Laboratory 11 - Queuing system with preemption with multiple servers

Giuseppe Esposito
s302179

December 4, 2022

1 Preliminary analysis

In this laboratory, we were asked to build a simulator whose aim is to reproduce a queuing system with preemption, with multiple servers. In this system, we have 2 servers which has to serve different client arriving in the system in randomly generated instances of time. The arriving clients can be of different types: high priority or low priority. The client with high priority will be served first and the low-priority clients will have to wait until all the high-priority clients are served.

2 Assumptions

- The rationale behind the generation of arrival or departure events with different setups is the same as the one described in laboratory 9.
- The interarrival time and the service time are generated with the following setup: $\mu_{lp} = \mu_{hp} = [0.4, 0.8, 1.4, 2.0, 2.4, 2.8]$ $\lambda_{hp} = \lambda_{lp} = 1$ or $\lambda_{hp} = \frac{1}{2}, \lambda_{lp} = \frac{3}{2}$
- the distribution of the generated times follow the requested processes ['exponential distribution', 'hyperexponential distribution', 'deterministic process']
- **conflict**: I defined a conflict as the generation of a high-priority client and the 2 servers are working on the job of 2 low-priority clients.
- the queue has a fixed max-length to 1000
- when a conflict occurred, the job of the client that is in the first server is interrupted, and the time his job needs to be completed is saved as service time and the client is added at the head of the queue
- when a conflict occurred if the queue has reached its max-size, the last low-priority client is dropped from the queue.

3 Input parameters

- Simulation time: the time that the simulation has to last
- Distribution type (distribution): possible distributions by means of which I generate random instances of time.
- different parameters to generate different instances of time during the simulation both for inter-arrival time and for service time.
- Confidence level of the different confidence intervals is always set to 95

4 Output metric

he metrics under analysis are: the average delays for each level of utilization and the average delay for each batch of the plotted average delays array for each departure event

5 Main algorithm

5.1 General flow

The algorithm has the following flow: an initial arrival client is generated and his job starts, then a service time is randomly generated for that client and the time is updated. Then the clients are continuously generated with a randomly assigned label that is 'high-priority' or 'low-priority' the clients' job starts until one of the servers is idle or until a conflict has occurred. The conflict occurs when the 2 'low-priority' clients have been serving and a 'high-priority' client is generated. In that case, the corresponding server attribute of the client is set to None and the remaining service time he needs to complete his job is saved in the client object, then the high-priority just generated starts and all the high-priority clients will be served until they expire.

Unfortunately, the script does not work properly so I couldn't report the resulting graphs.