Network QoS (371-2-0213)	**Summer Semester, 2024**

Assignment 1

Due date: Wednesday, September 25th 2024, noon @ Moodle

1. *(20 points)* Consider algorithm AAP_ε for $\varepsilon = 1$, and assume all edges have capacity $\log 4D$. We will compare the performance of AAP_ε with the performance of an optimal solution over the same network, but where all edges have unit capacity (instead of $\log 4D$). This type of analysis is usually referred to as *resource augmentation* analysis, i.e., how good can an algorithm be if it has more resources than the optimum (which has less resources). Intuitively, it should be clear that AAP_ε should be able to do better against such a restricted optimum, but the question is how much better.

   Prove that the the solution produced by AAP_ε is within a constant factor of the one produced by such a restricted optimal solution (try to prove the smallest constant possible).

   Note that the standard analysis we saw in class guarantees a competitive ratio of $O(\log D)$. This means that by having resource augmentation, we are able to *prove* a better performance

2. *(20 points)* In this question we will prove a lower bound on the competitive ratio of deterministic algorithms. One method for doing this is the following: Describe an adversary that generates the arrival sequence, such that *any* deterministic online algorithm does relatively bad, and where there is a way to do well for the same arrival sequence, had we known it all in advance (this would be the optimum). Specifically, we will prove a lower bound on the performance of deterministic algorithms for the admission control problem considered in class.

   Prove a lower bound of $\Omega(\log n)$ on the competitive ratio of any deterministic algorithm for the admission control problem considered in class. Aside from the proof, use a small diagram to depict the requests arrival sequence.

   Hint: read [1], but pay attention to the fact that the notation there is *different* than the one we used in class.

3. *(20 points)* In this question we consider some relations between the FIFO model and the various restricted bounded-delay models (all assuming cut-through).

   (a) *(10 points)* In the bounded-delay model, prove that if the arrival sequence is 2-bounded then it has agreeable deadlines.

   (b) *(10 points)* Let P be a set of packets, each with some release time and weight. Consider the following notation:

      i. $O_{\text{fifo}}(B)$ is the optimal solution in the FIFO model problem with buffer size B, given P

      ii. $O_{\text{u-delay}}(\delta)$ is the optimal solution in the uniform bounded-delay model problem with uniform delay δ, given P (i.e., all packets have delay δ)

   Prove that, for the case where $\delta = B$, $O_{\text{u-delay}}(\delta) = O_{\text{fifo}}(B)$. [1]

[1] I.e., if we take buffer size B in the FIFO model, and let δ be the *same* B in the uniform bounded-delay model, the optimal solutions are the same.

4. *(20 points)* Browse around the Internet, and look for Markov Modulated Poisson Processes (MMPP). Answer the following questions for a 2-state MMPP:

   (a) *(5 points)* How does this stochastic process work? Explain each of the parameters, and write down a diagram to better explain your answer.

   (b) *(5 points)* What is such a process useful for in a networking context? Specifically, what kind of traffic does it model? Explain why this is so.

   (c) *(10 points)* Write down a short program in *Python* that generates traffic consisting of unit-size packets according to a 2-state MMPP that is an ON-OFF process. Choose the parameters so that the traffic would have an average rate of r packets/time-unit, and average burst size of B packets (recall packets are of unit-size). r and B should be the arguments of your program, as well as a time horizon T during which traffic is generated. Your answer to this section should include the following:

   - your documented(!) code (to be submitted on Moodle),
   - An explanation as to why your choice of parameters for the MMPP indeed generates traffic with the required r and B statistics (r and B are target traffic parameters, and your choice of internal MMPP parameters should generate such traffic).
   - Using your code, generate 2 sequences of packets, S_1 and S_2, such that S_1 has average rate $r = 1$ and average burst size $B = 20$, and S_2 has average rate $r = 1.5$ and burst size $B = 3$. Submit 2 plots, one for each sequence, showing the buffer occupancy of a queue that has the MMPP-generated sequence as the arrival process, and a service rate of 1 packet/time-slot, for T=500 time slots. In each plot, you should have 100 ticks, where for each one you show the maximum average queue occupancy in the last 5 time slots (with non-overlapping intervals).
   - If your 2 plots look significantly different, explain why they are different. If you plots look similar, explain why they are similar.

   A few notes: (a) It is recommended that you find a paper that makes use of such traffic, and understand how MMPP is used in the paper. (b) Since the MMPP is a stochastic process, it will (probably) not generate traffic that *exactly* conforms with r and B. However, you should provide a sufficient explanation as to how/why you have chosen your parameters so that it the resulting traffic "comes close" to correspond to parameters r and B.

5. *(20 points)* In this question you will be required to implement a 3-color marking module in *Python*, for unit-sized packets. Assume each packet has the following fields:

   - *arrival time*
   - *size* (in this question, this should be set to 1 for all packets)
   - *color* $\in \{0, 1, 2\}$ (0=green, 1=yellow, 2=red).

   (a) Implement an (r, B) regulated token bucket, for a positive integer B, and some real number $r > 0$, implementing the algorithm specified in Algorithm 1 below.

   (b) Next, implement a two-rate token bucket with peak parameters (PIR, PBS) and committed parameters (CIR, CBS), performing 3-color marking, implementing the algorithm specified in Algorithm 2 below (as explained during the lecture).

   (c) Finally, feed your 2-color marking module with the MMPP traffic generated in the previous question. Choose the parameters (of the MMPP and the 3-color marking module, were the MMPP average rate should be at least 1) such that roughly 10% of your traffic is colored red, and 30% of your traffic is colored yellow. (you should generate a sufficiently long stream of packet, to obtain the required statistics, of no less than 500 time slots)

Your answer to this section should include the following:

- your documented(!) code (to be submitted on Moodle),
- the statistics of your 3-color marking module, given the MMPP traffic, including:
  - (i) number of time slots simulated,
  - (ii) MMPP parameters,
  - (iii) average rate and burst size for these MMPP parameters (these should be derived similarly to what you did in the previous question),
  - (iv) peak TB parameters, and percentage of red traffic, and
  - (v) committed TB parameters, and percentage of yellow traffic,
- an explanation of why these peak and committed parameters indeed yield the required percentage, and
- for each color (green, yellow, and red), a plot showing the *cumulative percentage* of traffic of that color (since the beginning of the simulation). All plots should appear on the same figure. (x-axis should be time, and y-axis should be percentage).[2]

Algorithm 1 two-color-TB(r, B): Input: a sequence of packet arrivals

```
for every packet p in the sequence do
    if there are enough tokens in the bucket then
        mark p as conforming
        reduce appropriate amount of tokens from bucket
    else
        mark p as non-conforming
    end if
end for
```

Algorithm 2 three-color-TB(PIR, PBS, CIR, CBS): Input: a sequence of packet arrivals

```
for every packet p in the sequence do
    mark p with the appropriate color
end for
```

[2] Note that if your simulation is correct, each plot should converge to the right percentage.

6. *(10 points)* OPTIONAL BONUS!!! Write 2 good questions for the final exam, related to any of the material covered in class.

   A good question is a question requiring understanding of the topic, but does not require an overly lengthy answer (10 lines or less should suffice for an answer).

   Such questions could be be related to modifications of algorithms we've seen, calculations of parameters, variants and/or special cases of models/problems we've seen, lower bounds, etc.

   Each good question can get up to 5 points, for a total of no more than 10 points. If you submit more than 2 questions, you will get credit for the best two (assuming they're deemed good questions)!

   Each question should be accompanied by a complete answer. A question to which no answer is provided will not be considered.

References

[1] Awerbuch, Azar and Plotkin: Throughput-competitive online routing. In FOCS 1993, pp. 32-40.