EECS 122: Introduction to Communication Networks
Homework 9 Solutions

Solution 1.

a) The TCP congestion control algorithm causes each competing connection to get roughly the same window size (W) on average. A connection's throughput is W/RTT. To provide roughly equal bandwidth (throughput) to each connection, TCP could map each connection onto n independent subconnections, where each subconnection has its own window and runs the TCP congestion control algorithm independently. The source would divide the data to be sent among the subconnections, and the sink would merge it back together. Now each subconnection ends up with roughly the same window size, so each connection gets a throughput of $(W/\text{RTT}) \cdot n$. If n is chosen to be proportional to RTT, then each connection gets a throughput proportional to W, which as we said before is roughly the same for all competitors.

b) Let's try to roughly emulate the behavior of n independent subconnections with a truly single connection. Let's assume the connections use congestion avoidance but not slow start (which is just a performance optimization that would complicate our analysis). When there are n independent subconnections, each one increases its window by one packet per RTT, so the sum of the n windows increases by n packets per RTT. When a loss occurs, only one of the subconnections resets its window. If the individual windows were about the same, then the sum of the windows goes down by a factor of about $(n-1)/n$. Therefore, a single new-style connection that increases its window by n packets per RTT and decreases its window by a factor of $(n-1)/n$ when it detects a loss should get about the same throughput that n old-style connections would have gotten. If n is still chosen to be proportional to RTT (and now n need not be an integer), then the new algorithm will allocate roughly the same bandwidth to competing connections. [Note that this new-style algorithm violates the IETF standards for TCP, which require the normal congestion avoidance algorithm.]

Solution 2.

a) The true mean RTT is $0.8 \cdot 10 \text{ ms} + 0.2 \cdot 100 \text{ ms} = 28$ ms. We have assumed that the estimated mean is close to the true mean, so the timeout is set close to 56 ms. 80% of the packets have RTTs less than the timeout, and 20% have RTTs greater than the timeout, so 20% are mistakenly believed to be lost.

b) The true mean deviation is $0.8 \cdot |10 \text{ ms} - 28 \text{ ms}| + 0.2 \cdot |100 \text{ ms} - 28 \text{ ms}| = 28.8$ ms. We have assumed that the estimated mean and mean deviation are close to the true values, so the timeout is set close to $28 \text{ ms} + 4 \cdot 28.8 \text{ ms} = 143.2$ ms. All the packets have RTTs less than the timeout, so none are mistakenly believed to be lost.

c) In the derivation below, each statement is equivalent to the one before it. We begin by making substitutions based on the code:

$$\begin{gathered}
\mathit{timeout} \geq \mathit{samp} \\
\mathit{mean} + c \cdot \mathit{dev} \geq \mathit{samp} \\
\mathit{oldmean} + a \cdot \mathit{err} + c \cdot [\mathit{olddev} + b \cdot (|\mathit{err}| - \mathit{olddev})] \geq \mathit{samp} \\
\mathit{oldmean} + a \cdot \mathit{err} + c \cdot [\mathit{olddev} + b \cdot (|\mathit{samp} - \mathit{oldmean}| - \mathit{olddev})] \geq \mathit{samp}
\end{gathered}$$

Then we split into two cases and collect terms:

$$\begin{aligned}
& \mathit{samp} < \mathit{oldmean} \wedge \\
& (1 - a + cb)\mathit{oldmean} + (a - cb - 1)\mathit{samp} + (c - cb)\mathit{olddev} \geq 0 \\
\vee\ & \mathit{samp} \geq \mathit{oldmean} \wedge \\
& (1 - a + cb)\mathit{oldmean} + (a + cb - 1)\mathit{samp} + (c - cb)\mathit{olddev} \geq 0
\end{aligned}$$

which can be rewritten:

$$\begin{aligned}
& \mathit{samp} < \mathit{oldmean} \wedge \\
& (1 - a + cb)(\mathit{oldmean} - \mathit{samp}) + (1 - b)(c)\mathit{olddev} \geq 0 \\
\vee\ & \mathit{samp} \geq \mathit{oldmean} \wedge \\
& (1 - a + cb)\mathit{oldmean} + (a + cb - 1)\mathit{samp} + (1 - b)(c)\mathit{olddev} \geq 0
\end{aligned}$$

For the *samp* $<$ *oldmean* case, all the factors are positive, so the inequality is simply true. (Recall that we are given $0 \leq a, b \leq 1$ and $c \geq 0$.) For the other case, notice that the history of RTT samples can cause *dev* to become arbitrarily large, but because of the way *dev* is updated, it can never be negative. Therefore, the $(1 - b)(c)\mathit{olddev}$ term can be any nonnegative number including zero, so we can drop that term, and the resulting inequality is always true iff the original inequality is always true.

$$\mathit{samp} < \mathit{oldmean} \ \vee\ (1 - a + cb)\mathit{oldmean} + (a + cb - 1)\mathit{samp} \geq 0$$

samp is nonnegative, and because of the way *mean* is updated, it is always nonnegative also, but it can be zero. The coefficient $1-a+cb$ is nonnegative. Therefore, our condition is always true iff $a+cb-1>=0$, or:

$$c \geq \frac{1-a}{b}$$

TCP uses $a=1/8, b=1/4, c=4$, which satisfies the condition, so TCP guarantees that the timeout is never less than the latest sample.

Solution 3. The bandwidth of the signal is 5 MHz. From Nyquist's sampling theorem, we know that we have to sample this signal at at least twice this frequency, that is, at 10 MHz. In order to achieve a signal-to-quantization-noise ratio of at least 48 dB, we have to use at least $\frac{48}{6}=8$ bits per sample. At a sampling rate of 10 MHz, this results in a digital signal with bit-rate $8 \cdot 10 = 80$ Mbps. In practice, digital video is almost always compressed, so its bit-rate is lower.