) estimate

重要

- **Loss**
  - Timeout
  - Duplicate acknowledgments
- **Mark**
  - Packets marked by routers with large queues

Drop-tail Queuing



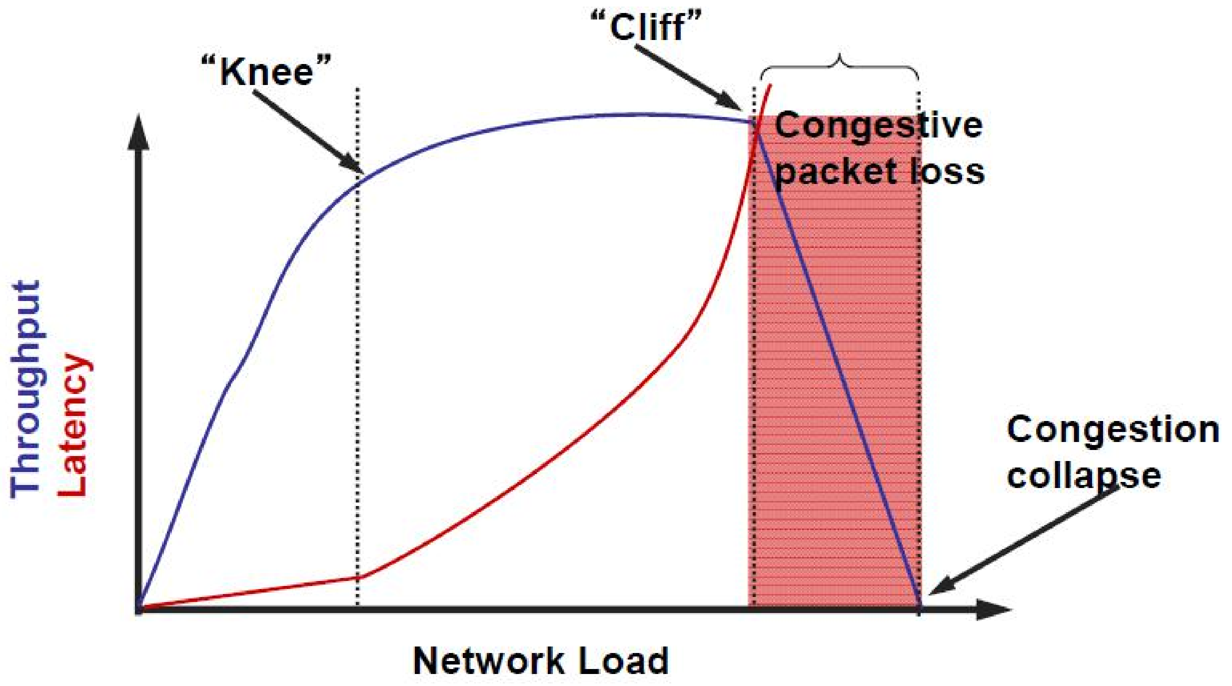
Congestion Collapse

- **Rough definition:** “When an increase in network load produces a decrease in useful work”

- **Why does it happen?**
  - Sender sends faster than bottleneck link speed
  - Packets queue until dropped
  - In response to packets being dropped, sender retransmits
  - Retransmissions further congest link
  - All hosts repeat in steady state…

Proactive vs Reactive

- **Congestion avoidance**: try to stay to the left of the "knee"
- **Congestion control**: try to stay to the left of the "cliff"



重要

Congestion Control

南方科技大學
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- **Flow control**
  - To prevent resource exhaustion at end node
  - Basic idea: receiver advertises sliding window awnd with each ACK

- **Congestion control**
  - To prevent resource exhaustion within network
  - Basic idea: source calculates congestion window cwnd from indication of network congestion (losses, delay, mark)

- **Sender TCP window =** min { awnd, cwnd}

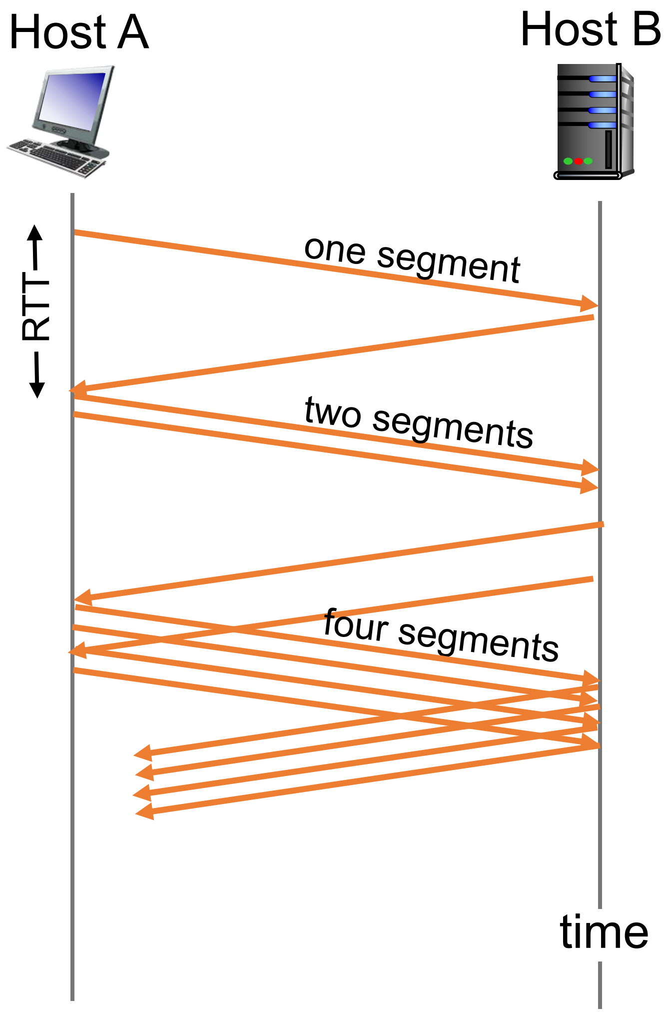
Congestion Control Algorithms

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Tahoe (Jacobson 1988)
    - Slow Start
    - Congestion Avoidance
    - Fast Retransmit
- Reno (Jacobson 1990)
    - Fast Recovery
- Vegas (Brakmo & Peterson 1994)
    - New Congestion Avoidance
- RED, REM…

1. Slow Start

- Goal: quickly find the equilibrium sending rate
- When connection begins, increase rate exponentially
  - initially **`cwnd`** = 1 MSS
  - double **`cwnd`** every RTT
- When to stop exponential increase?
  - First packet loss detected
  - Threshold encountered (ssthresh)



2. Congestion Avoidance

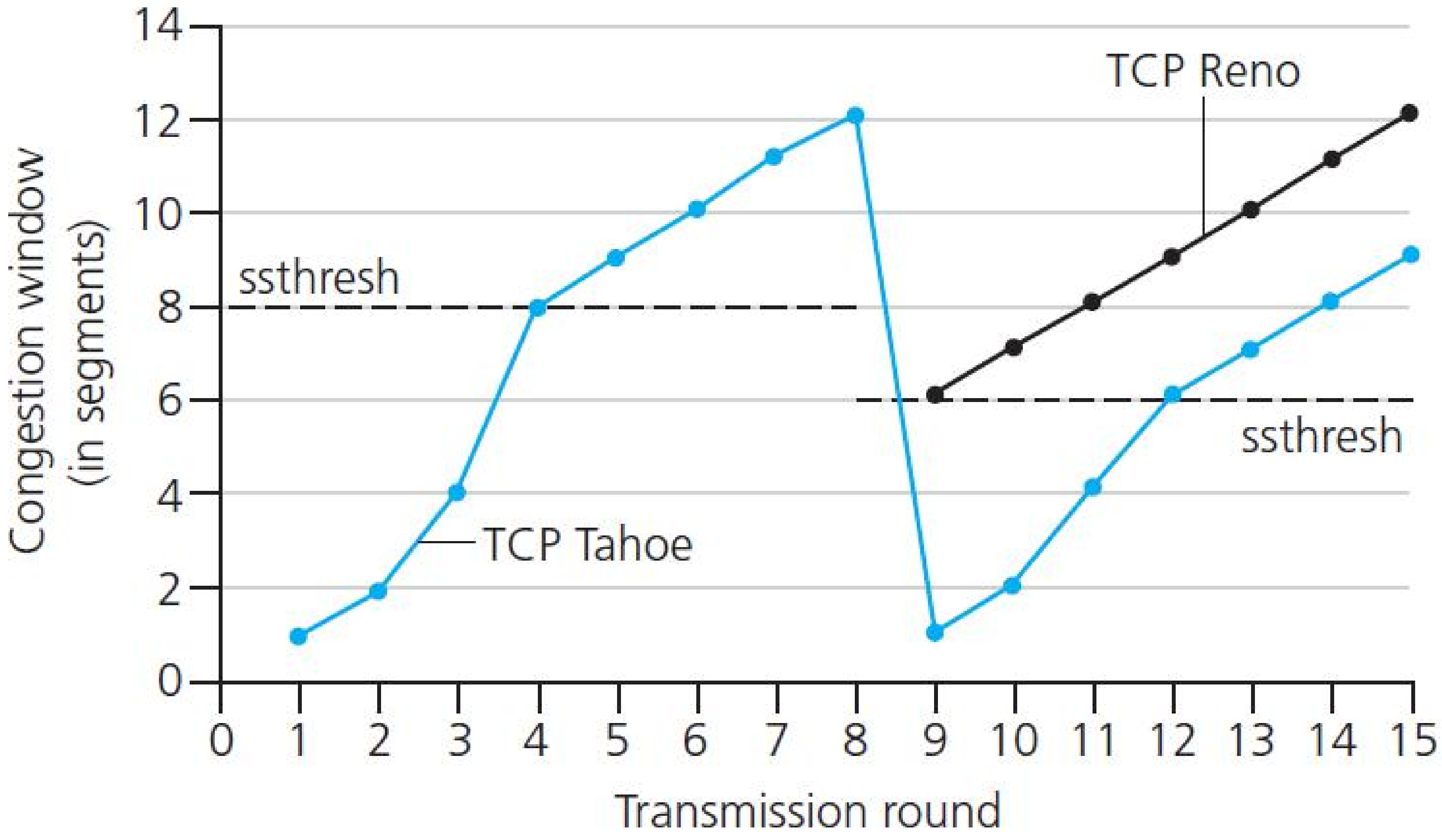
- Goal: detecting and reacting to loss

- Loss indicated by timeout
  - **`cwnd`** set to 1 MSS
  - Window grows exponentially (as in slow start) to threshold, then grows linearly

- Loss indicated by 3 duplicate ACKs: TCP RENO
  - Dup ACKs indicate network capable of delivering some segments
  - **`cwnd`** is cut in half window then grows linearly

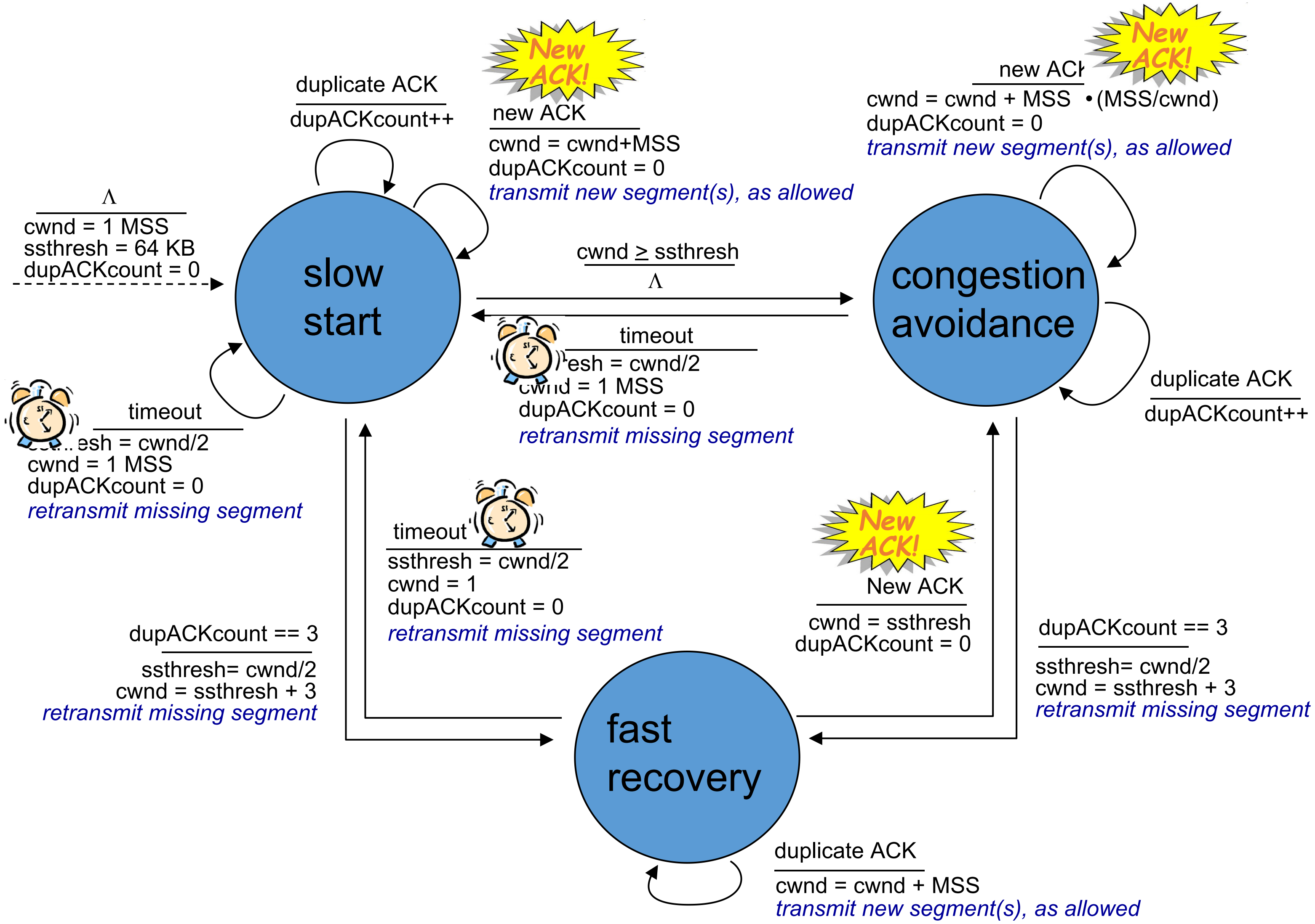
- TCP Tahoe always sets **`cwnd`** to 1 (timeout or 3 duplicate acks)

Evolution of Cwnd

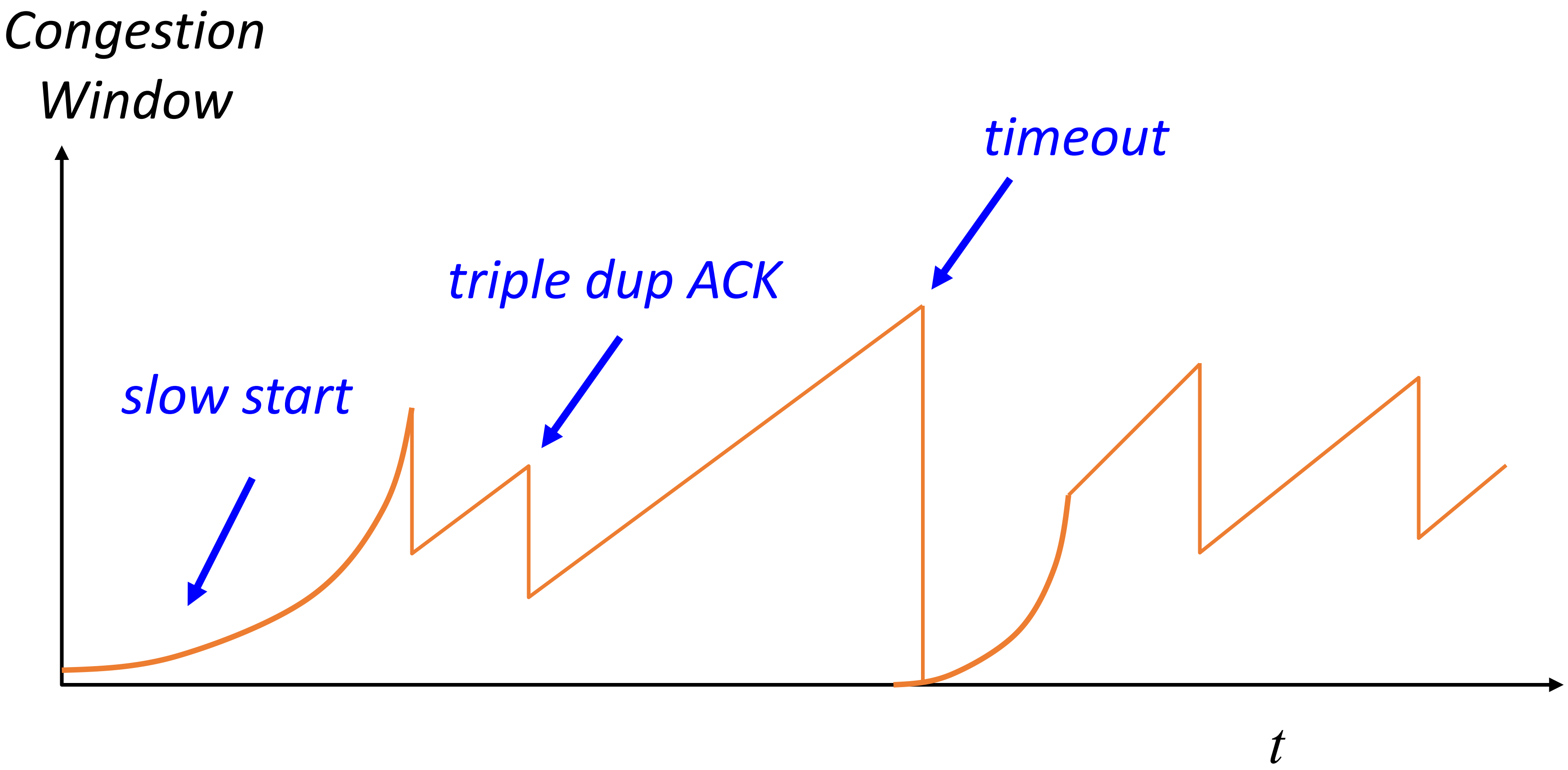


on loss event, `ssthresh` is set to 1/2 of `cwnd` just before loss event

State Transition Diagram

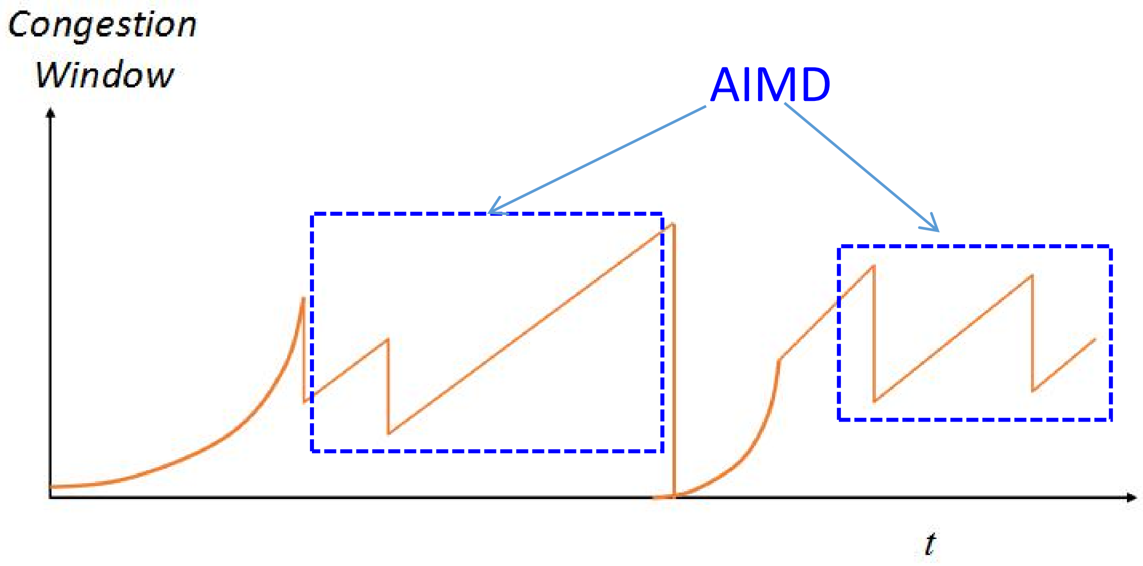


TCP Sawtooth Behavior



AIMD

- AIMD: Additive Increase, Multiplicative Decrease
  - Mechanism working in congestion avoidance and fast transmission



Discussion

- TCP flows can be
  - Elephant flows
  - Mice flows

- Which type of flows are dominant
  - In terms of number of flows?
  - In terms of bandwidth consumption?

- How does Tahoe/Reno work on mice flows?

Contents

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Basics of Transport Layer
- Reliable Transportation
- Flow Control
- Congestion Control
- Multipath TCP

Multipath TCP

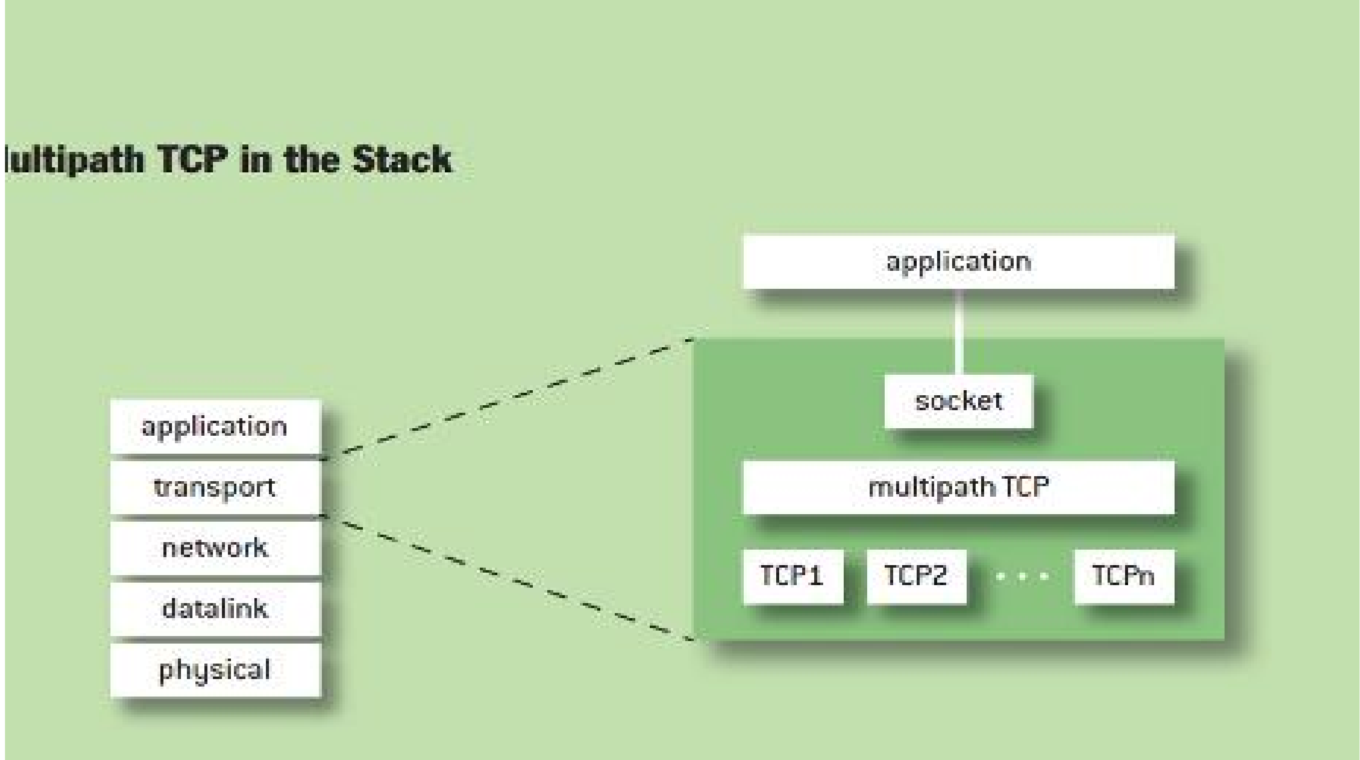
- Regular TCP
  - Single IP connection

- Modern infrastructures
  - **High-end server**: multiple Ethernet cards
  - **Data centers**: rich topologies with many paths
  - **Mobile users**: WiFi and cellular at the same time

- Benefits of multipath
  - Higher throughput
  - Failover from one path to another
  - Seamless mobility

Working with Unmodified Apps

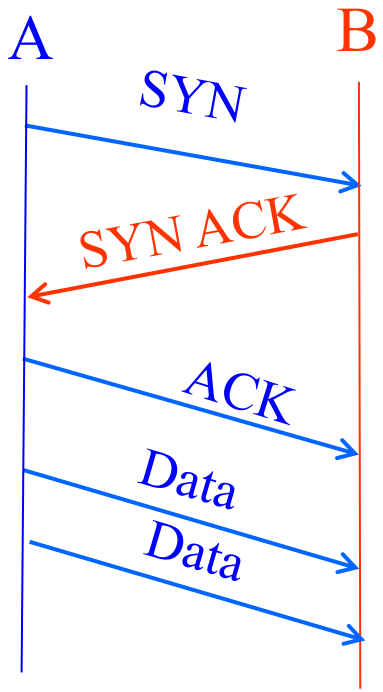
- Present the same socket API and expectations (IP, port, protocol)



Working with Unmodified Apps

- Establish the TCP connection in the normal way
  - Create a socket to a single remote IP address/port
  - And then add more subflows, if possible

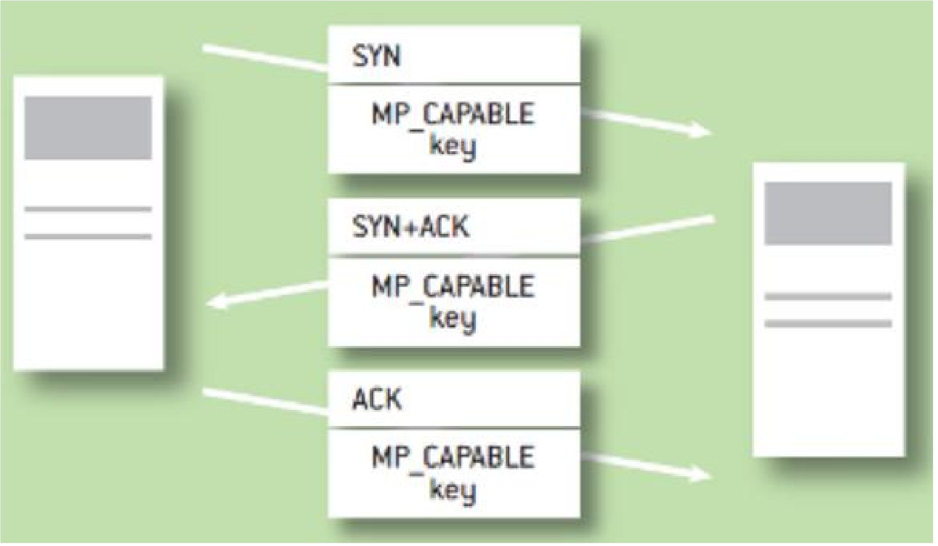
Each host tells its *Initial Sequence Number (ISN)* to the other host.



Negotiating MPTCP Capability

- How do hosts know they both speak MPTCP?

- During the 3-way SYN/SYN-ACK/ACK handshake
  - If SYN-ACK doesn’t contain MP_CAPABLE, don’t try to add any subflows!



Adding Subflows, Idealized

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- How to associate a new subflow with the connection?
  - Use a token generated from original subflow set-up
- How to start using the new subflow?
  - Simply start sending packets with new IP/port pairs
  - … and associate them with the existing connection
- How could two end-points learn about extra IP addresses for establishing new subflows?
  - Implicitly: one end-point establishes a new subflow, to already-known address(es) at the other end-point

Sequence Numbers

- Challenges across subflows
  - More out-of-order packets due to RTT differences
  - Access networks that rewrite sequence numbers
  - Middleboxes upset by discontinuous TCP byte stream
  - Need to retransmit lost packets on a different subflow

- Solution: two levels of sequence numbers
  - Sequence numbers per subflow
  - Sequence numbers for the entire connection

- Enables
  - Efficient detection of loss on each subflow
  - Retransmission of lost packet on a different subflow

Read More on MPTCP

南方科技大学 SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- Improving Datacenter Performance and Robustness with Multipath TCP [SIGCOMM 2011]

- Design, Implementation and Evaluation of Congestion Control for Multipath TCP [NSDI 2011]

Summary

SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

- What is transportation layer for?
- Major issues
  - Reliable transmission
  - Flow control
  - Congestion control
- New infrastructures and applications drive new extensions