5 complete days = 5 x 24 = 120 hours = **432000** s

**648000** user sessions have been completed

Sessions arrived at a rate of **90** sessions per minute

* Each user’s session starts executing in the ***WebServer***. There is only **one** resource for the *WebServer*, whose **service time** is **30ms** in average. After the *WebServer* executes, the user session can follow two different paths, depending on whether the user is the winner of a previous bet or not.
* If the user is a winner of a previous bet, which happened **25%** of times, the user executed in the ***WinnerPaymentServer*** and credit is transferred to the user’s account. The **utilization** of the *WinnerPaymentServer* is **37,5%.** After transferring the credit, the session continues in the ***PlayerEngagementServer***, which offers new betting possibilities to the user in events that will happen in the next **30** **minutes**.
* If the user is not a winner of a previous bet, which happened **75%** of times (because the user lost the previous bet or because the user did not bet the last time), the user session moves directly to the ***PlayerEngagementServer****.*This server checks all the possible bettable events in a championship that will be resolved in the next **30** **minutes** and generates lists with the events that are most likely to captivate the user to do a new bet.
  + A user session iterates, in average, 2 times its execution in this *PlayerEngagementServer.*server (because an average user is interested in 2 different championships).
  + Each iteration of this kind of “bet recommendation” service needs a lot of CPU to discover the preferences of the user. The service time of a job in the *PlayerEngagementServer*are 400ms and it has 2 servers working in parallel*.*
  + Each iteration of the *PlayerEngagementServer*also needs to perform a request to the *Database*to retrieve the data about the existing bettable sport events in a concrete championship. We have observed that the demand (Dk) of a user session in the *Database*are 120ms.
* After executing in the *PlayerEngagementServer* , the user may have been convinced to bet on one of the elements in the list, or he/she may leave the system (we have not been able to directly measure the frequency with which users take any of these two options). If the user has been convinced to bet, the session continues its execution in the *BettingServer*.
* The *BettingServer* executes the bet of the user in the system. There is a single resource for executing the *BettingServer.*We have not been able to directly observe the service time of the *BettingServer.* However, we could observe that there were, in average, 0.265 jobs in the *BettingServer*; that the average time between a job arrived to the *BettingServer*and its execution was completed (including the time it spent waiting for service in a queue) was 0.2525 seconds, and that the utilization of the server is 20.91%

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.