COMP 431 - Problem Set 6

Joe Puccio

April 16, 2015

Collaborators: Fred Landis, Max Daum, Spencer Byers.

1.

No, this additive-increase additive-decrease algorithm does not retain the property that a set of two connections sharing a link would receive equal shares of the link's capacity. Because both connections are subtracting the same constant amount when a loss occurs, they will maintain the same sharing proportion as when they started, and so in most instances the algorithm will not result in equal sharing of link capacity. However, this additive-increase additive-decrease algorithm does result in equal sharing when the initial rates are equally proportioned, because the connection usage will only vary along the line of equally sharing proportions.

2.

The latency of slow start TCP (that is, the dynamic congestion windows) given T links would be given by:

$$2RTT + \frac{O}{R} + (T-1)\frac{S}{R} + P(T\frac{S}{R} + RTT) - (2^P - 1)\frac{S}{R}$$

We must add a $(T-1)\frac{S}{R}$ term because the entire object's transmission delay will increase by this much when you have $T-1$ routers between the end systems, and then we must multiply the $P\frac{S}{R}$ term by T because the time to return the ACK for the first segments in each window will increase by a factor of the number of links.

3.

Assumptions and explanation for (a) and (b): Assuming TCP with slow-start and that we have no packet loss. Also assuming the HTTP GET requests are sent with the last ACK in the TCP 3-way handshake. Also assuming, for slight simplification, that the image references are sent in the first packet of the base page and therefore the requests for the 10 embedded images can be sent with the ACK for the first packet in the base page (we claim this is reasonable because often resources are linked to at the top of HTML documents). Finally, assuming that payloads can be variable length, and therefore smaller packets must be sent if the data is not exactly divisible by the MSS. Stepping through the problem logically, for keep-alive HTTP requests (that is, persistent TCP connections), we see that we must first incur a 2RTT delay, which is due to the TCP 3-way handshake, plus the ACK for the first packet of the base page, and this first packet adds an additional $\frac{S}{R}$ transmission delay. Throughout the rest of the transmission for the base page, our window size will increase (by slow start) and transmit 9 full packets and part of a 10th (due to the MSS and base page size). Because we assumed that the requests for the images went out with the ACK for the first packet of the base page, the server will immediately begin transmitting the images in the window of

size 8, once the last packet of the base page has been transmitted (the window of sizes 1,2,4 have taken care of the majority of the base page). This process continues for the remaining images, with the window size doubling unbounded (because we assume no loss, and therefore there is no threshold to consider) until all images have been transmitted. We must take note of the fact of stalling delays due to the ACK of the first packet arriving after the window has been fully transmitted. With all this taken into account, we arrive at:

a)

Rate	Persistent (s)	Non-persistent (s)
28000b/s:	15.861142857142859	19.014285714285712
$10^5 b/s$:	4.7572799999999997	8.8017599999999998
$10^6 b/s$:	1.0514239999999999	6.788672
$10^7 b/s$:	0.90300159999999996	6.6188672000000004

b)

Rate	Persistent (s)	Non-persistent (s)
28000b/s:	20.745142857142856	72.738285714285709
10^5b/s:	10.514240000000001	67.886719999999997
10^6b/s:	9.0300159999999998	66.188671999999997
10^7b/s:	9.0030015999999993	66.018867199999988

Note that this approach takes into account variable size packets, and will differ from a solution that only considers fixed size packets.

c)

We are tasked with explaining the total response time of a web page with M embedded objects of size O with transmission rate R and a client capable of x parallel connections. Well, we know that the basepage is going to take $\frac{O}{R}$ time to transmit, and, because all of these objects are being sent over the same link, we will not be able to overlap transmission and thus the M embedded objects will take an additional $M\frac{O}{R}$ time to transmit (thus, we know there must be an $\frac{O}{R} + M\frac{O}{R}$ or $(M+1)\frac{O}{R}$ term. Next, we know that we must have $2RTT$ for the 3-way TCP handshake and the propagation of the GET response for the basepage, moreover because the HTTP requests are not keep-alive (that is, not the TCP connection is non-persistent), we know that each object is going to also incur a 2RTT delay, however because the client may make up to x simultaneous connections, this 2RTT delay for every object is cut down by a factor of x (because x handshakes can be done in parallel), so thus $2\frac{M}{x}$ RTTs are incurred for the embedded objects (the basepage cannot be done in parallel with other connections because in order for the other connections to start, the basepage must be fully downloaded). Thus, we know that the response time must include a $2RTT + 2\frac{M}{x}RTT$ term which can be rewritten as $2(\frac{M}{x} + 1)RTT$. Lastly, we know that there are additional delays due to bit stream stalling during, for instance, slow start, and so there is an added and independent stalling, which we'll call SSL. Putting these together, we achieve:

$$(M+1)\frac{O}{R} + 2(\frac{M}{x} + 1)RTT + SSL$$

4.

Our estimated RTT given the 4 sample RTT's is

$$EstimatedRTT = (1-x)^3 sampleRTT_4 + x(1-x)^2 sampleRTT_3 + x(1-x)sampleRTT_2 + xsampleRTT_1$$

which, given x = .1, is

$$EstimatedRTT = (.729)sampleRTT_4 + (.081)sampleRTT_3 + (.09)sampleRTT_2 + (.1)sampleRTT_1$$

We can generalize this for any n samples to achieve:

$$EstimatedRTT = x^0(1-x)^{n-1}sampleRTT_n + x(1-x)^{n-2}sampleRTT_{n-1} + ... + x(1-x)sampleRTT_2 + x(1-x)^0$$

which we can compact to be

$$EstimatedRTT = (1-x)^{n-1}sampleRTT_n + x\sum_{i=1}^{n-2}(1-x)^i SampleRTT_{i+1}$$

which can be rewritten as

$$EstimatedRTT = (1-x)^{n-1}sampleRTT_n + x\sum_{i=1}^{n-1}(1-x)^{i-1} SampleRTT_i$$

We can see why this formula is in the class of exponential moving average equations because, in the limit as $n \to \infty$, the oldest sample terms' weight dies off exponentially (with each additional term, their coefficients get a $(1-x)$ factor smaller), while the newer terms have significantly smaller exponents on their subzero coefficients (therefore, they are more heavily represented in the sum than the old terms).

5.

a) $k = MIN(i : 3^0 + 3^1 + 3^2 + ... + 3^{i-1} \geq \frac{O}{S}) = MIN(i : \frac{1}{2}(3^i - 1) \geq \frac{O}{S}) = MIN(i : i \geq \log_3(\frac{2O}{S} + 1))$
so $k = \log_3(\frac{2O}{S} + 1)$

b) $q = MAX(i : RTT + \frac{S}{R} \geq 3^{i-1}\frac{S}{R}) = MAX(i : 3^{i-1} \leq 1 + \frac{RTT}{\frac{S}{R}}) = MAX(i : i \leq \log_3(1 + \frac{RTT}{\frac{S}{R}}) + 1)$
so $q = \log_3(1 + \frac{RTT}{\frac{S}{R}}) + 1)$

c) Assuming the TCP implementation is using slow start, then our latency is going to be

$$2RTT + \frac{O}{R} + P(\frac{S}{R} + RTT) - (3^P - 1)\frac{S}{R}$$