Performance Evaluation, Homework 1

Joël M. Fonseca 227334, Francis Damachi 217575

March 4, 2019

1 Variability of performance metrics

To test the variability of the performance metrics we ran 100 simulations with an arbitrary configuration defined as follows: $C = 3$, $AP = 2$ and $S = 1$. We can observe from Figure 1 that each metric varies within some interval.

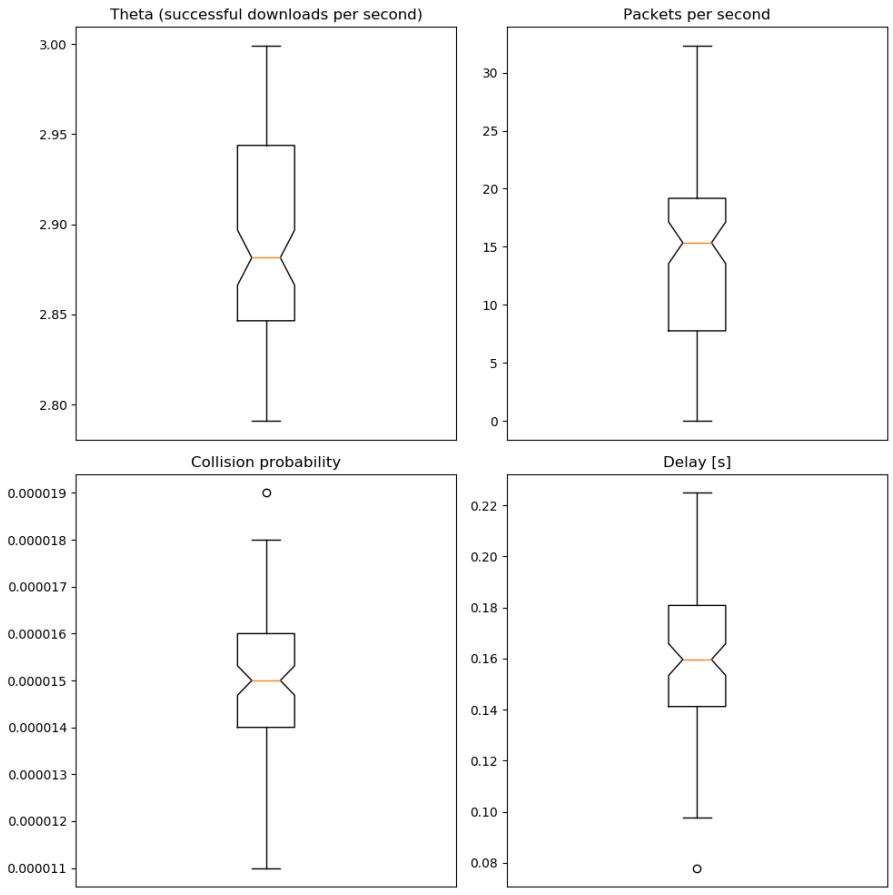

Figure 1: Variability of performance metrics over 100 simulations with arbitrary configuration.

2 Performance for AP=S=1

2.1 Simulation setup

For this question, we ran the simulation by varying the load factor C in the range between 1-1000. The results of these simulations are shown on the following 4 plots. We plot the sucessful download requests per second (theta), the delay, the packets per second and the collision probability. All of these plots are with respect to the number of requests per second.

2.2 Discussion

2.2.1 Observations

First of all, we can see that on the first plot from Figure 2, if we restrict the load factor between 1-170, we see that there seems to be a linear growth with the number of successful downloads achieved. As we reach approximately the value of 170, the graph grows slower. Meaning that as when we increase the number of requests per second from 170 we don't get a linear increase but rather a sub-linear growth. Similarly, if we compare the plot from Figure 2 to the plot from Figure 5, we see that the graph displaying the number of packets changes its shape from approximately $C = 170$. Furthermore, it seems that at the same mark of 170 number of requests, the collision probability reaches its peak value of 1. This might explain the reason why we aren't achieving as many successful downloads anymore because due to packets colliding with each other.

2.2.2 Congestion collapse

To conclude our analysis we can see a glimpse of congestion collapse in Figure 2. As the delay increases due to the increase of load we reach a point where the number of successful downloads increases less quickly. Perhaps if we were allowed to run the experiment with more than 1000 number of requests, we would have seen the number of successful downloads decrease dramatically.

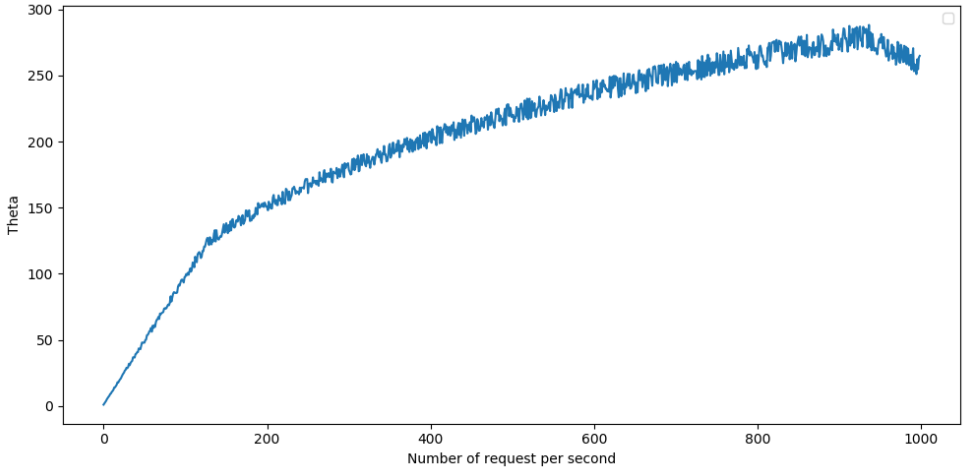

Figure 2: The number of successful downloads (Theta) as the number of requests per second increases.

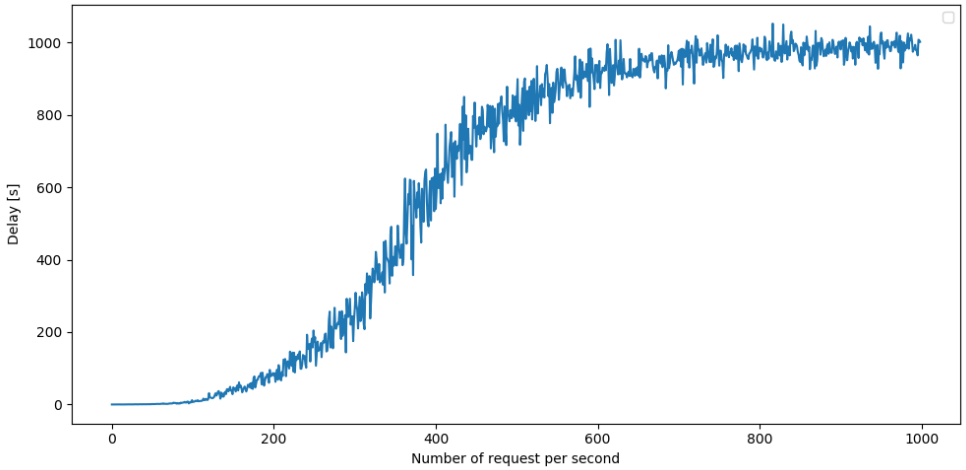

Figure 3: The delay in seconds as the number of requests per second increases.

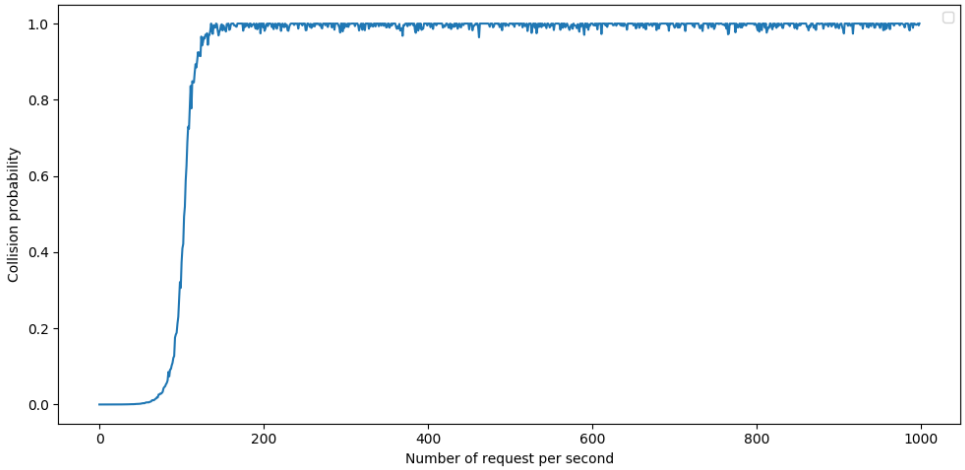

Figure 4: The collision probability as the number of requests per second increases.

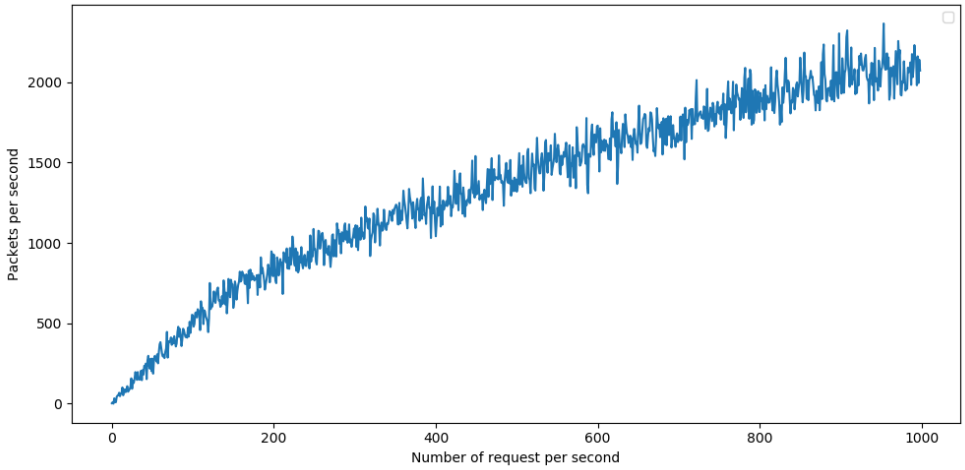

Figure 5: The achieved packets per seconds as the number of requests per second increases.

Suppose that doubling the number of access points is a valid solution. Then, one way to verify it is to freeze all the remaining parameters and double the number of access points for a few iterations. In this way, we can see if we get an improvement in the throughput.

3 Doubling the number of access points only

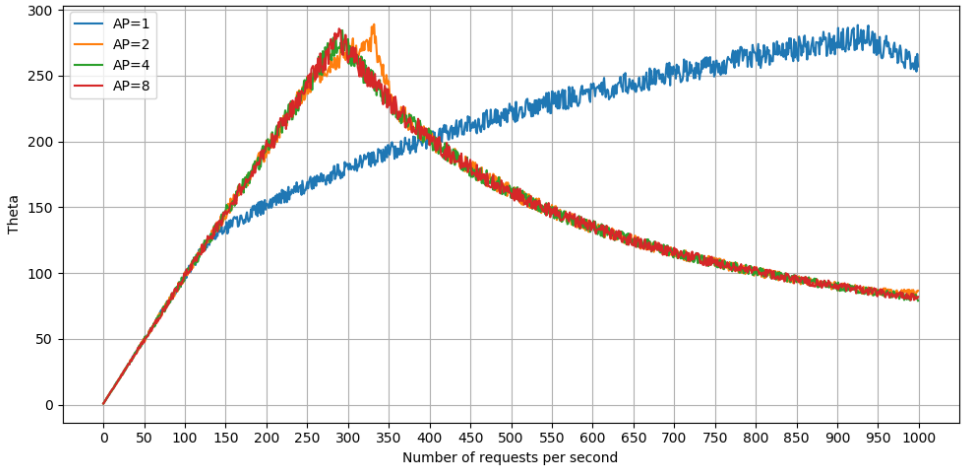

Figure 6: Effect of doubling the number of access points on the number of successful download requests per second by keeping the number of servers unchanged.

From Figure 6 we see that doubling the number of access points works only up to a certain point. Indeed, around 300 requests per second, we observe a congestion collapse from which the performance becomes worst as the number of requests per second increases for $AP \neq 1$. After 400 requests per second, we see that there is no advantage to having more than one access point as long as we keep the number of servers unchanged. Hence, we see that doubling only the number of access points and freezing the remaining parameter does not lead to a linear growth in the throughput. However, a more sophisticated approach involving all the parameters may lead to such goal.

4 Engineering Rule

4.1 Observations

From Question 3, we know that we cannot only change the number of access points, AP, depending on the number of requests per second, C, but we should also consider the number of servers, S. In our problem, the optimal solution should be composed of the minimal number of access points and servers as these can represent a significant cost.
The procedure is rather simple: when plotting the behaviour for doubling the AP like in Figure 6, for example, we should look at a break in the linear throughput. The break indicates that a change should occur, either on the AP or S side, depending on what we plot, in order to maintain the linear growth. Notice that by doubling values, we might miss the behaviour compared to if we had simply increased linearly the values. Indeed, maybe $AP = 3$ is sufficient for a specific C rather than having $AP = 4$ and therefore avoid wasting resources.

4.2 Experimentations

One way to come up with a rule is to think about how many requests an AP and a S can serve. Figure 7 contains the analysis of increasing S linearly while keeping $AP = 1$. Figure 8 shows the inverse: it increases AP linearly while keeping $S = 1$. For both figures, the blue (respectively red) vertical dashed line represents the critical value of C at which an AP (respectively S) should be added in order to keep a linear throughput.

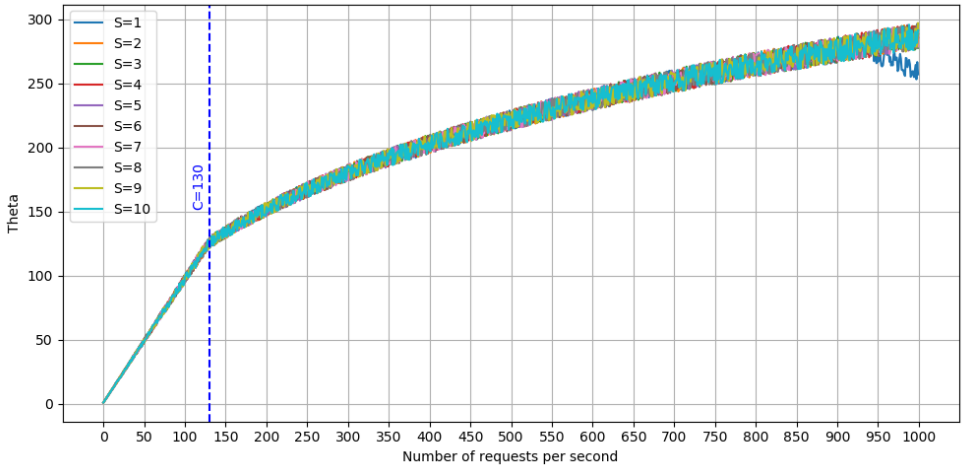

Figure 7: Analysis of linear increase of S with $AP = 1$.

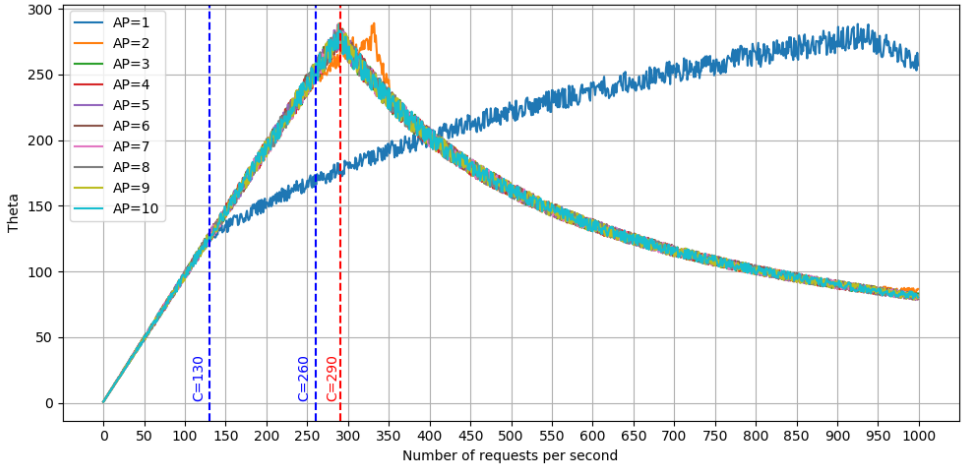

Figure 8: Analysis of linear increase of AP with $S = 1$.

From the above figures, we see that an AP can handle a load of $C = 130$ and that a S can handle a load of $C = 290$ before having a decrease in performance. To verify this, Figure 9 contains the same content of Figure 8 but with $S = 2$.
We see that our assumption works. Indeed, at $C = 130, 260, 390, 520$ we see a change in performance for the configuration with $AP = 1, 2, 3, 4$ respectively. Hence, the rule of adding one AP every 130 requests seems appropriate. We also see that, the critical load for the server is at $C = 580$ which is double the load we had for the configuration where $S = 1$. We know the limitation comes from the server side as every $AP > 4$ configuration is performing worst afterwards.

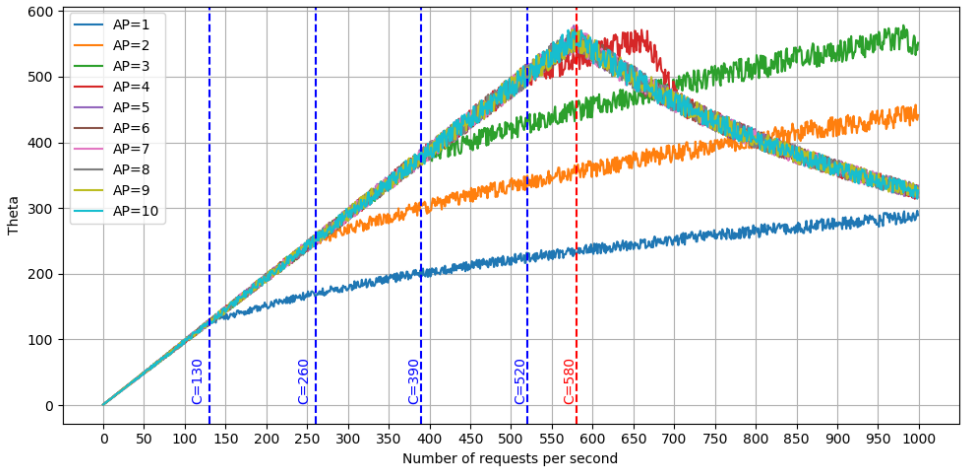

Figure 9: Analysis of linear increase of AP with $S = 2$.

4.3 Rule

Let $f(c)$ (respectively $g(c)$) be the number of access points AP (respectively the number of servers S) needed for c number of requests per second. Then, our engineering rule is defined as:

$$f(c) = \text{ceil}(c/130) \quad \text{and} \quad g(c) = \text{ceil}(c/290). \tag{1}$$

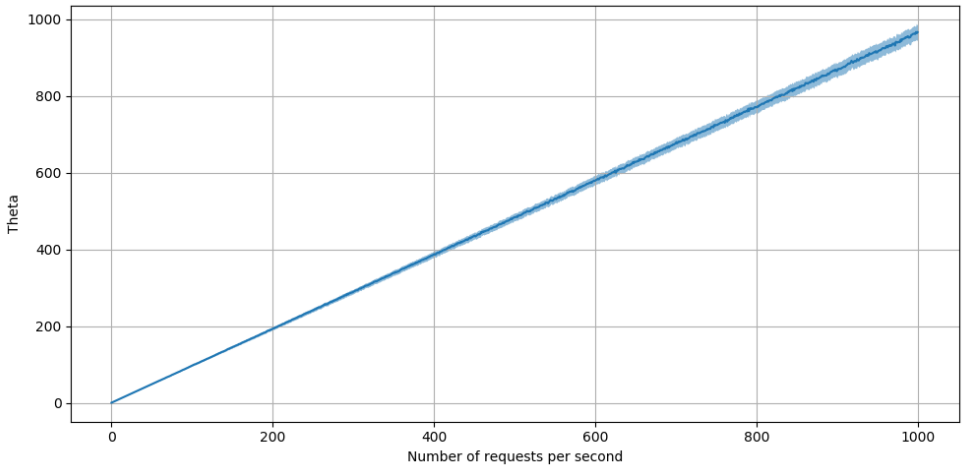

Figure 10: Engineering rule result averaged over 100 simulations.

4.4 Optimization

It is very likely that $C = 130$ is the congestion point for the AP and $C = 290$ for the S. One way to optimize our rule is to increase the number of corresponding AP and S just before reaching the congestion point. In this way, delays can be further reduced. Note that this was not implemented in our solution, but can be tried out as further work if a deeper analysis on the delay metric is of interest.