

EEC 687 Mobile Computing (Spring 2007)

TCP in Fixed Networks

Chansu Yu

Cleveland State University

Contents

- Physical layer issues
 - Communication frequency
 - Signal propagation
 - Modulation and Demodulation
- Channel access issues
 - Multiple access / Random access / Asynchronous
 - 802.11 / Bluetooth
 - Capacity / Energy / Fairness / Directional
- System issues
 - Embedded processor
 - Low power design
- Network issues
 - Location management
 - Mobile IP / Cellular IP
 - MANET routing / Clustering
 - Multicast
 - Interoperability
 - Network reliability (TCP)
 - Quality of service (QoS)

Internet Protocol (IP)

- Packets may be delivered out-of-order
- Packets may be lost
- Packets may be duplicated

3

chansuyu@gmail.com

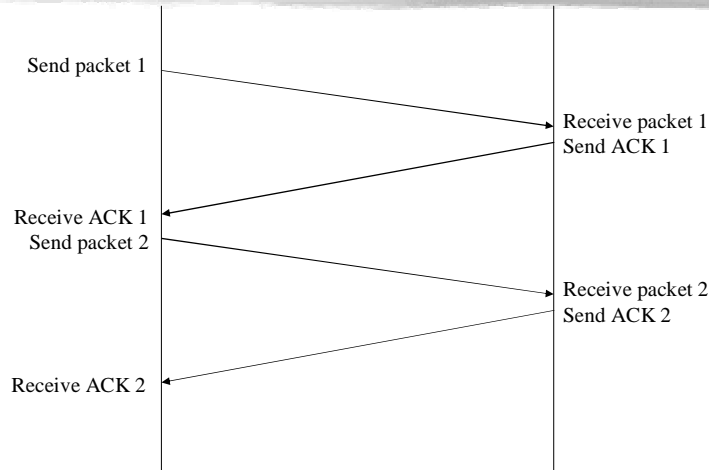
Transmission Control Protocol (TCP)

- Reliable ordered delivery
- Implements congestion avoidance and control
- Reliability achieved by “positive acknowledgement with retransmissions”
- End-to-end semantics
 - Acknowledgements sent to TCP sender confirm delivery of data received by TCP receiver
 - Ack for data sent only **after** data has reached receiver

4

chansuyu@gmail.com

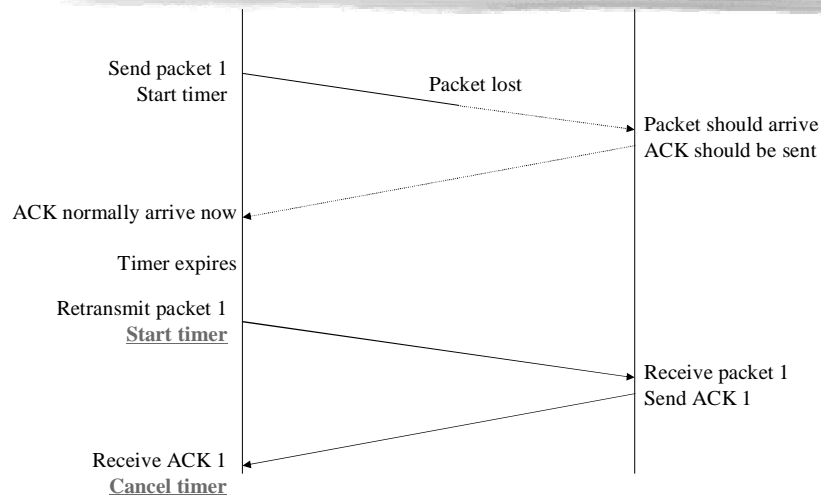
Reliable, Ordered Delivery: Positive Acknowledgement



5

chansuyu@gmail.com

Reliable, Ordered Delivery: Timeout & Retransmission



6

chansuyu@gmail.com

Issues in TCP

1. Fast packet loss detection
 - Retransmission timeout (RTO)
 - Duplicated acknowledgements (Dupacks)
2. Efficient delivery
 - Sliding window with cumulative acknowledgement
3. Flow problems
 - End-to-end flow problem
 - Flow problem in intermediate systems (congestion)

7

chansuyu@gmail.com

1. Reliable, Ordered Delivery: Fast Packet Loss Detection

- Two methods
 - A. Retransmission timeout (RTO)
 - B. Duplicate acknowledgements (Dupacks)

8

chansuyu@gmail.com

A. Fast Packet Loss Detection Using Retransmission Timeout (RTO)

- ❑ At any time, TCP sender sets retransmission timer for only one packet
- ❑ If acknowledgement for the timed packet is not received before timer goes off, the packet is assumed to be lost
- ❑ RTO (timeout value) dynamically calculated

9

chansuyu@gmail.com

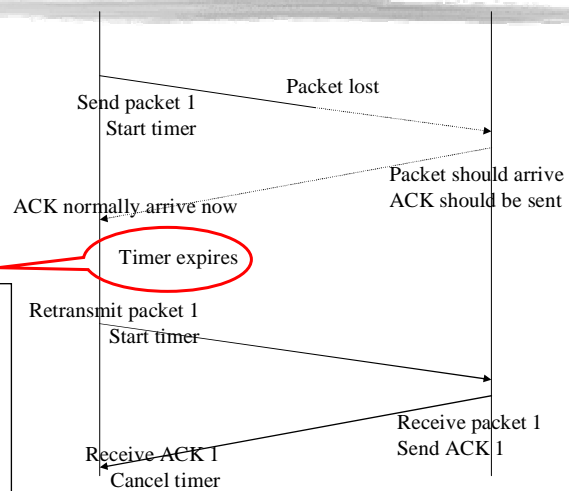
Retransmission Timeout (RTO) Calculation

Q1: Which value will you use ?

Q2: How to set the value in dynamic Internet ?

Q3: If a retransmission occurs, is an ack for the original or for the retransmission (important because RTT is calculated) ?

A1: RTT (round trip time)
 A2: adaptive retransmission algorithm (changing RTT and thus, changing time-out values)
 A3: if ACK arrives, is it for the original packet or for the retransmitted packet (acknowledge ambiguity) ?
 A4: timer backoff strategy



10

chansuyu@gmail.com

Retransmission Timeout (RTO) Calculation

- ❑ Large variations in the RTT necessitates a larger RTO (why???)
- ❑ RTO = mean + 4 mean deviation
 - Standard deviation σ : $\sigma^2 = \text{average of } (\text{sample} - \text{mean})^2$
 - Mean deviation $\delta = \text{average of } |\text{sample} - \text{mean}|$
 - Mean deviation easier to calculate than standard deviation
 - Mean deviation is more conservative: $\delta \geq \sigma$ (right???)
- ❑ Karn's algorithm – no ack ambiguity

$$\begin{aligned} \text{left} &= (a+b)/2 \\ \text{right} &= \text{sqrt}(a^2+b^2)/2 \\ 4*\text{left}^2 &= a^2+b^2+2ab \\ 4*\text{right}^2 &= a^2+b^2 \end{aligned}$$

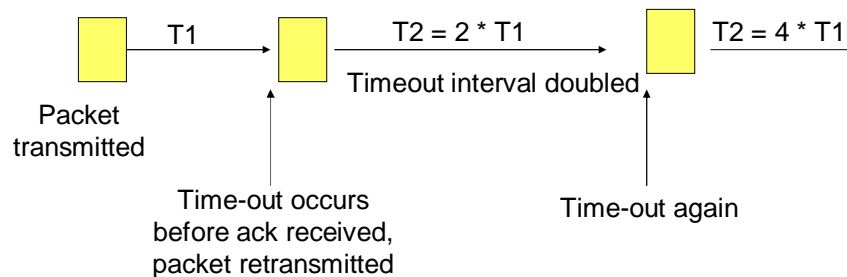
thus, left >= right

11

chansuyu@gmail.com

Retransmission: Exponential Backoff

- ❑ Double RTO on each timeout (why???)



12

chansuyu@gmail.com

B. Fast Packet Loss Detection Using Dupacks

- ❑ Timeouts can take too long
 - how to detect packet loss sooner and initiate retransmission sooner?

- ❑ If the receiver receives out-of-order packet
 - What does the receiver do ??? ← do nothing
 - “**Duplicated acks (dupacks)**”
 - What are the benefits - Fast Retransmission

13

chansuyu@gmail.com

Dupacks

- ❑ Dupacks may be generated due to
 - packet loss, or
 - out-of-order packet delivery

- ❑ TCP sender assumes that a packet loss has occurred if it receives **three dupacks** consecutively (why not one/two???)
 - | | | | | | |
|----|----|----|---|---|---|
| 12 | 11 | 10 | 9 | 8 | 7 |
|----|----|----|---|---|---|
 - Packet Loss Detection: Receipt of packets 9, 10 and 11 will each generate a *dupack* from the receiver.
 - Fast Retransmission: The sender, on getting these dupacks, will *retransmit* packet 8.

14

chansuyu@gmail.com

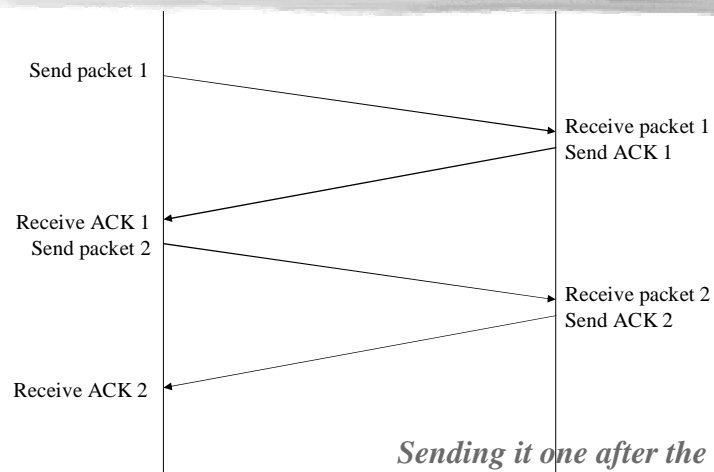
Issues in TCP

1. Fast packet loss detection
 - Retransmission timeout (RTO)
 - Duplicated acknowledgements (Dupacks)
2. Efficient delivery
 - Sliding window with cumulative acknowledgement
3. Flow problems
 - End-to-end flow problem
 - Flow problem in intermediate systems (congestion)

15

chansuyu@gmail.com

2. Efficient Delivery



16

chansuyu@gmail.com

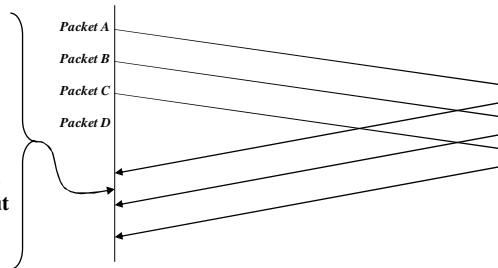
Efficient Delivery: Sliding Window

- ❑ A simple positive acknowledgement protocol wastes a substantial amount of network bandwidth
- ❑ Sliding windows protocol allows the sender to transmit multiple packets before waiting for an acknowledgement

Some packets are sent and ack'ed

Some packets are sent but not ack'ed yet

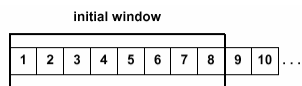
Some packets are not sent yet but can be sent without delay because they are within the window



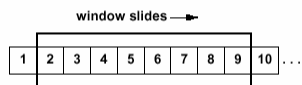
17

chansuyu@gmail.com

Sliding Window Protocol



(a)



(b)

What if the sender receives ACK for packet 3?

What does it mean if the window size is 1?

What does it mean if the window size is ∞ ?

Figure 13.3 (a) A sliding window protocol with eight packets in the window, and (b) The window sliding so that packet 9 can be sent when an acknowledgement has been received for packet 1. Only unacknowledged packets are retransmitted.

18

chansuyu@gmail.com

Sliding Window Basics

❑ Cumulative acknowledgements

- Good: Easy to generate and unambiguous
- Bad: the sender does not receive information about all successful transmissions, but only about a single position in the stream that has been received

❑ An acknowledgement ack's all contiguously received data

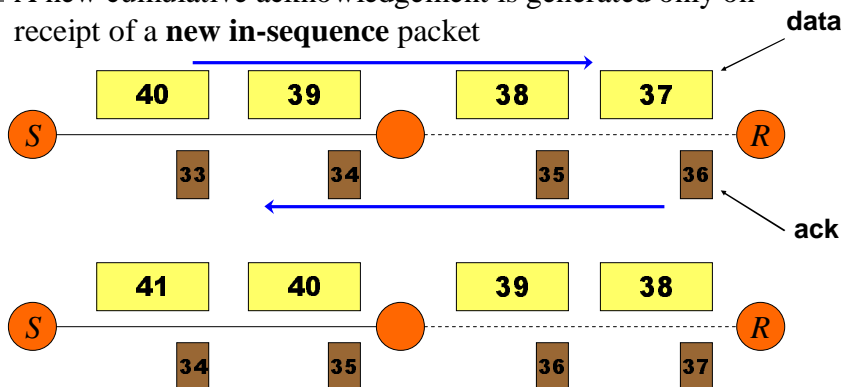
❑ TCP assigns byte sequence numbers

19

chansuyu@gmail.com

Cumulative Acknowledgements

- ❑ A new cumulative acknowledgement is generated only on receipt of a **new in-sequence** packet



20

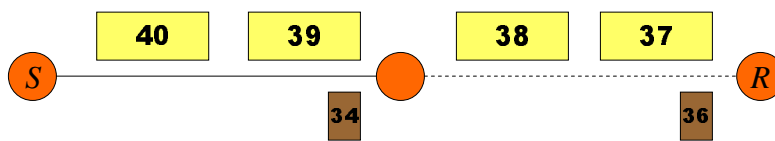
chansuyu@gmail.com

One More Optimization

□ If the sender receives ACK_{36}

- It means ???
- Do we need to send acks for all packets?
- “Delayed acks” (e.g., ACK_2 , ACK_4 , ACK_6 , ...)

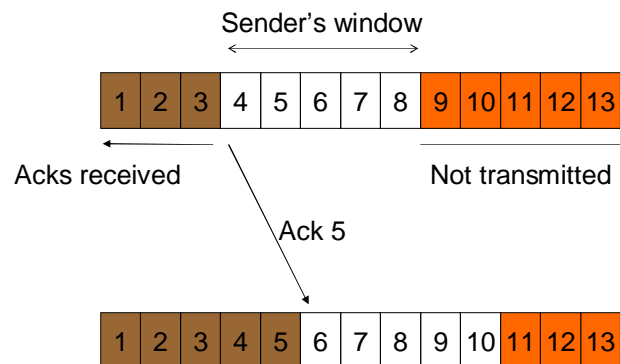
*It means packets 1 to 36 have been received
->Do we need to send acks for all packets?*



21

chansuyu@gmail.com

Delayed ACKs



22

chansuyu@gmail.com

Issues in TCP

1. Fast packet loss detection

- Retransmission timeout (RTO)
- Duplicated acknowledgements (Dupacks)

2. Efficient delivery

- Sliding window with cumulative acknowledgement

3. Flow problems

- End-to-end flow problem
- Flow problem in intermediate systems (congestion)

Window size ($1 \sim \infty$)
if 1, it takes long but perfect control
(if a packet is not successful,
do not add more traffic)
if ∞ , no control over traffic

23

chansuyu@gmail.com

3. Flow Problems

A. End-to-end flow problem

- Control a source that sends more traffic than the receiver can tolerate
 - Between source and destination
- => Easier to solve: The receiver informs its status to the sender
("advertised window size")

B. Flow problem in the intermediate systems

- Control a source that sends more traffic than the intermediate systems can tolerate
- Called "congestion"

24

chansuyu@gmail.com

Window-based Flow Control

- ❑ Congestion simply means increased delay
 - Respond by retransmissions
 - Incurs more congestion
 - Thus, we need to reduce the window size
- ❑ Window size is determined as the minimum of
 - receiver's advertised window - determined by available buffer space at the receiver \leq "flow control"
 - congestion window (cwnd) - determined by the sender, based on feedback from the network \leq "congestion control"
- ❑ On detecting a packet loss, TCP sender
 - assumes that network congestion has occurred
 - drastically reduces the congestion window

25

chansuyu@gmail.com

Determining Window Size for Congestion Control

- ❑ Window size bounds the amount of data that can be sent per round-trip time (RTT)
 - For example, what is the maximum throughput (bandwidth)???
 - cwnd=1 packet (1000 bits), RTT=1 sec \Rightarrow 1 packet/sec = 1000 bps
 - cwnd=1 packet (1000 bits), RTT=0.1 sec \Rightarrow 10 packets/sec = 10000 bps
 - cwnd=10 packets (10000 bits), RTT=1 sec \Rightarrow 10 packets/sec = 10000 bps
 - "Bandwidth \leq cwnd/RTT"

26

chansuyu@gmail.com

Determining Window Size for Congestion Control

- ❑ $cwnd=10$ packets (10000 bits), $RTT=1$ sec \Rightarrow 10 packet/sec = 10000 bps
 - If the network supports more than 10000 bps, the network is not fully utilized
 - W should be increased

 - If the network supports up to 1000 bps, the network is fully overloaded
 - W should be reduced

27

chansuyu@gmail.com

Determining Window Size for Congestion Control

- ❑ Ideal window size = delay * bandwidth (delay-bw product)
 - What if window size < delay*bw ?
 - Inefficiency (wasted bandwidth)

 - What if window size > delay*bw ?

 - Queuing at intermediate routers
 - increased RTT due to queuing delays

 - Potentially, packet loss

28

chansuyu@gmail.com

Determining Window Size for Congestion Control

❑ Congestion control used in TCP (cwnd changes)

➤ Multiplicative decrease

- congestion window size

➤ Slow start

- When congestion ends, it can increase the window size multiplicatively, which may result in oscillation
- “Additive” recovery (slow start)

29

chansuyu@gmail.com

Determining Window Size for Congestion Control

❑ Slow Start

- For each ACK arrived, increase the congestion window size by one
- but it actually is not “slow”
- cwnd grows “**exponentially**” with time during slow start

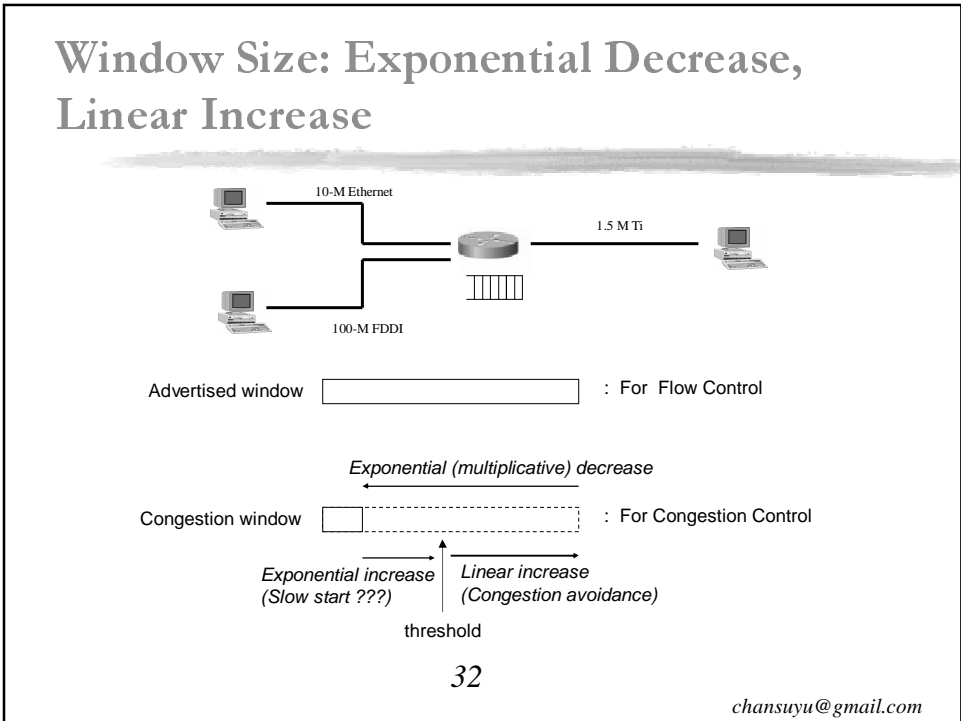
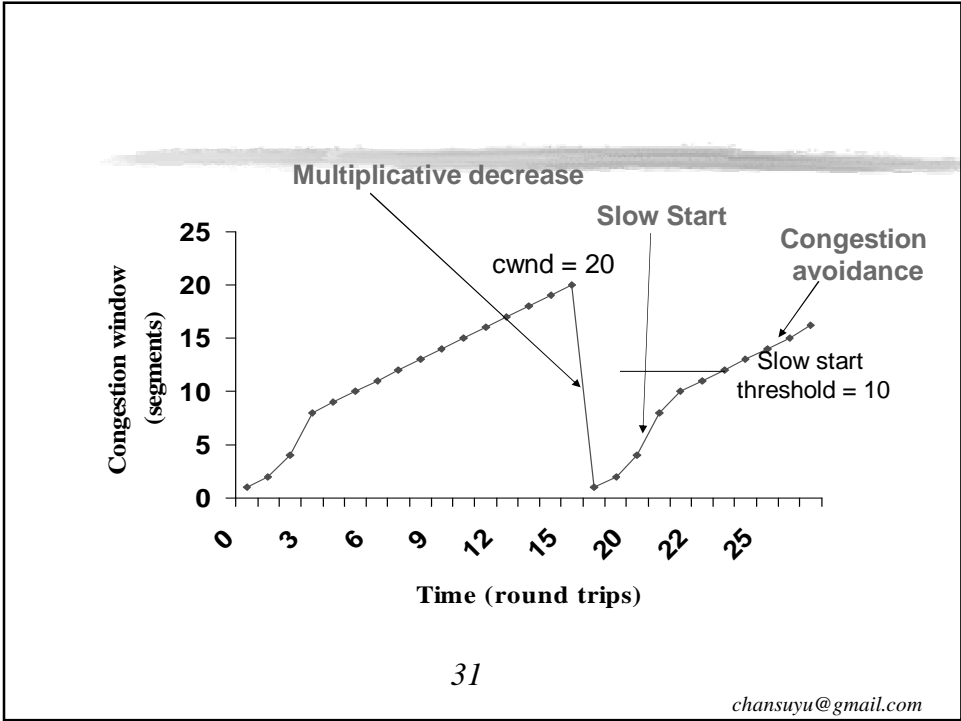
❑ When cwnd reaches slow-start threshold (half the window size before packet loss), congestion avoidance is performed

❑ Congestion avoidance

- cwnd increases **linearly** with time (increase one if all ACKs arrive)
- Rate of increase could be lower if sender does not always have data to send

30

chansuyu@gmail.com



Format of a TCP Segment

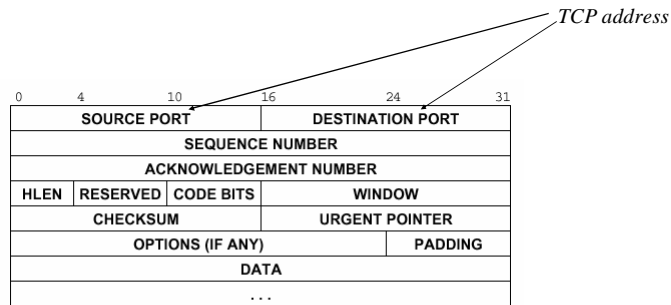


Figure 13.7 The format of a TCP segment with a TCP header followed by data. Segments are used to establish connections as well as to carry data and acknowledgements.

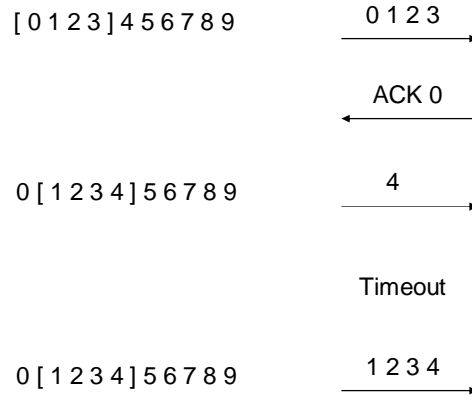
Decimal	Keyword	UNIX Keyword	Description
0			Reserved
1	TCPMUX	-	TCP Multiplexor
7	ECHO	echo	Echo
9	DISCARD	discard	Discard
11	USERS	sysat	Active Users
13	DAYTIME	daytime	Daytime
15	-	netstat	Network status program
17	QUOTE	qotd	Quote of the Day
19	CHARGEN	chargen	Character Generator
20	FTP-DATA	ftp-data	File Transfer Protocol (data)
21	FTP	ftp	File Transfer Protocol
22	SSH	ssh	Secure Shell
23	TELNET	telnet	Terminal Connection
25	SMTP	smtp	Simple Mail Transport Protocol
37	TIME	time	Time
43	NICNAME	whois	Who Is
53	DOMAIN	nameserver	Domain Name Server
67	BOOTPS	bootps	BOOTP or DHCP Server
77	-	rje	any private RJE service
79	FINGER	finger	Finger
80	WWW	www	World Wide Web Server
88	KERBEROS	kerberos	Kerberos Security Service
95	SUPDUP	supdup	SUPDUP Protocol
101	HOSTNAME	hostnames	NIC Host Name Server
102	ISO-TSAP	iso-tsap	ISO-TSAP
103	X400	x400	X.400 Mail Service
104	X400-SND	x400-snd	X.400 Mail Sending
110	POP3	pop3	Post Office Protocol Vers. 3
111	SUNRPC	sunrpc	SUN Remote Procedure Call
113	AUTH	auth	Authentication Service
117	UUCP-PATH	uucp-path	UUCP Path Service
119	NNTP	nntp	USENET News Transfer Protocol
123	NTP	ntp	Network Time Protocol
139	NETBIOS-SSN	-	NETBIOS Session Service
161	SNMP	snmp	Simple Network Management Protocol

Figure 13.16 Examples of currently assigned TCP port numbers. To the extent possible, protocols like UDP use the same numbers.

Review of TCP -- Sliding Window: the maximum range of data sent but not acknowledged



An Example: Sliding Window Size = 4 bytes



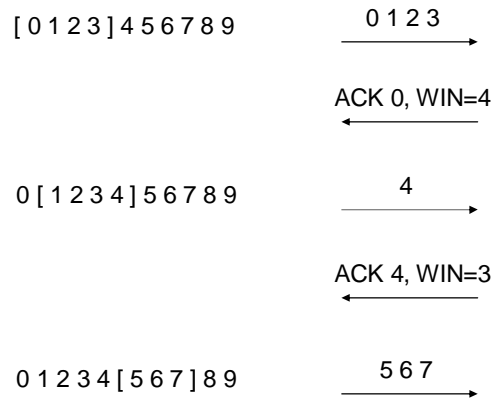
35

chansuyu@gmail.com

Review of TCP – Flow Control: The receiver advertises its available buffer size to the sender in each TCP acknowledgment



An Example: Initial Sliding Window Size = 4 bytes

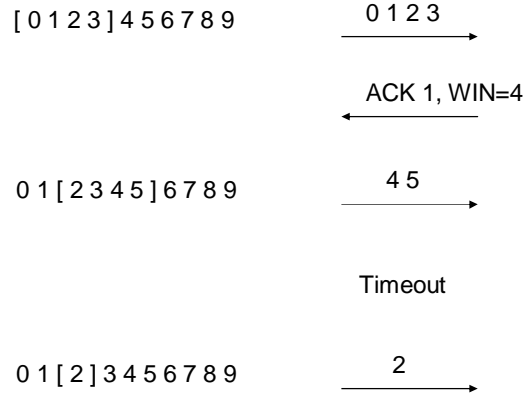


36

chansuyu@gmail.com

Review of TCP – Congestion Control: Using timeout as the indication of network congestion.

An Example: Sliding Window Size = 4 bytes

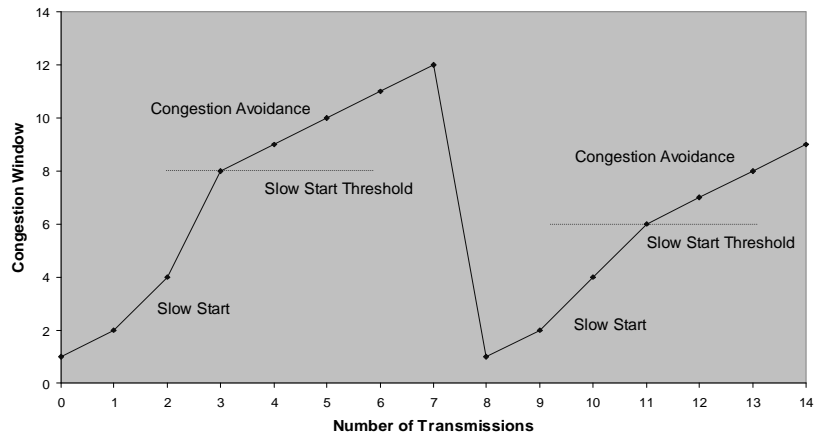


37

chansuyu@gmail.com

Review of TCP—Congestion Control: Additive Increase and Multiplicative Decrease

TCP Tahoe, Reno



38

chansuyu@gmail.com

TCP Summary

- ❑ Error Detection – using TCP Checksum field
- ❑ Error Correction - Acknowledgements & Retransmissions
 - Roundtrip time estimation is critical
- ❑ Sliding Windows – can send multiple segments w/o starting to receive some acknowledgements
- ❑ Flow Control by adjusting the window
- ❑ Congestion Control
 - *Key Assumption: Packet loss is due to congestion*
 - ➔ *Reduces the number of segments sent when detects losses*

39

chansuyu@gmail.com

TCP Summary

- ❑ Slow Start:
 - Starts with a slow transmission rate at the start of a new connection or after a congestion condition
 - Exponentially increases congestion window up to threshold
- ❑ Fast Retransmit:
 - TCP acknowledgements:
 - Number of segments correctly and *sequentially* received
 - *Only sent upon receipt of a segment*
 - Multiple acknowledgements for the same number of segments indicate a gap rather than congestion → no need to wait: immediately retransmit lost segment

40

chansuyu@gmail.com