Hard deadline: May 15, 2023, 5 pm IST
No extension can be given as I have to upload grades by May end and going through the assignments will take time. Hence, the assignment is uploaded a month before the deadline so that you can plan your time well.
Instructions: Write the simulation in codes in Python, C++ or MATLAB. You need to submit a write-up (word/latex) where you discuss your approach, explain different parts of the codes (and logic behind them) and report your findings from the simulations. At the end of the report you must also add the code and instructions to run that, so that we can run it if needed.
One question on simulating a Geo/Geo/1 queue has already been uploaded. That would be worth 6 **points**. Here, we upload two more simulation questions and a little "research-y" question.
Recall TCP. Here, we shall simulate a simplified version that has the essence of TCP. Using that we shall try to understand the interplay between schedulers like max-weight and TCP. As I had said in the class, analysis of that interplay is often intractable and hence, simulations (and real experiments) are the only way forward.
In these problems, we focus on that side of TCP which deal with timeouts and not dupACKs, since simulating a scenario that results in dupACKs would involve multi-hops and may be a little hard on you. It does not mean that those scenarios are not important.

Q. 1 [8 points] **Interplay between the MAC layer and a TCP like transport layer**

Consider K discrete-time Bernoulli flows of packets arriving at a buffer which is served by a single server whose service capacity is Binomial($N, \frac{K}{N}$), where $N > K$. This means that if $S(t)$ is the realization of Binomial($N, \frac{K}{N}$) at time t, the server can serve $S_i(t)$ number of jobs from flow i while satisfying $\sum_i S_i(t) \leq S(t)$.

Out of K flows, R flows have Bernoulli p arrivals. Remaining flows are also independent Bernoulli, but their rates change in the following way.

For each of these $K - R$ flows maintain average delay in the following way. For flow i, if the average delay up to packet n is $T_n^{(i)}$ and the delay (from arrival to service) of the $n+1$th packet is $D_{n+1}^{(i)}$ then update the average delay as $T_{n+1}^{(i)} = 0.9\ T_n^{(i)} + 0.1\ D_{n+1}^{(i)}$.

Suppose the arrival rate for flow i at time t is $p^{(i)}(t)$ and $n^{(i)}(t)$ be the number of packets served so far. If at time t there is at least one un-served packet that arrived on or before $t - \lceil 1.2\ T_{n(t)}^{(i)} \rceil$, update $p^{(i)}(t+1) = \frac{2p^{(i)}(t)}{3}$. If at time t, r packets are served and all of them arrived after $t - \lceil 1.2\ T_{n(t)}^{(i)} \rceil$, then update $p^{(i)}(t+1) = \max(0.9, p^{(i)}(t) + \delta)$ for $0 < \delta < 1$. Generate an arrival in slot $t+1$ for this flow according to Bernoulli $p^{(i)}(t+1)$.

Consider the following service policies.

Processor sharing: If $S(t) > l\ K$ and $S(t) \leq (l+1)\ K$ for some integer $l \geq 0$, then serve l packets from each flow first. Then, serve one additional packet from flows $1, 2, \ldots S(t) - l\ K$.

Water-filling or minimize the max-queue: Let flow i have $Q_i(t)$ number of packets at time t. Serve the flows in such a way that at $t+1$, the gap $\max_i Q_i(t+1) - \min_i Q_i(t+1)$ is minimized. (Think: there is an iterative way of doing it.)

Max-weight scheduling: Check class notes.

- In your report, write a short criticism of this caricature TCP model.
- Pick any of the $K - R$ TCP flows. Compare its throughput (define it) and average delay under the three scheduling policies described above.
- Also compare the server utilization, i.e., the number of packets really served at t divided by $S(t)$. (Take a time average of that fraction.)

  Try different values of p. Pay close attention to p close to 1 and p close to 0.

  Try out the following (K, R) pairs: $(5,1)$, $(20,1)$, $(5,4)$, $(20,15)$, $(50,1)$ and $(50,40)$.

  Try $N = \lceil 1.1K \rceil$, $N = \lceil 1.5K \rceil$ and $N = \lceil 3K \rceil$

Q. 2 [5 points] **Finite buffer** Repeat the previous problem, but with the change that the buffer size is limited. Here, any packet arriving to a full buffer will be dropped. Though a dropped packet will not contribute to the update of $T_n^{(i)}$, but it will contribute to the number of un-served packets that arrived on or before $t - \lceil 1.2\ T_{n(t)}^{(i)} \rceil$.

Simulate for different buffer sizes $\alpha\frac{Kp^2}{1-p}$ for $\alpha = 1.2$, 1.5 and 3. (The formula $\frac{Kp^2}{1-p}$ is for the average queue-length in the infinite buffer case. It comes from Kingman's bound and Little's theorem, in case you are interested to know. We are trying to explore, how big a buffer should be.)

Q. 3 [5 points] If you do the above simulations carefully, you will make some interesting observations. You will be given marks for collecting your thoughts and putting them down one or two exciting directions that the observations lead you to. This problem is intentionally kept open ended so that you can get a flavor of research, which is also an objective of this graduate course. (Note that the this particular problem is not so well explored. So, if you pursue carefully, you may even get a paper out of it!)