MERRY GO ROUND PROJECT

Performance Evaluation of Computer Systems and Networks

Bruk Tekalgne, Leonardo Fontanelli

UNIVERSITÀ DI PISA

Contents

OBJECTIVES

Considering a Merry Go Round, that has **N** seats, where children pay one coin for a ride that takes **T** units of time.
The children come to the Merry Go Round and queues up waiting for a ride. The owner may decide to start a ride even if the Merry Go Round is not full, the children may decide to drop out from the queue with an increasing probability after a threshold time **Q**.
Regarding the interarrival time of the children to the MGR, there are two kind of scenarios that has been taken into account: the exponential distribution of interarrival times with single arrival and the burst arrival, where the interarrival are still exponential, but the number of children in one arrival is geometrically distributed.
So, the main objectives of this project are to model the system with the software Omnet++, implementing the behavior of the MGR given the factors defined from the request. Choose reasonable value for the parameters. Evaluate the earnings per unit of time of the MGR, depending on the threshold he/she decides to start a ride on the MGR and the threshold **Q**.

MODELLING AND CODING

MODELLING

According to the design theory of the software Omnet++, we modelled the Merry Go Round with 3 modules: Entry, MGR, Exit.

- **ENTRY**: The Entry module is responsible of generating children and sending them to the MGR module.
  According to the requests, we defined two scenarios for the interarrival of the children: Exponential interarrival and burst arrival (still exponential interarrival between two arrivals).
- **MGR**: The MGR is a compound module, which is composed by 2 submodules: **Owner** and **IN_MGR**. The first one is the handler of the events in the Merry Go Round, receives notification at every child arrival, decide when is the time for a ride and collects the money earned.
  The second one is the core of the MGR, it has the queue for the children that are waiting for the ride, and N seats for the ride.
- **EXIT**: The Exit module only receives children that have enjoyed the ride in the Merry Go Round.

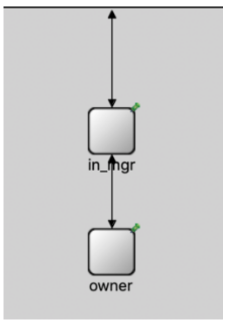

Figure 1 Inside MGR compound module

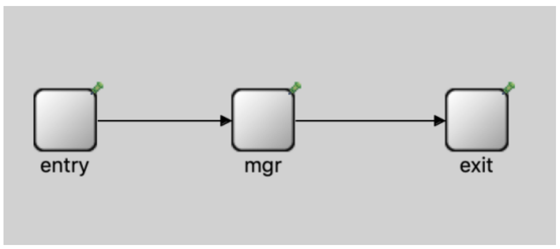

Figure 2 General Omnet Model

CODING

We used Omnet++ 5 to design and implement the system and then simulate the requested behavior. The code has been organized having three parts Entry module, Exit module and Mgr module. The latter one is compound module containing two sub modules In_mgr and owner module. Each of them are described as follows:

- **ENTRY MODULE:**
    - **initialize():** initialize a self-message `new cMessage("starter");` and pass it to the `scheduleAt(simTime(), msg);` for starting the arrival function of the children to the MGR. Here a signal is registered to collect the number of children generated each time it generates a child or children. Its purpose is to kind of see the load balancing in the exponential interarrival and burst arrival over time.
    - **handleMessage(`cMessage` *msg):** when the self-message is received, it calls the function GenerateChild(msg);
    - **GenerateChild(`cMessage` *msg):** this function is responsible of creating the children, that are modelled as an extension of `cMessage`, with the definition of the new field "**int waitingThreshold**" that indicates the remaining time patience of the children in the queue.
      We were asked to design two kinds of interarrival, one is the exponential and the second is the burst (still exponential, but the children arrive in groups), hence we defined a variable **"burstArrival"** in the .ini file in order to switch from a kind of interarrival to another with the boolean value specified in the configuration file. In the burst arrival (when the "burstArrival" variable is set to true from the configuration file), we used the function **"geometric(p, rng);"i.e. rng = 3,** to generate the number of children, geometrically distributed, in one arrival. For the interarrival time (for both cases) we used the function **"exponential(mean, rng);" i.e. rng = 2,** for generating the exponential distribution of interarrival gap.
      Regarding the waiting threshold of the children, is composed by two parts: one is fixed and it is the **Q** time, which is a parameter from the .ini file specifying the minimum waiting time of a child, the second one is represented by the probability that the children drop out from the queue, and in the code we represent it by the function **"exponential(mean, rng);"**. The summation of the two parts will be the resulting **waitingThreshold** of the children.
- **IN_MGR MODULE:**
  As previously said, this is a submodule of the compound module MGR.
    - **Initialize():** initialize the **seat_num, ride_time** variables with the values set in the .ini file. Initialize a self-message `new cMessage("timer");` and pass it to the `scheduleAt(simTime() + 1, timer);`, in this way we'll have a timer for a continuous check of the threshold of the children at fixed duration of time.
    - **handleMessage(`cMessage` *msg):** in this function we have many choices depending on which message we receive and the related event.
      For a message named **"kid"**, the event is a new arrival of a children in the MGR and we call the function **"queueUp(msg);"** to take and store it in the **childrenLine** , queue data -structure used to simulate the queue of children waiting.
      For a message **"start"**, the start signal from the owner, in this case the ride must start, and we call the **"startRide(msg);"** to handle it.
      For a message **"EndRide"**, which is a self-message coming from the function

"startRide(msg)", we need to stop the ride of the MGR and we call the **"endRide(msg)"** to do it.
For a message **"timer"**, initially created from the inizialize() function, we call **"checkThreshold()"** and we schedule again the timer with `scheduleAt(simTime() + 1, timer);` for the next check.

- **queueUp(**`cMessage` ***msg):** with this function we put in the FIFO queue (represented by a cQueue childrenLine) the children arrived to the MGR that are waiting for a ride. And we send a notification to the Owner module that a new child has arrived.
- **startRide(**`cMessage` ***msg):** we've received the "start" message from the Owner Module. Given that we have **N** number of seats in the MGR, with this function we take **N** children from the queue (or everyone if there are less than **N** children) and we put them in the MGR (represented by a cQueue sits). We create a selfmessage **"EndRide"** and we use it as a timer to trigger the end of the ride and to call the relative function "endRide(msg)";
- **endRide(**`cMessage` ***msg):** when this function is called, the event that we want to handle is the end of the ride of the MGR. So now the children have to come out from the "sits" queue. We do this with a "for" iteration and then we notify the Owner module that the ride is over with a `new Ctrl_M("done");` in this of message we store the number of children that have enjoyed the ride as a payment done to the owner.
- **checkThreshold():** this function is periodically triggered by the timer started in the initialize function. With a "for" iteration we scan the queue of the children, we decrement the field of the waiting threshold and if they reach zero, they are discarded from the queue. And, if there at least one exclusion from the queue, the Owner is informed of this event with a message. Also the new updated number of children in the queue.

- **OWNER MODULE:** This is the second submodule of the compound module MGR.
  - **initialize():** initialize the owner's minimum sit number threshold with the one specified in the .ini file. We register the signal of the "earnings per unit of time", which is the one that gives us the performance statistic of the MGR. We create a new timer with `new cMessage("takeStat");` and we put `scheduleAt(simTime() + 10, msg);` in order to take the statistic every 10 seconds.
  - **handleMessage():** depending on the kind of message that we receive, we select a different choice to handle the event. When the Owner module receives a `"C_arrive"` this means that a new child has come, so we update the number of children stored that are waiting for the queue and we call the **sendStartMessage();** to see if there are condition to start the ride of the MGR.
    When the Owner receives a `"done"` message the ride of the MGR is over, so now we have to collect the money of the children that have enjoyed the ride and enable again the Owner to give again the "start" signal.
    The `"takeStat"` message, as previously said, is the one that gives the possibility of storing the earnings per unit of time and emit them for the statistical analysis.
    We emit the statistic storing two variables **money** and **lastMoney**, the latter will be the quantity of money collected in the before "emit", in this way we obtain the earning per unit of time.
  - **sendStartMessage():** in this function, first of all we check if there are the condition to start a ride: that there isn't a ride in progress and if is reached the minimum number of

sits for the owner’s threshold. If the conditions are satisfied, the ride can start and we send a **new** Ctrl_M("start"); to the IN_MGR module.

FACTORS DEFINITION

In this part we’re going to define the factors: the variable that we put under study or that are going to affect our model.

- **Number of seats (N):** This is the number of seats that has the Merry Go Round. So that in every ride, N children at maximum can sit. We fixed this value to 10 for every simulation.
- **Ride time (T):** This is the time (in seconds) that the MGR needs for completing a ride. We fixed this value to 5s.
- **Children’s Threshold (Q):** Represent the minimum patience of the children expressed in seconds. In total the patience of the children is composed by Q and a random overhead value coming from the “exponential” function.
- **Owner’s Threshold (P):** Represent the patience of the Owner before starting the MGR, even if is not full. We modelled this patience as the minimum number of children. For example, given 10 seats, if P = 6, the Owner will start the ride if there are at least 6 children in the queue.
- **Mean Interarrival (“meanInterArrival”):** it is the number given to exponential random number generator as an average value around which the generated numbers are going to be distributed. This one is not specified in the question, but it is of great significance as we will discuss it latter in calibration.

VALIDATION

The purpose of this part is to “validate” the model, so that the result obtained from the simulation can be considered reliable. In order to validate the model, given that we have the exponential distribution both from the interarrival and the children’s patience, computing the results manually to see if there are confirmations in the simulation, it’s very difficult. So our choice has been to set some “stress” values of the factors from the configuration file, to see if the behavior is the one that we expected.

- **FIRST TRIAL: QUEUE**
  The first configuration is to see if every generated child, queues up correctly at the MGR. So, the idea is to put very high values for the Q time and very high value for the threshold of the owner (P > 10), in this way the ride will never take place. The purpose of this trial is to see every child queueing up to the MGR.
- **SECOND TRIAL: DROP-OUT**
  The second configuration is to try to force the dropout of the child, to see if the children drops out correctly from the Playground. The purpose of this trial is that every generated child, will have an expiry of the threshold and will drop out from the queue. In order to obtain this, we put very high value of owner's threshold (P>10, like before), and very low Q time patience of children. The behavior of the model confirms that every child after a certain time, drops out the queue.

- **THIRD TRIAL: EVERY CHILDREN SERVED**
  The third configuration is to have every children served by the MGR, to check the correct behavior of the owner, and to check that every children will reach the exit module.
  So, to obtain this behavior we put very high Q time patience of the child and the owner's threshold equal to the number of the seats (P = N = 10).
  Running the simulation, we can see that every child, after a certain amount of time, will be served by the MGR and will reach correctly the exit.
- **MONEY COLLECTION**
  This is only to confirm the correct behavior of the money collection from the owner.
  In this way, we don't need to impose any particular configuration from the .ini file, we only need to check at the end of the simulation period if "money = generatedChild – childrenInTheQueue – childrenDropped – occupiedSits"

These trials have been repeated also with the "burst" scenario, for a confirmation of the correct behavior of the model.

CALIBRATION OF THE SIMULATOR

Mean Inter Arrival time

This value has a significant effect on the results we measure. Our region of interest is where the following condition holds.

$$\text{Mean interarrival} * \text{Sit Num} \sim \text{ride\_time}$$

To put it in simple terms, the average number of children arrived at the queue equals the total sit number in the MGR. this condition allows us to see the effect of minimum child patient and owner sit number threshold in the system. So, we choose the Mean Interarrival to 0.5 second. We check our theory and observe that it is correct like the following.

I) When we set mean interarrival a small number (in relative context) like 0.1s regardless of the value of child patience or owner minimum sit threshold the earning per unit of time is ~1.999 dollar per second. It is difficult to obtain variations by varying the children patience and owner minimum sit number threshold.

II) When we set mean interarrival a large number like 2.5 second we get the earning per unit of time is < 0.5 dollar per second regardless of the value of patience and owner minimum sit threshold. Even if we increase the children threshold to significantly large value, i.e. 100 sec, the earning per unit of time remains below 0.5.

But when we set the value of MeanInterarrival to 0.5 second we get ranges of values from ~0.25 to ~1.95 by manipulating the children patience and owner threshold.

TEST DEFINITION

We defined 4 different tests, in order to compute the performance of the model with different configurations of owner sit threshold and children patience. As already defined, the objective of this work is to evaluate the earnings per unit of time, varying the **Q** (children) and **th** (owner) thresholds. Hence the test will differentiate each other for different values of the thresholds.

The purpose is to mix "low" and "high" values of both and to compare the contributions of changes of these two properties in the system. Also address if there is an interplay between the two factors and also the contribution of changing these parameters.
For both exponential interarrival and burst arrival with exponential interarrival, we fixed the values of the sits N = 10, and the ride time T = 5s. Those values will be the same for every simulation and for both scenarios. The test cases and parameters below are common for both settings.

- Test 1: for this we set Q = 2s, th = 1.
- Test 2: in this configuration we set Q = 2s, th = 10.
- Test 3: in this one we set Q = 10s, th = 1.
- Test 4: in the last one we put Q = 10s, th = 10.

Note: we set the minimum child patience to 2s. In the algorithm we implement decrement and check if patience time is below 0. So we intend to make the child wait at least 1 sec before leaving the playground.

DURATION SELECTION

We ran the model with different duration for every test, we computed the sample mean and the sample variance for each one.
In this case we choose for the analysis the Test 1 because it shows large variation in the end result. From this table came out that the best duration is 75000, because it has the best value for the variance. So from now on, we'll use that value for our simulations.

Table i: Test 1 duration analysis

duration	15000	23000	30000	38000	45000	60000	75000
Run ID							
0	0.8842	0.878	0.8764	0.8785	0.872911	0.879683	0.251033
1	0.86893	0.874826	0.8835	0.882289	0.890022	0.880367	0.249433
2	0.88527	0.878696	0.871933	0.883474	0.893667	0.88935	0.248767
3	0.92933	0.913391	0.9056	0.903763	0.900956	0.89605	0.239568
4	0.88347	0.874304	0.8694	0.865316	0.869689	0.872417	0.241434
5	0.89073	0.888217	0.900433	0.905184	0.897022	0.898983	0.236235
6	0.8884	0.872565	0.866233	0.864763	0.856978	0.86785	0.233169
7	0.89713	0.891435	0.883667	0.883684	0.877289	0.895883	0.2513
Average	0.8909	0.8839	0.8821	0.8834	0.8823	0.8851	0.2439
variance	0.0003	0.0002	0.0002	0.0002	0.0002	0.0001	0.0001
CI _ L 99%	0.8906	0.8837	0.8819	0.8831	0.8820	0.8849	0.2438
CI _ U 99%	0.8913	0.8842	0.8824	0.8836	0.8826	0.8852	0.2439

Warmup period

The time for selection the range of period that the system will reach steady state is done based on the convergence of values taken from each run ID. Similar to the above case we select the case with the most varying earning per unit of time i.e. Test 1. As shown in the graph below the value tends to get close to the aggregate in 15,000 seconds. For safety we decided to take 20,000 second as a warmup duration. Adding the duration of 75,000s we select above we run the experiment with a total duration of 95,000s.

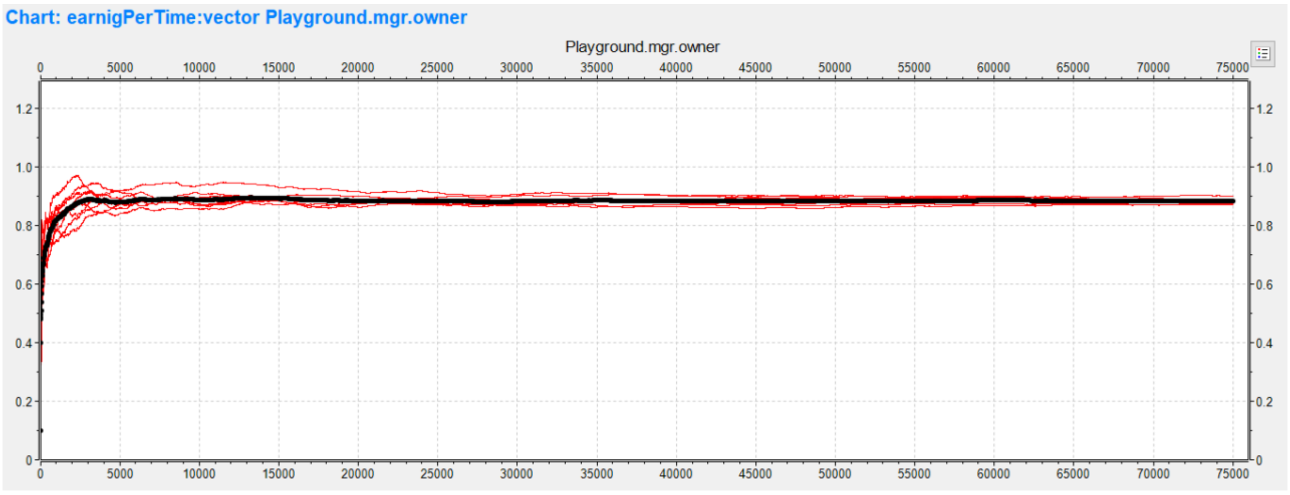

Figure 3 warm up duration for test 1

Result obtained

We run the simulation and obtain the results which we are going to discuss below each with in a separate topic and combine them at the end.

1 EXPONENTIAL INTERARRIVAL (individual arrival)

We gathered the earning per unit of time of each runId of test scenarios as shown in the table below. We call it individual arrival because one child arrive at an instant.

Table ii: Results for individual arrival with exponential interarrival

configs	Test 1	Test 2	Test 3	Test 4
runs				
0	0.892654	0.251033	1.918118	1.924943
1	0.888042	0.249433	1.92809	1.932676
2	0.894707	0.248767	1.92813	1.926276
3	0.87451	0.239568	1.920331	1.92641
4	0.880523	0.241434	1.921317	1.927076
5	0.894147	0.236235	1.92625	1.925343
6	0.871124	0.233169	1.916478	1.923077
7	0.881243	0.2513	1.92729	1.933076
average	0.885	0.244	1.923	1.927
variance	0.000	0.000	0.000	0.000
StDev	0.009	0.007	0.005	0.004
CI L 99%	0.874	0.235	1.918	1.923
CI U 99%	0.895	0.252	1.929	1.932

The confidence interval above is calculated based on the assumption that the values are normally distributed around the mean. Furthermore, we tested this assumption by plotting the QQ plots for each scenario. Below we present QQ plot for Test 1. Also each result is an obtained in an Independent and Identical process.

Table iii: QQ plot inputs for test 1

id	results	normal quantiles	residuals	sorted residual
1	0.892654	-1.53412	0.008036	-0.01349
2	0.888042	-0.88715	0.003423	-0.01011
3	0.894707	-0.48878	0.010089	-0.0041
4	0.87451	-0.15731	-0.01011	-0.00338
5	0.880523	0.157311	-0.0041	0.003423
6	0.894147	0.488776	0.009529	0.008036
7	0.871124	0.887147	-0.01349	0.009529
8	0.881243	1.534121	-0.00338	0.010089

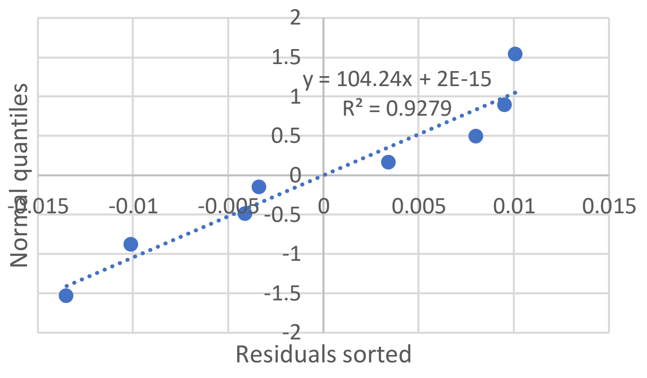

Figure 4: Normality QQ plot for Test 1

Impact analysis

We have analyzed the impact of changing the owner minimum sit and child min patience values and the interplay between the two as shown below.

Table iv: average earning obtained with respective condition

		-1	1
		owner th sit	
	Child pat. Sec	1	10
-1	2	0.8846187	0.243867
1	10	1.9232502	1.92736

Table v: impact and interplay analysis

I	O th	Chi pat Q	both	earning	(Ei - E')
1	-1	-1	1	0.8846	0.130
1	1	-1	-1	0.2439	1.002
1	-1	1	-1	1.9233	0.460
1	1	1	1	1.9274	0.466
4.979096	-0.636642	2.722124	0.644861	sum of ys	2.058
1.24	-0.16	0.68	0.16	qi=total/4	
4*qi~2	0.101	1.852	0.104		
percentage	4.924	90.024	5.052		

From the analysis we see that 90% of the increase is due to the increase in the children patience and the owner minimum sit threshold number is very nearly equal to the combine effect of both factors which is 5%. The negative sign at owner th indicates the decrementing effect of increasing th on earning per unit time.

Visualizing the obtained result, we plot the graphs below based on the above data.

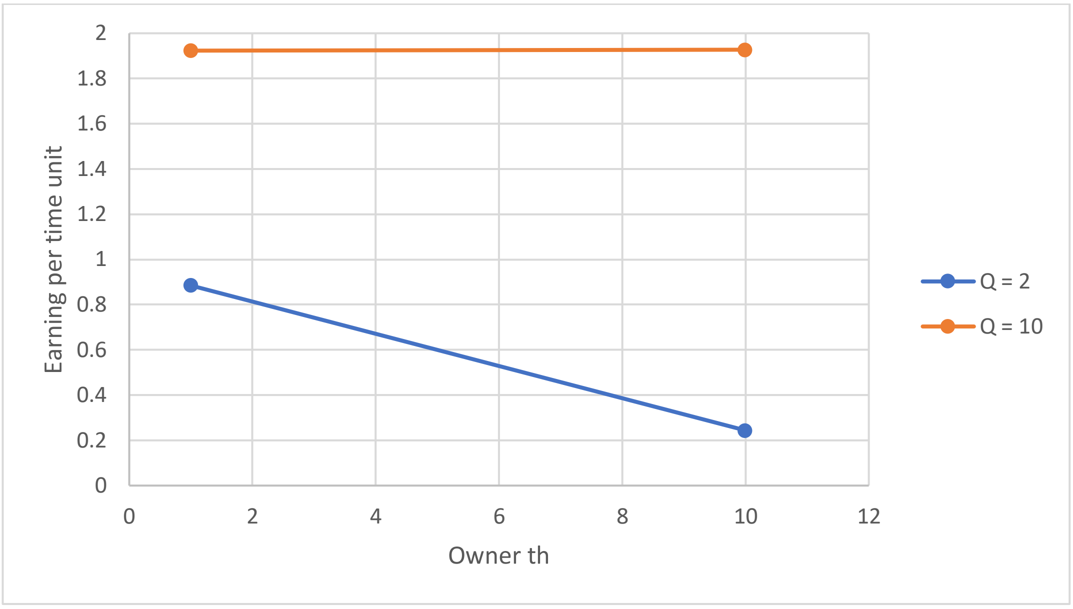

Figure 4 fixed minimum child patience plot

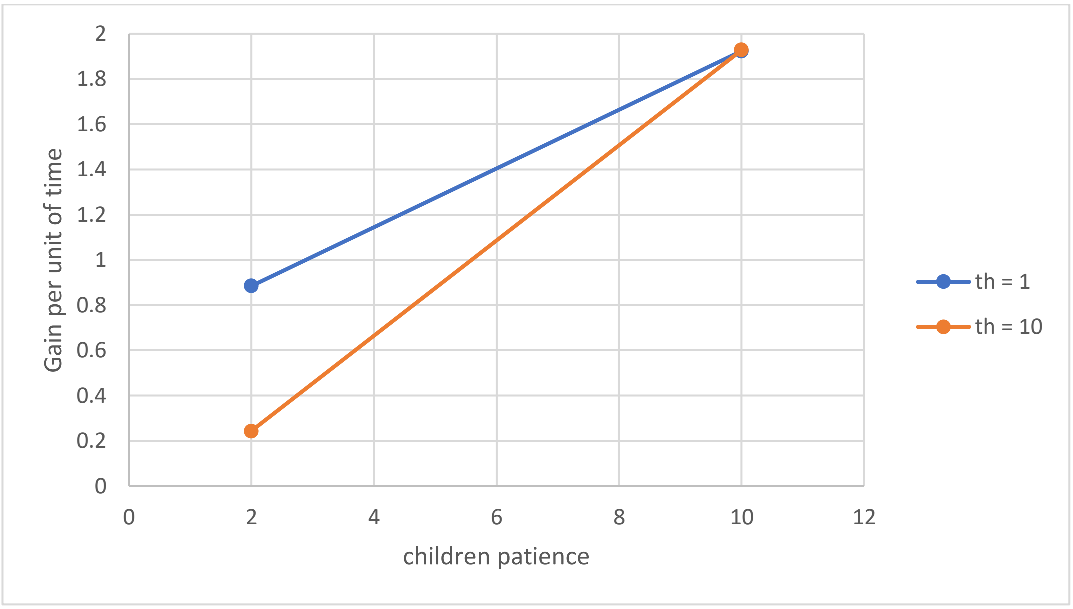

Figure 5 fixed owner minimum sit threshold plot

Our observation is as long as the children patience is high the impact of the owner changing the minimum sit threshold increasing or decreasing the th will have insignificant effect on the earning per unit time. But when the children patience in going low the earning can be kept high by decreasing the minimum sit threshold. As depicted in the graph of ***Fig 5,*** earning per unit time is almost equal for th of both 1sit and 10sit at minimum patience of 10 second. Then they diverges when the child patience goes down from 10s to 2s. Eventually 1 sit will have higher earning per unit of time that 10 sit minimum.

2 BURST ARRIVAL

Before going in to result discussion let us first find a way to equate the two cases.

Load Balancing

In this case the number of children arrived at an instant is no longer 1. Instead, the number is geometrically distributed. In order to balance the load and make the average number of children generated at an instant equal to that of exponential interarrival case. We made the p value equal to 0.5. We have taken the statistics of the number of children generated and plot for the duration as in the figure 6 below. The red line is the mean of the value generated.

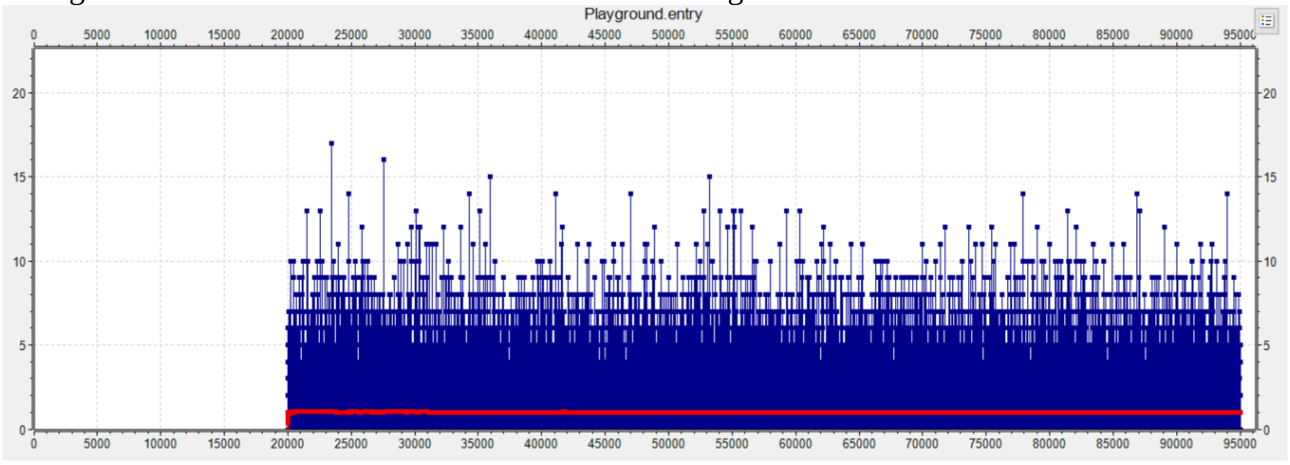

Figure 6 Burst arrivals and their mean

Result collected

We gathered the earning per unit of time of each runId of test scenarios as shown in the table below.

configs	Test 1	Test 2	Test 3	Test 4
runs				
0	0,822717	0,586988	1,81878	1,80193
1	0,826836	0,584589	1,82096	1,80857
2	0,837102	0,579536	1,82662	1,81072
3	0,837742	0,565391	1,81612	1,81439
4	0,825117	0,572724	1,81392	1,80636
5	0,825277	0,580323	1,81448	1,80108
6	0,823144	0,580589	1,81400	1,80383
7	0,82365	0,569791	1,81420	1,80251
average	0,828	0,577	1,817	1,806
variance	0,000	0,000	0,000	0,000
StDev	0,006	0,007	0,005	0,005
CI L 99%	0,820	0,569	1,812	1,801
CI U 99%	0,835	0,586	1,823	1,812

Table vi: Burst interarrival result collected

The confidence interval above is calculated based on the assumption that the values are normally distributed around the mean. Furthermore, we tested this assumption by plotting the QQ plots for each scenario. Below we present QQ plot for Burst Test 1. Note here, similar to the exponential interarrival case each RunID stands for an IID processes.

Table vii: QQ plot inputs for test 1

id	results	normal quantiles	residuals	sorted residual
1	0,8236	-1,53412	-0,00556	-0,00556
2	0,8395	-0,88715	0,010398	-0,00536
3	0,8263	-0,48878	-0,00282	-0,00511
4	0,8275	-0,15731	-0,00165	-0,00282
5	0,8240	0,15731	-0,00511	-0,00165
6	0,8238	0,48878	-0,00536	0,00431
7	0,8334	0,88715	0,004308	0,00580
8	0,8349	1,53412	0,005799	0,01040

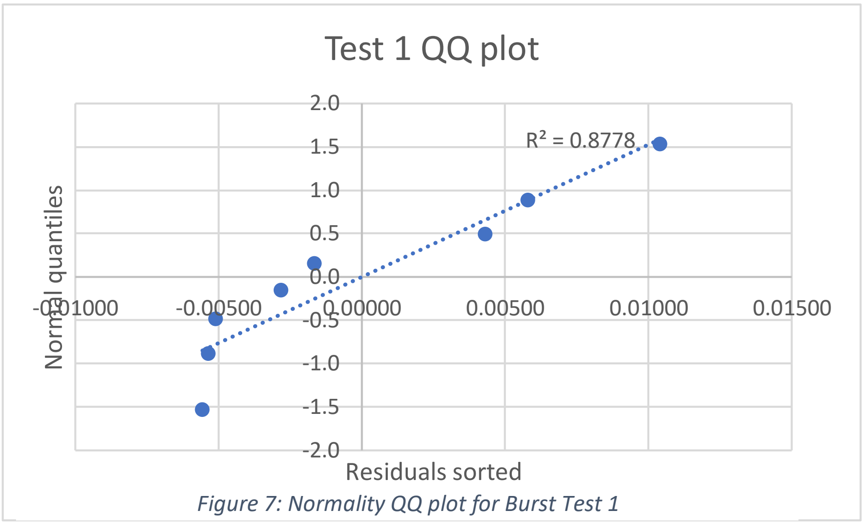

Figure 7: Normality QQ plot for Burst Test 1

Impact analysis

Like the previous scenario, we've analyzed the impact of every factor in the earning per unit of time. The results are the followings:

Table viii: average earning obtained with respective condition

		-1	1
		owner th st	
	Child pat. Sec	1	10
-1	2	0,828	0,577
1	10	1,817	1,806

I	O th	Chi pat Q	both	earning	(Ei - E')
1	-1	-1	1	0,827698	0,184
1	1	-1	-1	0,577491	0,462
1	-1	1	-1	1,817384	0,314
1	1	1	1	1,806173	0,301
5,02874625	-0,2614185	2,218367	0,238995	sum of ys	1,262
1,26	-0,07	0,55	0,06	qi=total/4	
4*qi~2	0,017	1,230	0,014		
percentage	1,354	97,514	1,132		

Table ix: impact and interplay analysis

From this kind of analysis, we can clearly see that the children patience Q has a 97,5% impact on the final result, the owner threshold has a 1,3% impact which is similar to the combined factor. The negative sign at owner th indicates the decrementing effect of increasing th on earning per unit time.

Now, having the latter result. We visualize the finding as below:

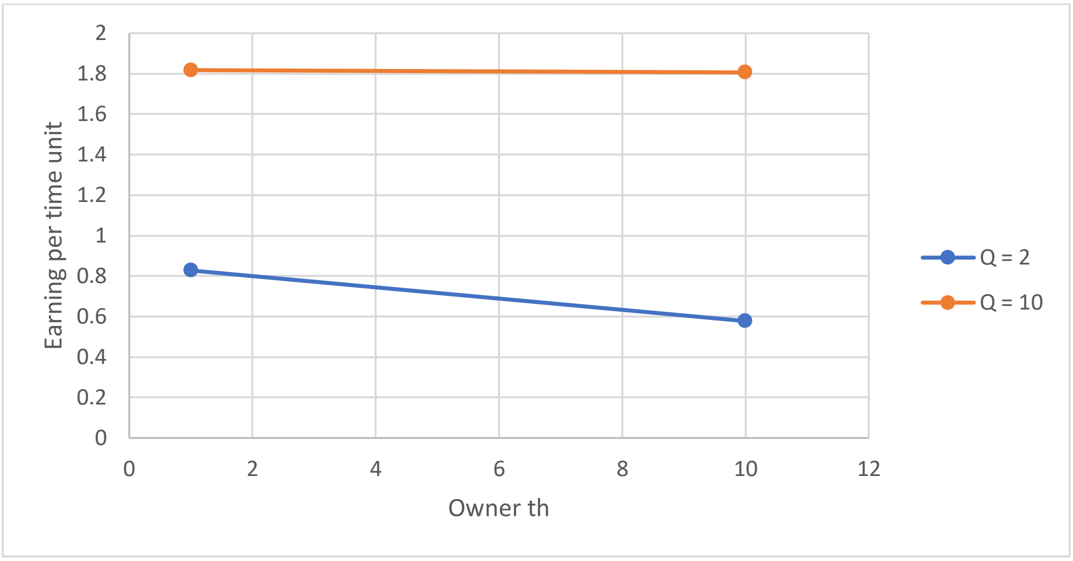

Figure 8 fixed minimum child patience plot

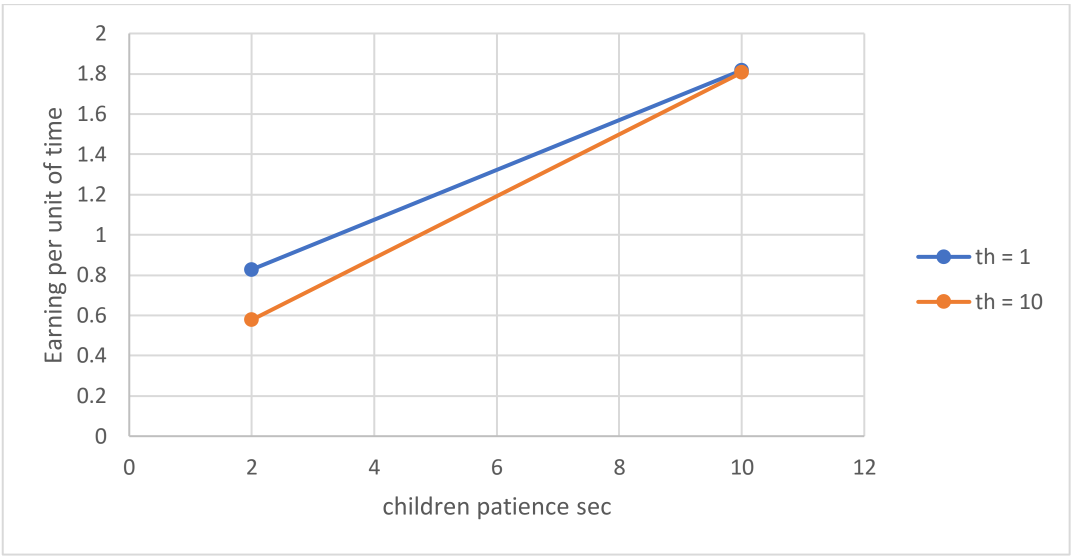

Figure 9: fixed owner minimum sit threshold plot

From these observations, in ***Fig. 8*** we can see that fixing the average child patience, the owner sit threshold has a slight impact when Q=10, and practically no consequence in the earning when Q=2. The more interesting case is in the ***Fig. 9,*** where with the owner sit threshold fixed to 10 and 1 the earning value for the children patience equal to 10s is the same, and diverges when they decrease to 2s.

Case Comparison and Explanation

In general at low owner sit number threshold, for both high and low patience of children, individual arrivals have higher earning per unit of time compared to burst arrivals. Also at high sit number threshold and higher children patience the earning is better for individual arrival case. This is because at higher burst arrived children are likely to drop out in groups than individually arrived children when their patience run out. Which will result in a large reduction in the children in the queue.

On contrary on low children patience (Q) and higher owner minimum sit threshold (th) things are very different. In this case burst arrivals have **higher** earning per unit of time. The explanation for this is that since the children are arriving at the same time and also it is likely that they have the same patience they tend to leave at the same time. But with only fewer arrivals the owner threshold will be passed and the children are in the game. Considering the dropout, the child at the front of the queue wait time is approximately equal to **sit-number-threshold** times the **average interarrival time**. Note here that Burst arrival can be assumed as an **exponential arrival with 0s interarrival time for much of the arrivals** which decreases the average interarrival time a lot. This results in a child at the front of the queue waiting for a much smaller time in comparison with individual arrival and the smaller the time they wait the less they drop out. This results in low dropout at higher minimum sit number and as a result higher earning per unit of time for burst arrivals.

Combining the two results and making the comparison we observe that the effect of decreasing the minimum sit number threshold is relatively small at burst arrival cases compared to individual arrivals. As the below graph shows the range of values for the four test cases become narrower in burst arrival cases in comparison to individual arrival.

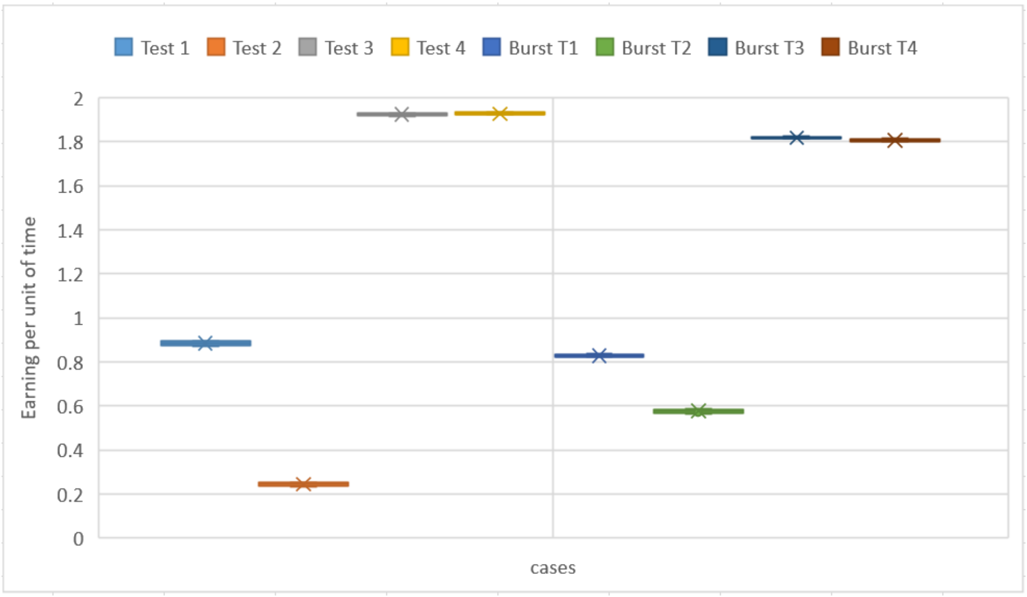

Figure 9 Box and whiskers plot for the individual and burst arrival cases

Conclusion

Generally for lower patience of children it is recommended to decrease the minimum sit number threshold for either cases.

Reference

1. Couse note on Data Analysis
2. OMNeT++ Simulation Manual, https://doc.omnetpp.org/omnetpp/manual/, visited from Dec 2018 to Jan 2019