CSCI 690 Computer Networks
Homework 7
Solution

Saad Mneimneh

Problem 1: A simple AIMD
Consider a simple congestion control algorithm that uses linear increase and multiplicative decrease (no slow start). Assume the congestion window size is in units of packets rather than bytes, and it is one packet initially.

(a) Give a detailed description of this algorithm.

ANSWER: Initially, $cwnd = 1$

$$on\ ACK:\ \ cwnd = cwnd + 1/cwnd$$

$$on\ timeout:\ \ cwnd = \min(1, cwnd/2)$$

(b) Assume that for every group of packets sent, only one cumulative ACK is returned. Plot the congestion window size as a function of time (units of RTT) when the following packets are lost: 9, 25, 30, 38, and 50. For simplicity assume the timeout is equal to 1 RTT.

RTT	1	2	3	4
sent	1	2-3	4-6	7-10

Packet 9 is lost. *cwnd* is reduced to 2.

RTT	5	6	7	8	9
sent	9-10	11-13	14-17	18-22	23-28

Packet 25 is lost. *cwnd* is reduced to 3.

RTT	10	11
sent	25-27	28-31

Packet 30 is lost. *cwnd* is reduced to 2.

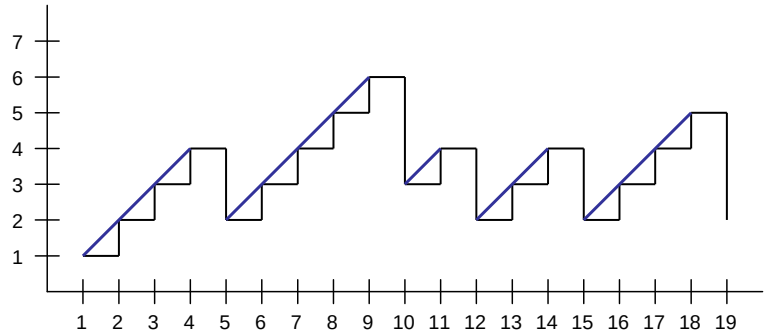
RTT	12	13	14
sent	30-31	32-34	35-38

Packet 38 is lost. *cwnd* is reduced to 2.

RTT	15	16	17	18
sent	38-39	40-42	43-46	47-50

Packet 50 is lost. *cwnd* is reduced to 2.

RTT	19
sent	50



(c) What is the effective throughput achieved for this connection if each packet is 1KB and the RTT is 100ms?

ANSWER: It took 19 RTTs to send 50 packets, i.e. 50KB. The effective throughput is $50 \times 1024 \times 8/1.9 \approx 215$ Kbps.

(d) Explain why the changing *cwnd* each time an ACK arrives using the following formula

$$cwnd = cwnd + MSS \times (MSS/cwnd)$$

is not correct for such algorithm (or in general).

ANSWER: The formula is accurate if each new ACK acknowledges one new MSS-sized segment. However, an ACK can acknowledge either small size packets (smaller than MSS) or cumulatively acknowledge many MSSs worth of data.

Problem 2: Fairness index
Suppose a congestion control scheme results in a collection of flows that achieve the following throughput rates: 100 KBps, 60 KBps, 110 KBps, 95 KBps, and 150 KBps.

(a) Calculate the fairness indes for this scheme.

ANSWER: $\frac{(100+60+110+95+150)^2}{5(100^2+60^2+110^2+95^2+150^2)} = 0.9$

(b) Add a flow with throughput rate of 1000 KBps to the above, and recalculate the fairness index.

ANSWER: $\frac{(100+60+110+95+150+1000)^2}{6(100^2+60^2+110^2+95^2+150^2+1000^2)} = 0.3$

Problem 3: MIMD (optional)
This problem is inspired by a question asked in class. Consider an congestion control algorithm that always uses slow start to increase the window size. Therefore:

$$on\ ACK:\ cwnd = cwnd + 1$$

$$on\ timeout:\ cwnd = cwnd/2$$

(a) Under what circumstances is the fast retransmit mechanism not effective?

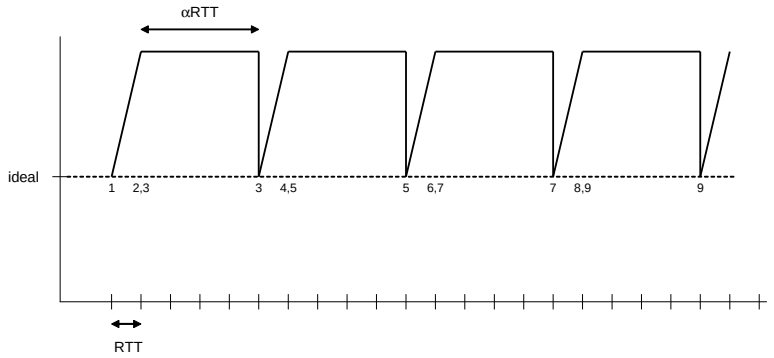
ANSWER: Fast retransmit uses 3 dupacks instead of a timeout to declare a lost packet. When the window size is very large (compared to the ideal size), many packets will be lost. Therefore, at some point, dupacks will stop being generated and we must wait for a timeout.

(b) Suppose that the timeout is equal to $\alpha \times RTT$ (in practice α could be as large as 20). Show by constructing an example that the effective throughput of this congestion control algorithm could be as low as (assuming a unit of bandwidth)

$$\frac{2}{\alpha + 1}$$

ANSWER: We will explore the idea mentioned in part (a). With MIMD, it is possible for the window size to double in one RTT. If the window size was ideal, this doubling will cause half of the packets to be lost. Therefore, there will be no dupacks and we have to wait for the timeout to detect the loss. For simplicity, we will assume that the doubling and halfing is intantaneous.
The numbers in the figure below indicate the batches of ideal window sizes that succeed. Therefore, batch 1 requires 1 RTT, batches 2 and 3 (having failed on batch 3 upon doubling) require (1+α) RTTs, etc... We can see that half the batches require 1 RTT and the other half require α RTTs. Ideally, however, each batch should require 1 RTT to transmit. Therefore, the effective throughput is

$$\frac{B \times RTT}{0.5 \times B \times RTT + 0.5 \times B \times \alpha RTT} = \frac{2}{1 + \alpha}$$



Problem 4: Random Early Detection
Consider a RED gateway with $MaxP = 0.02$, and with an average queue length halfway between the two thresholds. Recall that p is computed as $\frac{Q_{avg} - Q_{min}}{Q_{max} - Q_{min}} MaxP$ and $p_{count} = \frac{p}{1 - count \times p}$, where *count* is the number of packets that are queued since the last drop while $Q_{min} \leq Q_{avg} \leq Q_{max}$.

(a) Find the drop probability p_{count} for $count = 1$ and $count = 50$.

ANSWER: Since Q_{avg} is half way between the two thresholds, then $Q_{avg} = (Q_{min} + Q_{max})/2$. Therefore, $p = 0.5MaxP = 0.01$. $p_1 = \frac{0.01}{1-0.01} = \frac{0.01}{0.99} = 1/99 = 0.01$. $p_{50} = \frac{0.01}{1-50*0.01} = 0.02$.

(b) Calculate the probability that none of the first 50 packets are dropped. Note that this is $(1 - p_1) \times \ldots \times (1 - p_{50})$.

ANSWER:

$$(1-p_1)(1-p_2)\ldots(1-p_{50})$$
$$(1-\frac{0.01}{0.99})(1-\frac{0.01}{0.98})\ldots(1-\frac{0.01}{0.5})$$
$$(\frac{0.98}{0.99})(\frac{0.97}{0.98})\ldots(\frac{0.49}{0.5}) = \frac{0.49}{0.99} = 0.495$$

Problem 5: Fair Queueing
Consider a router that is managing three flows, on which packets of constant size arrive at the following real times:

flow A: 1, 2, 4, 6, 7, 9, 10

flow B: 2, 6, 8, 11, 12, 15

flow C: 1, 2, 3, 5, 6, 7, 8

All three flows share the same outbound link, on which the router can transmit one packet per time unit (therefore, we can think of every packet having a length of 1, and the router having a service rate $c = 1$). Assume there is an infinite amount of buffer space.

(a) For each packet, give the real time when it is transmitted by the router if the router is an ideal round robin system.

(b) For each packet, give the real time when it is transmitted by the router if the router implements Fair Queueing. Ties are to be resolved in the order A, B, C.

PS: there will be quite an amount of bookkeeping. So try to keep track of information using a table to compute $V(t)$, a_i^k, S_i^k, F_i^k, and which packets are still in queues, etc...

t|arrivals|$V(t)$|#flows(ideal sys)|F_p|t_{next}|sent|A's queue| B's queue|C's queue

---

t	arrivals	v(t)	# flows (ideal sys)	F	t_next	sent	A's queue	B's queue	C's queue
1	A1 C1	0	2	1 1	3	A1			C1
2	A2 B1 C2	0.5	3	2 1.5 2	3.5	C1	A2	B1	C2
3	C3	0.833333	3	3	3.5	B1	A2		C2, C3
3.5		1	3		5		A2		C2, C3
4	A3	1.166667	3	3	5	A2	A3		C2, C3
5	C4	1.5	2	4	6	C2	A3		C3, C4
6	A4 B2 C5	2	3	4 3 5	9	A3	A4	B2	C3, C4, C5
7	A5 C6	2.333333	3	5 6	9	B2	A4, A5		C3, C4, C5, C6
8	C7 B3	2.666667	3	7 4	9 9	C3	A4, A5	B3	C4, C5, C6, C7
9	A6	3	3	6	12	A4	A5, A6	B3	C4, C5, C6, C7
10	A7	3.333333	3	7	12	B3	A5, A6, A7		C4, C5, C6, C7
11	B4	3.666667	3	5	12	C4	A5, A6, A7	B4	C5, C6, C7
12	B5	4	3	6	15	A5	A6, A7	B4, B5	C5, C6, C7
13		4.333333	3		15	B4	A6, A7	B5	C5, C6, C7
14		4.666667	3		15	C5	A6, A7	B5	C6, C7
15	B6	5	3	7	18	A6	A7	B5, B6	C6, C7
16		5.333333	3		18	B5	A7	B6	C6, C7
17		5.666667	3		18	C6	A7	B6	C7
18		6	3		21	A7		B6	C7
19		6.333333	3		21	B6			C7
20		6.666667	3		21	C7			
21		7	0		0				

The above table shows how Fair Queueing simulates the ideal system. The first column t denotes the real time. The second column *arrivals* shows the packet arrivals. The third column $V(t)$ denotes the virtual time, as described in class, this is computed as follows:

$$V(t_j) = V(t_{j-1}) + \frac{t_j - t_{j-1}}{\#\ flows}$$

where the number of flows is taken from column 4 one row before. This is the number of flows in the ideal system during (t_{j-1}, t_j). The number of flows is updates every time there is an arrival, and every time some packets leave the ideal system (we keep track of this using t_{next}). The fifth column F computes the finish time of every packet. This is as usual $\max(F^{k-1} + V(t)) + 1$ (every packet has length 1), where F^{k-1} is the finish time of the previous packet of the same flow. Column six t_{next} computes the next time (real time) when a packet (or more) will leave the ideal system.

$$t_{next} = ([F_{min} - V(t)] * (\#\ flows) + t)/c$$

where F_{min} is the smallest finish time greater than $V(t)$ (i.e. $F_{min} - V(t)$ is always positive). When time t is reached such that t is equal to some t_{next}, we know that some packets will leave. These are the packets that have their finish time equal to $V(t)$. At this point, we also update the number of flows. Finally, every time the Fair queueing scheduler picks the packet with the smallest finish time to transmit. The rest of the columns show the packet sent and the state of the three queues.

ANSWER to part (a): The tuples $(sent, t+1)$ show the real time when each packet leaves the Fair queueing system.

ANSWER to part (b): To find when packets leave the ideal system, we look at the virtual times $V(t)$ and map their corresponding real time.

packets	V(t)	real time
A1, C1	1	3.5
B1	1.5	5
A2, C2	2	6
A3, B2, C3	3	9
A4, B3, C4	4	12
A5, B4, C5	5	15
A6, B5, C6	6	18
A7, B6, C7	7	21