Lab6Answers

Xiao Wang, Meng-Chieh Lin

November 2019

1 Work distribution

Xiao Wang (wang3702@purdue.edu):
server component, performance evaluation, and write-up
Meng-Chieh Lin (lin1055@purdue.edu):
client component, performance evaluation, and write-up

2 Using semaphores protect the shared producer/consumer audio buffer so that it is not corrupted by concurrent access. Describe your method in Lab6Answers.pdf.

To avoid the buffer corruption, we designed two global parameters to control the access. One is buffer_end, which is used to record the current received buffer size, another is buffer_start, which is used to record the current played audio size. Since we designed the queue in an array, we used buffer_end%buffer_size and buffer_start%buffer_size to record the starting point and ending point. Moreover, we will check if the buffer is enough to play or receive when we do the playing or receiving. By doing so, we successfully avoid the reading and writing corruption.

3 Plot the time series measurement logs using gnuplot, MatLab, or Mathematica.

The following figures are transmissions of file pp.au from machine pod1-2 to escher02. The unit of time is second, Lambda is packet per second, and load is bytes. About the parameters we choose for the sender(server) and the receiver(client), payload size is 1488 bytes, initial lambda is 11, block size is 4096 bytes, gamma is 4, buffer size is 49152 bytes, and target buffer size is 24576 bytes. In control-param.dat file, we choose parameter α as 1, δ as 0.5, ϵ as 0.1, and β as 1.0. This file is used for all figures in the report.

Figure 1 to Figure 8 are the benchmarks for pp.au.

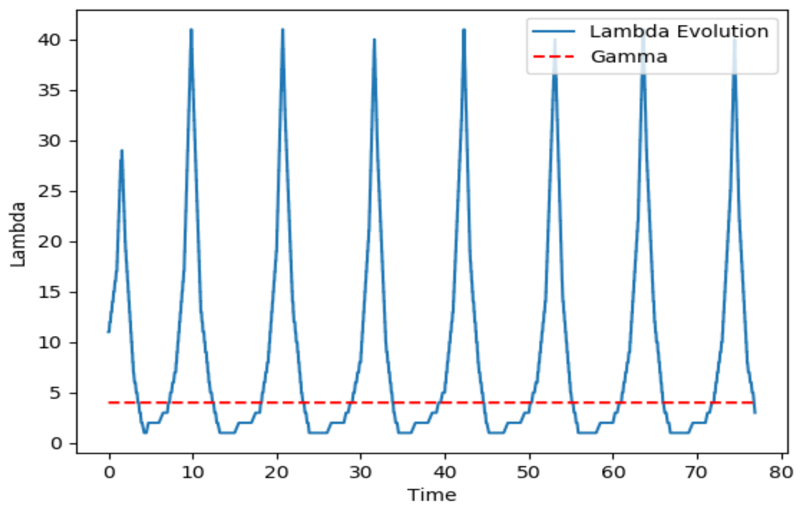

Figure 1: Sender mode: A

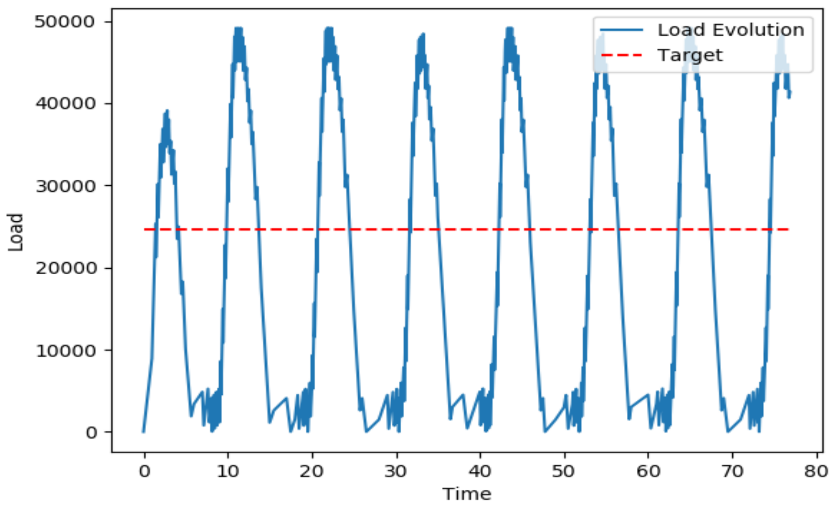

Figure 2: Player Mode: A

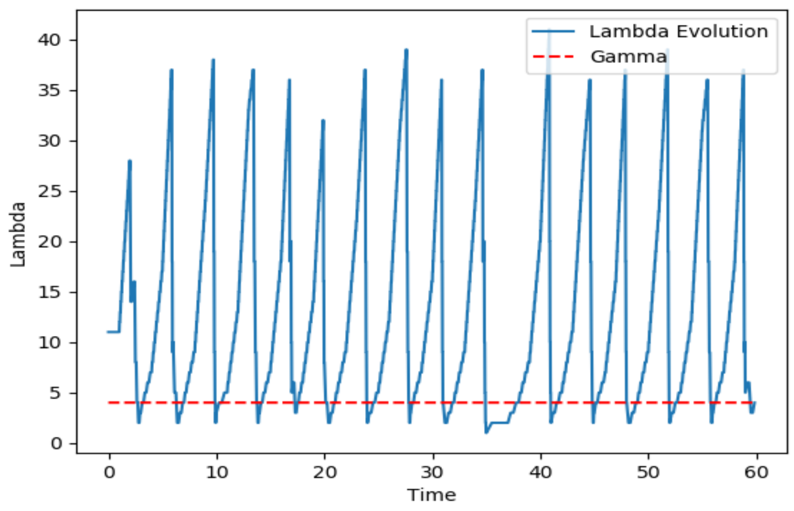

Figure 3: Sender Mode: B

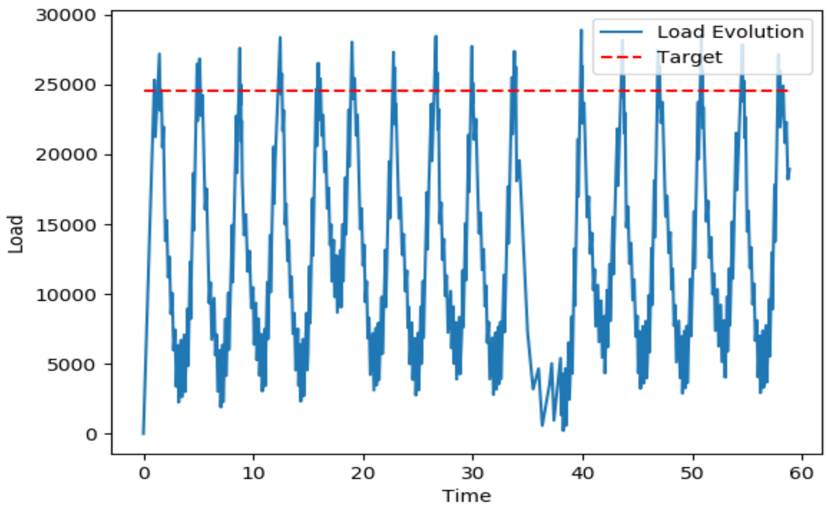

Figure 4: Player Mode: B

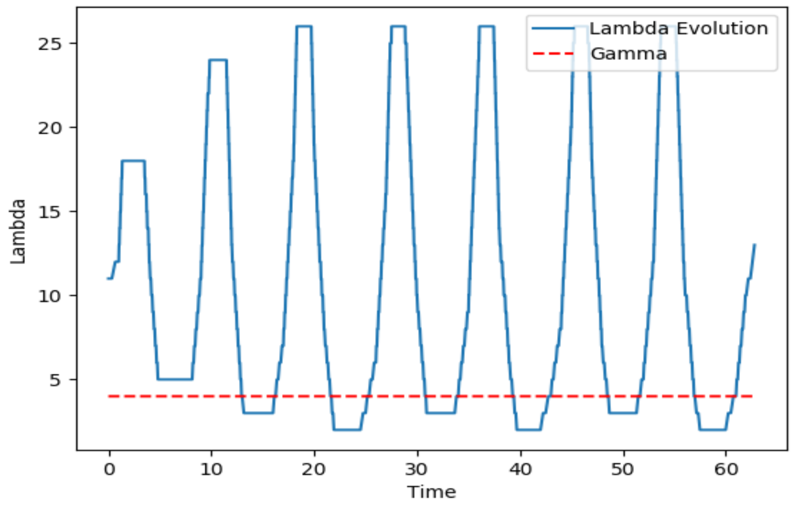

Figure 5: Sender Mode: C

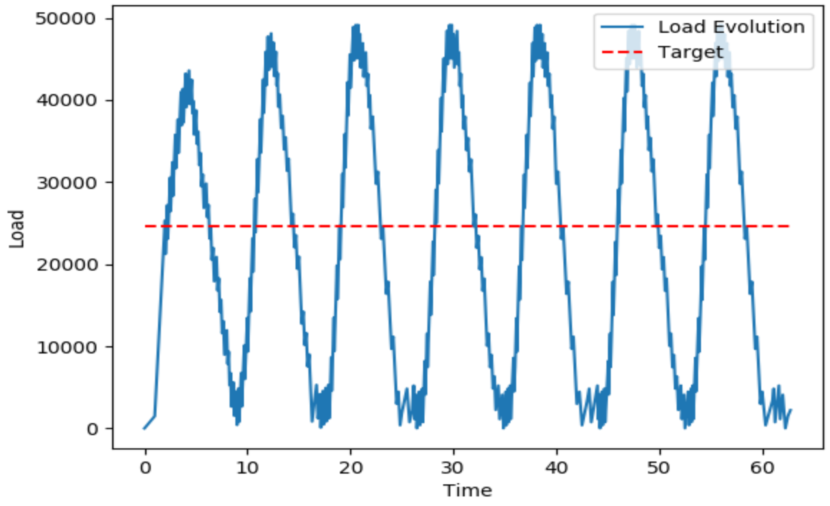

Figure 6: Player Mode: C

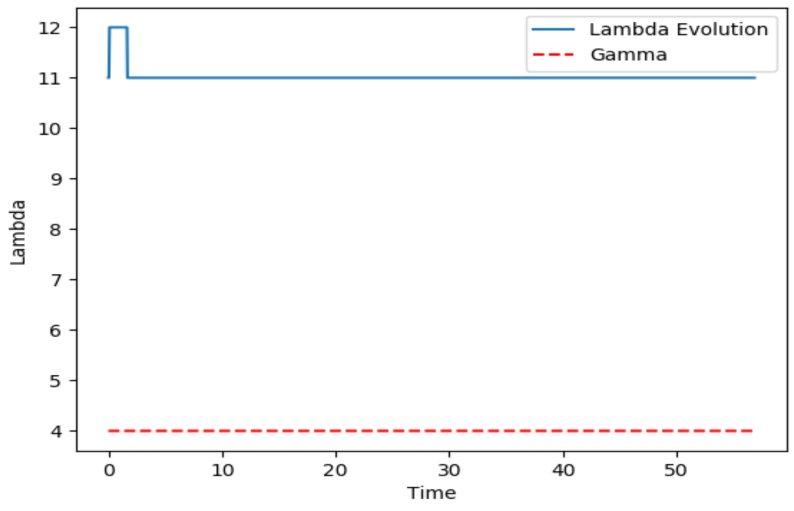

Figure 7: Sender Mode: D

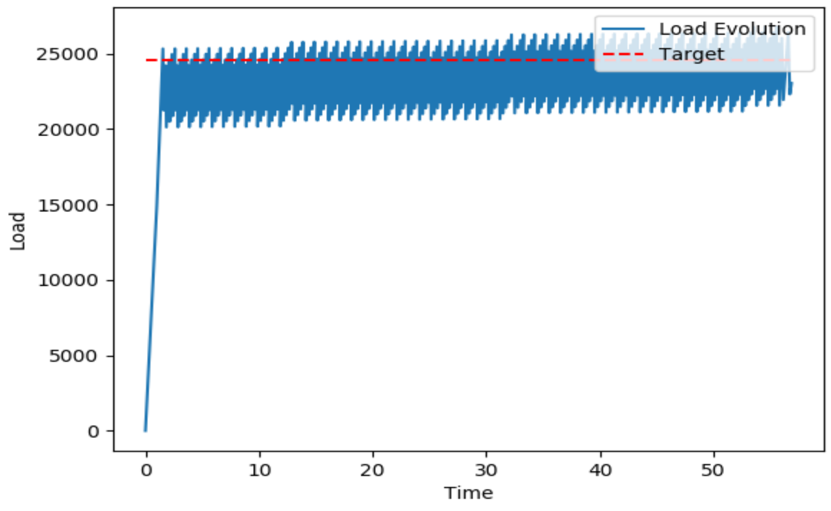

Figure 8: Player Mode: D

The figures (Figure 9 to Figure 16) are transmissions of file kline-jarrett.au from machine data.cs to pod2-1.

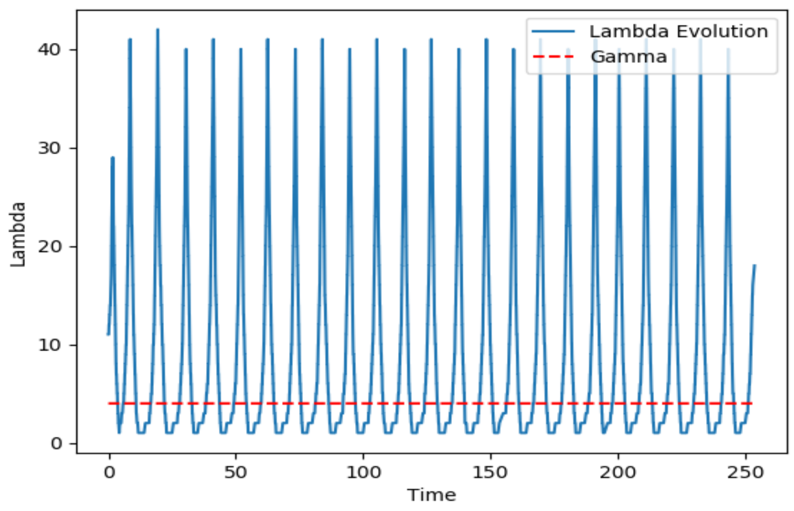

Figure 9: Sender mode: A

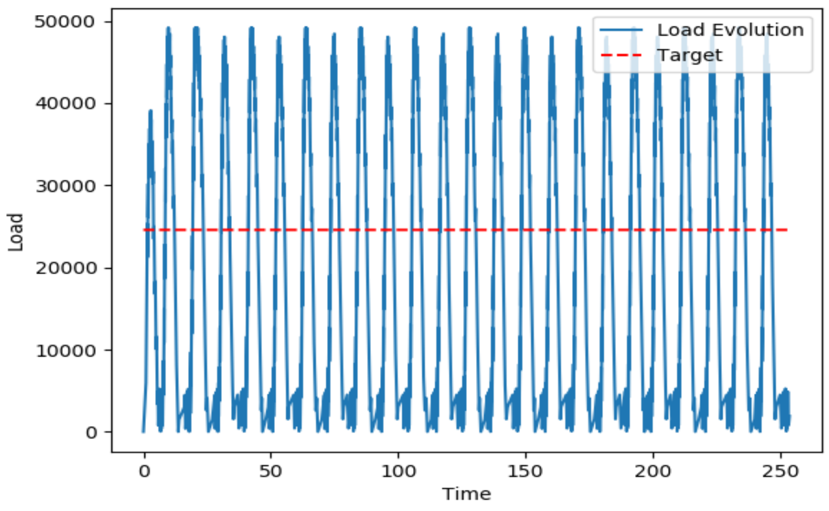

Figure 10: Player Mode: A

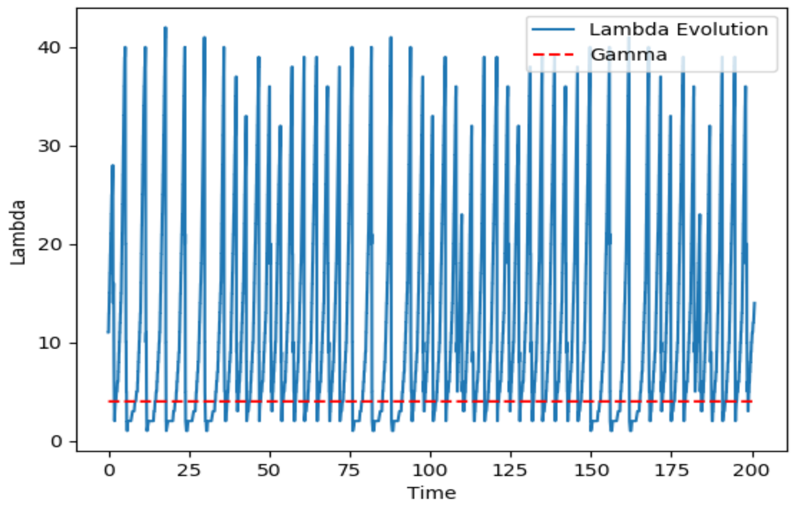

Figure 11: Sender Mode: B

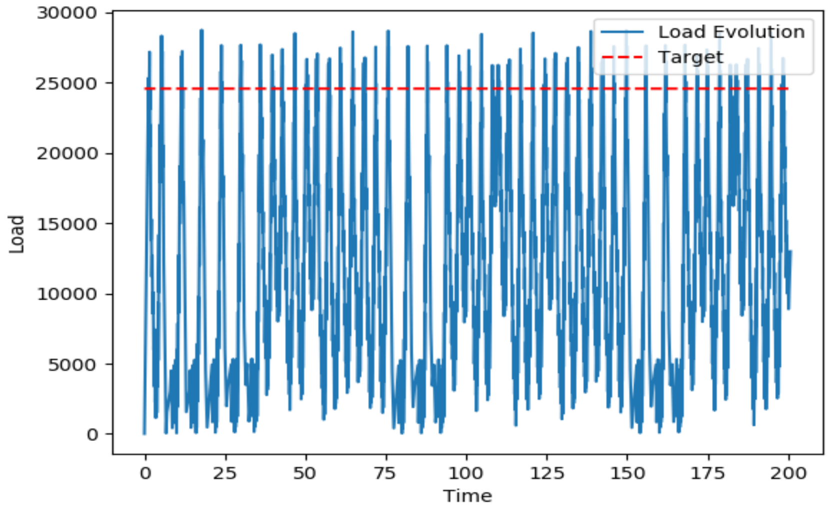

Figure 12: Player Mode: B

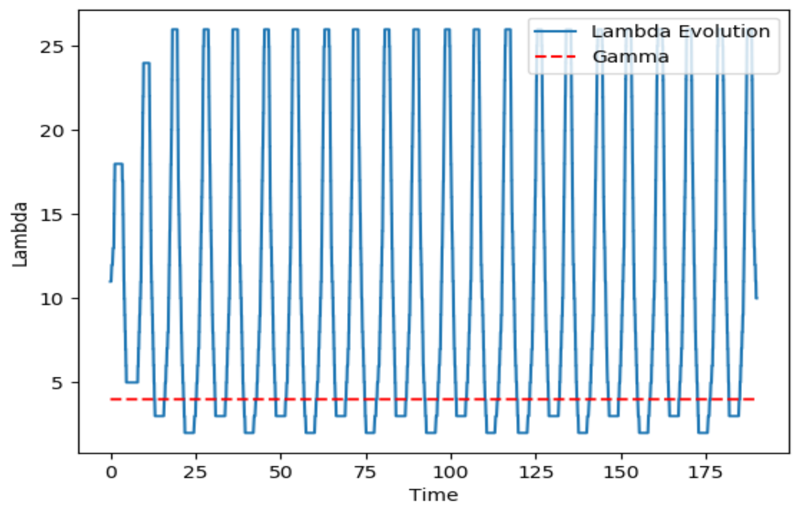

Figure 13: Sender Mode: C

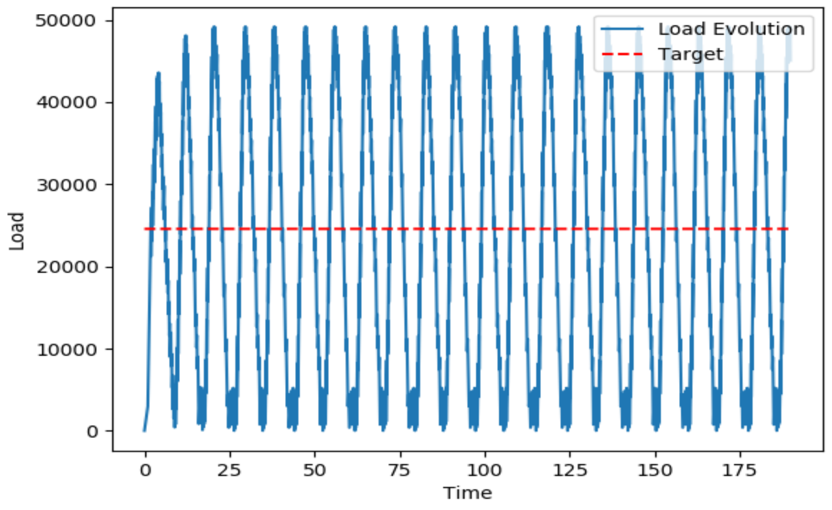

Figure 14: Player Mode: C

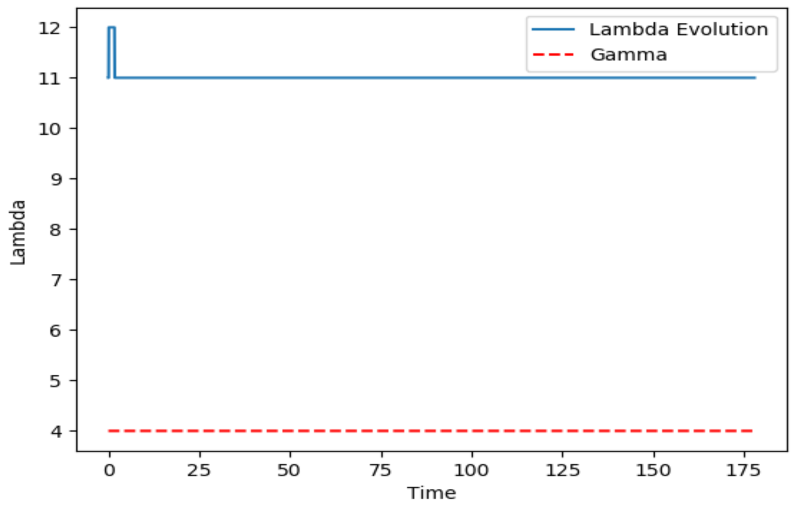

Figure 15: Sender Mode: D

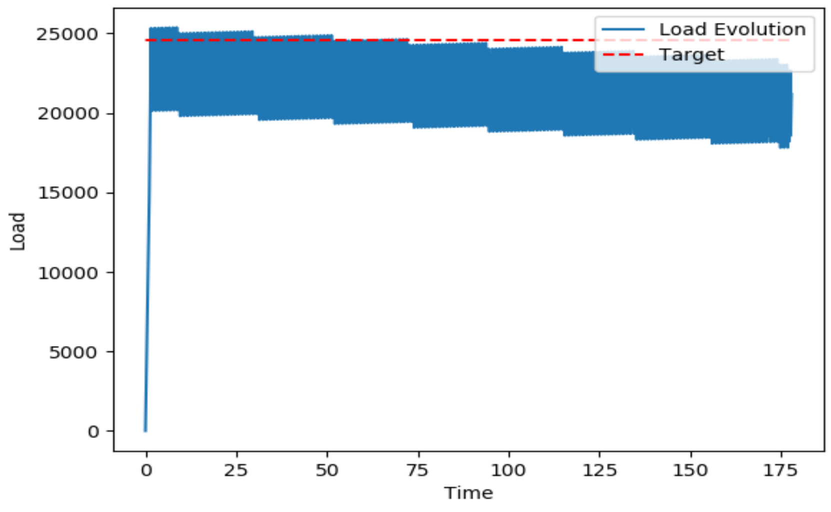

Figure 16: Player Mode: D

According to the result of four modes discussed in the lecture, our result from mode A to mode C pretty makes sense. Only the result of mode D shows a small oscillation when the load evolution converges to the target on receiver side. We think it is because of additional small latency while running other codes, such as dealing with received bytes. The latency causes the entire pace slower than the specified gamma on receiver side, resulting in a small oscillation of load around the target.

4 Compare subject audio quality perception with the numerical performance results.

Although the modes used are different, we still have a good perception of the audio.

5 Extend your benchmark by testing with 3 concurrent clients. Check if performance is impacted by additional load at the server.

The figures (Figure 17 to Figure 24) are transmissions of file pp.au from machine pod1-2 to escher02, escher03, and escher04. The parameters in this part are as the same as session 2.

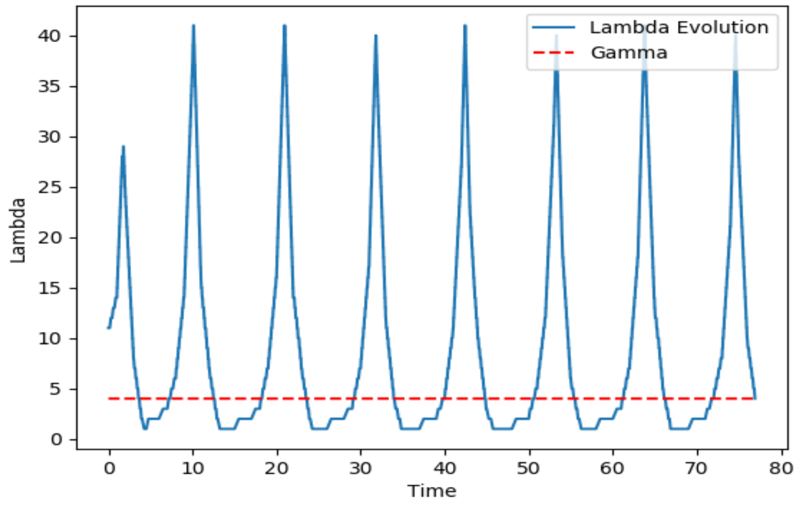

Figure 17: Sender mode: A

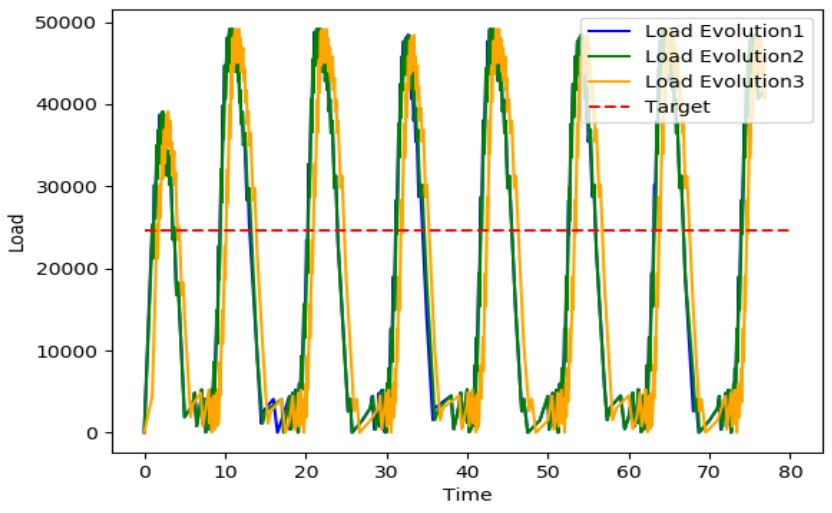

Figure 18: Players Mode: A

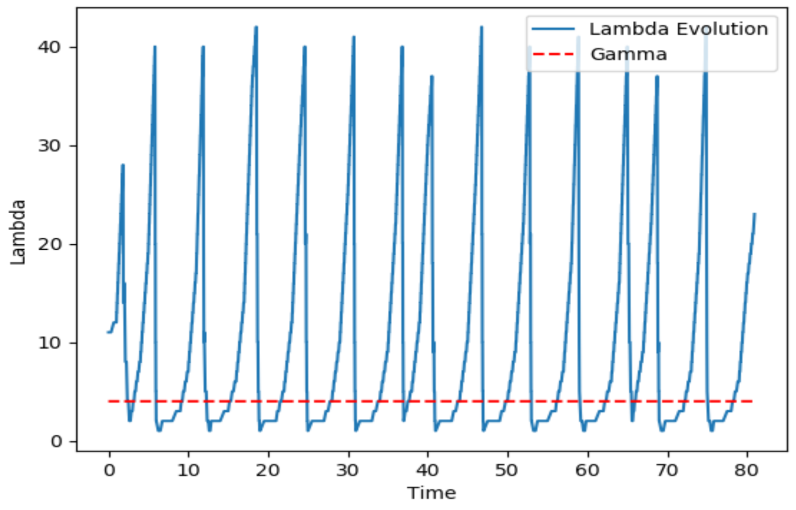

Figure 19: Sender Mode: B

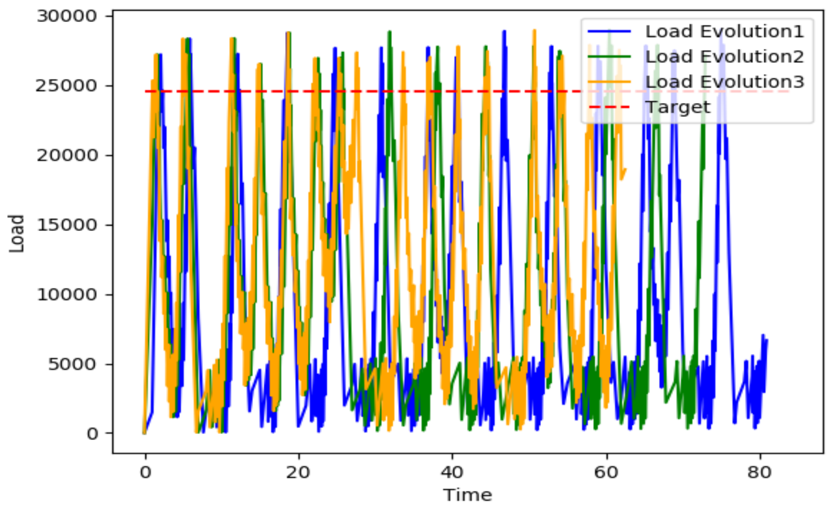

Figure 20: Players Mode: B

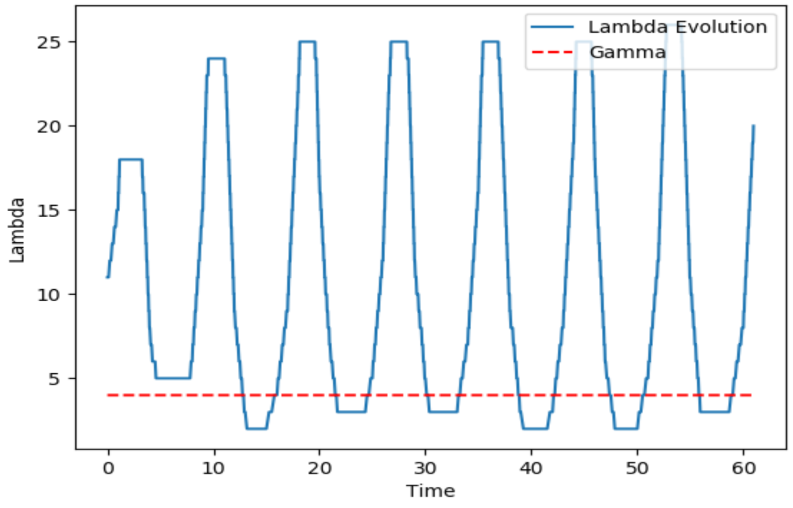

Figure 21: Sender Mode: C

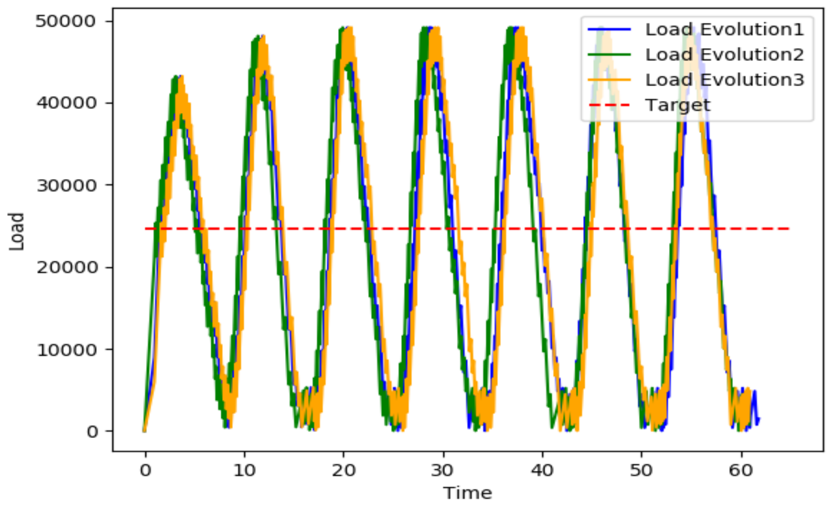

Figure 22: Players Mode: C

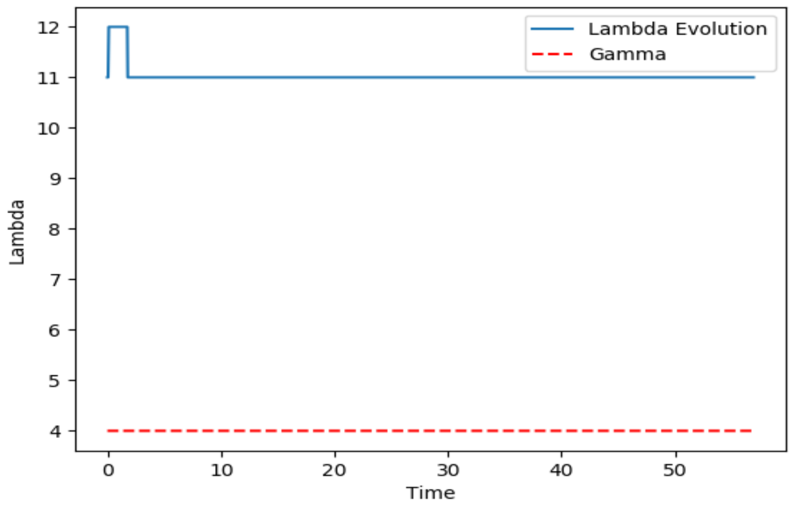

Figure 23: Sender Mode: D

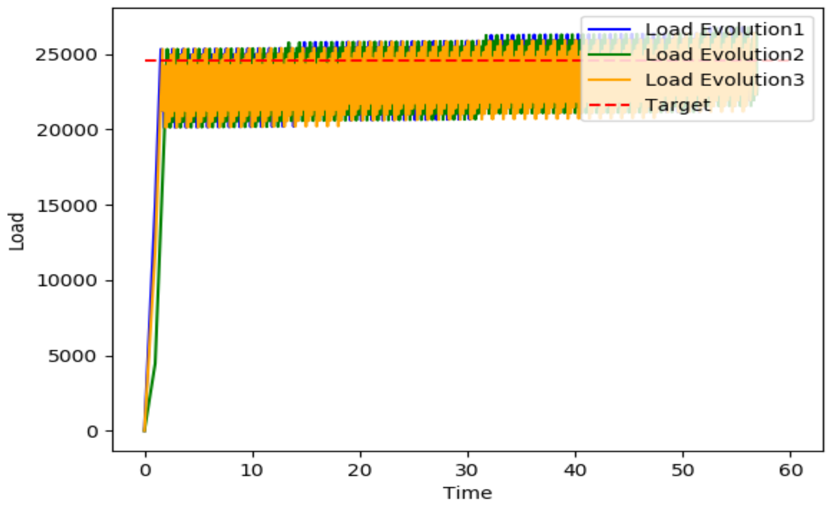

Figure 24: Players Mode: D

Based on observation, the impact of three concurrent transmissions on our benchmark is not evident, meaning that concurrent transmissions do not strongly affect the congestion control.