We are going to study the performance a software system that executes sessions from its users. The system is a digital platform for education. The users are the students, and the sessions are learning sessions. There are **2000** students.

The system is composed of four service centers: An ***AuthenticationServer***, a ***ContentDeliveryServer***, a ***ContentWritingServer***, and a ***LoggingServer***. All the service centres have only **one** **resource**. We are interested in knowing the **average** **service** **time** of each service center, but we could not directly measure it. However, we have been able to measure the following:

* In the system, a learning session finishes, on average, every **0.3956 seconds** (hint, this is the System Throughput).
* Each students’s learning session executes once in the *AuthenticationServer*, which checks whether the session comes from an authorized student of the platform. We have seen that the AuthenticationServer has been busy the **0.8%** of time.
* After the execution in the *AuthenticationServer*, the learning session executes in the *ContentDeliveryServer*. An execution of the *ContentDeliveryServer* delivers a chunk of educational content in a webpage. The service time of the *ContentDeliveryServer* is **300** **milliseconds**. After each execution of the *ContentDeliveryServer*, the student spends **5 minutes**, on average, reading the delivered material. We call this group of two activities (getting content from the *ContentDeliveryServer* and reading it) a "learning chunk". On average, a student **iterates three times** on this "learning chunk" before moving to any other activity. After the three "learning chunks" iterations, two things may happen:
  + Alternative 1) The student may want to finish the session in the educational platform. This happens **25%** of times. In that case, the execution goes to the *LoggingServer* which saves that the user has been connected to the platform and the amount of time connected. After that logging operation, the student's learning session finishes. We have observed that there were, on average, **0.0414 jobs** in the *LoggingServer*; that the average time between a job arrived at the *LoggingServer* and its execution was completed (including the time that it spent waiting for service in a queue) was **0.1046 seconds**; and that its utilization was **3.956%.**
  + Alternative 2) The student writes a comment or question in the educational platform about the content that he/she has read. This happens the remaining **75%** of times. This operation executes in the *ContentWritingServer*. The demand of the *ContentWritingServer* per user session is **0.48 seconds**, on average, and its throughput **0.4747**. After executing in the *ContentWritingServer*, the student finishes the learning session **50%** of times (in that case, the execution goes to the *LoggingServer* and the session finishes), or he/she returns to the *ContentDeliveryServer* to continue with the "learning chunks" iterations (the other 50% of times).
* When a student finishes a learning session, the student leaves the platform for 1 hour, on average, and he/she does other things. After that hour, the student returns to the platform for another learning session.

A) Use the operational laws to calculate the average service time Sk of the *AuthenticationServer, ContentWritingServer*, and the *LoggingServer*

B) Model the system that has only one resource in each service center using Queueing Networks (in JMT or your preferred Queueing Network simulation engine). Take a screenshot of the model (ONLY one screenshot with the queueing network model; do not include the value of the parameters). Simulate the model to calculate the system **response time**, the **utilization** of each service center, and the **throughput** of each service center. Use the simulation results to calculate how much time a student is **waiting on average for a reply** from the system during a complete learning session (clarification: the student is not waiting for a reply from the system when he/she is reading the content provided by the *ContentDeliveryServer* or when he/she is not in a learning session. The student is waiting for a reply from the system when the execution is any of the other service centers).

Hint1: In the cases that, from a service center (e.g., *ContentWritingServer*) a job can go to more than one service center, use Probabilistic Routing.

Hint2: Use the exponential distribution for all times and rates (frequencies) you need to model.

Hint3: When a request iterates X times in a group of service centers, you can model it adding an additional loopback arc (see slide 4 in document 2DV608-Performance2.pdf), having the loopback arc probability (X-1)/X .

C) Assume that most of the education moves online due to a pandemic that closes the campus. In that case, more courses will be published on the digital platform for education and, therefore, we expect that the number of students that will use the digital platform will double (from 2000 to 4000). Moreover, we also expect that online learning becomes more intensive for each student, causing that the average number of iterations in "learning chunks" before moving to the next activity increases from 3 to 5.

 In that situation, you suspect that some components of your system will saturate.

Discover the minimum number of resources of each server that we will need to avoid saturation (to discover these values, you can use the results from different simulations of the queueing network and calculations).

Simulate the new system to calculate the system response time, utilization of each service center, and throughput of each service center.

**Bonus points** (10%) (you need some previous knowledge on probability and statistics for this): The information about the *ContentWritingServer* throughput=0.4747 is redundant. You could have obtained the same value "0.4747" using the rest of the information in the exercise (but it is not easy). Show how the "*ContentWritingServer* throughput=0.4747 " can be calculated from the rest of the data to get 10% extra in your assignment.

Good luck!

Hint1: In the cases that, from a service center (e.g., *Database*), a job can go to more than one service center, use Probabilistic Routing.

Hint2: Use the exponential distribution for all times and rates (frequencies) you need to model.

Hint3: Use the figure above, the structure and probabilities are the same.

Hint4: If the question about *“how much time a user is waiting”* does not sound familiar to you, check slides 24-25 in Part 3. We calculated the proportion of time that a user is waiting in that exercise, which is something similar to what it is asked here.

1. Use the **operational laws** to calculate the service time **Sk** of the **WinnerPaymentServer**, the **Database**, and the **BettingServer**; and to calculate the **utilization** of the **Webserver**.
2. Model the System using Queueing Networks (in JMT or in your preferred Queueing Network simulation engine). Add screenshots of: the **structure of the network** and about all the information you add to each component (**service times**, **routing probabilities**, etc.). Simulate the model to calculate the **System Response time**, the **Utilization** and **Throughput** of each of the five components in the system and show screenshots of the results.
   1. In the cases that, from a service center (Webserver) a job can go to more than one service center, use Probabilistic Routing.
   2. Use the exponential distribution for all times and rates (frequencies) you need to model.
   3. When a request iterates X times in a group of service centers, you can model it adding an additional loopback arc (see slide 4 in performance 2), having the loopback arc probability (x-1)/x and the arc that leaves the service center probability 1/x.
3. Upgrade the functionality of the system in order to gain the attention of more users. The upgrade is the following : The **PlayerEngagementServer** will check more championships in which a user can bet and, therefore, it will iterate 4 times its execution in average (with its corresponding request to the Database in each iteration), instead of the previous 2 times. With this upgrade we expect that the number of users that are interested in our betting system will double; that is, the system will receive 180 sessions per minute, instead of the previous 90 sessions per minute. Your intuition makes you think that these increments in the number of users and in number of iterations for each user could saturate the PlayerEngagementServer and the Database. Therefore,
   1. Use the operational laws to calculate whether any of these two servers saturate.
   2. If any of them saturate, use the operational laws to calculate the minimum number of parallel servers for the PlayerEngageentServer and the Database that will be necessary to avoid system saturation.
   3. Update your previous Queueing Network model in 2 with the new number of iterations, new number of users per minute, and new number of servers for the PlayerEngagementServer and the Database. Simulate the model to calculate the System Response time and add a screenshot of the result.