TCP and Congestion Control (Day 2)

Yoshifumi Nishida
nishida@csl.sony.co.jp
Sony Computer Science Labs, Inc

Today's Contents

- Part1: TCP Issues and Solutions
- Part2: Congestion Control
- Part3: Simulating TCP

Part 1: TCP Issues and Solutions

- Long Fat Networks
- Ambiguity of Acknowledgment
- Connection Setup Overhead
- Security Vulnerabilities

Long Fat Network (1)

- What are "Long Fat Networks"?
  - A network with large bandwidth and long delay.
    - ex. High-capacity satellite channels
- TCP performance
  - TCP performance is calculated by Window Size and RTT.

$$\text{TCP peformance} = \frac{\text{Window Size}}{\text{Round-Trip Time}}$$

  - Required Window Size for networks.

Required Window Size = Round-Trip Time × Maximum Transfer Rate of the network.

  - But Maximum window size is limited to 65,535 bytes.
    - The window size in TCP header has only 16 bits.

Long Fat Network (2)

- 65,535 bytes window size is not enough for Long Fat Networks!
  - Example of Long Fat Networks.

Transfer rate	RTT(msec)	Required Window Size (bytes)
1.54Mbps (T1)	500	95,500
45Mbps (T3)	60	337,500

Long Fat Network (3)

- Window Scale Option
  - Extension to specify large window size
    - defined in RFC1323: TCP Extensions for High Performance.
  - Option Format:

Window Scale Option

kind=3	len=3	shift count
1 byte	1 byte	1 byte

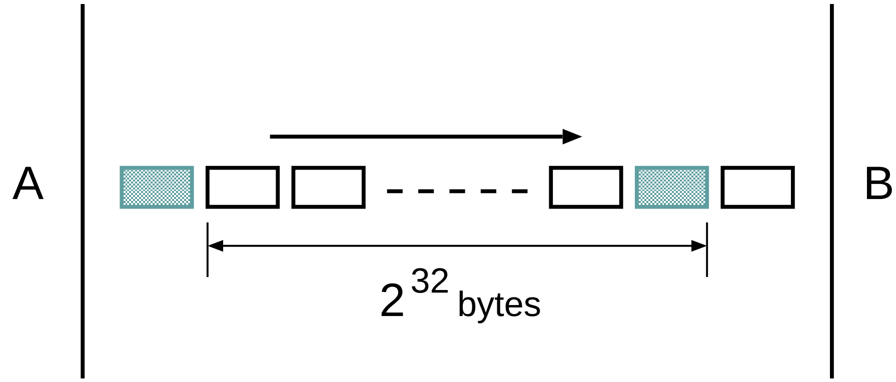
  - The window size is treated as:

$$\text{Window Size} = \text{Value in Window Size field} \times 2^{\text{shift count}}$$

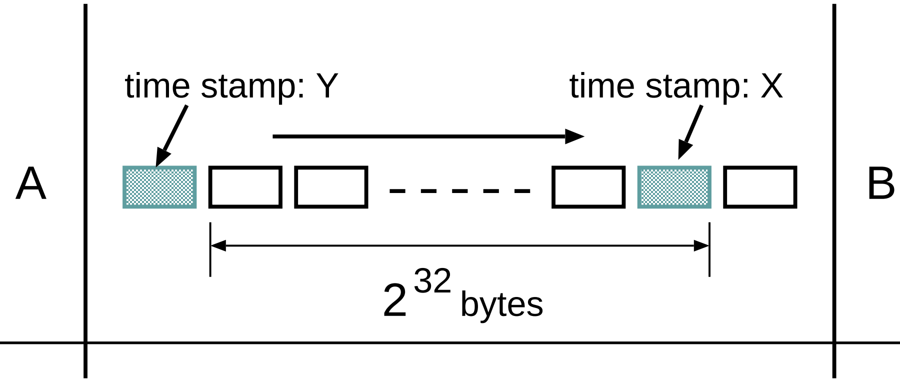
  - Max value of shift count is limited to 14.
    - Maximum window size is 1,073,725,440 (65535 * 2^14) bytes with this option.

Long Fat Network (4)

- Sequence Number Wrap Around
  - Another issue for Long Fat Networks.
  - 32-bit sequence number space may wrap around in LFNs.

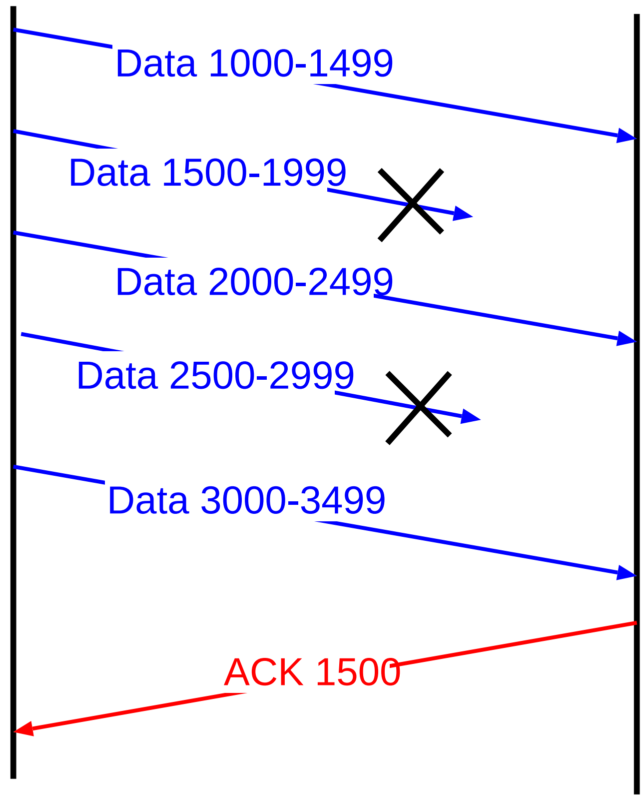


- Time Stamp Option
  - Provides transmit time information.
    - TCP can identify each packet with Time Stamp and Sequence Number.



Ambiguity of the Acknowledgment (1)

- Cumulative ACK style is ambiguous, when multiple packets are lost.
  - TCP cannot identify which packets are lost exactly.
    - Causes poor performance over lossy networks (ex. wireless networks)

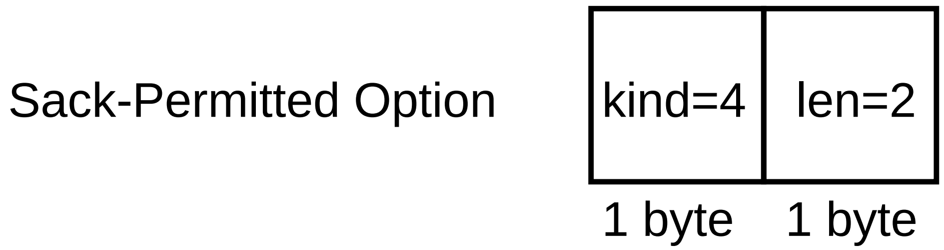


Ambiguity of the Acknowledgment (2)

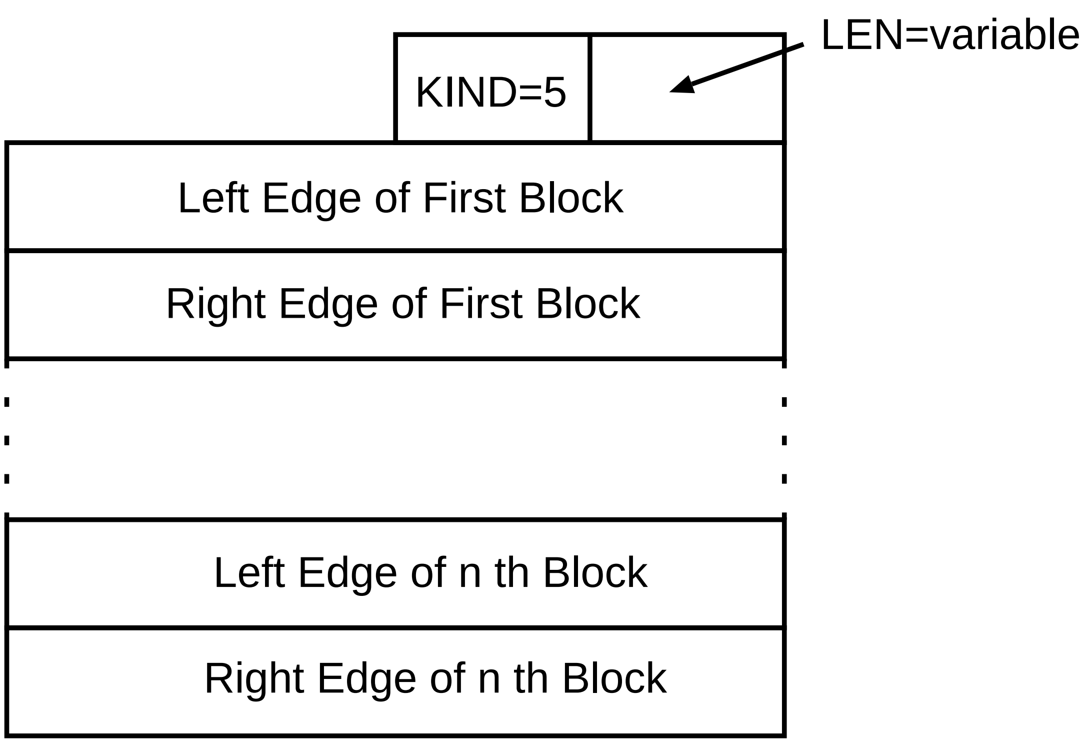
- Selective Acknowledgment Options
  - Provides precise information about packet arrivals.
  - Two options are defined in RFC2018.
- SACK Permitted Option
  - Used in a SYN packet to indicate that SACK option can be used.
- SACK Option
  - Used in an ACK packet to indicate which packets were received precisely.

Ambiguity of the Acknowledgment (3)

- SACK Permitted Option

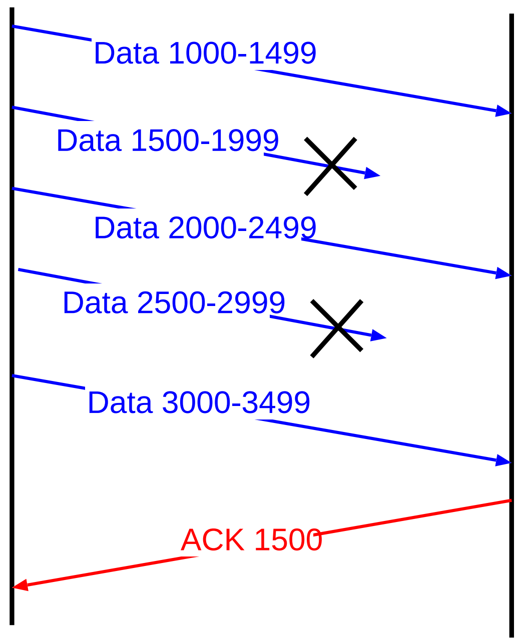


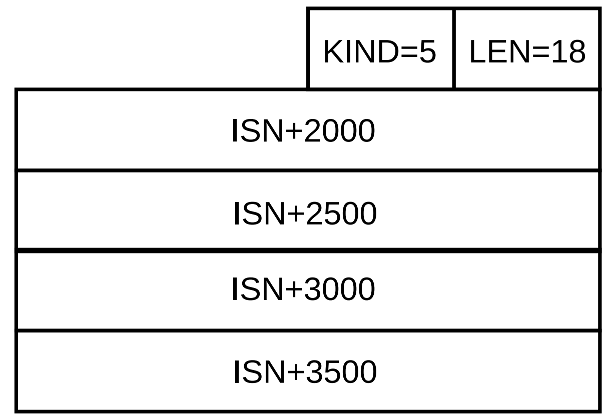
- SACK Option



Ambiguity of the Acknowledgment (4)

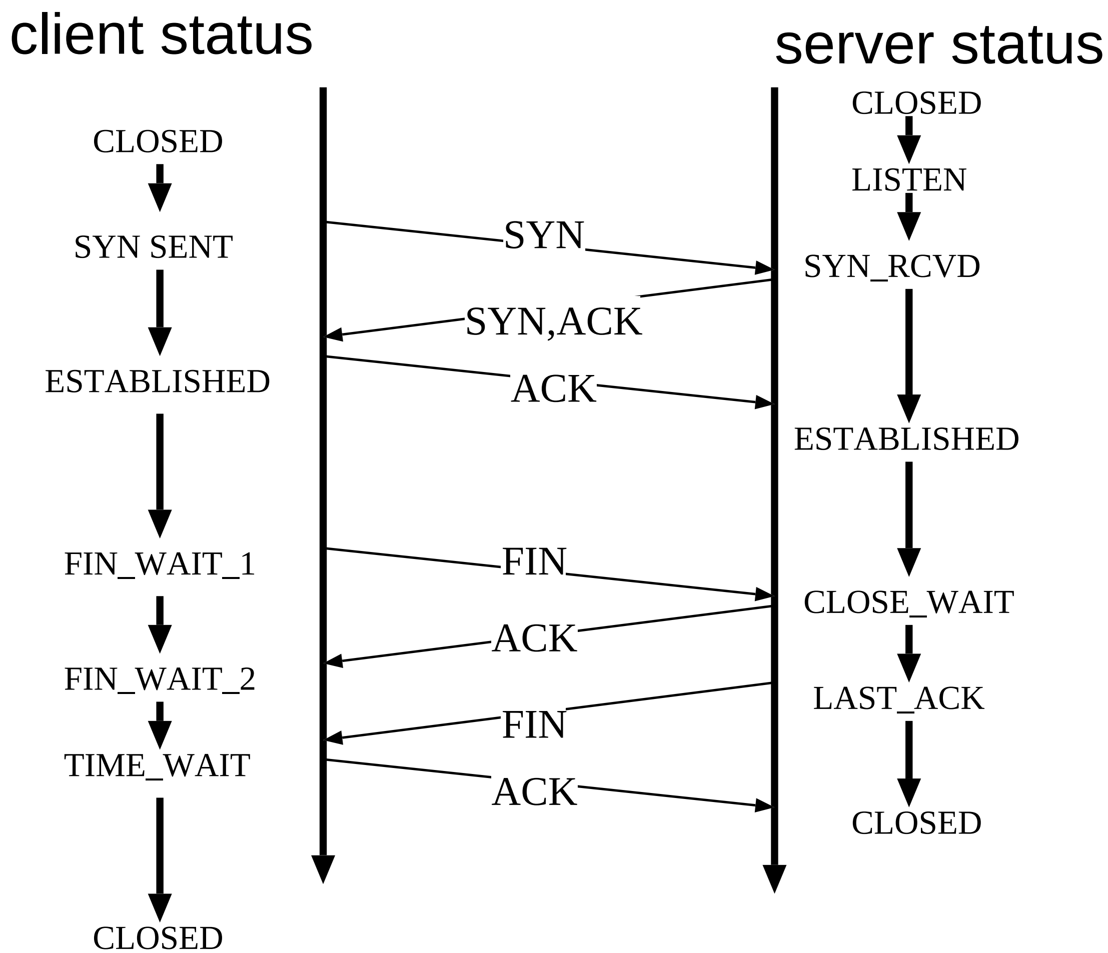
- Example of the SACK option





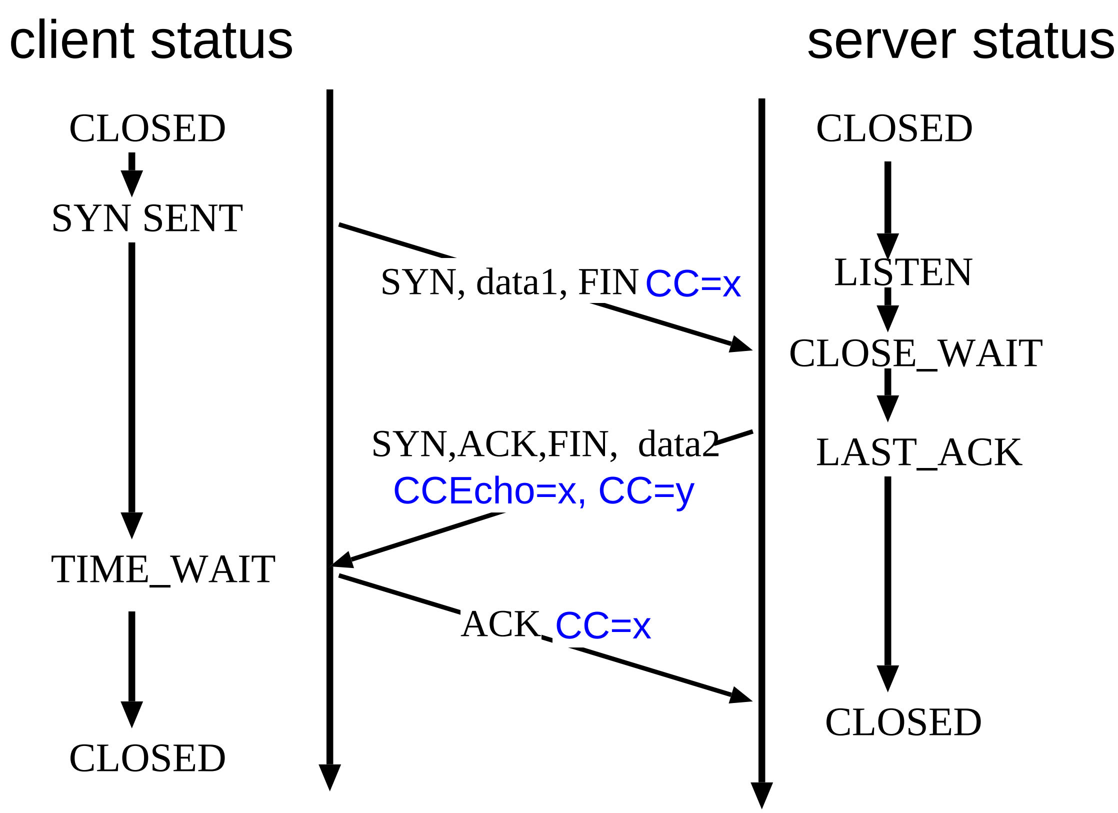
Connection Setup Overhead (1)

- TCP is not suitable for a transaction service.
  - TCP requires 3 packets for connection setup.
  - TCP requires 4 packets for connection termination.



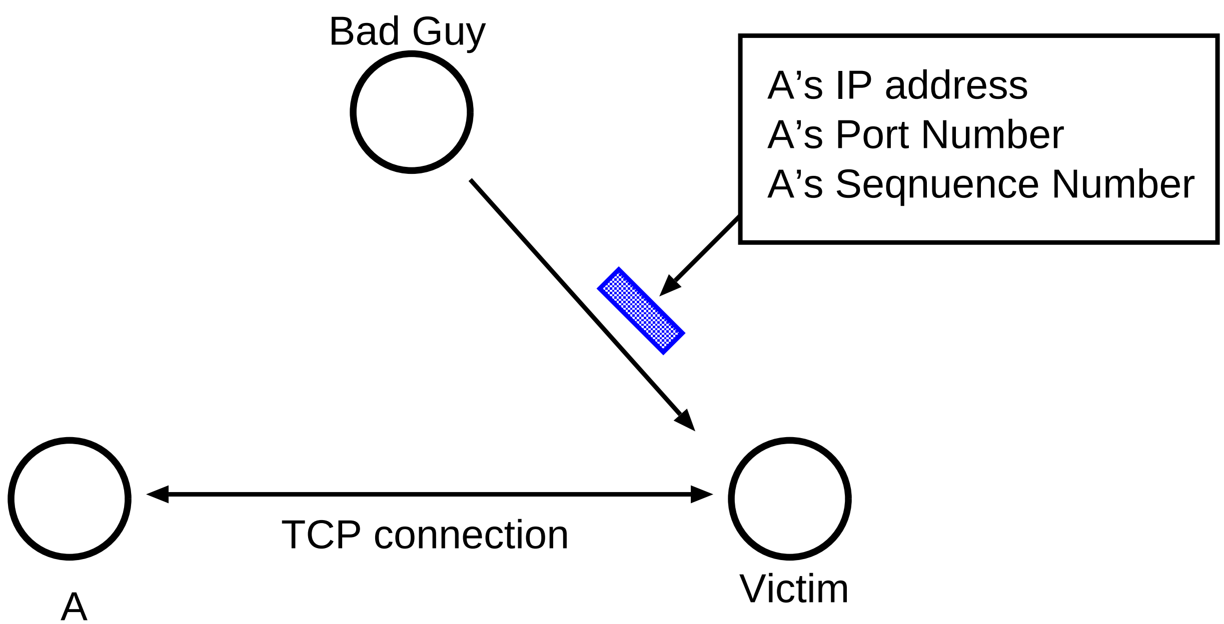
Connection Setup Overhead (2)

- T/TCP option
  - TCP extension for transactions
    - Exchange data with 3 packets.
    - Use Connection Count (CC) to bypass 3 way handshake
    - Defined in RFC1644.



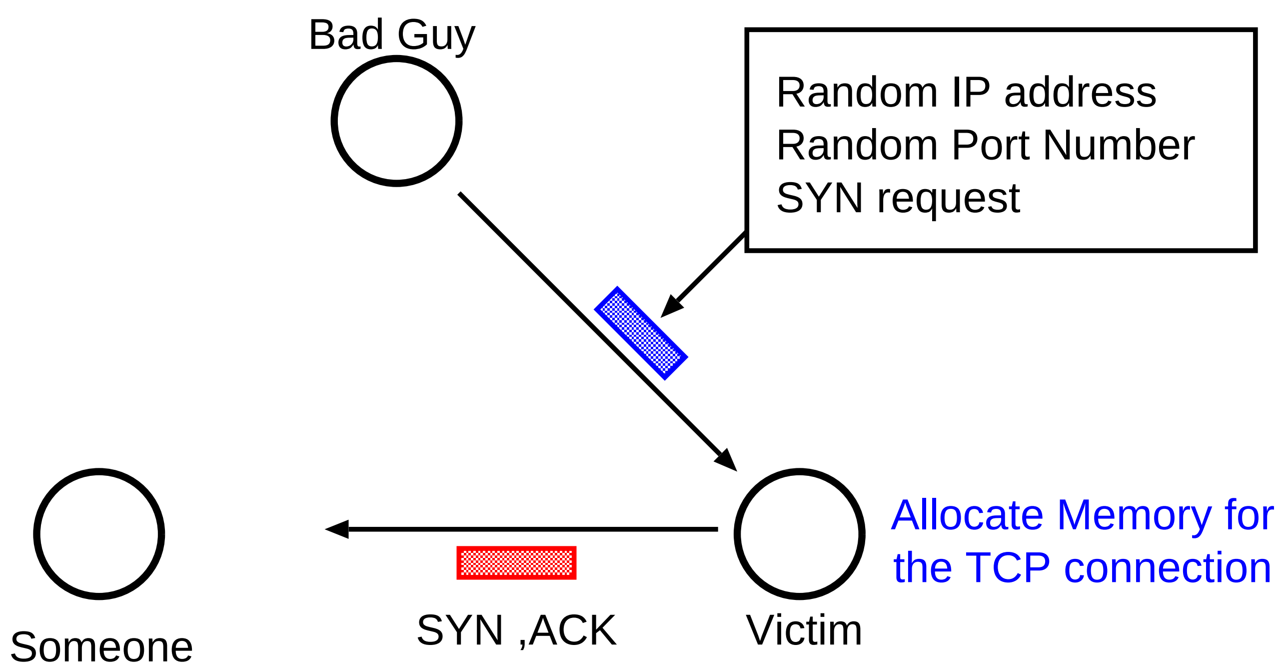
Security Vulnerabilities (1)

- Sequence Number Attack
  - If someone can guess Sequence Number used in your TCP connections...
    - He can "hijack" your TCP connection.
      - TCP checks IP address and Port Number and Sequence number.
  - But most of current implementations use cryptic algorithms to generate ISN (Initial Sequence Number).



Security Vulnerabilities (2)

- SYN Flood Attack
  - Denial of Service Attack
  - Send a large number of SYN packets with Random source IP address
  - Cause memory overflow on the victim
    - TCP allocates memory when it receives SYN packets.

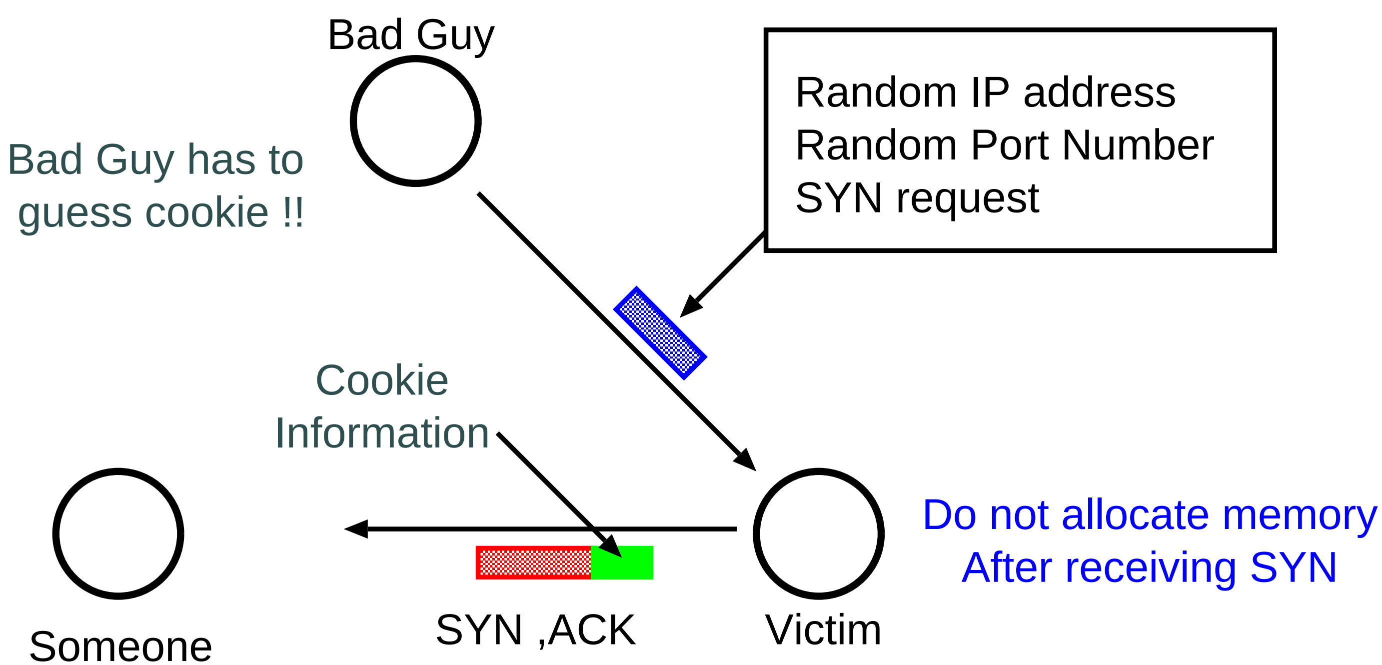


Security Vulnerabilities (3)

- Protection against SYN Flood Attacks
  - IP level solution
    - Use IPsec
      - Allows TCP connection only to authenticated hosts
    - Use IP filter
      - Filters out IP addresses that do not look legitimate

Security Vulnerabilities (4)

- Protection against SYN Flood Attacks
  - TCP level solution
    - SYN Cache
      - Reduces the memory size allocated after receiving SYN packets
    - SYN Cookie
      - Sends back ACK with Special Sequence Number in response to SYN packets.
      - Does not allocate memory at all after receiving SYN.

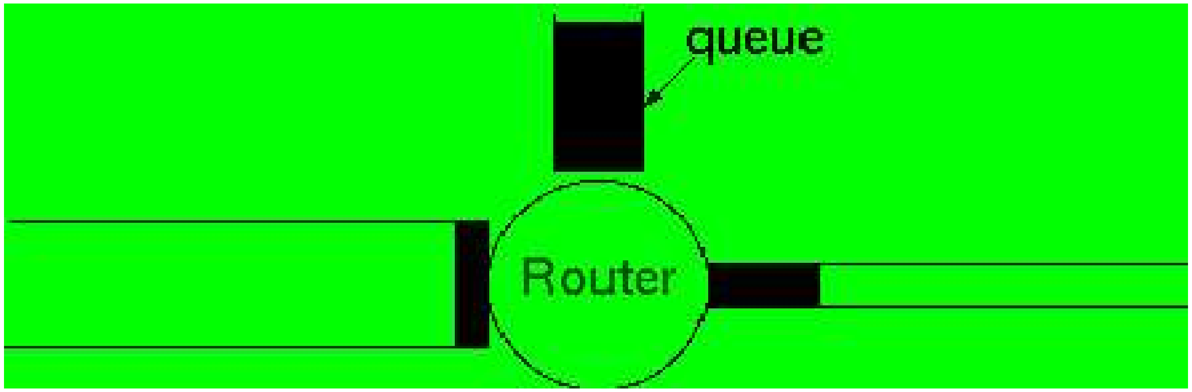


Part 2: Congestion Control

- How does congestion happen?
- Why congestion is difficult?
- Congestion Control by TCP

How does congestion happen?

- Congestion occurs when there is too much traffic in the networks
- Routers have queuing capability.
  - If a router cannot transmit packets at a given instance, it stores packets in the queue and waits for the next chance to transmit.
  - Queue has limited size
  - If queue data exceeds limit, packet will be discarded.



Congestion Tends To Get Worse

- If congestion occurs..
  - Packet transfers are delayed
  - Packets are discarded

    ↓
  - Some protocols/applications try to retransmit data
  - Users try to retransmit the data or request the same data again and again

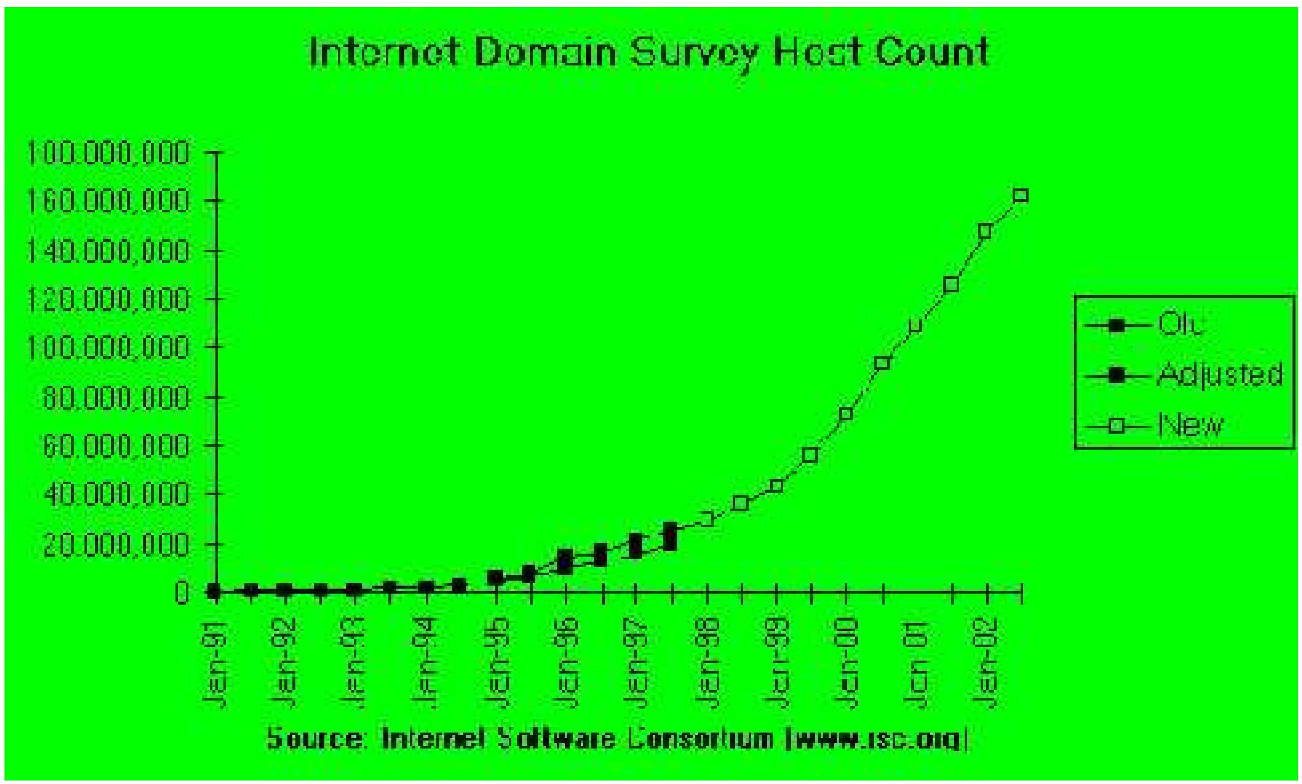
    ↓
- The ratio of valid data is decreasing...

  ↓
- Congestion Collapse
  - We cannot use network!

Why is congestion control difficult? (1)

- Internet is designed to be autonomous.
  - No central control.
  - There is no way to control each user's behavior.

- Internet is very huge and still expanding.



Why is congestion control difficult? (2)

- The status of the Internet is hard to grasp
  - It is difficult to determine how many user/application share the network exactly.
  - It is difficult to determine the source of the congestion exactly.
  - It is difficult to determine the capacity of the networks exactly.
  - It is difficult to determine how much networks are congested exactly.
  - It is difficult to determine why packets are lost exactly.

Congestion Control by TCP

- Autonomous control by end-nodes.
  - No central control
- Simple estimation algorithms for network conditions.
  - Selects appropriate transfer rate for each network.
    - Avoid congestion as much as possible.
  - Detects congestion
    - Avoid congestion collapse as much as possible.

TCP Congestion Control Concept (1)

- Primary concept
  - There is no way for TCP to determine the network condition exactly.
  - TCP regards ALL packet losses as congestion.
- Transmission control with simple algorithms.
  - If packets are NOT lost..
    - TCP assumes network is NOT congested → Increases transfer rate.
  - If packets are lost..
    - TCP assumes network is congested → Decreases transfer rate.

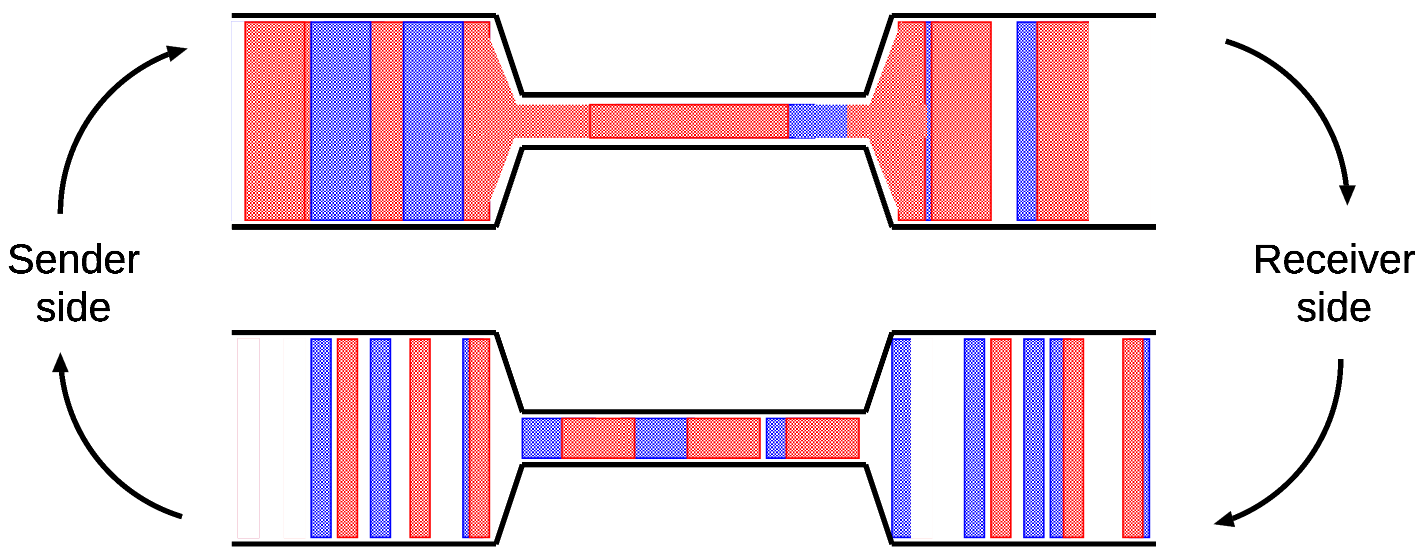
      ↓
  - TCP increases transfer rate until packet loss occurs.
    - TCP tries to estimate the limit of the network by causing packet loss.

TCP Congestion Control Concept (2)

- How to control transfer rate?
  - Introduces new variable "congestion window (cwnd)" in sliding window scheme.
  - Adjusts the amount of data being injected into the networks
- How to determination Window Size?
  - Window Size = min(advertised window, congestion window)
    - Advertised Window is used for flow control, which is sent from receiver side.
    - Congestion Window is used for congestion control, which is decided on sender side.

TCP Congestion Control Concept (3)

- Self-Clocking
  - Uses an arrival of ACK as a trigger of new packet transmission.
    - Packet arrval interval will change according to the characteristics of the transit networks.
  - Adjusts transfer rate to the network capacity automatically.
    - No need for complex mechanism for controlling transfer rate!



History of TCP Congestion Control

- 3 major versions of TCP congestion control
  - TCP congestion control scheme has been deployed with BSD Unix.
  - Tahoe
    - Implemented in 4.3BSD Tahoe, Net/1 (around 1988)
    - Slow Start and Congestion Avoidance
    - Fast Retransmit
  - Reno
    - Implemented in 4.3BSD Reno, Net/2 (around 1990)
    - Fast Recovery after Fast Retransmit
  - NewReno
    - No reference implementation (around 1996)
    - New Fast Recovery Algorithm

Tahoe TCP

- Two major congestion control schemes
  - Slow-Start and Congestion Avoidance
    - Increases Window Size
  - Fast Retransmit
    - Detects congestion

Slow-Start and Congestion Avoidance (1)

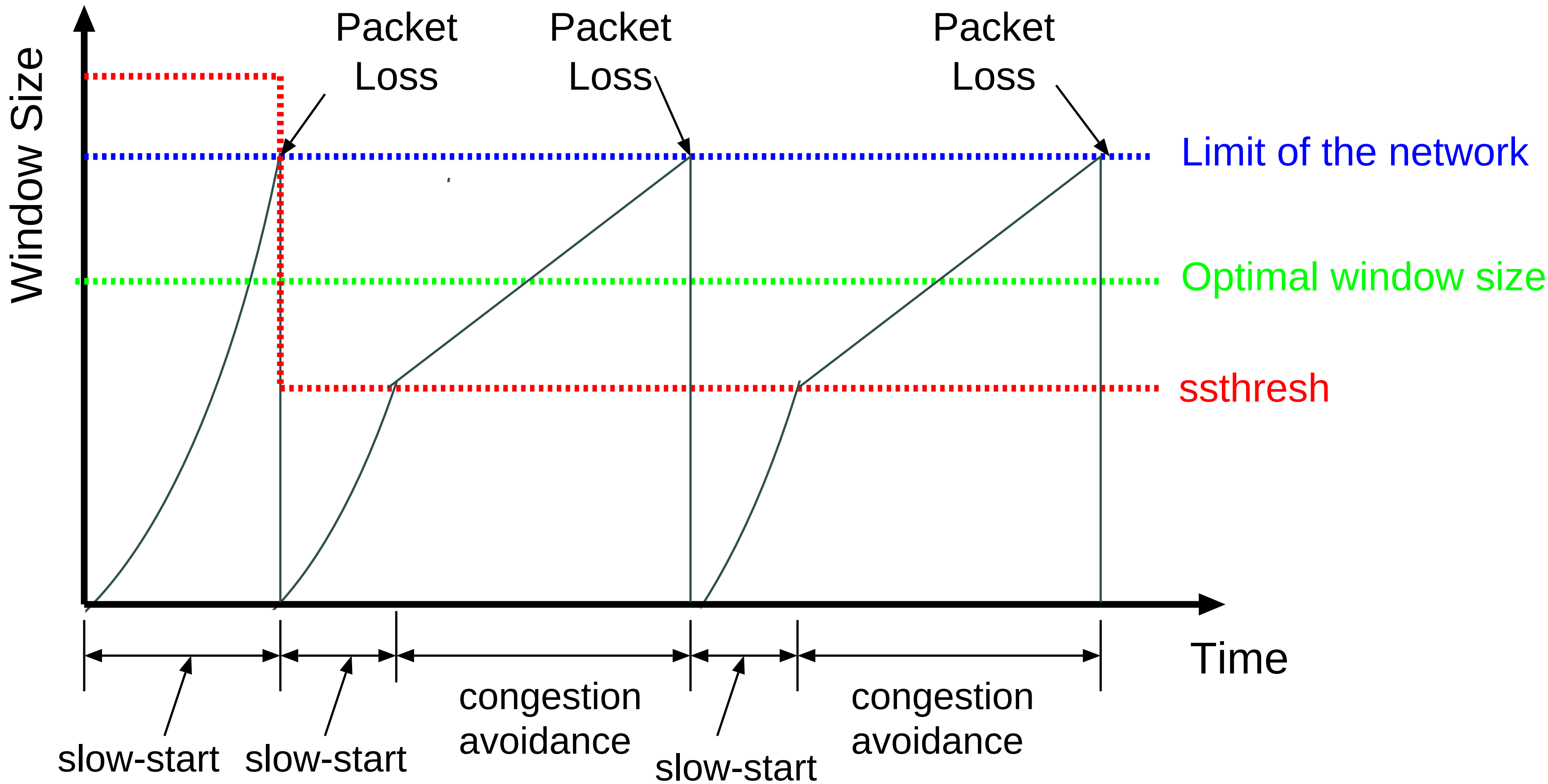
- Two communication phases for increasing congestion window
- Slow Start
  - Used at the beginning of a transfer, or after timeout.
  - Starts from minimum window size
  - Increases congestion window size by MSS bytes for each ACK received
  - Increases window size exponentially
- Congestion Avoidance
  - Increases congestion window size by MSS / cwnd bytes for each ACK received.
  - Increases window size linearly

Slow-Start and Congestion Avoidance (2)

- Transition from Slow-start to Congestion Avoidance
- TCP keeps a variable "ssthresh" to determine which algorithms are used.
  - If cwnd < ssthresh then do slow-start
  - If cwnd > ssthresh then do congestion avoidance
- Algorithms for "ssthresh"
  - Initial value: arbitrarily high value (ex. advertised window size)
  - When TCP detects packet loss, it will be set to cwnd/2.

Slow-Start and Congestion Avoidance (3)

- cwnd variation of Tahoe TCP

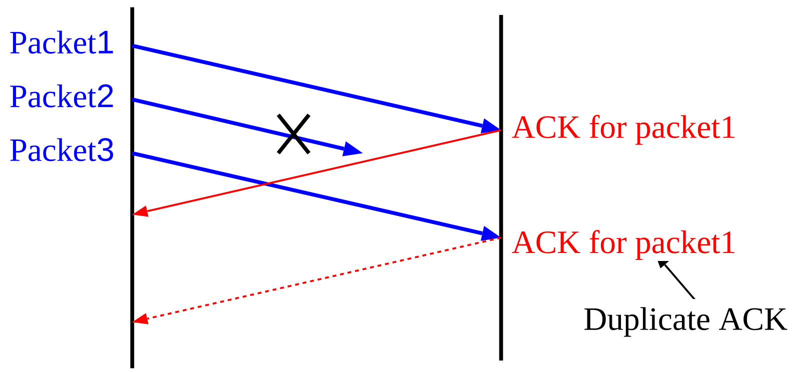


Slow-Start and Congestion Avoidance (4)

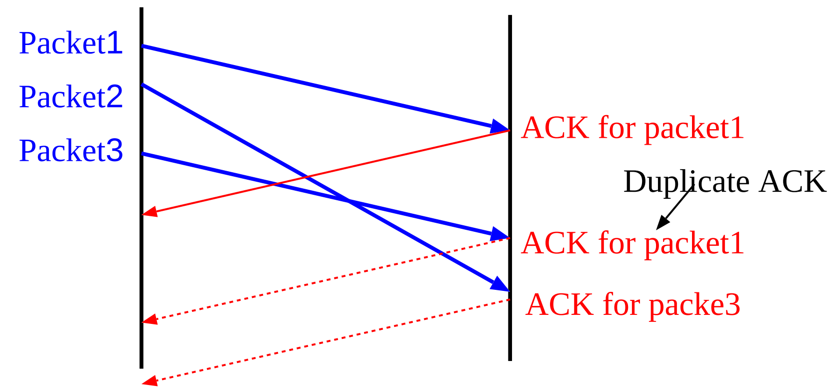
- Goal of slow-start and congestion avoidance
  - Keep window size around optimal size as much as possible.
  - Slow-Start
    - Increase window size rapidly to reach maximum safety transfer rate as fast as possible.
    - Maximum safety transfer rate:
      - Half of the transfer rate that caused packet loss
  - Congestion Avoidance
    - Increase window size slowly to avoid packet losses as long as possible

Fast Retransmit (1)

- Retransmit packets without waiting for retransmission timeout
- Fast retransmit uses "duplicate ACK" to trigger retransmission packets.
  - Duplicate ACK:
    - ACKs that are the same as previous ACK
    - Duplicate ACKs are generated by packet loss or packet disorder.



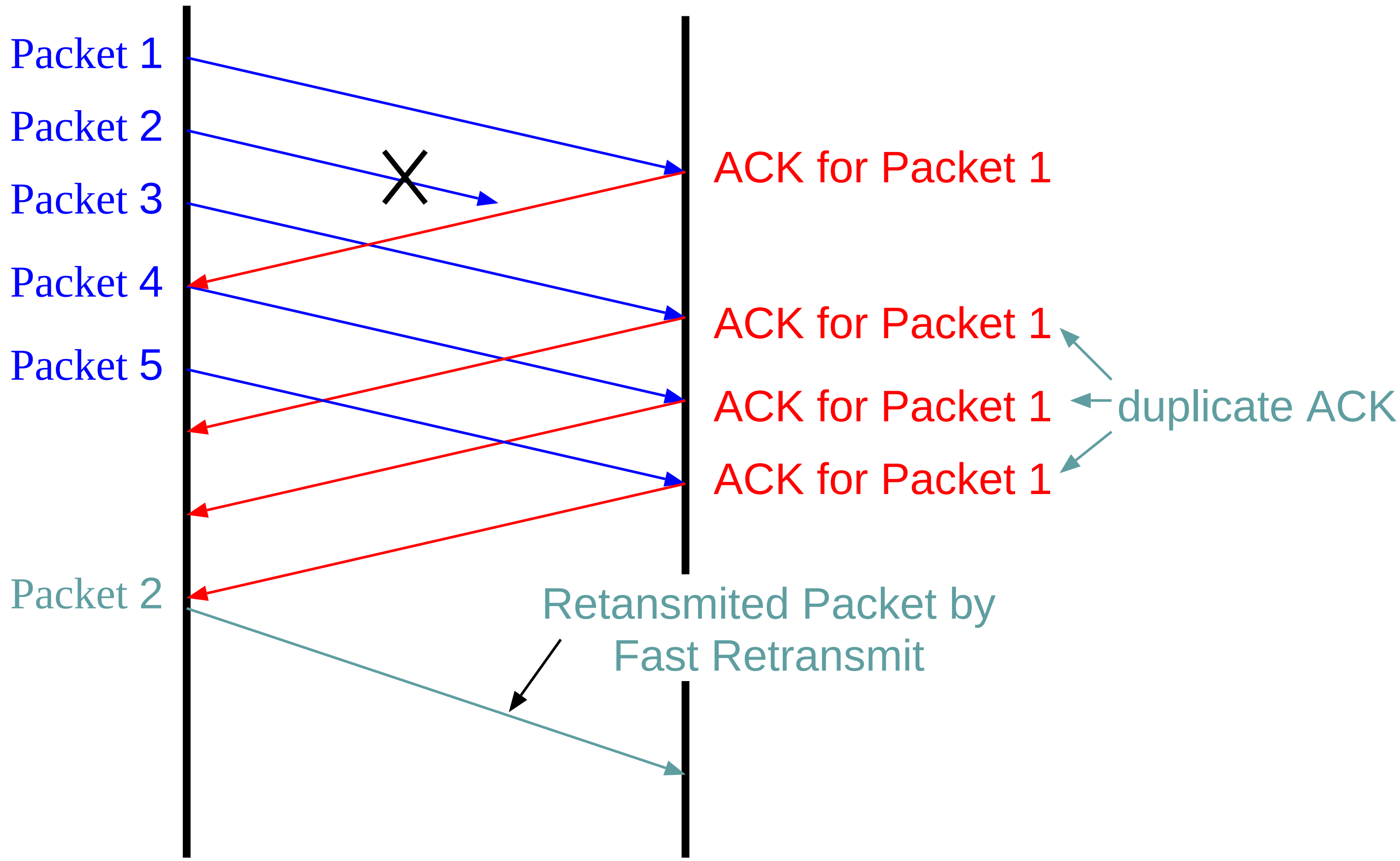
duplicate ACK generated by packet loss



duplicate ACK generated by packet disorder

Fast Retransmit (2)

- TCP cannot determine whether duplicate ACK is generated by packet loss or packet disorder.
- But TCP assumes that 3 successive duplicate ACKs are caused by packet loss.



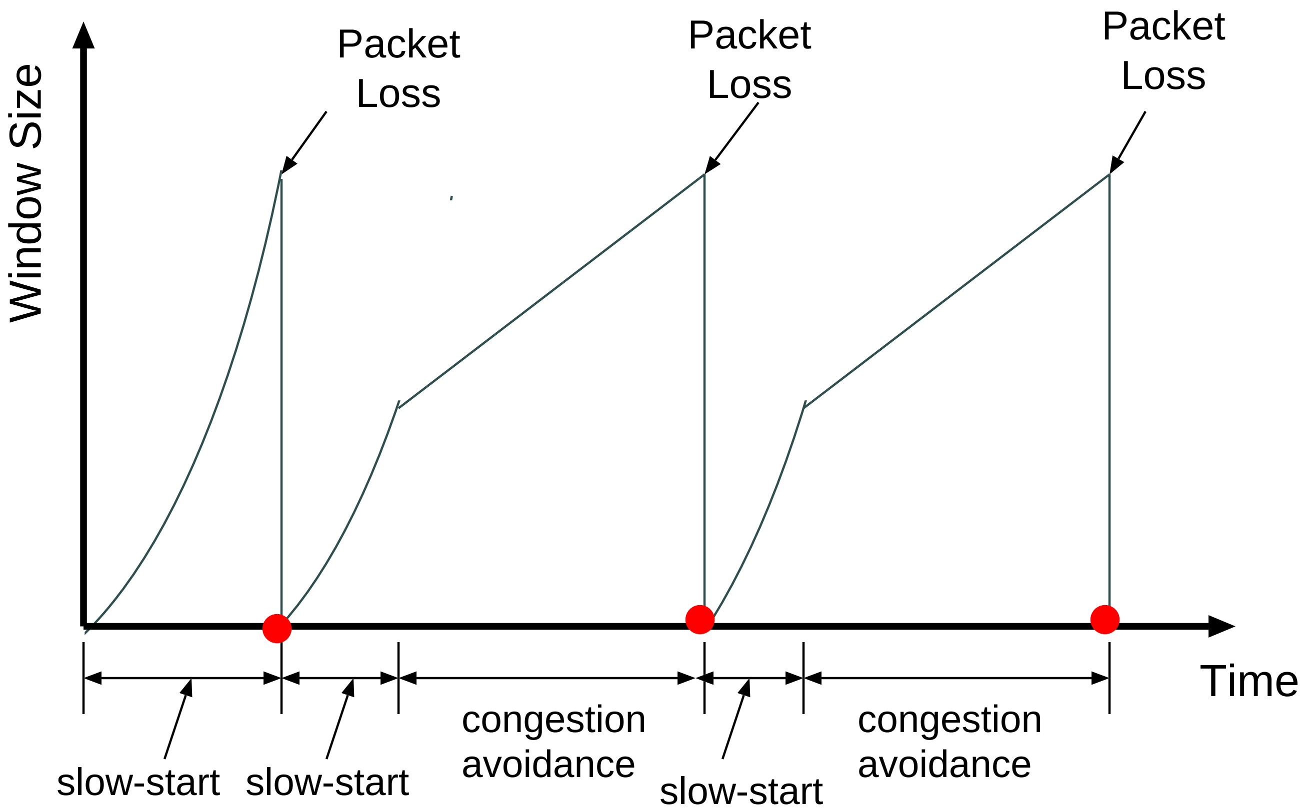
Reno TCP

- Performance improvement for Tahoe TCP.
  - Tahoe TCP is very sensitive to packet loss.
  - 1% packet loss rate may cause 50-75% decrease in throughput

- Introduced the "Fast Recovery" algorithm.
  - Recovers transfer rate quickly after packet loss

Fast Recovery (1)

- Problem of Tahoe TCP
  - Window Size is set to minimum value after packet loss.



- Congestion estimation by Tahoe TCP
  - Every packet loss is assumed to be serious congestion.

Fast Recovery (2)

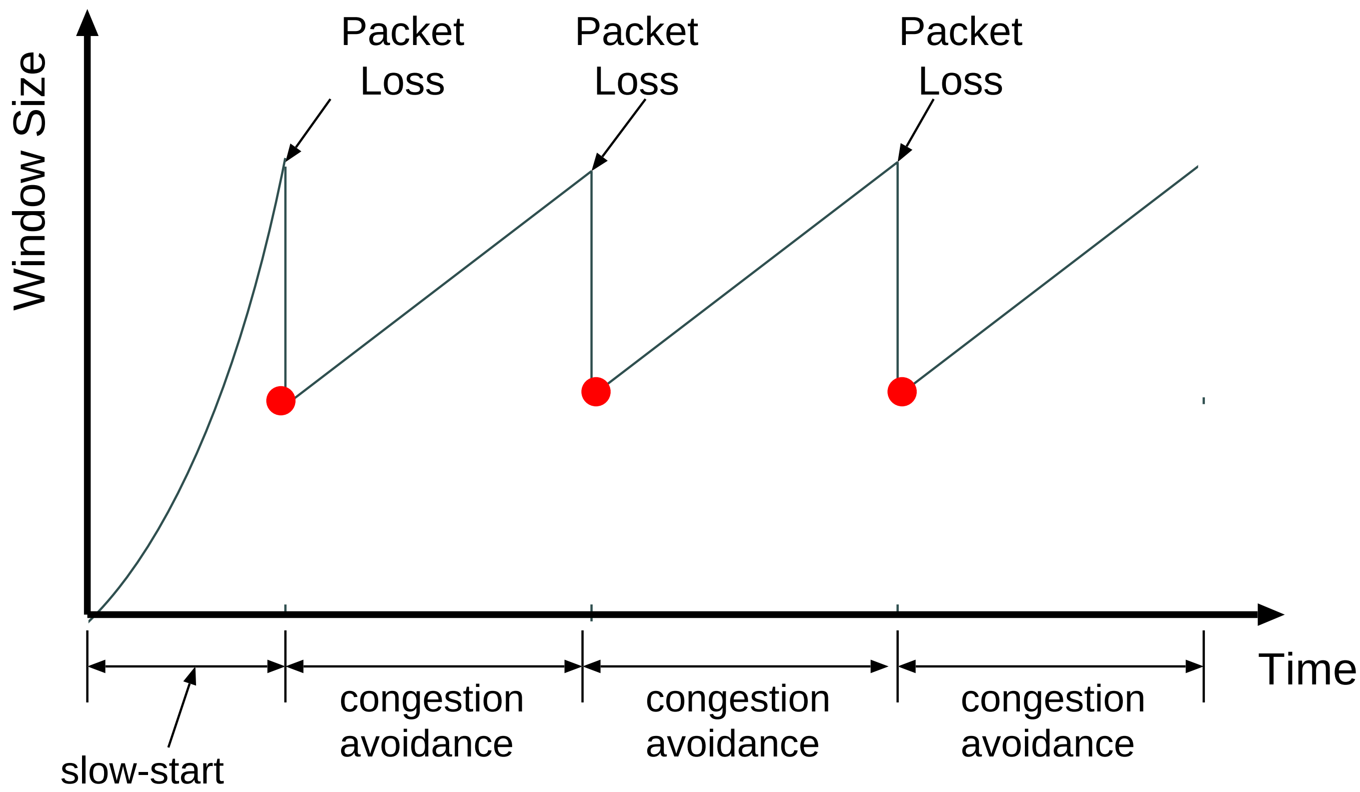
- Congestion estimation by Reno TCP
  - If packet loss was found by Retransmit Timeout,
    - Congestion is serious.
      - Window Size should be set to minimum value and do Slow-start.
  - If packet loss was found by Duplicate ACK,
    - Congestion is not serious.
    - Because..
      - At least 3 packets could arrive at the receiver after packet loss.
      - At least 3 packets have left the network, so there may be a chance to transmit a packet

.

    - So, Window Size is set to half of the current cwnd value and transits to Congestion Avoidance phase.

Fast Recovery (3)

- Example of cwnd variation of Reno TCP



  - After packet loss, TCP halves congestion window and enters Congestion Avoidance phase.

Problem of Reno TCP

- If two or more segments are lost in the current window, Fast Recovery algorithm cannot retransmit all lost packets.
  - TCP has to wait for retransmit timeout.
- Selective ACK option can solve this problem, but it has not been widely implemented yet.
  - Selective ACK requires a modification to both data sender and receiver.

NewReno TCP

- Performance improvement for Reno TCP.
  - Improves performance against multiple packet loss in the window.
  - Does not need Selective ACK.
  - Requires modification to only data sender.

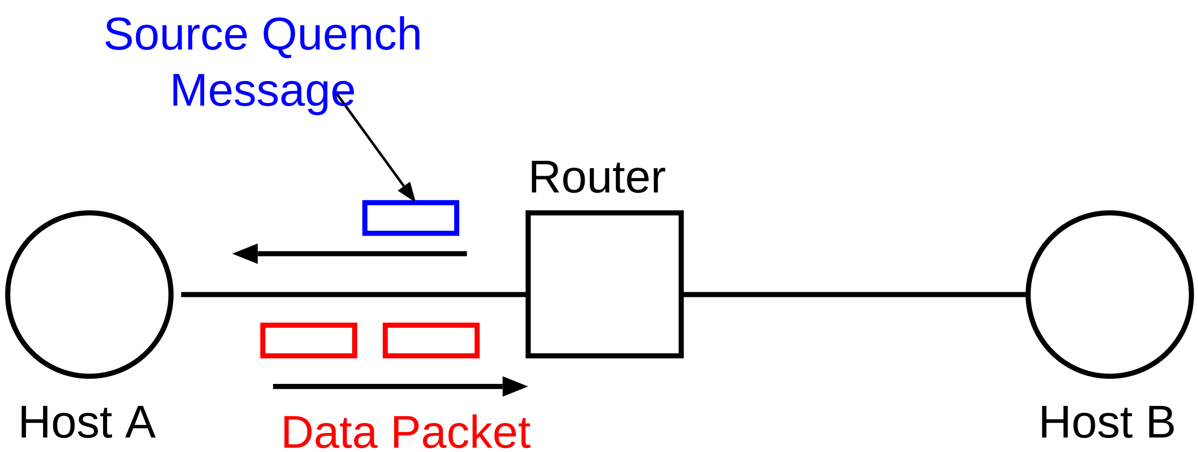
- NewReno is a bit more aggressive scheme than Reno.
  - Reno retransmit packets in response to either retransmit timeout or 3 duplicate ACKs.

Congestion Control with routers

- Advantage for using routers
  - End nodes can only determine congestion by sensing packet losses.
  - Router knows more about congestion than end nodes
    - If queue length in the router exceeds a certain threshold, we can assume network is becoming congested.
    - But, how do the routers tell the end nodes?
- ICMP source quench
- Explicit Congestion Notification (ECN)

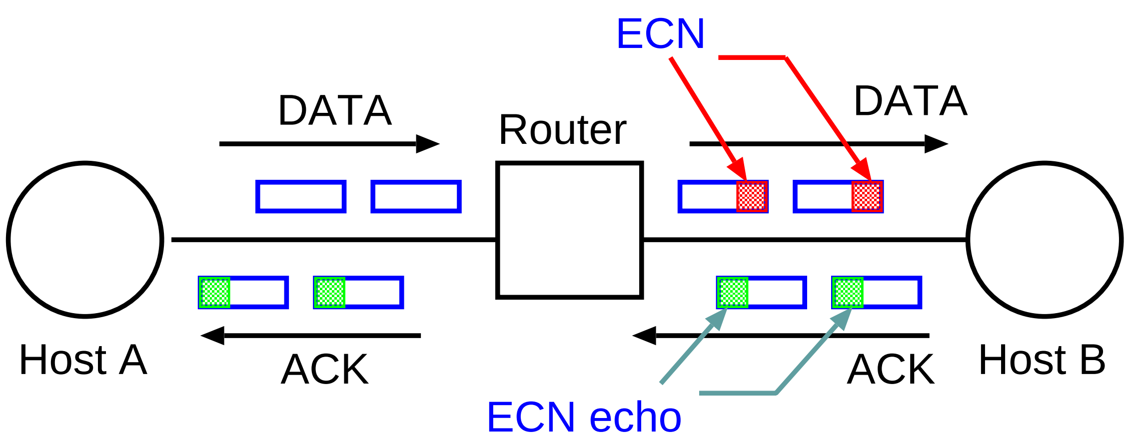
ICMP Source Quench

- If router finds that network is congested, router sends back "ICMP Source Quench" message to the data sender.
  - Data sender should set window size to minimum after receiving Source Quench.
  - Cons.
    - More traffic is generated in times of congestion.
  - Pros.
    - Can tell occurrence of congestion quickly.



Explicit Congestion Notification (ECN)

- If router finds that network is congested, router marks "ECN bit" in the IP header.
  - Receiver sends back ACK with "ECN echo" after receiving ECN packets
  - Sender should reduce Window Size after receiving ECN echo.
    - Cons.
      - ECN is a bit slower than Source Quench.
    - Pros.
      - Can find congestion before packet loss occurs
      - Does not add any traffic in the networks



Part 3: Simulating TCP

- Why simulation is necessary?
  - Analyze theoretical aspects
  - Can perform experiments easily rather than configuring real networks.
  - Easy to implement new functions
    - Does not require the knowledge of kernel coding

Network Simulator (1)

- ns: Network Simulator
  - http://www.isi.edu/nsnam/ns/
  - Can be used on major OSs (Linux, FreeBSD, NetBSD, Windows...)
  - Supports lots of networking technologies
    - Application-level protocols
      - HTTP, telnet, FTP
    - Transport protocols
      - UDP, TCP, RTP, SRM
      - Supports various TCP versions: Tahoe, Reno, NewReno..
    - Router Mechanisms
      - Various queuing mechanism: CBQ, RED, ECN
    - Link-layer mechanisms
      - CSMA/CD
  - High extensibility
    - Lots of protocol functions are provided as C++ object class

Network Simulator (2)

- nam: Network Animator
  - http://www.isi.edu/nsnam/nam/
  - Can visualize output of ns simulator

Summary

- TCP provides a reliable service between end-nodes.
  - Packet Retransmission based on Acknowledgment
- TCP plays an important role in congestion control in the Internet.
  - Autonomous Control by end-node
    - Simple estimation for network condition
- Congestion Control is one of the important topics for the future of the Internet.
  - TCP is NOT the perfect solution, but provides some essential hints.