Performance Evaluation of Round Robin CQI based Cellular Network

Università di Pisa

Antonio Le Caldare, Vincenzo Consales, Vincent Della Corte

year 2017/2018

Contents

Chapter 1

Modeling

1.1 Introduction

In these paragraph we describe how we modeled the Celluar Network described in the specifications.

- **A Web Server**, which generates data in the form of packets to be trasmitted to users. For our pourposes we have defined the class `UserPackets` which includes a `start_time` field and its interface (named `UserPacket_m`) includes a getter/setter method to update this field.

  The size of each packet is an integer RV $\sim U(3, 75)$, since the service demand has to be uniform and consistent with the frame size. Moreover the packet interarrival time to the antenna has to be an exponential RV, so each packet is generated properly to satisfy this requirements.

- **An Antenna**, which has FIFO **infinite** queue for each user. Packets received form **Web Servers** are stored inside queues and then are sent in a unicast way according to the Round Robin policy (which is described in the next section).

- **A Mobile Station**, which personifies a generic user connected to the antenna. On each timeslot it sends a channel quality indicator (CQI), which is a number between 1 and 15 that define the number of bytes the antenna can pack into a Resource Block (RB).

  CQIs are integer RVs generated according the following scenarios:

  1. Uniform, each user generates a RV $\sim U(1, 15)$
  2. Binomial, each user generates a RV $\sim Bin(n, p_i)$, where n is the number of users, and $0 < p_i < 1$ depends on the user i.

To build our model and to run simulations we used the framework **OMNeT++ v5**, so each item described before is defined by a `*.ned` file. Each `Mobile Station` computes some statistics: slotted throughput (related to each time slot) and response time of received packets. The `Antenna` compute also statistics about the frame filling, which we will describe later.

The `CellularNetwork.ned` file shows how the previous modules are connected to obtain the network. Since frames are sent in a unicast way, there are multiple instances of the Web Server module, one for each Mobile Station, seeded in a different way in order to have IID RV.

The obtained network, by setting $n = 10$, is the following:

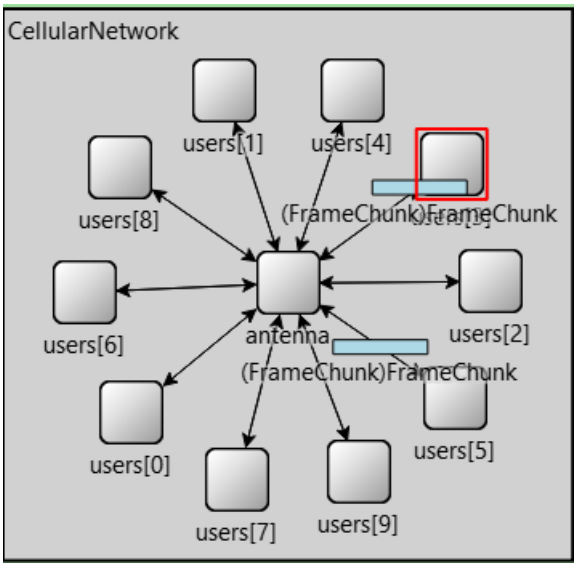

Figure 1.1: Simulated Network (omnet++)

1.2 Frame Chunks

Packets are delivered to users by enveloping that inside RBs. As requested by specifications $RBsize_{\mathrm{i,t_j}} = f(CQI_{\mathrm{i,t_j}})$ where i is index of *user*, and t_{j} is index of *timeslot*. To do that, the scheduler receives CQIs from the `Mobile Stations` at beginnig of each time slot, then compute every $RBsize_{\mathrm{i,t_j}}$ and fills the frame according to scheduling policy. Note that a frame can carry RBs for different users so, generally speaking, a frame in a specific time slot t_{j} can have RBs of different size. To cope these requirements we defined a new module `Frame Chunk` wich groups all RBs addressed to a specific `Mobile Station`. By introducing this module we can consider a frame as a collection of `Frame Chunks`. To deliver the whole frame we must send each `Frame Chunk` to its user in a unicast way.

1.3 Schedulers

In this section we will analyze the frame filling policy wich are defined in the module `Scheduler`. Let's consider a user $0 \leq i < n$. Starting from user i, the scheduler allocates a new `Frame Chunk` and fills it with packets taken from the $\texttt{FIFOQueue}_{\texttt{i}}$. A `Frame Chunk` is considered full if it contains 25 RBs because it corresponds to the whole frame. At the next time slot will be served the user $(i+1)\text{mod } n$. If the frame is not full the scheduler must allocate others `Frame Chunks` in order to fill the residual space using one of the two following policies.

- **Round-Robin Frame Fill** The residual space in the frame is filled by considering the following $\texttt{FIFOQueues}_{\texttt{j}}$ $j \in \{(i+1)\text{mod } n, (i+2)\text{mod } n, \ldots\}$
- **Best CQI based Frame Fill** The residual space in the frame is filled by considering the users with the best(highest) CQI in a decreasing order. In the case of $CQI_{\text{i}} = CQI_{\text{j}} \mid i < j$ is selected the user i.

We will analyze the effect of both policy to performace regarding the throughput and the response time for each user with varying workloads.

Chapter 2

Simulation

2.1 Introduction

How we said in the previous chapter we built the model inside the framework **OMNeT++ v5**. The definition of network *(CellularNetwork.ned)* and their components can be found in the directory *RRCellNet/src*. We decided to analyze it when 10 `Mobile Stations` are connected to the `Antenna`. In order to simplify the model we represented the network with a *Star Topology*. `Mobile Stations` receive packets through unicast channel `without transmission delay`.

We recall that `Web Servers` are sources of packets addressed to `Mobile Stations`. In order to have true random and independent arrivals to `Antenna` we need a 10 RNG, one for each `Web Server`. We also need RNG to generate packets with random size so others 10 RNG are neeeded. We need also a RNG for each `Mobile Station` in order to generate random CQIs at each timeslot. Overall the model requires 30 RNGs each of them initialized with a different seed in order to have independent pseudo-random variables. The seed-set is changed at every repetition to have different independent experiments. At the end of all repetions the results are aggregated by computing the mean and 95% confidence interval.

Let's consider for example the mean throughput for a rate λ^*. By running each repetition we get the values $X_1, X_2, \ldots X_{10}$. X_i is a random variable which represents the mean throughput for the repetition i. By the CLT theorem we can say that X_i is a normal RV since it is obtained by summing up a huge number of *slotted throughput*. We can estimate the mean $\bar{X}$ and a 95% confidence interval by using the *Student's t distribution* because X_i are normal RV.

$$\bar{X} = \frac{X_1 + X_2 + \ldots + X_{10}}{10} \qquad S^2 = \frac{1}{9}\sum_{i=1}^{10}(X_i - \bar{X})^2 \tag{2.1}$$

$$CI_{0.95} = \left[\bar{X} - \frac{S}{\sqrt{10}}t_{0.025,9},\ \bar{X} + \frac{S}{\sqrt{10}}t_{0.025,9}\right] \tag{2.2}$$

Similar consideration can be done for others quanties wich we will analyze during simulations. Once we have computed the mean $\bar{X}$ and its confidence interval we can do a box plot for that quantity at varying workload λ as required by specifications. Data are exported from simulation to csv files through a bash script *exportdata.sh* and plots are done through an R script *analyze_csv.r*. These script are both included in the directory *RRCellNet/simulations*.

All parameters for simulations are summarized here and can be found in the file *RRCellNet/simulations/omnetpp.ini*. We will use them in the following chapters unless otherwise specified.

- **Number of resource block**, $\#RB = 25$
- **Number of users**, $n = 10$
- **Number of RNG**, $\#RNG = 30$
- **Max RB size**, $RBsize_{\max} = 93$
- **Max packet size**, $packetsize_{\max} = 75$
- **Timeslot period**, $T_{\text{slot}} = 1\text{ms}$
- **Number of repetions**, $\#REP = 10$
- **Simulation time**, $ST = 60\text{s}$
- **Warmup period**, $WP = 0.3\text{s}$

2.2 Warm-Up Period Estimation

In the previous section we saw that the warm-up time is 0.3s, but we haven't specified yet how that value comes out.

In this section we will illustrate how we have estimated the length of the warm-up period. We followed this approach: for each scenario and for each user we plotted the graph of the throughput and the response time and then for each repetition we applied the sliding moving average and then we computed the sample mean of the

latter in order to see how many time it was required to converge around that value. We chosen the worst case among all of them, which graph is the following:

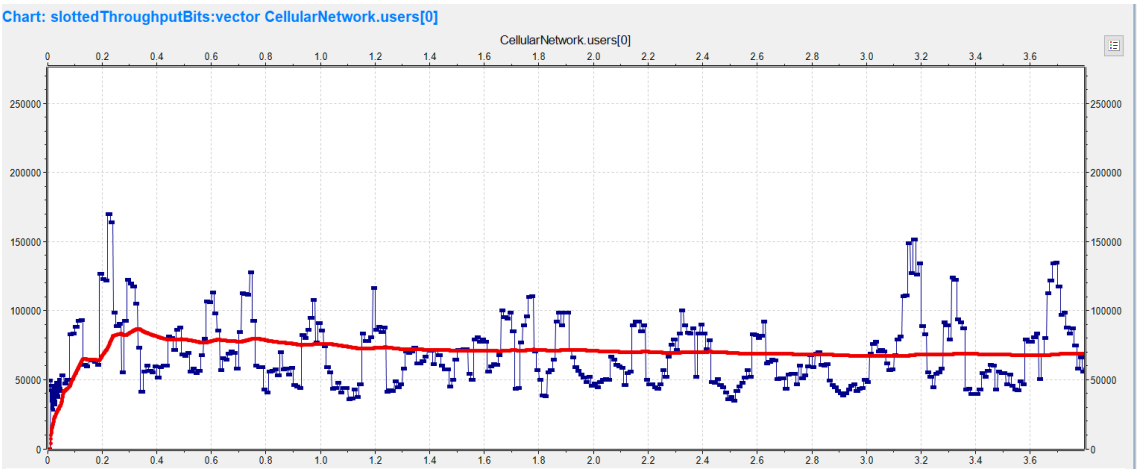

Figure 2.1: Worst Case for the Warm-Up period estimation

So we saw that in the worst case the graph converged to the sample mean after **0.3s** and we chosen that value as **warm-up time**.

Chapter 3

Validation

3.1 Introduction

After building our model inside the **OMNeT++** we can proceed with the simulation in order to analyze the quantities wich we are interested. First of all we will simulate the model in very simple cases in order to be sure it reproduces the system's behavior correctly. We decided to validate our model by *removing randomness*. This choice allows us to do some easy computation by hand and to verify if the result of simulation is consistent with them. Our model will be considered a good replica of the system if it will pass all the *validation tests*. During the validation we will use the first scheduler: **Round-Robin Frame Fill** because it generates simulation which are easier to analyze and to compare with analitical results.

3.2 1st test: fixed CQI, fixed λ rate, fixed packet size, 1 user

In this test we just one `Mobile Station` connected to the antenna wich always generates the same CQI for each timeslot. Inside the `Web Server` the λ rate is fixed and also the packet size. This is a very simple system with deterministic arrivals and deterministic service demand. We can compute the traffic that `Web Server` sends to the antenna as

$$th_{\text{in}} = \frac{packetsize \times 8}{1/(1000\lambda)}[bps] \tag{3.1}$$

If the system is in a stable state the output throughput and the input throughput must be equal. The maximum output throughput is the one we have by setting all parameters to maximum values.

$$th_{\max} = \frac{\#RB \times RBsize_{\max} \times 8}{T_{\text{slot}}} = \frac{25 \times 93 \times 8}{0.001} = 18.6 \text{ Mbps} \tag{3.2}$$

We can derive very easily the $\lambda_{\max}$ rate wich produces the max throughput allowed by antenna.

$$\lambda_{\max} = \frac{\#RB \times RBsize_{\max}}{1000 \times T_{\text{slot}} \times packetsize_{\max}} = \frac{25 \times 93}{1000 \times 0.001 \times 75} = 31 \text{ ms}^{-1} \tag{3.3}$$

For $0 < \lambda < \lambda_{\max}$ the system is a stable state and so $th_{\text{in}} = th_{\text{out}}$. For higher λ the FIFOQueue grows indefinitely because the frame is not able to carry as much data in a time slot.

Let's consider all the simulations with the following parameters:

- $\lambda \in \{1, 2, \ldots, 32\}$
- $packetsize = 75$
- $CQI = 15$

We can se in the graph a linear behavior regarding the throughput until the system is not saturated. Note that the response time for each packet is zero since there is no queueing until $\lambda < \lambda_{\max}$ and grows indefinitely when the system saturates. This is not surprising since $th_{\text{in}} \propto \lambda$ when $\lambda < \lambda_{\max}$ and packet's size is fixed.

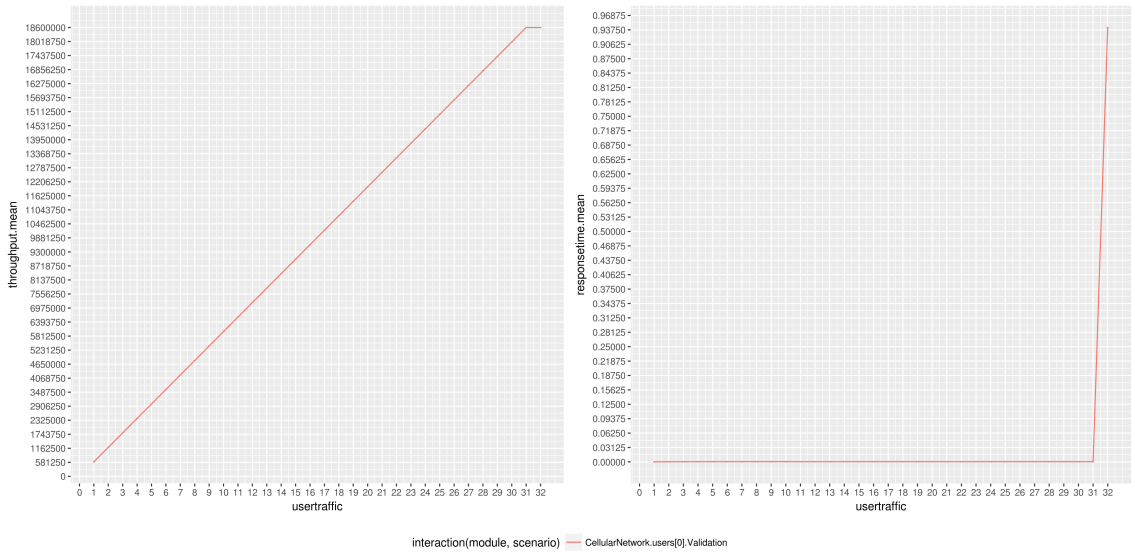

Figure 3.1: 1st validation scenario: throughput, response time

3.3 2nd test: fixed CQI, fixed λ rate, fixed packet size, 2 users

In this test there are two `Mobile Stations` connected to antenna. As in the previous scenario all parameters are fixed. We have two indipendent flow of data from `Web Servers` to `Antenna` so the input throughtput could by computed as

$$th_{\text{in}} = \frac{packetsize_0 \times 8}{1/(1000\lambda_0)} + \frac{packetsize_1 \times 8}{1/(1000\lambda_1)} \tag{3.4}$$

For these simulation we have chosen the following parameters:

- $packetsize_0 = packetsize_1 = 40$
- $CQI_0 = 6$
- $CQI_1 = 15$
- $\lambda = \lambda_0 = \lambda_1$ and $\lambda \in \{1, 2, \ldots, 32\}$

We can compute the max *slotted thorughput* for both users.

$$\begin{aligned} slotth^0_{out} &= \frac{\#RB \times RBsize_0 \times 8}{T_s} = \frac{25 \times 20 \times 8}{0.001} = 4 \text{ Mbps} \\ slotth^1_{out} &= \frac{\#RB \times RBsize_1 \times 8}{T_s} = \frac{25 \times 93 \times 8}{0.001} = 18.6 \text{ Mbps} \end{aligned} \tag{3.5}$$

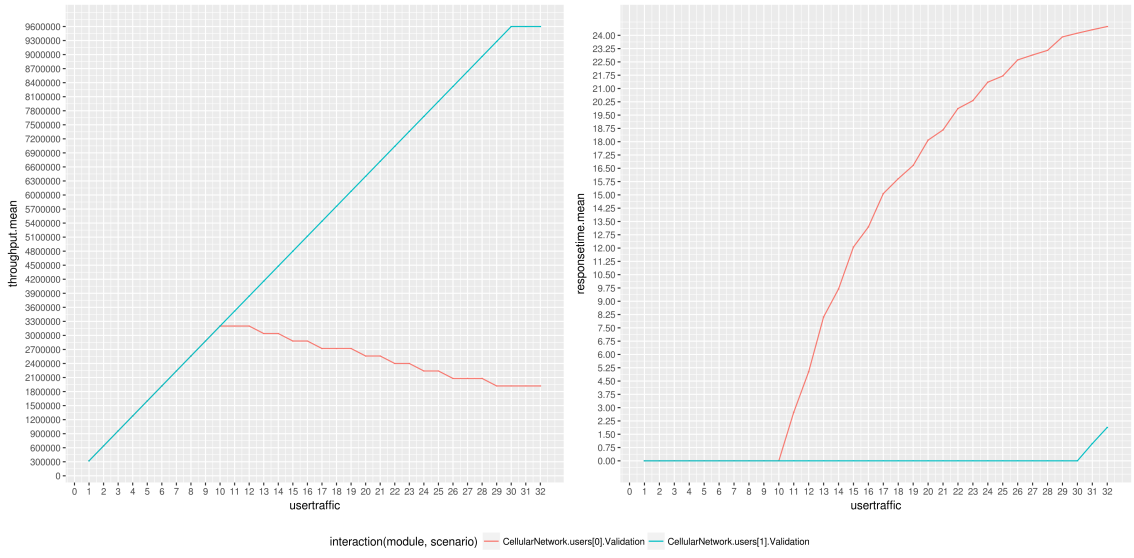

Figure 3.2: 2nd validation scenario: throughput, response time

At full load we expect that users compete to fill the frame. If the scheduler were fair, at full load, the average throughput would be $th^i_{out} = (slotth^i_{out})/2$ since there are 2 users and the scheduler follow a round robin scheme, which is fair in principle.

We see in the graph that throghput grows linear ad is equal for both users when $1 \leq \lambda \leq 10$. For $\lambda > 10$ the `Mobile User[0]` saturates, conversely `Mobile User[1]` continues to increase its throughput until it reaches saturation at $\lambda = 30$. When $\lambda = 10$ the input flow is $th^0_{in} = th^1_{in} = (40 \times 8)/0.0001 = 3.2$ Mbps and this result could be seen also in the graph. Infact when $\lambda \leq 10$ both users are in stable state so $th_{in} = th_{out}$. We can see that, when λ increases the mean throughput approches to the half of slotted throughput as said before. However there is a small difference between the expected mean throughput and the result of simulation. This oddity can be explained better by analyzing the following graph, which shows the mean resource block per frame assigned to users.

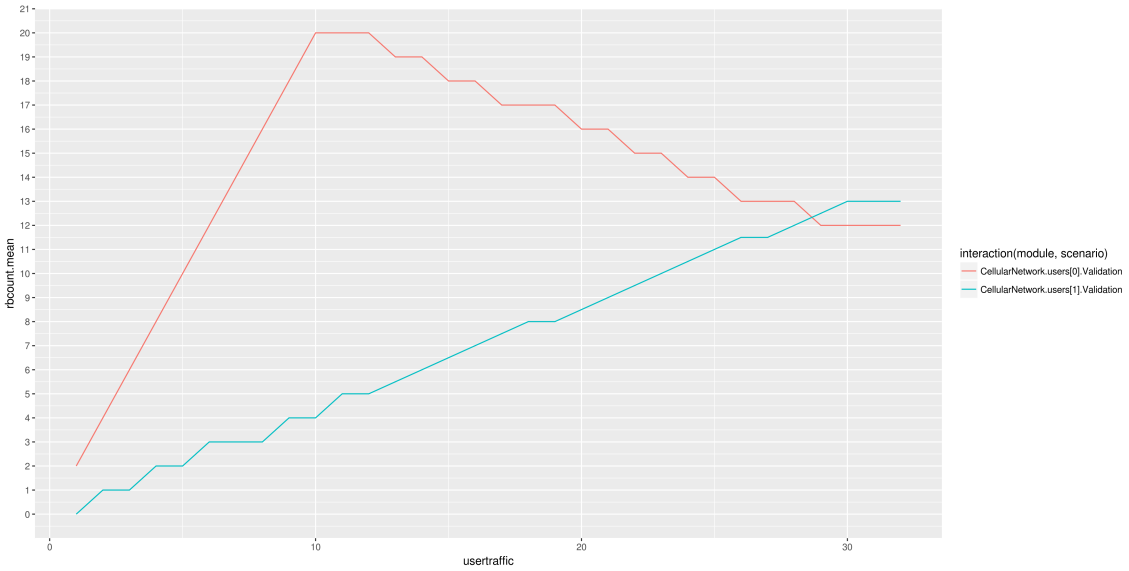

Figure 3.3: 2nd validation scenario: mean RB count

By math the mean RB would be $\#RB/2 = 12.5$ but the number of RB assigned per user is slightly different since, on average, 11 RBs are assigned to `Mobile User[0]` and 13 RBs are assigned to `Mobile User[1]`. This oddness is due to fragmentation of packet. In our simple model infact packets can not be fragmented so if a packet does not fill inside the last RB this RB is lost and it assigned to the next user. Note that in validation scenarios everything is fixed, also packet dimension, so there could be strangeness like that. However we can say that our model is quite accurate and can be used to simulate the network in more complex scenarios. There is another strange behavior that involves response time. When the input traffic is

too high we expect that response grows indefinitely since packets get queued. However, by observing the graph, the mean response time seems to approch to $ST/2$. In the following graph it is displayed the response time of packets addressed to `Mobile Station[0]` when the rate $\lambda = 40$, so it is very high.

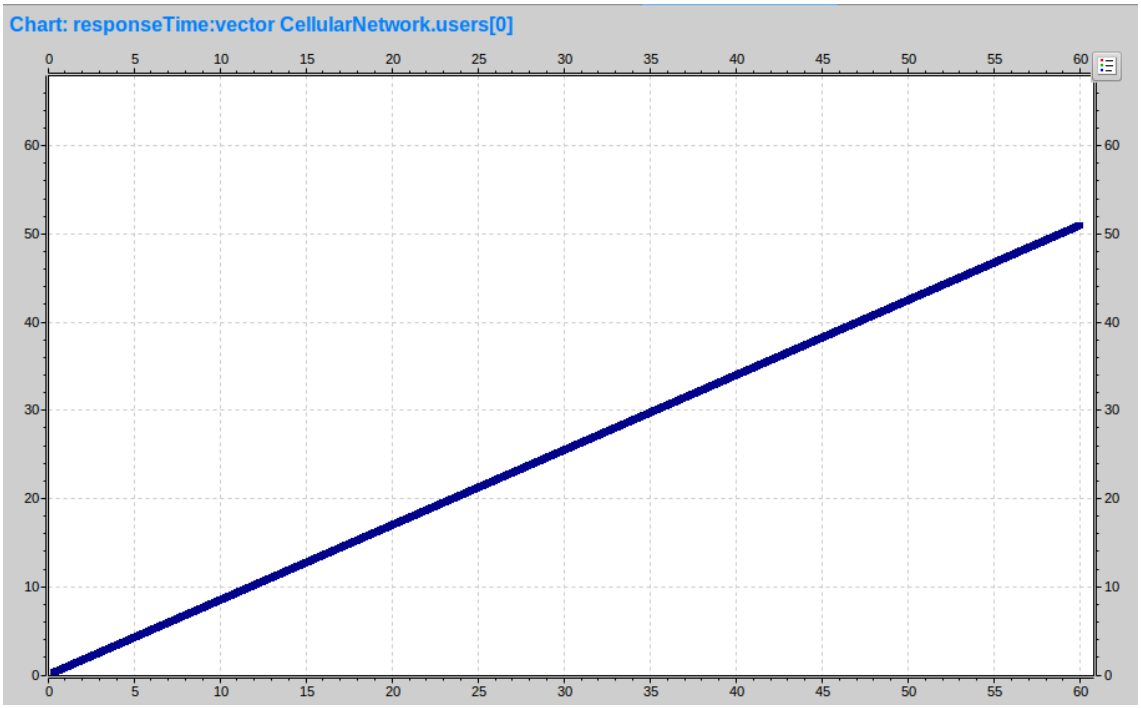

Figure 3.4: 2nd validation scenario: response time(vector)

We we can see cleary a linear relation between simulation time and response time when the system is unstable. We can describe the relation as:

$$RT(t) \approx \frac{\max(RT)}{ST} t, \quad 0 \leq t \leq ST \tag{3.6}$$

The previous formula has that meaning: *a packet which is generated at time t will have a response time about equal to* $RT(t)$. Now we can try to calculate the *mean response time*.

$$\begin{aligned} E[RT(ST)] = \frac{\sum_{t=0}^{ST} RT(t)}{ST} = \frac{\sum_{t=0}^{ST} \frac{\max(RT)}{ST} t}{ST} = \frac{\max(RT)}{ST^2} \sum_{t=0}^{ST} t = \\ = \frac{\max(RT)}{ST^2} \frac{ST(ST+1)}{2} = \frac{\max(RT)}{2} \frac{ST+1}{ST} \approx \frac{\max(RT)}{2} \end{aligned} \tag{3.7}$$

At the end we can observe that if the system is unstable $RT(t)$ has a linear behavior so if we calculate the mean at the end of simulation we obtain $E[RT(ST)] \approx ST/2$.

3.4 3rd test: NoFramingTest, uniform CQI, exponential interarrivals, fixed packet size, 10 users

This is a test to validate eventually the model shown in the previous chapter. It's too difficult to build a full matematical model starting from the requirements, but having no one is not helpful in explaining the simulation results. So we started to build simple models by leaving off some of the requirements and by considering only some specific conditions (like saturation). In our case one of the most difficult things to model is packet framing: as the requirements said, every RB cannot contain packets from two or more different users. This means that there will be some wasted space to consider. Furthermore, if a packet cannot completely fill the frame, it cannot be scheduled and the scheduler moves to the next user.

A question now arises: how this packet framing policy will influence our results? Intuition make us think that total antenna throughput will get somewhat worse, just thinking about the wasted space. To confirm this intuition we set up another scenario, called NoFramingTest, which will be running using the following parameters:

- $\#users = 10$
- $CQI_i \sim U(1, 15)$
- $packetsize_i = 25$ B
- $interarrivalrate_i \sim exp(\lambda)$

The most strange parameter here is *packetsize*. This is a simple way to have no framing at all in our simulation. We could end up fixing $packetsize = 1$ B, which would be the simplest way, however simulation would be too heavy to run at higher traffic rates (we tried, but our PCs started frying). Recalling that we need just to check throughput in saturation, we can suppose that every frame is only filled with packets from the currently selected user (using RR policy). Studying the formula that defines the total frame size of each client in our model:

$$\begin{aligned} rbsize_i &= f(CQI_i) \\ framesize_i &= \#RB \times rbsize_i \end{aligned} \tag{3.8}$$

The number of packets that will fill the frame can be computed as:

$$\#packets_i = framesize_i / packetsize \tag{3.9}$$

where *packetsize* is fixed, our goal is to find a *packetsize* such that $\#packets_i$ is a natural number (all packets fill exactly the frame) for each client (i). As we said, $packetsize = 1$ is a solution, however another interesting solution is $packetsize = 25$. This is due to the fact the $\#RB = 25$ is fixed and used to compute every client frame size. Note that this is valid only if frame is filled every time by a single client (as the previous assumption). Lets check the results:

INSERIRE GRAFICO THANTENNA QUI
INSERIRE GRAFICO FILLRB QUI
(dovrebbe essere esattamente a zero quando i client saturano)

A simple mathematical model to compute antenna total throughput can be:

$$E[th_{antenna}] = \frac{\sum_{i=1}^{\#client} \#RB \times E[rbsize_i]}{\#client \times (1/T_{slot})} = \frac{\sum_{i=1}^{10} 25 \times E[rbsize_i]}{10 \times (1/T_{slot})} \tag{3.10}$$

$rbsize_i$ is a RV variable with an unknown distribution, however

$$rbsize_i = f(CQI_i), \quad CQI_i \sim U(1, 15), \quad \forall i \in [1, 10] \tag{3.11}$$

This means we can simply use the general formula for functions of RVs

$$E[X] = \sum_{x_i : g(x_i) = y_i} g(x)p(x) = E[g(x)] \tag{3.12}$$

Considering that

$$f(CQI_i) = [3, 3, 6, 11, 15, 20, 25, 36, 39, 50, 63, 72, 80, 93, 93]$$

our result, that will be used also in other computations, is

$$E[rbsize_i] = \frac{1}{15} \times \sum_{i=1}^{10} f(CQI_i) = 40.6 \text{ B}, \quad \forall i \in [1, 10] \tag{3.13}$$

which is constant for all clients. In this case the initial model can be simplified to

$$E[th_{antenna}] = \frac{\#RB \times E[rbsize_i]}{1/T_{slot}} \times 8 = 8.12 \text{ Mbps} \tag{3.14}$$

The mean **matches exactly** the simulation results!

This test is not only for a validation purpose, but it is a starting point to study how fragmentation impacts network performances. We will see this later.

Chapter 4

Simulation: Uniform Scenarios

4.1 Introduction

In this chapter we will consider `Mobile Stations` which, at each timeslot, generate CQIs with an integer uniform distribution. In the validation scenario we proved that scheduler is fair so we expect that all users will have similar values for the mean throughput and the mean response time. In this simulation we will consider the following parameters:

- $n = 10$
- $packetsize_i \sim U(3, 75), \quad 0 \leq i \leq n-1$
- $CQI_i \sim U(1, 15), \quad 0 \leq i \leq n-1$
- $\lambda = \lambda_i = 0.1 + 0.5k, \quad k \in \{1, 2 \ldots, 10\}, \quad 0 \leq i \leq n-1$

This scenario is almost similar to the NoFraming Validation test, but the main difference is that *packetsize* is a uniform RV. As we said before we expect to get worse throughput result due to the fact that some RB space will be left empty because packets cannot be fragmented.

We will simulate the scenario with both scheduler and we will check if our first intuition is true or not and the impact of **framing** in the average performance metrics.

4.2 Uniform CQI, Fair Scheduler

In this simulation we will consider the basic **Fair Scheduler**. We remember that, in this case, if $currentUser$ empties his queue or has a packet that is bigger than RB_{size} the scheduler will consider the next users.

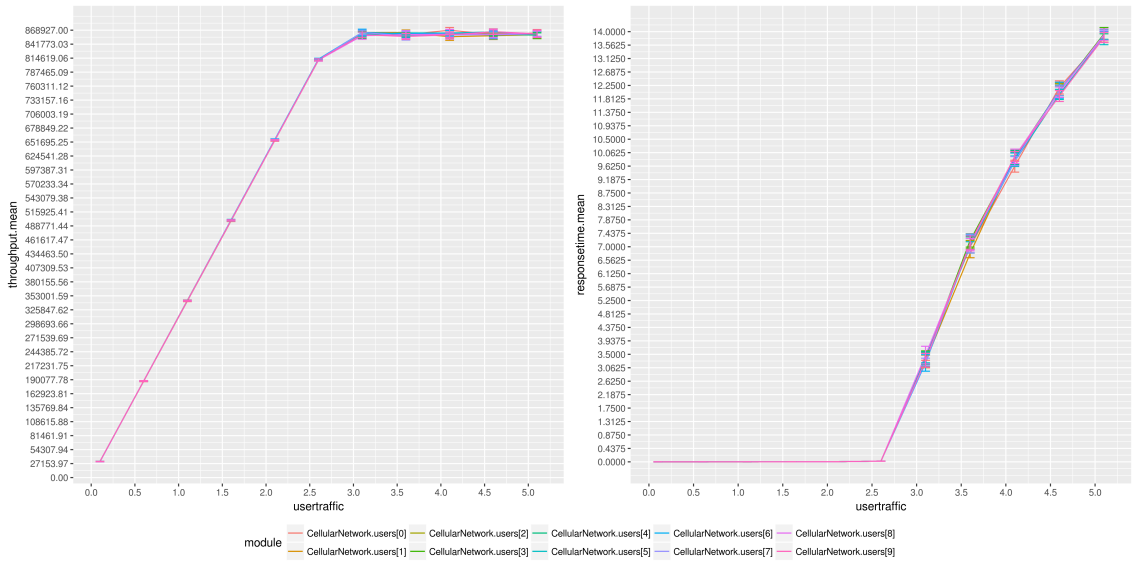

Figure 4.1: Uniform scenario - Fair: throughput, response time

The first thing we can notice is that every client has the same saturation point (which depends on λ): this can be explained considering that scheduler is fair and every client generates, on average, the same amount of packets (which are also the same size on average). Response times are stable before reaching the common saturation point and after tends to arise indefinitely, until reaching a maximum. This maximum is the same shown in the 2_{nd} Validation test and has the same explaination. So the only valid values for Response Times are generated between $0 < \lambda < \lambda_{sat}$, which is common for each client.

In the graph we notice that $\lambda_{sat} = 3.1$. If we compare this rate with the maximum rate computed in the 1st Validation Test we observe that $\lambda_{sat} = \lambda_{max}/10$. This is not surprising since there are 10 `Web Server` wich push traffic to the Antenna. Each one is a *Poisson process* with rate λ so the superimposition of 10 them is a *Poisson process* with a rate $\Lambda = 10\lambda$. Actually the system saturates before $\lambda = 3.1$, infact in validation we considered the best case where $CQI = 15$ so RBs had the maximum size. Here RB_{size} is a RV however $\lambda_{sat} = 3.1$ is a raw approximation confirmed by the results of the simulation.

The most important result, here, is the antenna total throughput. Lets see:

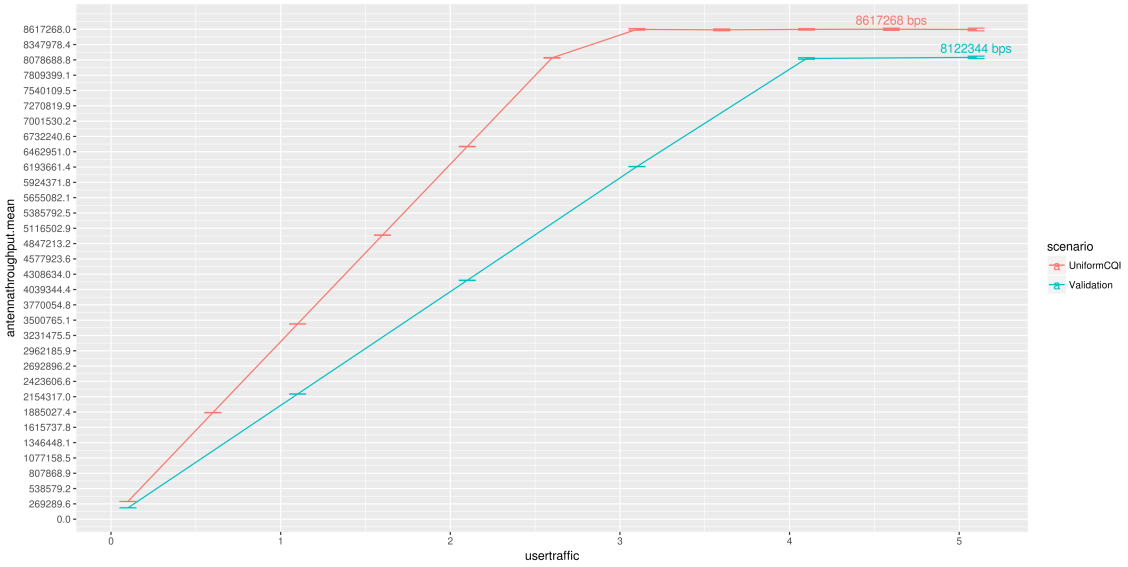

Figure 4.2: Antenna throughput: UniformCQI, Validation-NoFramingTest

Comparing to the 3rd NoFraming Validation test result, our global throughput is **higher**. This is a very very strange result that **destroys our first intuition** about the impact of framing.

How can the throughput get higher? Thinking about our model, we came up with a possible explanation. Framing policies, as we seen before, are basically two:

- One RB cannot contain traffic from 2 or more different clients
- If a packet cannot entirely fit the frame, it cannot be scheduled

The first policy cannot give us an higher throughput and this is already proven: we are telling that some frame space is eventually wasted, and this leads always to a worse or equal throughput result. So lets focus on the second policy: our intuition (hopefully correct this time) suggests us that framing, as a measure of how much RBs are not filled due to framing, is uniformly distributed for each client. Lets remember that the remaining space is allocated to other clients using a Fair policy (FairScheduler) and $framesize_i$ depends on $rbsize_i = f(CQI_i)$. Now lets try to analyze a single iteration of the scheduler algorithm, applied to 2 clients in the following state:

$$\begin{aligned} \#RB_{free} &= 1, & currentUser &= 1 \\ CQI_1 &= 1, & rbsize_1 &= 3 \\ CQI_2 &= 13, & rbsize_2 &= 80 \end{aligned}$$

Both have one packet of $packetsize = 75$ (which is the maximum we can have) in backlog. The first client cannot fit his packet into the frame chunk, so remaining RBs (in our case just 1) are allocated to the client 2, which can now fit his packet into his frame chunk. The main factor, here, is $framechunksize_i = remainingRBfor_i \times rbsize_i$: we are telling that client with the best CQI can fit more likely his packets into the frame, despite of the current serving user, due to the fact that his $framechunksize_i$ is bigger than the other clients. This reminds us a bit of BestCQI Scheduler policies.

However we must consider that the previous case does not describe completely all the possible behaviors of the system. Infact, lets consider this other case:

$$
\begin{aligned}
&\#RB_{free} = 1, \quad currentUser = 1 \\
&CQI_1 = 3, \quad rbsize_1 = 6, \quad backlog = 1 \text{ packet of 75 bytes} \\
&CQI_2 = 1, \quad rbsize_2 = 3, \quad backlog = 1 \text{ packet of 3 bytes} \\
&CQI_3 = 13, \quad rbsize_3 = 80, \quad backlog = 1 \text{ packet of 75 bytes}
\end{aligned}
$$

Client 1 packet cannot be scheduled ($75 > 6$), so we go next to the second user which can now fit his packet into the new frame chunk ($3 = 3$). As we can see here, the RB is allocated to the user with the worst CQI (client 2) and not the best (client 3). So we can also deduce that a client with a small packet in backlog is more likely to fit his packet into the frame.

Combining this result with the previous we can infer that a packet is more likely to fit if:

- The packet is small
- The user CQI is high

We can add another case:

$$
\begin{aligned}
&\#RB_{free} = 1, \quad currentUser = 1 \\
&CQI_1 = 3, \quad rbsize_1 = 6, \quad backlog = 1 \text{ packet of 75 bytes} \\
&CQI_2 = 1, \quad rbsize_2 = 3, \quad backlog = 1 \text{ packet of 75 bytes} \\
&CQI_3 = 2, \quad rbsize_3 = 3, \quad backlog = 1 \text{ packet of 75 bytes}
\end{aligned}
$$

None of the packets cannot be inserted into the framechunk, so the RB is completely wasted. This is the case that lowers the throughput, due to the fact that packet sizes are small and CQIs are low. However, in our scenarios (Uniform, UniformBest, Binomial ...) the number of users is high enough (10 users) to get, more likely, at least a small packet and/or at least a good enough CQIs to not waste the remaining space.

Note that we have tried to analyze just few cases and do a very raw approximation of antenna total throughput mean value tendency, so we can't make an exact model of the system in order to proof this result analytically. At the end of simulation we can summarize these results about the Uniform Scenario with Fair Scheduler:

- Saturation rate $\lambda = 3.1$
- Antenna throughput $th_{antenna} = 8615055 \pm 14206$ bps, *conf lvl 95%*
- User throughput $th_i = 861343 \pm 4968$ bps, *conf lvl 95%*

4.3 Uniform, Best CQI scheduler

Now we will simulate the scenario by using the second scheduler. If there were not **framing** the performance of both scheduler would be the same but, as seen before, **framing** is present and has a great impact in performance. We can expect this scheduler will have better performance in throughput since it fills the residual space inside the frame by using packets addressed to users with higher CQIs.

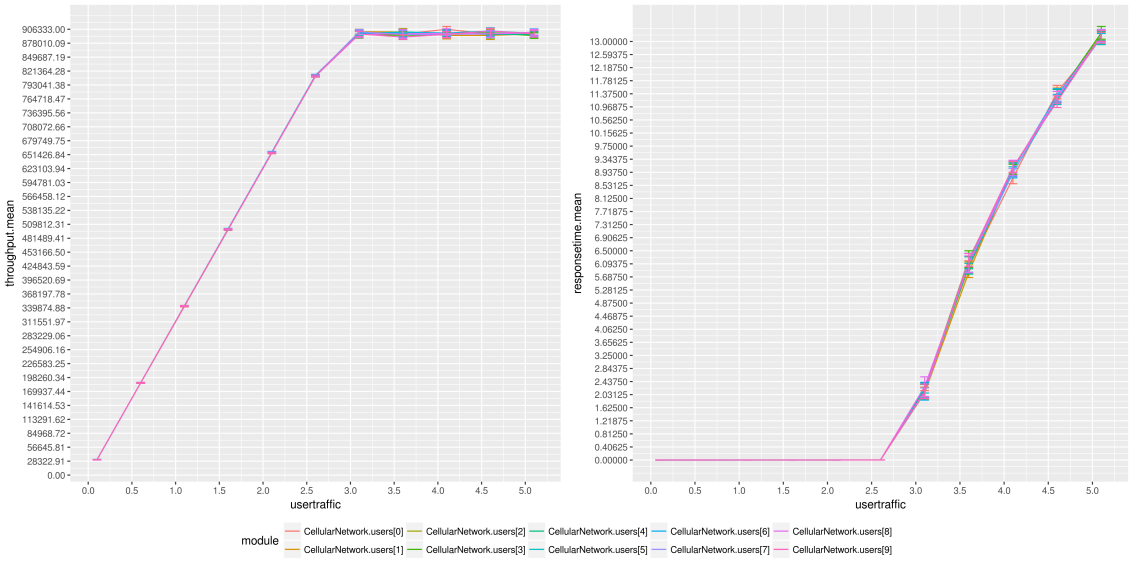

Figure 4.3: Uniform scenario - BestCQI: throughput, response time

Result for Uniform Scenario with Best CQI Scheduler in saturation:

- Saturation rate $\lambda = 3.1$
- Antenna throughput $th_{antenna} = 8981747 \pm 14355$ bps, *conf lvl 95%*

- User throughput $th_i = 897148 \pm 4772$ bps, *conf lvl 95%*

We can see cleary an increase of throughput per each user and smaller the response time with respect to fair scheduler. The saturation point is the same at about $\lambda = 3.1$ but the throughput as we said is higher. The reason is that the residual space in the frame is used better since it is selected the user with the best CQI in order to carry as much data as possible. We can see also that the throughput distribution is fair among the users if we plot the lorenz curves for throughput and response time.

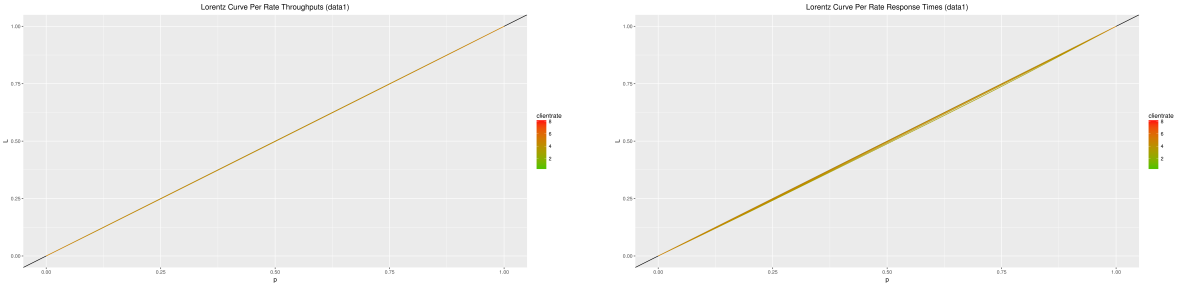

Figure 4.4: Uniform scenario - BestCQI: lorenz curves

At the end in order to choose the best scheduler for that scenario we decided to compare the empirical cdf of throughput near the saturation point $\lambda = 3.1$.

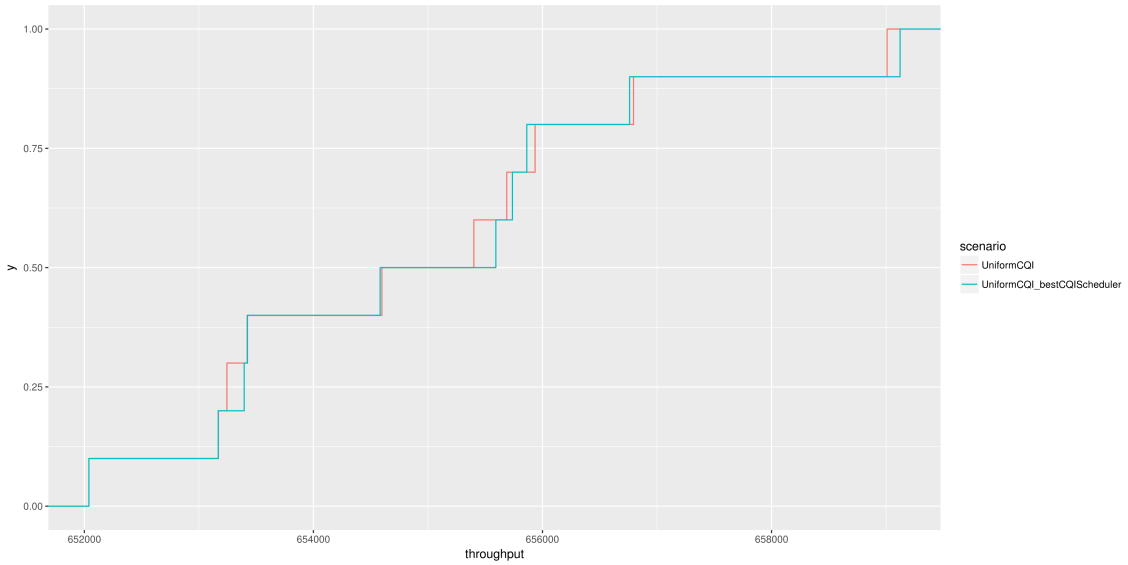

Figure 4.5: Uniform scenario: ECDF comparison $\lambda = 2.1$

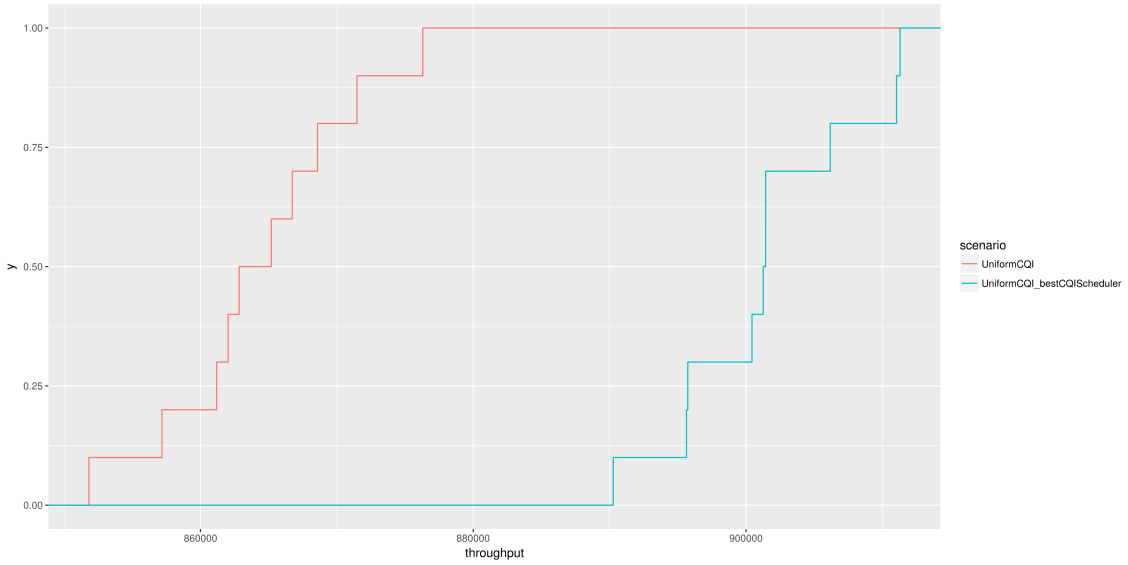

Figure 4.6: Uniform scenario: ECDF comparison $\lambda = 3.1$

We observe that the Best CQI Scheduler is definitely better than the Fair one when $\lambda \geq \lambda_{sat}$ otherwise the performance are about the same. The fairness is garanted by both scheduler in all *arrival rates* but this is not surprising CQI and service demand have the same uniform distribution for all users. The result is that each users exploits an equal slice of available resources to the antenna.

Chapter 5

Simulation: Binomial Scenarios

This scenario models an enviroment where users have a fixed position in the space. There are some users which receive stronger signal from antenna, so they have an higher mean CQI, and others which are far from antenna and then they have smaller mean CQI. The requirements state that the distribution of CQIs must have a binomial distribution. To satisfy these, at each timeslot t_j, every `Mobile Station` i generates a RV $X_{i,t_j} \sim Bin(n-1, p_i)$, and then $CQI_{i,t_j} = X + 1$. By using this trick $CQI_{i,t_j} \in \{1, 15\}$ and it has a binomial distribution. In order to have different mean we have chose this values for parameters p_i:

Listing 5.1: omnet.ini - p parameters

```
1  CellularNetwork.users[0].cqi_binomial_p = 0.13
2  CellularNetwork.users[1].cqi_binomial_p = 0.22
3  CellularNetwork.users[2].cqi_binomial_p = 0.31
4  CellularNetwork.users[3].cqi_binomial_p = 0.40
5  CellularNetwork.users[4].cqi_binomial_p = 0.49
6  CellularNetwork.users[5].cqi_binomial_p = 0.58
7  CellularNetwork.users[6].cqi_binomial_p = 0.67
8  CellularNetwork.users[7].cqi_binomial_p = 0.76
9  CellularNetwork.users[8].cqi_binomial_p = 0.85
10 CellularNetwork.users[9].cqi_binomial_p = 0.94
```