CS 168 Introduction to the Internet: Architecture and Protocols
Fall 2024 Sylvia, Rob, and Peyrin

Discussion 9: Congestion Control

1 True False

1. UDP uses congestion control. **Solution:** False, TCP uses congestion control
2. Flow control slows down the sender when the network is congested. **Solution:** False, flow control ensures the sender doesn't overflow the receiver's buffer
3. For TCP timer implementations, every time the sender receives an ACK for a previously unACKed packet, it will recalculate ETO. **Solution:** False, only clean samples are used. For example, ACKs on a packet that have been retransmitted are not used since the sender cannot be sure which version the ACK is from.
4. CWND (congestion window) is usually smaller than RWND (receiver window). **Solution:** True
5. AIMD is the only ”fair” option among MIMD, AIAD, MIAD, and AIMD. **Solution:** True

2 Impact of Fast Recovery

Consider a TCP connection, which is currently in Congestion Avoidance (AIMD).

- The last ACK sequence number was 101.
- The CWND size is 10 (in packets).
- The packets #101-110 were sent at t=0,0.1,...,0.9 (sec), respectively.
- The packet #102 is lost only for its first transmission.
- RTT is 1 second.

Fill in the tables below, until the sender transmits the packet #116.

1. Without fast recovery:
   - On new ACK, $CWND+=\frac{1}{\lfloor CWND \rfloor}$
   - On triple dupACKs, $SSTHRESH = \left\lfloor \frac{CWND}{2} \right\rfloor$, then $CWND = SSTHRESH$.

Time (sec)	Receive ACK (due to)	CWND	Transmit Seq # (mark retransmits)
1.0	102 (101)	$10+\frac{1}{10}=10.1$	111
1.2	102 (103)	10.1	/
1.3	102 (104)	10.1	/
1.4	102 (105)	$\left\lfloor\frac{10.1}{2}\right\rfloor=5$	102 (Rx)
1.5	102 (106)	5	/
1.6	102 (107)	5	/
1.7	102 (108)	5	/
1.8	102 (109)	5	/
1.9	102 (110)	5	/
2.0	102 (111)	5	/
2.4	112 (102)	$5+\frac{1}{5}=5.2$	112-116

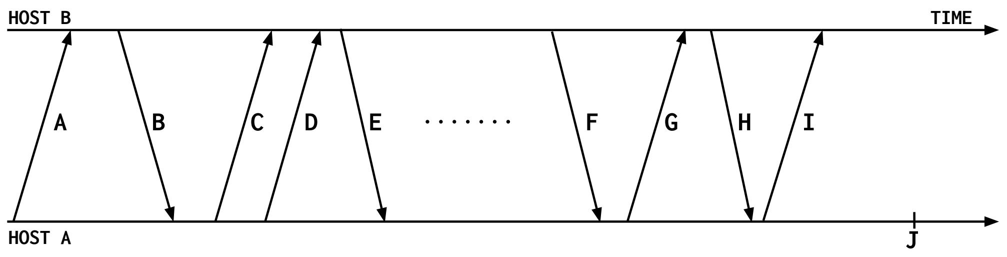
2. With fast recovery:

- On triple dupACKs, $SSTHRESH=\left\lfloor\frac{CWND}{2}\right\rfloor$, then $CWND=SSTHRESH+3$, enter fast recovery.
- In fast recovery, $CWND+=1$ on every dupACK.
- On new ACK, exit fast recovery, $CWND=SSTHRESH$

Time (sec)	Receive ACK (due to)	CWND	Transmit Seq # (mark retransmits)
1.0	102 (101)	$10+\frac{1}{10}=10.1$	111
1.2	102 (103)	10.1	/
1.3	102 (104)	10.1	/
1.4	102 (105)	$\left\lfloor\frac{10.1}{2}\right\rfloor+3=8$	102 (Rx)
1.5	102 (106)	9	/
1.6	102 (107)	10	/
1.7	102 (108)	11	112
1.8	102 (109)	12	113
1.9	102 (110)	13	114
2.0	102 (111)	14	115
2.4	112 (102)	SSTHRESH=5	116

Note: three dupACKs = 1 regular ACK + 3 ACKs of a sequence number that have been ACKed before already so we retransmit on the 4th ACK of seq # 102

3 Flags



The above figure shows the life cycle of a TCP connection with normal termination - that is, connection establishment, data exchange, and teardown.

1. For each of the arrows, choose whether it is a SYN, ACK, data, FIN or RST packet. A single arrow might have more than one of these flags set.

A:	SYN	D:	data	G:	FIN
B:	SYN + ACK	E:	ACK	H:	FIN + ACK
C:	ACK	F:	ACK	I:	ACK

2. When host A sends packet I, it sets a timer that ends at point J. What is the purpose of this timeout?

   Solution: Host A sets a timer when it sends the last ACK. This ensures that if the ACK is dropped, B can resend its FIN+ACK, and host A's connection will still be open to receive this message. Host B won't close its connection until it gets host A's ACK (or it times out).