Pset2 Report

Songtao He
songtao@mit.edu

In this report, we will quickly go through exercise A,B and C, and save most of our reasoning in exercise D.

1 Exercise A

In exercise A, 5 window sizes have been evaluated. The results are listed in table 1 and demonstrated in Figure 1. In this test, the scheme got the highest power score with a window size of 10. We repeat the evaluation three times with a fixed window size 20. The result is repeatable (table 2)

$WindowSize$	throughput	delay	power score
100	5.12 Mbps	1052 ms	4.86
50	4.79 Mbps	607 ms	7.89
20	3.29 Mbps	277 ms	11.87
10	1.96 Mbps	153 ms	12.81
5	1.07 Mbps	110 ms	9.72

Table 1: Result under Different Window Sizes

$WindowSize$	throughput	delay	power score
20	3.29 Mbps	276 ms	11.92
20	3.29 Mbps	277 ms	11.87
20	3.29 Mbps	276 ms	11.92

Table 2: Repeatablility

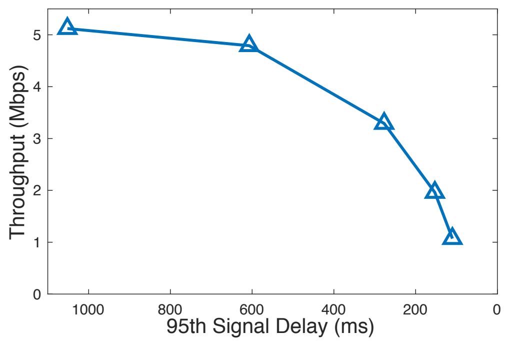

Figure 1: Throughput vs. 95th Signal Delay

2 Exercise B

In exercise B, an AIMD scheme is implemented. When an ACK is received, the window size will increase by $\frac{\alpha}{Win_{cur}}$. The 'MD' is triggered by a timeout of 50ms (We fixed this timeout value). The window size will be changed to βWin_{cur} when there is a timeout. We evaluated several combination of parameter α and β. The result is summarized in table 3. This scheme works better than the fixed window size scheme. The highest score we got is 16.22, where $\alpha = 2.0$ and $\beta = 0.3$.

$alpha$	β	throughput	delay	power score
1.0	0.7	4.62 Mbps	499 ms	9.25
1.0	0.5	4.37 Mbps	369 ms	11.84
1.0	0.3	4.13 Mbps	280 ms	14.75
2.0	0.5	4.76 Mbps	537 ms	8.86
2.0	0.3	4.64 Mbps	286 ms	**16.22**
3.0	0.3	4.81 Mbps	530 ms	9.07

Table 3: Result under Different Parameter Combination

3 Exercise C

In exercise C, a delay-based scheme is implemented. When an ACK is received, the scheme evaluates the RTT of this packet. If the RTT is smaller than RTT_{target}, the window size will increased by $\frac{\alpha}{Win_{cur}}$. If the RTT is larger than RTT_{target}, the window size will be changed to βWin_{cur}. We evaluated several combination of parameter α, β and RTT_{target}. The result is summarized in table 4. This scheme works pretty well. The highest score we achieved is 36.83.

RTT_{target}	α	β	throughput	delay	power score
80	1.0	0.92	2.80 Mbps	87 ms	32.18
80	1.0	0.95	3.06 Mbps	88 ms	34.77
80	1.0	0.98	3.67 Mbps	103 ms	**35.63**
80	1.0	0.99	4.03 Mbps	125 ms	32.34
80	2.0	0.95	3.40 Mbps	100 ms	34.00
80	2.0	0.98	4.00 Mbps	114 ms	35.09
70	1.0	0.98	3.35 Mbps	91 ms	**36.81**
70	2.0	0.98	3.75 Mbps	103 ms	36.41
60	1.0	0.98	2.82 Mbps	79 ms	35.69
60	2.0	0.98	3.29 Mbps	90 ms	36.55
60	2.0	0.97	3.02 Mbps	82 ms	**36.83**

Table 4: Result under Different Parameter Combination

4 Exercise D

In exercise D, we build a congestion-window based scheme. In the design of this scheme, we need to make several important decisions, such as control granularity, feedback function and window update mechanism. We describe these decisions as follow.

Control Granularity

Typically, congestion control systems work at two different granularities, per-packet controls (E.g., AIMD) or per-timeframe controls (E.g., Sprout). However, each of these two granularities has fundamental limitations. Per-packet control reacts nimbly when the throughput fluctuated heavily, but has a incomplete sense of large-scale link dynamic. Per-timeframe controls just lay in the opposite side of per-packet controls. These incomplete knowledge can lead to sub-estimation of the throughput and out-of-control of delay, which turns out to compromise the user experience. Thus, in this contest, the scheme adopts mixed granularity controls.

The scheme adjusts congestion window per ACK. The adjustment is calculated based on a chunk of history per-packet RTT. The estimation of chunk size is inspired by the idea of mixed granularity control. When a ACK is received, we look back in the history of ACK timestamps, and count the number of ACK within the latest 20ms[1] time window. Here, we use N_{20ms} to denote this number. Then, we calculate the chunk size using the following equation.

$$N_{chunk} = min(128, N_{20ms}) \tag{1}$$

The number 128 is not a magic number. The reason of using 128 is to reduce unnecessary computation due to we apply exponential decay on the history data. E.g. $0.97^{128} \approx 0.02$, which is small enough to be discarded.

When the link capacity is in a relatively stable state, we can always get enough samples to evaluate the link. When the link capacity is changing dramatically, e.g., recovering from an outage, the 20ms time window filter can make sure the N_chunk is small. This small N_chunk guarantees the agility of our control. To make sure the control system has enough observation of the environment, we force the sender to send heart-beaten packet every 20ms[2].

Feedback Function

We model the link capacity as the sum of a relatively stable link capacity and a capacity noise following a roughly centrosymmetry distribution. In figure 2, we study the characteristic of link capacity. We show two different throughputs calcucated at two different time granularities. The 200-step average represents the slowly evolving link capacity, whereas the 20-step average represents the capacity noise.

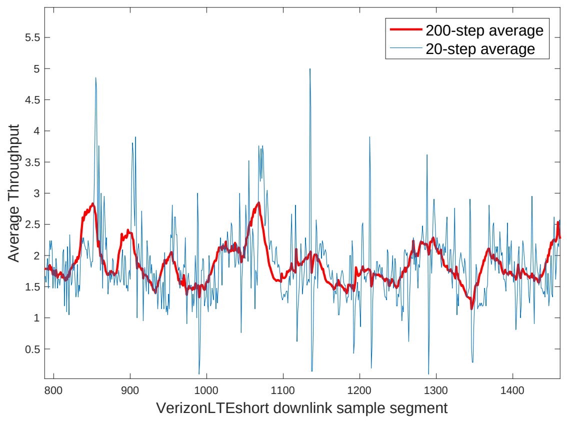

Figure 2: Example of Link Capacity Noise

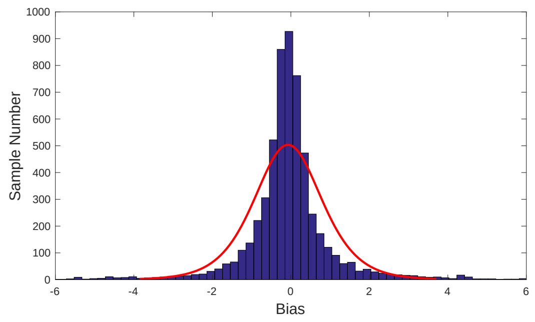

Figure 3: Distrubution of Link Capacity Noise

We then calculate the difference between these two throughputs and demonstrate the distribution in figure 3. In figure 3, we can conclude that the capacity noise is somehow following a centrosymmetry distribution. However, it is not a Gaussion noise[3]. The noise distribution has a larger varience than Gaussion noise. This can be caused by the *burst noise* in figure2. Thus, we regard this noise as the sum of a Gaussion noise and a burst noise. Although we only studied a small set of data, we believe this model is reasonable in practice.

We design our feedback function based on this observation and our control granularity decision. Basically, we use a low-pass filter to relieve the effect of Gaussion noise and a carefully designed feedback function to neutralize the burst noise. We first describe our feedback function, then we discuss the reason behind it.

Essentially, we split the feedback into negative feedback and positive feedback, and deal with them individually. The calculation of feedback is based on the history RTT. Let RTT_{avg} denote the average[4] RTT of the last N_{chunk} RTT smaples. We define the coefficient of positive and negative

[1]This number is fixed during all evaluations. We choice to use 20ms without any parameter optimizations.

[2]This is not a timeout

[3]The red curve in figure 3 is a Gaussion fit

[4]Calculated with an exponential decay of 0.97

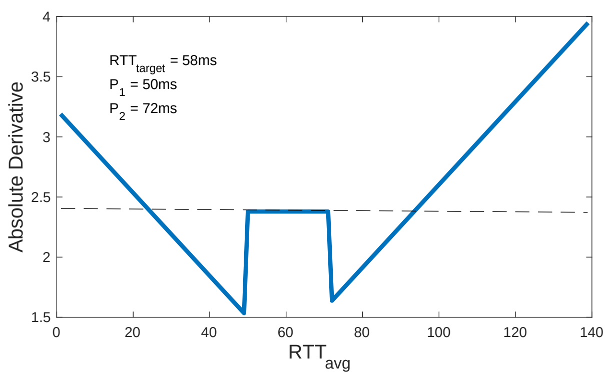

Figure 4: The Absolute Derivative of Feedback Function

feedback as follow,

$$C_{pos} = max(\frac{RTT_{target} * 2 - RTT_{avg}}{RTT_{target}}, 0.0) \quad (2)$$

$$C_{neg} = 1.0 + max(\frac{RTT_{avg} - RTT_{target}}{RTT_{target}}, -1.0) \quad (3)$$

where RTT_{target} is a parameter indicate the target RTT. We then define the positive and negative feedback functions.

$$F_{pos}(\alpha) = -C_{pos}min(RTT_{avg} - \alpha, 0.0) \quad (4)$$

$$F_{neg}(\alpha) = -C_{neg}max(RTT_{avg} - \alpha, 0.0) \quad (5)$$

Here the positive and negative feedback functions are functions of α. Actually, the α determines the effective scope of feedback function. Finally, the feedback function can be written as,

$$Feedback = F_{pos}(P_1) + F_{neg}(P_2) \quad (6)$$

where P_1 and P_2 are parameters. Essentially, when P_1 is smaller than RTT_{target} and P_2 is larger than RTT_{target}, they make the equation(7) a linear function of RTT_{avg} within the range $[P_1, P_2]$, and a quadratic function of RTT_{avg} when the $RTT_{avg} \notin [P_1, P_2]$.

It is straightforward to regard the average of history RTT as a low-pass filter for link capacity noise, whereas the reason of using equation(7) is not obvious. There is an simple explanation of equation(7). When RTT_{avg} is close to the RTT_{target}, we use linear feedback, which seems to be less aggressive. When the RTT_{avg} is far away from the RTT_{target}, we use quadratic feedback to pull it back aggressively. However, this interpretation of equation(7) is incomplete. Actually, the trick here is, when the equation(7) turns from the linear stage into the quadratic stage, the absolute derivative of it *decrease first instead of increase*.

Figure 4 demonstrates the absolute derivative of different RTT_{avg}. RTT_{avg} is essentially the result of a low-pass noise filter. When the RTT_{avg} is in a certain range, e.g., $[P_1, P_2]$, we assert that the link capacity is evolving moderately in time. Thus, we adopt a feedback which is proportional to the bias. When the RTT_{avg} is slightly drifting out of range $[P_1, P_2]$, we react conservatively by using a small ratio of feedback. This mechanism is designed to neutralize the influence of burst noise, which does exist in wireless link. Then, when the RTT_{avg} is increasingly far away from range $[P_1, P_2]$, the possibility of a dramatical link capacity change increases. Thus, we increase the ratio of feedback at the same time. This is supposed to be the true essence of equation(7).

Congestion Window Update

The effects of congestion window updates can not be observed immediately. It is also really hard to make a direct map between congestion window updates and the corresponding effects. In fact, the current measured RTT_{avg} is determined by the whole history of congestion window. It is impossible to figure out the detail underneath relationships. But somehow, we can try to make the system converge to the right point in a reasonable manner. We assume that the current RTT_{avg} is effected[5] by a chunk of congestion window history prior to the chunk we used in RTT_{avg} calculation. We apply the feedback on the average congestion window of this chunk (Win^*_{old}), instead of the current congestion window. This can be written as,

$$Win_{new} = Win^*_{old} + \beta Feedback(RTT_{target}, P_1, P_2) \quad (7)$$

Evaluation

In our scheme, we totally have four parameters, $RTT_{target}, P_1, P_2, \beta$. The choice of these parameters may not be magic. Most of them are supposed to have some connections with the delay requirement and the noise distribution. E.g., the choice of RTT_{target} should be roughly proportional to the delay requirement. Essentially, the range $[P_1, P_2]$ should be related to the distribution of noise. The bias between RTT_{target} and $\frac{P_1+P_2}{2}$ described the dissymmetry distribution of burst noise. E.g., the median number of the noise distribution in figure 3 is -0.08 (more negative burst noise). Thus, the value of RTT_{target} should be more closer to P_1, so that the absolute derivative of the first turning point in figure 4 is smaller than that of the second turning point. This makes the system more conservatively when reacts to a negative burst noise. Actually, the parameter setting in figure 4 can achieve a power score of 45.82. We list some parameter settings in table 5, somehow, they all almost follow the observations we just discussed. Figure 5 shows the scoreboard at 5pm ET October 23, 2016. It is very clear that the dots generated by this scheme formed the frontline of the scoreboard and surpass all other schemes.

[5]Not only, not completely, but somehow does have effect.

RTT_{target}	P_1	P_2	throughput	delay	power score
58	46	72	3.44	76	45.26
56	48	72	3.50	77	45.45
58	48	72	3.53	78	45.26
58	50	72	3.62	79	**45.82**
56	50	74	3.62	80	45.25
58	50	74	3.69	81	45.56
58	52	74	3.74	82	45.61
62	56	72	3.83	84	45.60

Table 5: Parameter Settings with a Power Score Higher than 45 (β is set to 0.04)

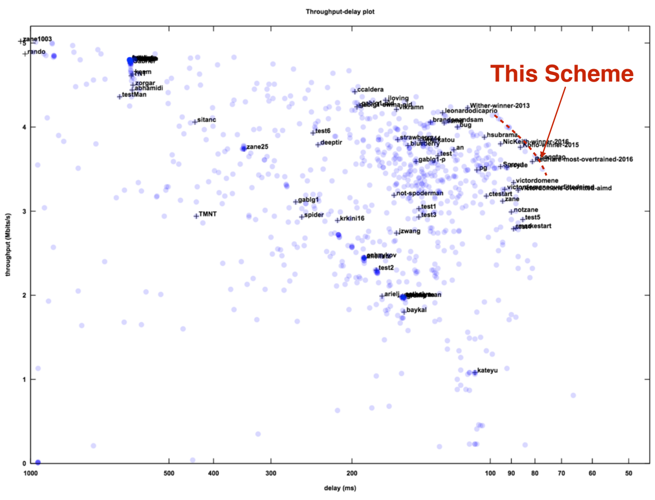

Figure 5: Scoreboard at 5PM ET Oct. 23, 2016

Discussion

The choice of parameters can lead to overfitting. However, if we can change these parameters dynamically based on the link character, then overfitting is not a problem. Fortunately, current evaluation has already shown some evidence and potentials that those parameters have connections with link characters, e.g. noise distribution. It is deserved to conduct further research towards this direction. A possible solution in practice is that we can regard the noise distribution as an inherit character of the wireless link, which could be measured at runtime and predictable in a short time. Thus, we can *deterministically* choose parameters based on this noise distribution at run-time.

5 Exercise E

During the contest, I totally used two names. The name 'test' was used for test. Later, I gave up this name and turn to use 'Songtao' as the final scheme name.