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:

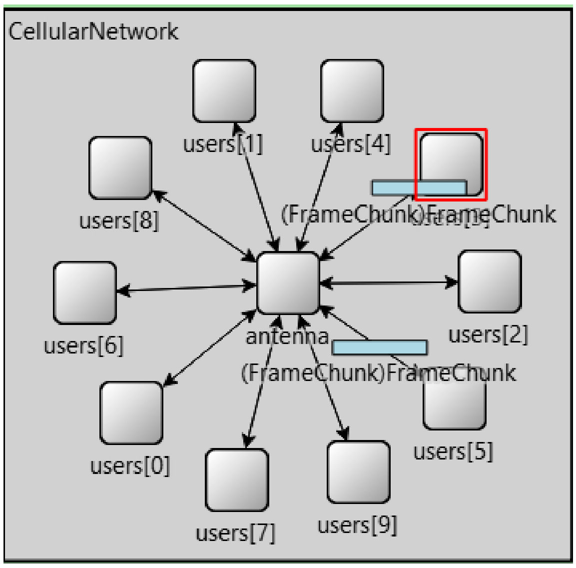

Figure 1.1: Simulated Network (omnet++)

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 $\mathtt{FIFOQueue}_\mathtt{i}$. 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.

- **Fair Frame Fill** The residual space in the frame is filled by considering the user j, where $j \neq currentUser$. Every user j has the same probability to be chosen.
- **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 $CQI_i = CQI_j \mid i < j$ 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 $X_1, X_2, \dots X_{10}$. 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**, $WP = 0.5\text{s}$

2.2 Warm-Up Period Estimation

In the previous section we saw that the warm-up time is 0.5s, 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:

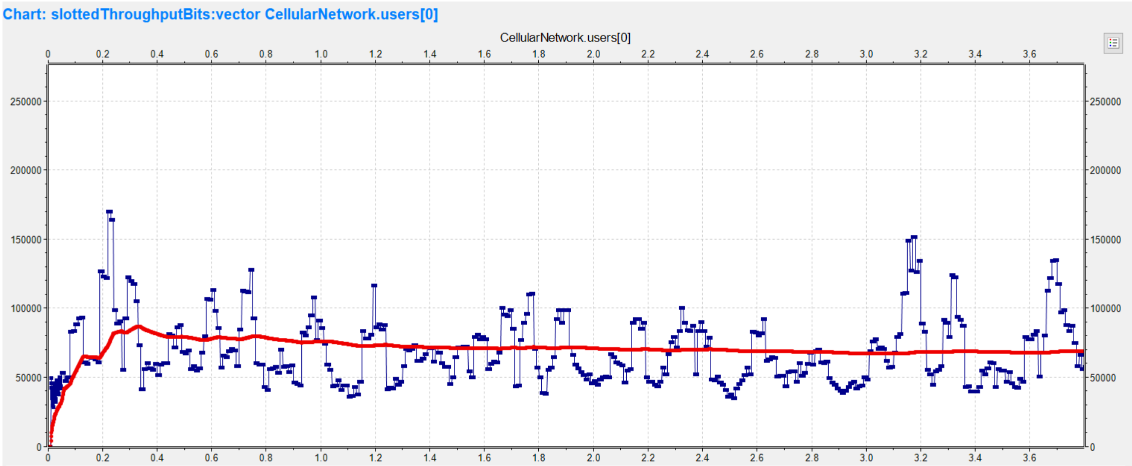

Figure 2.1: Worst Case for the Warm-Up period estimation

So we saw that in the worst case the graph converged to the sample mean after **0.5s** 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 1^{st} 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.

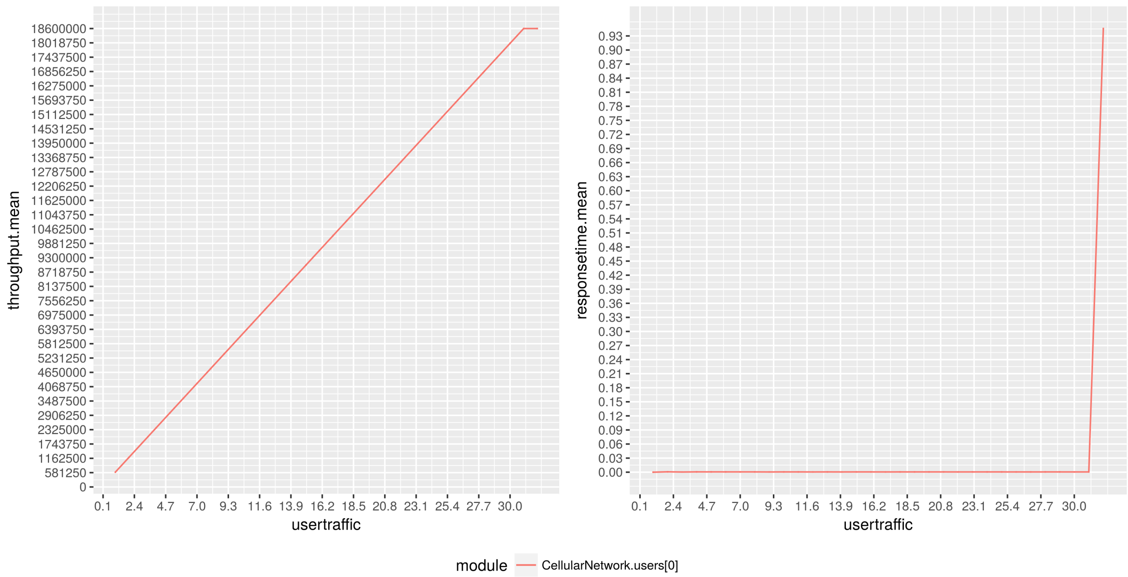

Figure 3.1: 1st validation scenario: throughput, response time

3.3 2^{nd} 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.

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

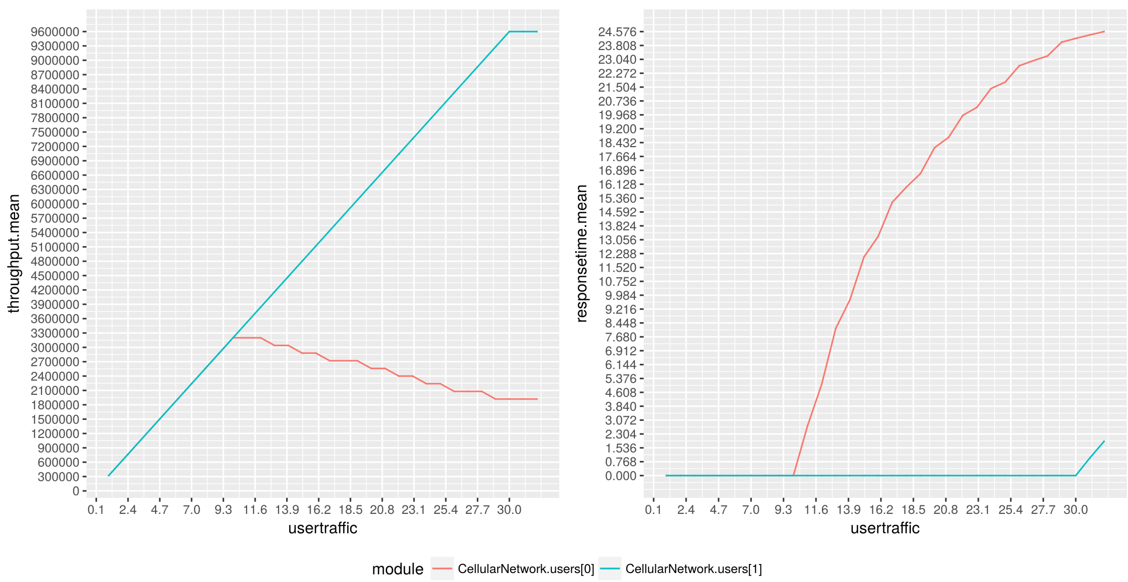

Figure 3.2: 2nd validation scenario: throughput, response time

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.

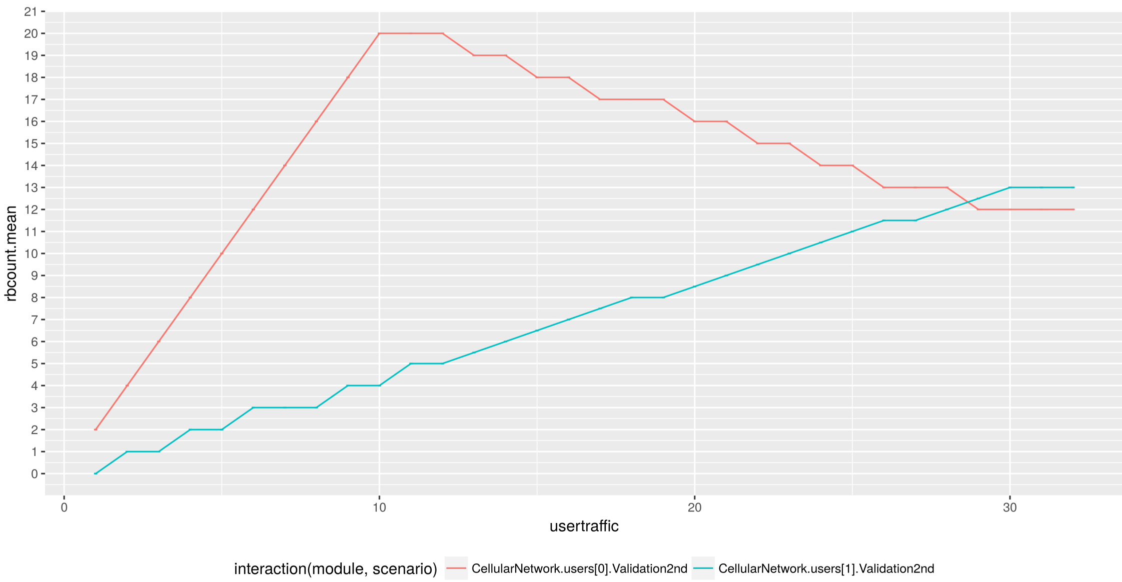

Figure 3.3: 2nd validation scenario: mean RB count

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, 12 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.

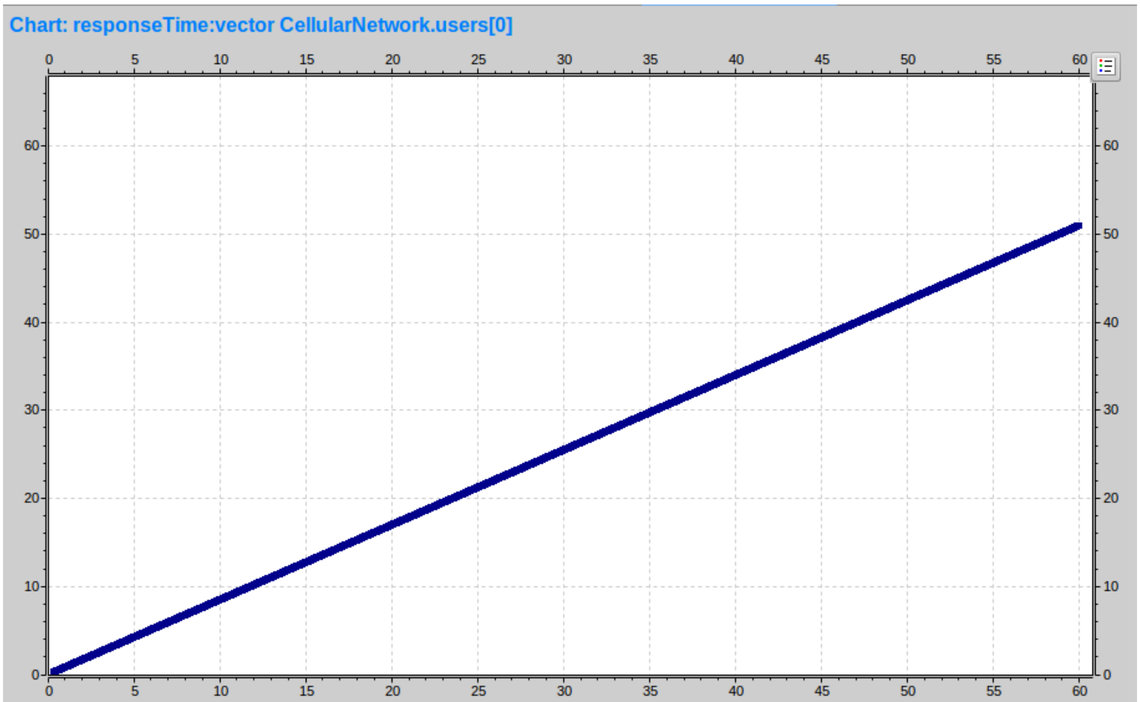

Figure 3.4: 2nd validation scenario: response time(vector)

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 $E[RT(ST)] \approx ST/2$. Note that if the simulation has an infinite duration, $ST = \infty$, $E[RT] \to \infty$ as expected.

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 *frame sharing*. As the requirements said, we have two *packet allocation policies* to implement: if a packet cannot completely fill the frame, it cannot be scheduled and the scheduler moves to the next user. Furthermore, every RB cannot contain packets from two or more different users. This means that there will be some wasted space to consider. The first policy is what causes *frame sharing*, because some totally free RB are used to empty to serve di other users.

A question now arises: how this packet allocation policies 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 frame sharing 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 throughput results:

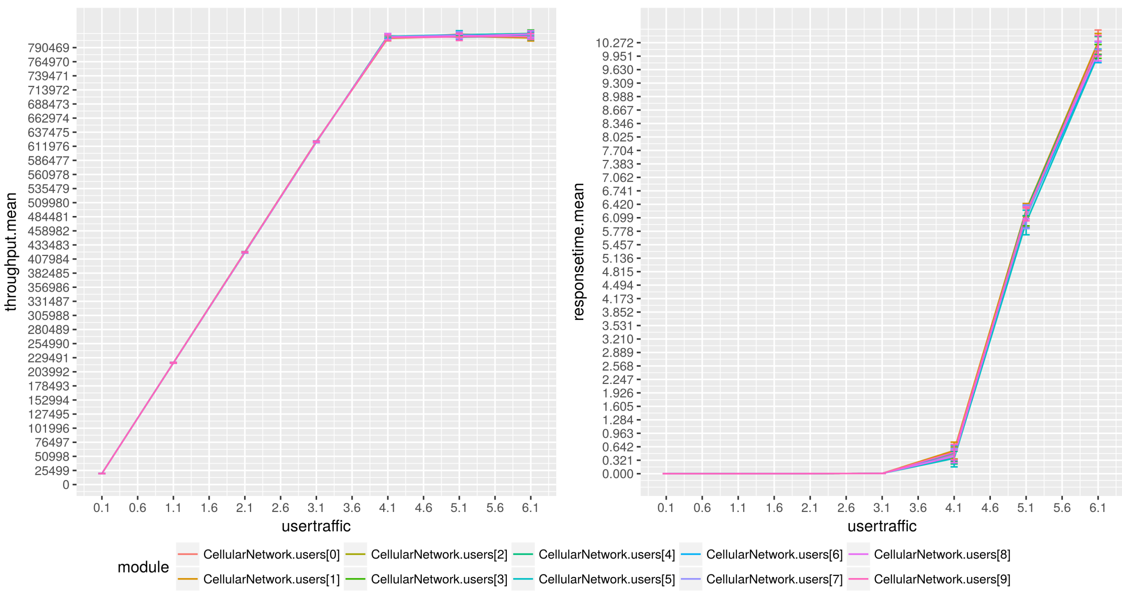

Figure 3.5: No-framing scenario: throughput, response time

First let's make sure that no *frame sharing* happens at the higher throughput rates: $lambda_{sat}$ is at about 4.1 and we can see from the next graph that there is no *frame sharing* at all over that rate! This graph measures basically how many RBs are shared to users other than the current served user.

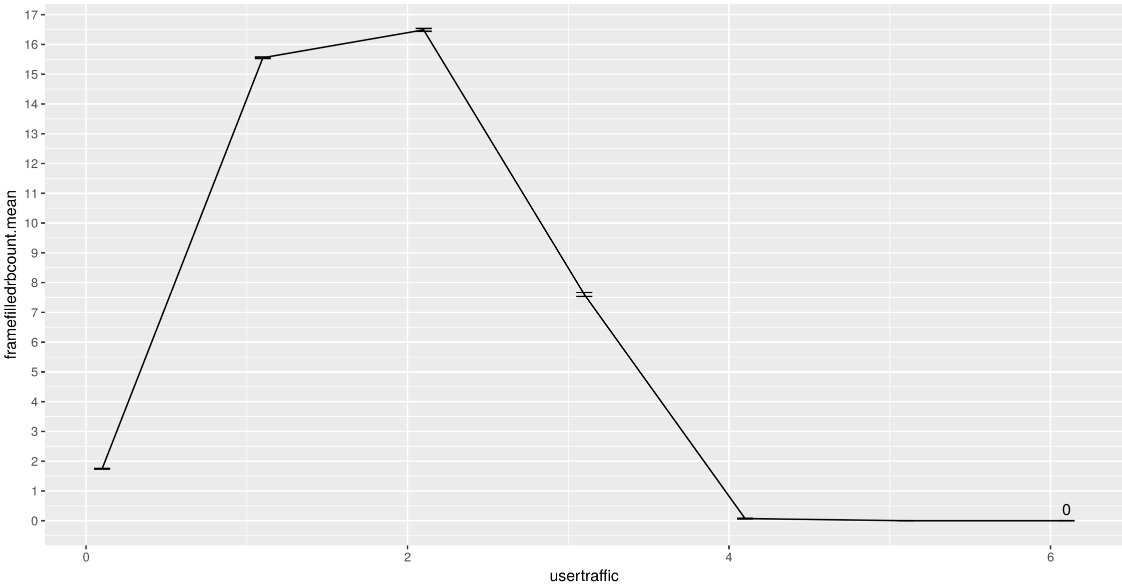

Figure 3.6: Mean nfill graph, NoFramingTest scenario

Note that for $\lambda_i < \lambda_{sat}$ *frame sharing* is still present because $nfill > 0$ and that is what we expected.

Now we can extract the throughput at λ_{sat}, which is $8127675 \pm 13109\ bit/s \simeq 8.128 \pm 0.131\ Mbit/s$.

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

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

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

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

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

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

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** the throughput simulated result!

This test is not only for a validation purpose, but it is a starting point to study how frame sharing 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. This setting simulate quite good the random way point scenario, where users move in the space. The user which moves n some slot receive strong signal, so it sends high CQI, in others receive weak signal so it sends low CQI.

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 packet allocation policies on 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.

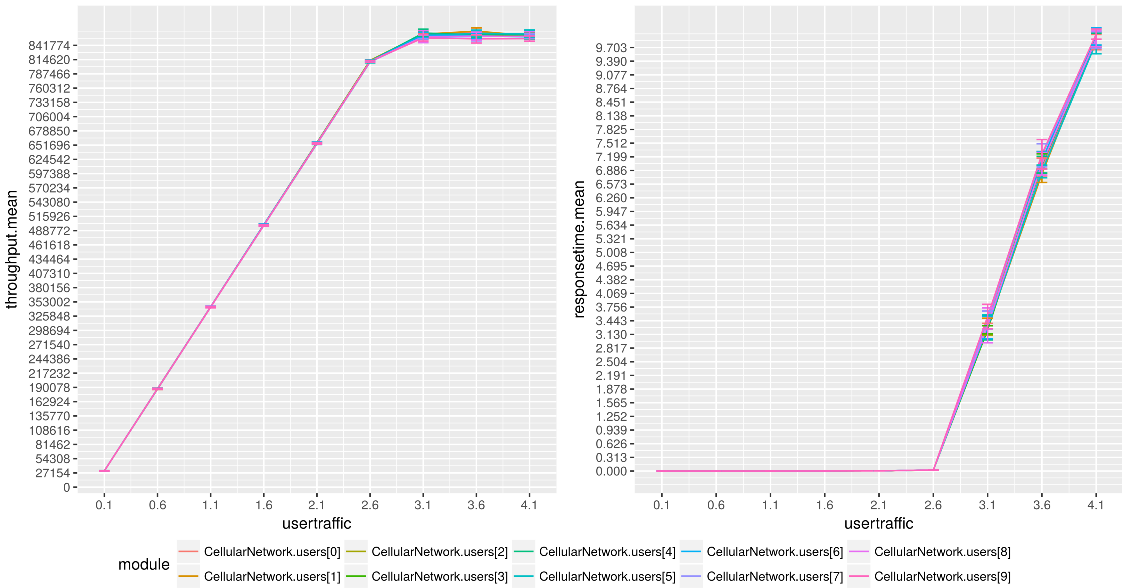

Figure 4.1: Uniform scenario - Fair: throughput, response time

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 1^{st} 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. In the following plot we can se the trend of response time for $0 < \lambda < \lambda_{sat}$. It is evident that when the saturation point is reached the response time grows exponentially.

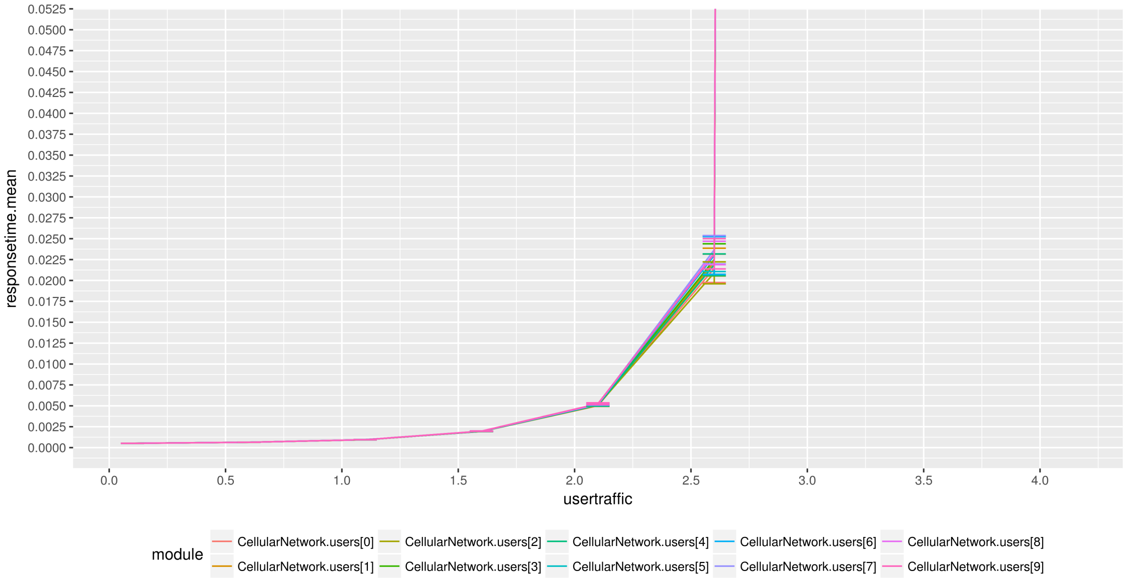

Figure 4.2: Uniform scenario: response time zoom

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

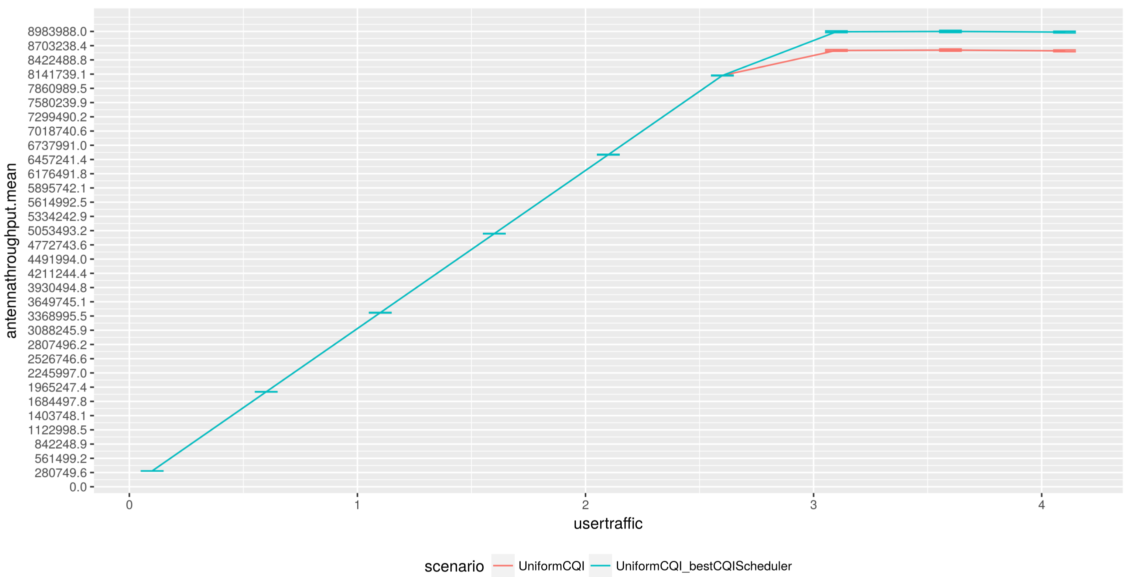

Figure 4.3: Antenna throughput: UniformCQI, Validation-NoFramingTest

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:

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

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:

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

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 prove 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 $th_{antenna} = 8616138 \pm 14206$ bps, *conf lvl 95%*
- User throughput $th_i = 861031 \pm 4076$ 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 **frame sharing** the performance of both scheduler would be the same but, as seen before, **frame sharing** is present and has a remarkable impact on performance. We can expect that this scheduler will have better performance in throughput since it fills the residual space of the frame by first serving users with the highest CQIs.

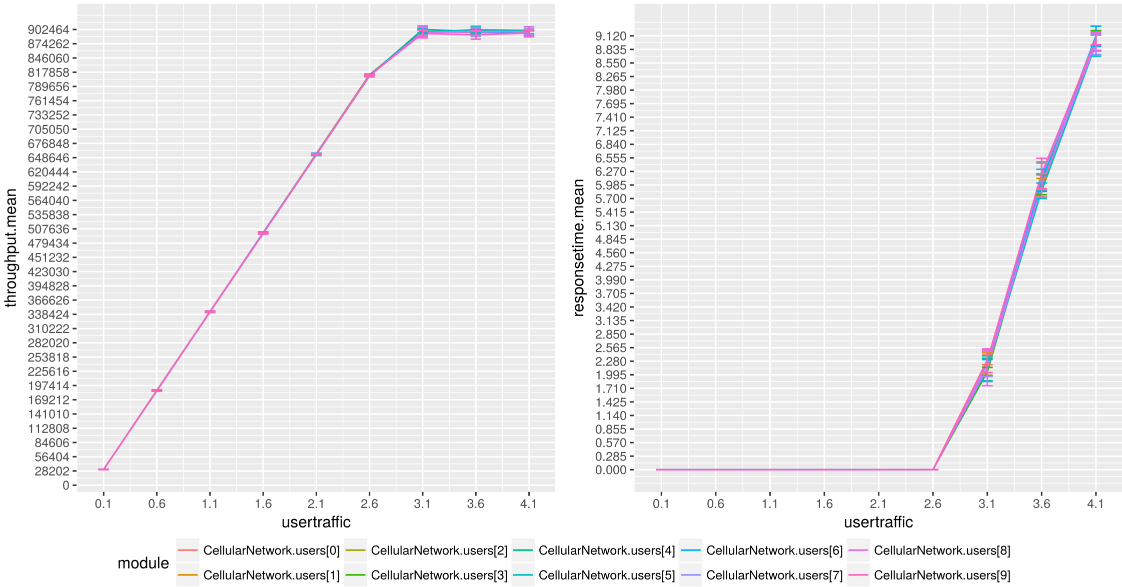

Figure 4.4: Uniform scenario - BestCQI: throughput, response time

By analyzing the response time graph in the interval $0 < \lambda < \lambda_{sat}$ we observe cleary that is is smaller than the previous case and it has a spike around saturation point.

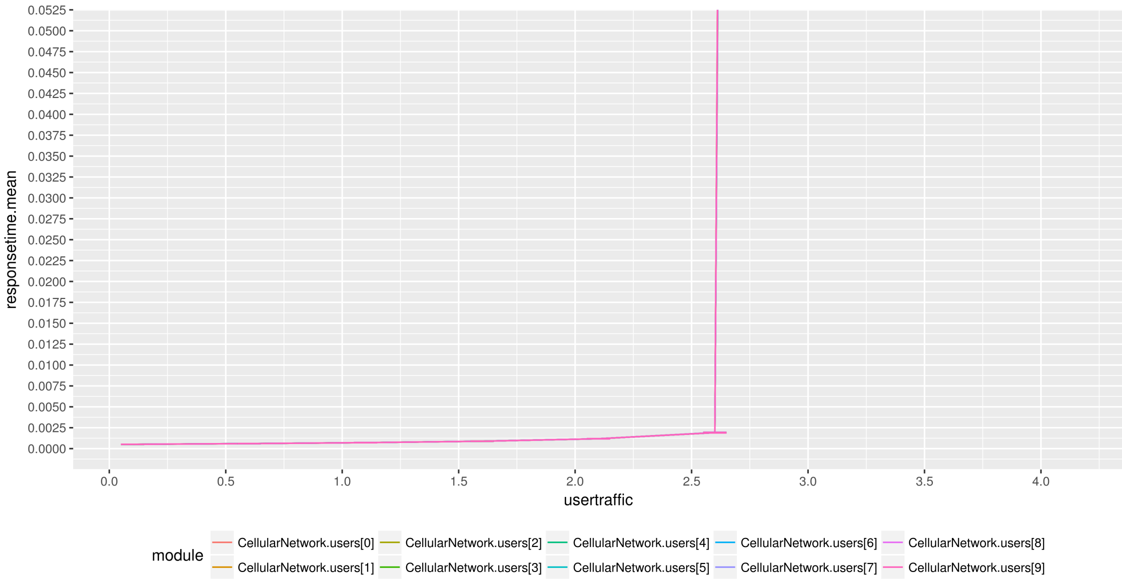

Figure 4.5: Uniform scenario - BestCQI: response time zoom

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 response times with respect to the 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 the best CQI user can use the higher resource block size at that moment. As a proof of this statement we can see how much RBs are used at a fixed lambda rate:

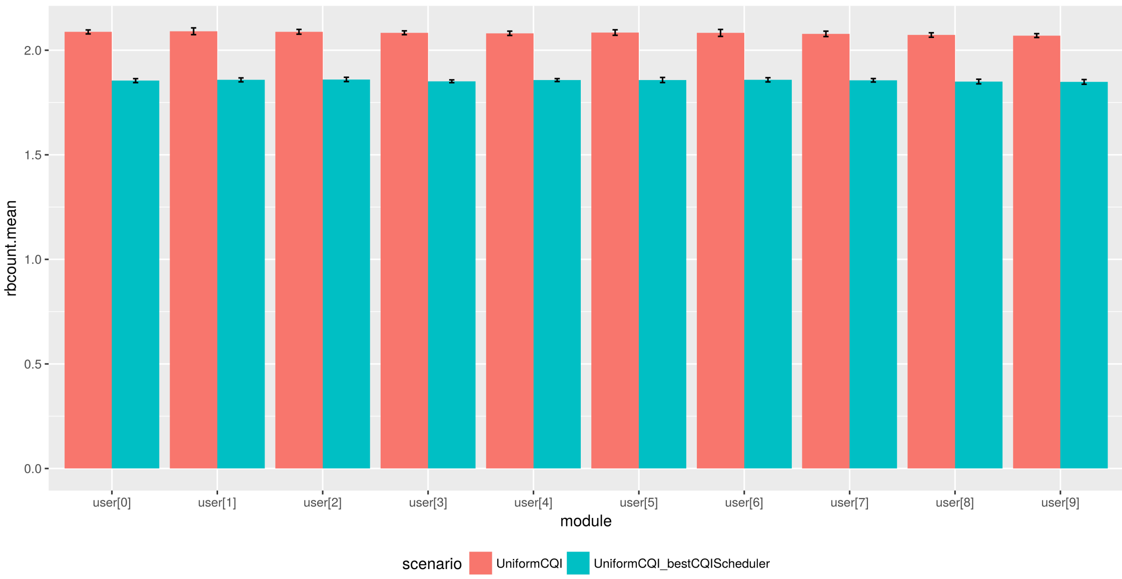

Figure 4.6: Mean Used Resource Blocks at $\lambda = 1.1$
for each user, comparing Uniform and Uniform BestCQI

As we can see all clients use less RB blocks because of the higher residual frame size, leading to a more efficient use of the transmission channel.

We can also see that the throughput distribution is fair among the users checking the Lorenz Curves for throughput and response time.

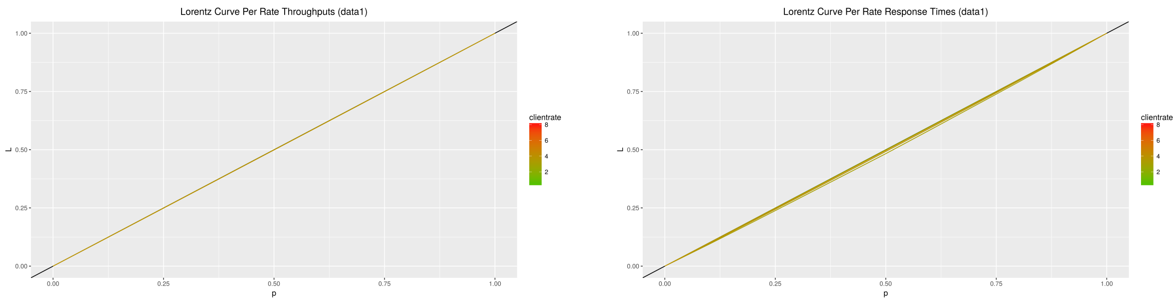

Figure 4.7: Uniform scenario - BestCQI: lorenz curves

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 $\lambda = 3.1$.

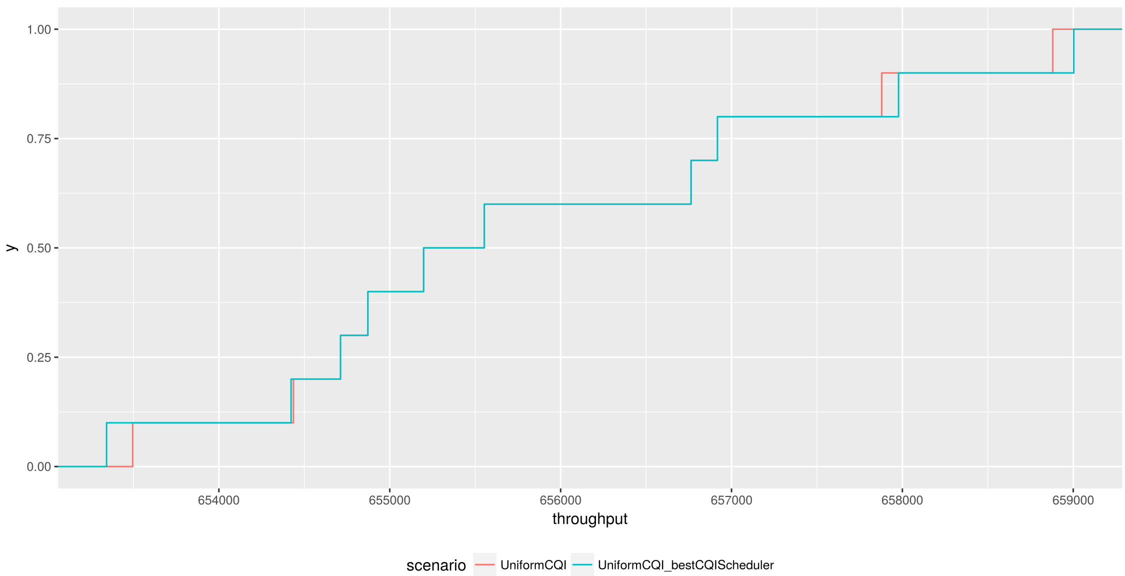

Figure 4.8: Uniform scenario: ECDF comparison $\lambda = 2.1$

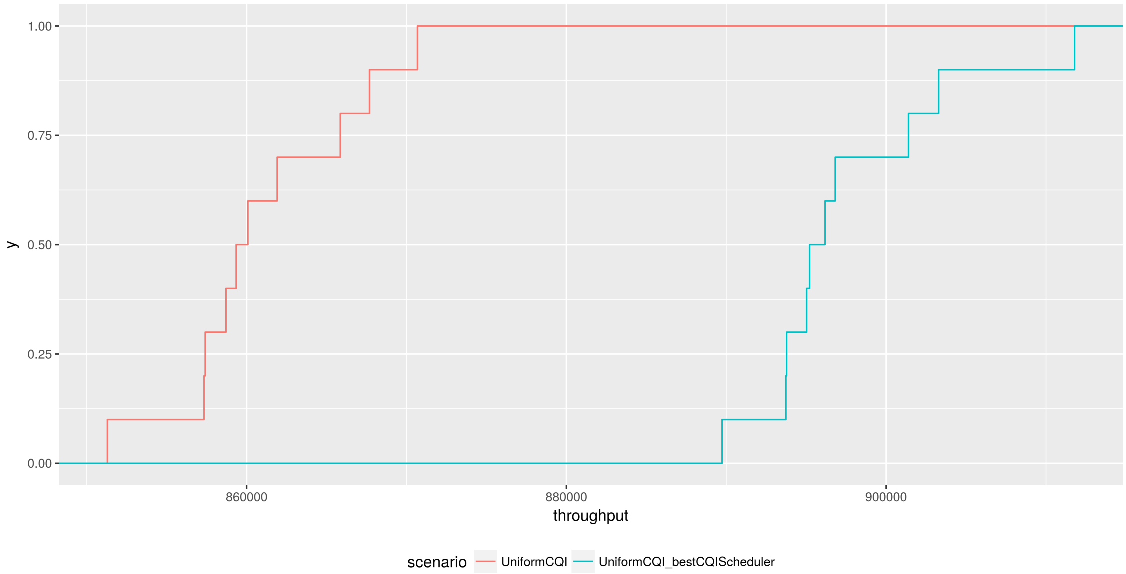

Figure 4.9: Uniform scenario: ECDF comparison $\lambda = 3.1$

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 because CQI and service demand has the same uniform distribution for all users. The result is that each user exploits an equal slice of available resources to the antenna.

4.3.1 Frame Sharing effects

As we said in the previous chapters, we suspected that this policy could negatively influence throughput and response time, but we proved this is wrong (starting from NoFramingTest results). In this case we can also see that all users benefit from a more efficient use of the channel: infact when a user leaves some unfilled frame space to another one, it let the other user to use more efficiently that space (but for itself). However CQIs are uniformly distributed, so there is a good chance that roles could switch in the next round. At the end, the policy leads to a sort of mutually beneficial channel sharing.

As a final comparision, let's check global antenna throughput values for NoFramingTest, Uniform Fair and Uniform Best CQI scenarios when all client arrival rates are beyond λ_{sat}:

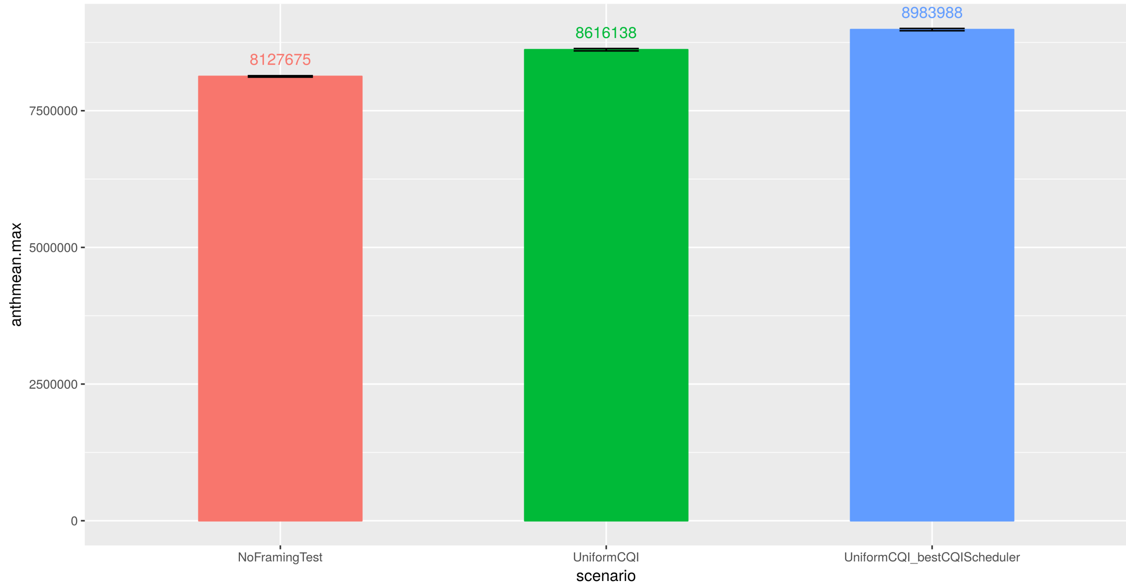

Figure 4.10: Uniform scenario: ECDF comparison $\lambda = 3.1$

At the end we can see that *Uniform Fair CQI* result lies between *NoFramingTest* and *Uniform BestCQI* results. This can be explained considering what we said before about the distribution of the residual frame size. “The user CQI is high” is the main policy of BestCQI scheduler. However, in Best CQI scenario, we can’t consider “The packet is small” because Best CQI user is **always** selected as first for the residual frame filling, and this increases the throughput for that scenario, w.r.t. Uniform CQI Fair scenario.

Chapter 5

Simulation: Binomial Scenarios

5.1 Introduction

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 $X_{i,t_j} \sim Bin(14, p_i)$, 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 means we chose the following values for parameters p_i.

Listing 5.1: omnet.ini - p parameters

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

As in the previous scenario we will analyze the performance about throughput and response time by using both schedulers. Before doing simulation we can do some conjectures. Note that in this scenario mean CQI are sensibly different so we expect that $user[9]$, that has highest probability to generate high CQI, will have highest throughput. However the two different scheduler will influence the throughput and response time in different ways.

These are the simulation parameters used in both binomial scenarios:

- $n = 10$
- $packetsize_i \sim U(3, 75), \quad 0 \leq i \leq n-1$
- $CQI_i \sim Bin(14, p_i) + 1, \quad 0 \leq i \leq n-1$
- $\lambda = \lambda_i = 0.1 + 0.5k, \quad k \in \{1, 2 \ldots, 19\}, \quad 0 \leq i \leq n-1$

5.2 Binomial, Fair Scheduler

In this scenario the scheduler is the basic Round Robin scheduler.

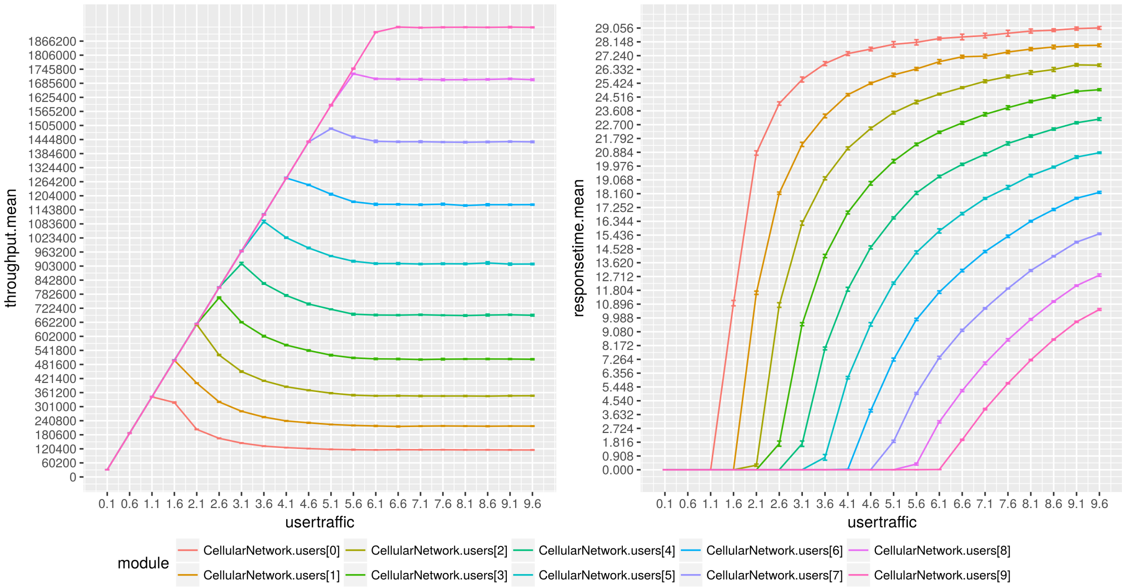

Figure 5.1: Binomial scenario - Fair: throughput, response time

Users have very different throughput as we expected. When $0 < \lambda \leq 1$, all users have the same throughput because the traffic is very low and the resources provided by the frame are quite enough to satisfy all users. However this is a very special case, we can see that as λ increases users having lower mean CQIs saturate. The scheduler is always fair but users with higher mean CQIs have bigger RBs inside the frame that can carry more data. Saturation points follow the distribution of CQIs among users, so users with lower CQI, on average, saturate before, otherwise users with higher CQI exploit better the frame and saturate with higher λ rate. By analyzing the throughput plot we observe that users have a small peak near the saturation point and then their throughput decrease and approches to an horizontal line. The explaination is quite easy. If λ rate is *small* users with higher mean CQI use few RBs to empty their queue, so the frame filling algorithm has free RBs to distribute among other users. When λ rate becomes *high* also users with higher mean CQI require the lots of RBs and the effect of residual frame fill is limited.

In the previous graph we observe that response time seems to approch to $ST/2$. As explained in the chapter 3.3 this strange behavior occurs when response time grows linear to infinity. In order to have a correct sight of what happen to response time we must zoom that plot.

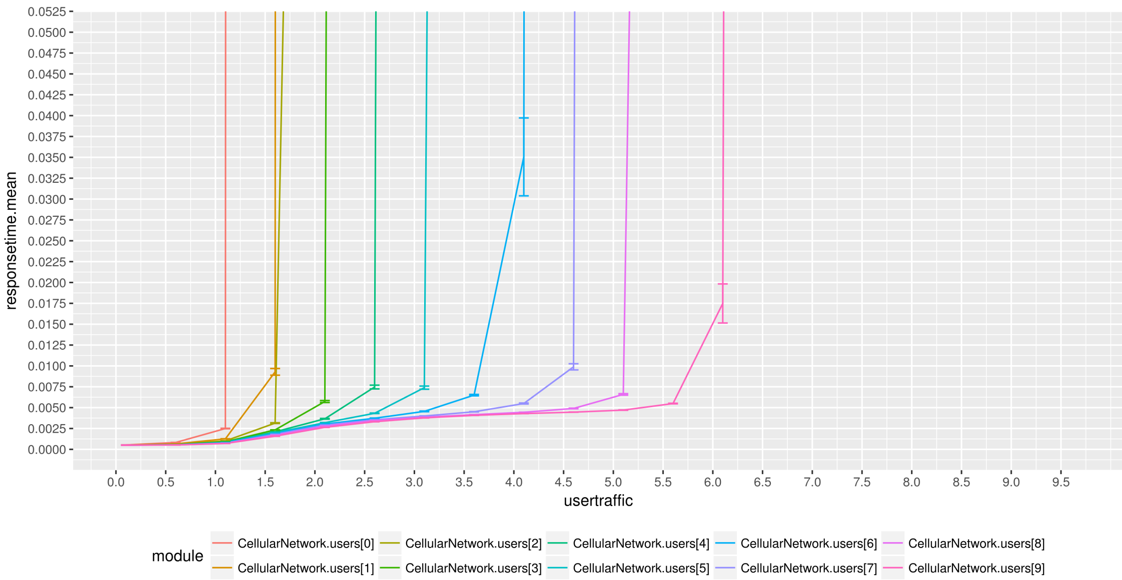

Figure 5.2: Binomial scenario - Fair: response time zoom

It is necessary to remark that the frame filling policy can have a strong impact on distribution of the throughput and response time. Let's consider, for example, a different policy:

- The residual space in the frame is filled by considering all users j, where $j = (currentUser + 1) \bmod n$ and so on, wrapping until user $(j-1) \bmod n$.

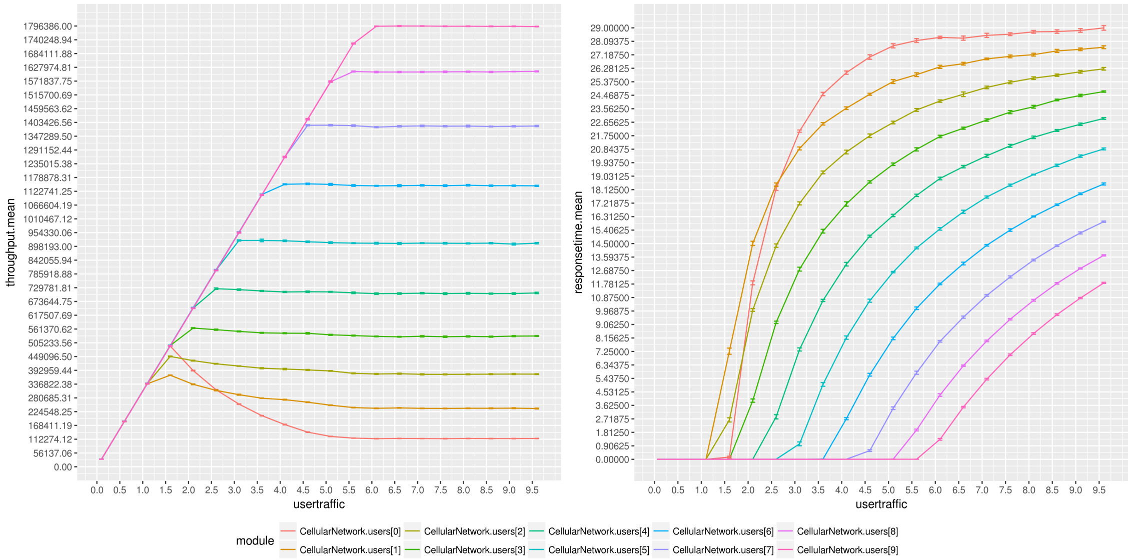

Figure 5.3: Binomial scenario, Modified scheduler - Simple: throughput, response time

If we consider the `user[0]` we can notice a very strange behavior. When $0 < \lambda \leq 1.5$ it has higher throughput and smaller response time than `user[1]` and `user[2]` and this seems senseless because `user[0]` has smaller mean CQI. However this odditity can be explained by observing that, in that range of traffic, `user[9]` requires a low number of RBs but the frame must be filled and the simple rule **always** chooses the next user first which is `user[0]`. When λ increases, `user[9]` requires more RBs, and the throughput of `user[0]` decreases.

In this condition the distribution of throughput is absoloutely no fair since there are privileged user, the ones near the antenna, and the others that receives signal of low power so cannot achieve the same throughput.

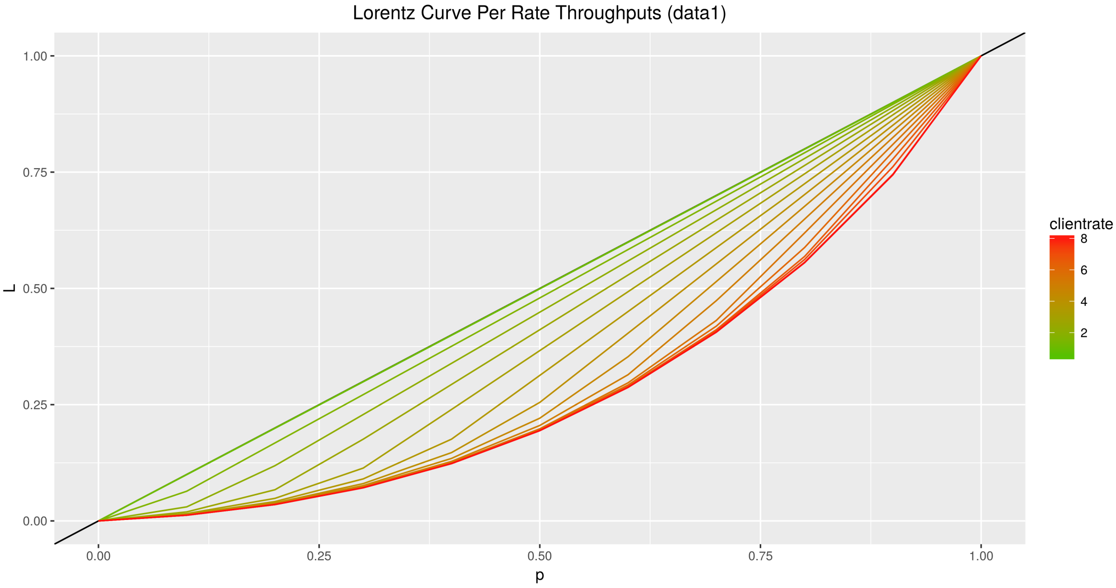

Figure 5.4: Binomial Fair - Throughput - Lorentz Curve

The following table shows the saturation point and the mean throghput per each user (confidence lvl 95%).

user	λ_{sat}	**throghput [bps]**
0	1.1	342855 ± 1027
1	1.6	499558 ± 1057
2	2.1	652748 ± 2600
3	2.6	767444 ± 3177
4	3.1	913755 ± 4954
5	3.6	1093554 ± 5318
6	4.1	1279993 ± 3140
7	4.6	1434389 ± 1780
8	5.6	1726384 ± 2218
9	6.1	1904459 ± 1880

5.3 Binomial, Best CQI Scheduler

Now let's turn on the BestCQI policy. We are somehow expecting the same base result as the Fair scheduler, this is because *frame sharing* pulls performances much less than the main filling policy (the $currentUser$ fills up the frame until queue is empty or next packet is too big to fit). However we expect an higher throughput for higher CQI users and lower throughput for lower CQI ones w.r.t the Fair Scheduler, mainly because residual frame filling now gives priority to higher CQI.

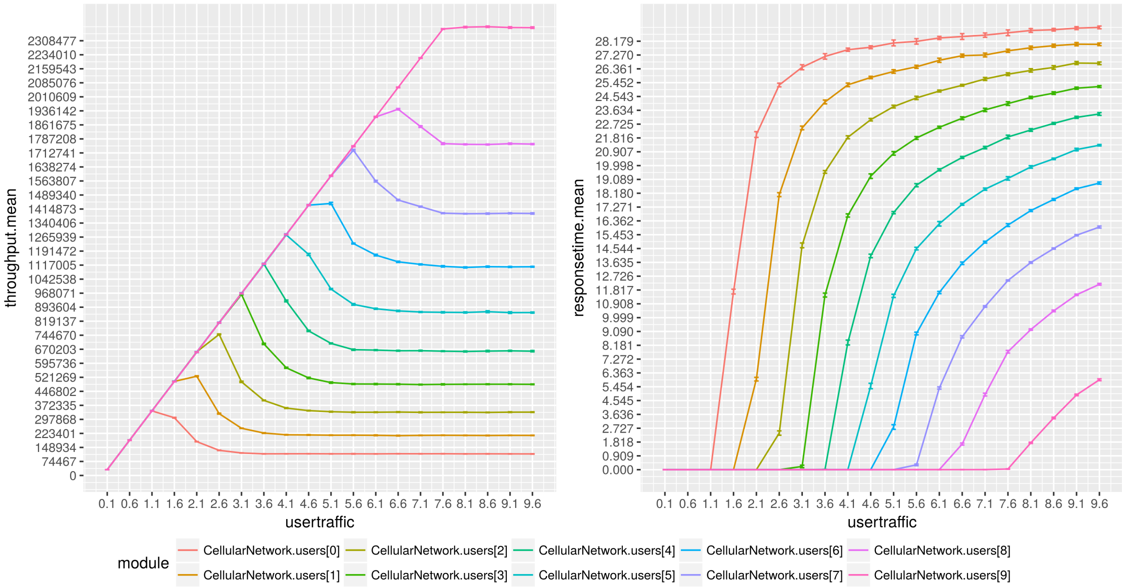

Figure 5.5: Binomial BestCQI scenario: throughput, response time

As we saw before, saturation points are different for each user and depend on CQI mean. However this time each throughput line has an shorter tail w.r.t. the previous experiment results, so this means that we have an higher value around the peak and lower values for higher λ. Instead, for `user[0]` and `user[9]` there is no difference in line shapes. How can we explain this results?

As λ gets higher, each user come to its saturation point: the first is `user[0]` which will use some RBs from the other users before reaching the peak, just as we saw in the last chapter. The next user to get to saturation point is `user[1]`, which will start to reclaim the residual frame and, eventually, start to use RBs from other users. This is also what we saw before, but there is an important difference: `user[1]` has priority over `user[0]` when doing the residual frame filling, because it has an higher CQI. This means that `user[1]` will eventually take *all* the residual RBs for himself at some point, leaving `user[0]` without extra RBs. In the same way, when `user[2]` start to reclaim its RBs and then the other users residual frames, it basically takes all extra RBs also from `user[1]`, and we can continue until `user[9]`, which, at the end, takes all! We basically explained:

- The higher peak, due to the higher number of reclaimed extra RBs (from other users residual frames)

- The shorter tail, which is caused by having less RBs when higher CQI users reclaim all the extra RBs

- The unchanged shape of `user[0]` and `user[9]`, the first because it is the first to reclaim residual frames from the other users, and the second because it is the higher CQI and reclaims all at the end (highier value of λ)

If we want to compare shapes, let's take, for example, `user[5]` and compare in both experiments, plotting them:

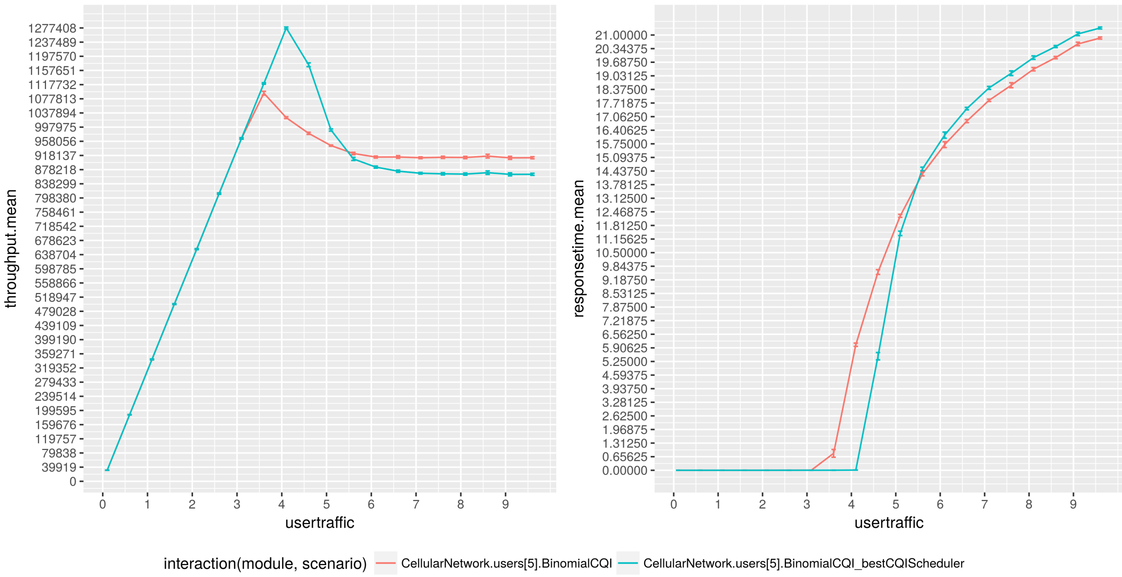

Figure 5.6: Binomial Fair/Binomial BestCQI comparison - throughput, response time

Let's check how simulated resource blocks are distributed among users at rate $\lambda = 4.1$ (around `user[5]` throughput peak), comparing the two experiments:

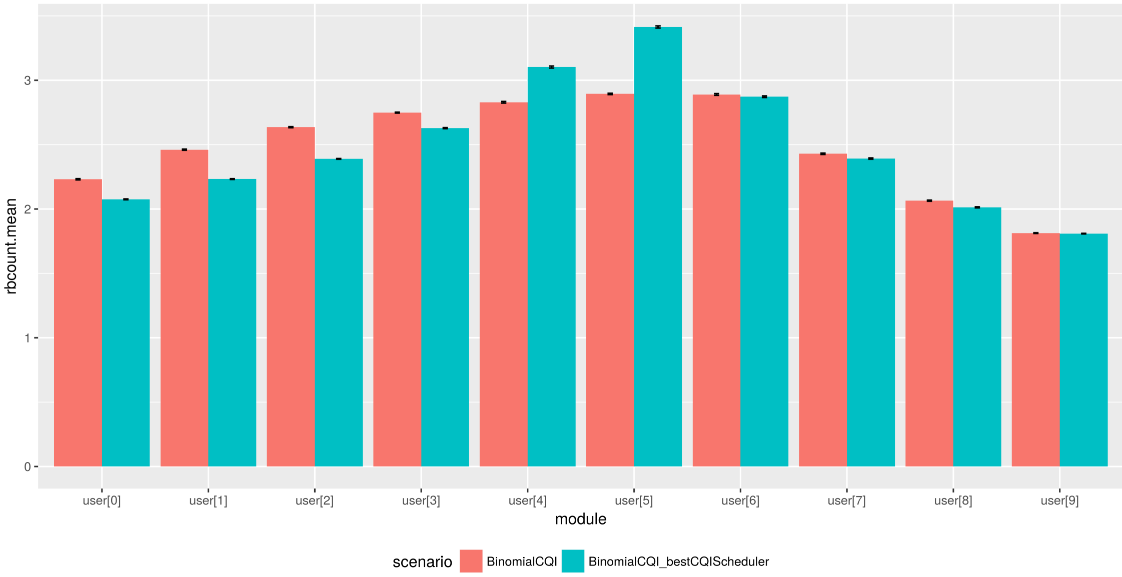

Figure 5.7: Binomial Fair/Binomial BestCQI comparison - Resource Block count

`user[5]` basically reclaimed all the missing RBs of lower CQI users, while instead they are fairly distributed among users when using the Fair CQI policy.

RB distribution unfairness can be confirmed also using a Lorenz Curve graph:

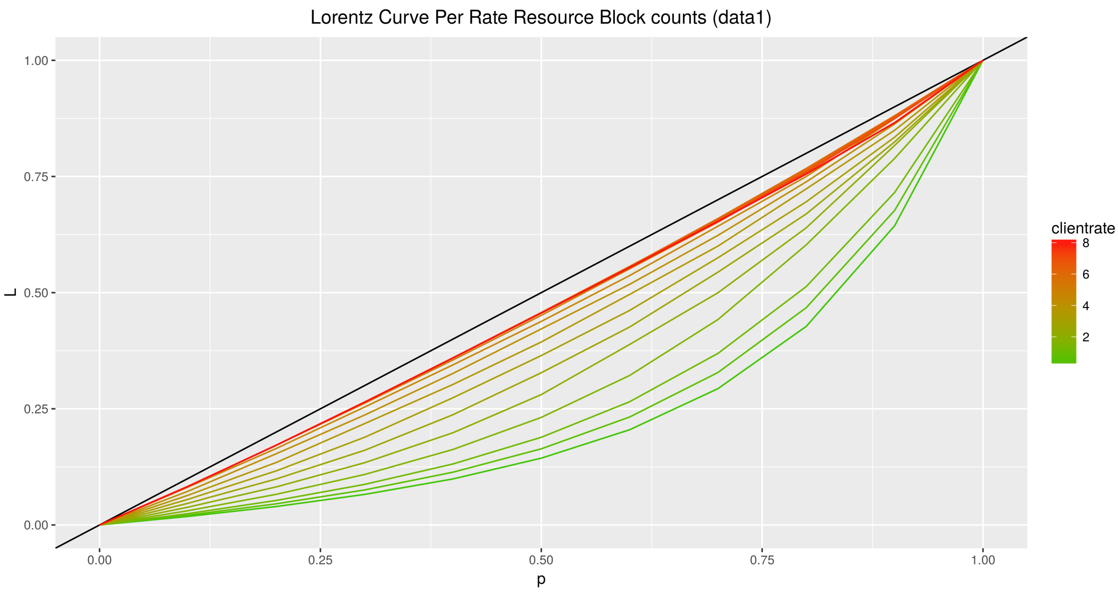

Figure 5.8: Binomial BestCQI scheduler - Resource Block - Lorenz Curve

The graph in 5.8 shows curves for each value of λ. We can see that fairness is higher at the end just because `user[9]` takes all the extra RBs and leaves the other users with nothing that their own RBs, which are of the same count for all (25). On the other hand Fairness is higher when λ is low because first users takes all the extra RBs from the higher CQI users, and the higher CQI users use less RBs due to the very low interarrival rate.

Throughput unfairness can also be shown using a Lorenz Curve graph:

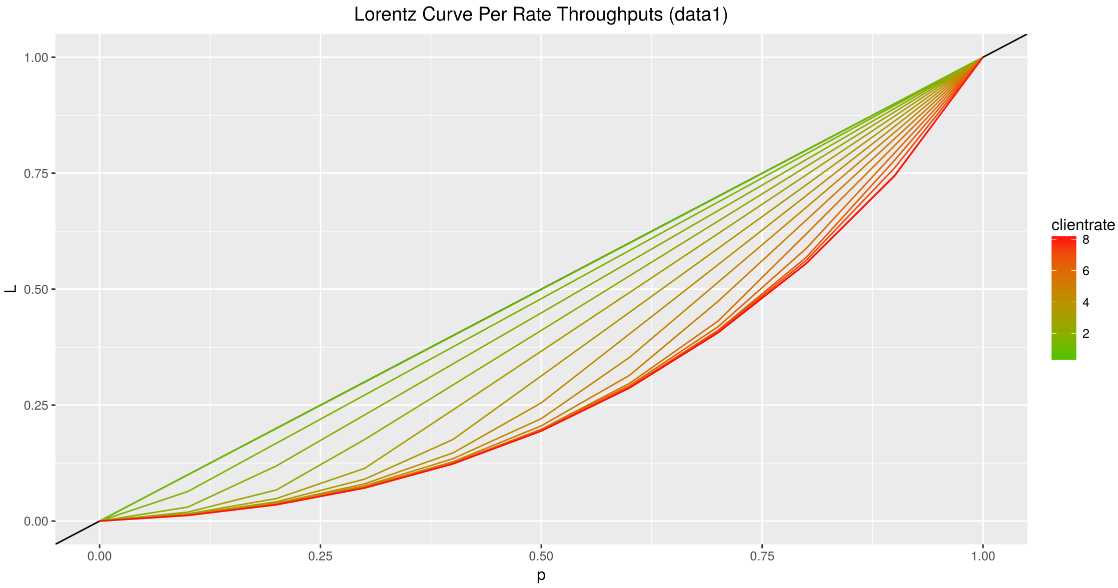

Figure 5.9: Binomial BestCQI scheduler - Throughput - Lorenz Curve

In this case we must consider that this measure heavly depends on the CQI distribution: the higher unfairness is shown when λ is high, because `user[9]` uses all extra RBs, and the lower unfairness is shown for low λ values, where no one has yet reached saturation.

Now let's take a zoom to the plot of response time. If we compare the following graph with the Binomial Fair one, we note that saturation points are shifted to right.

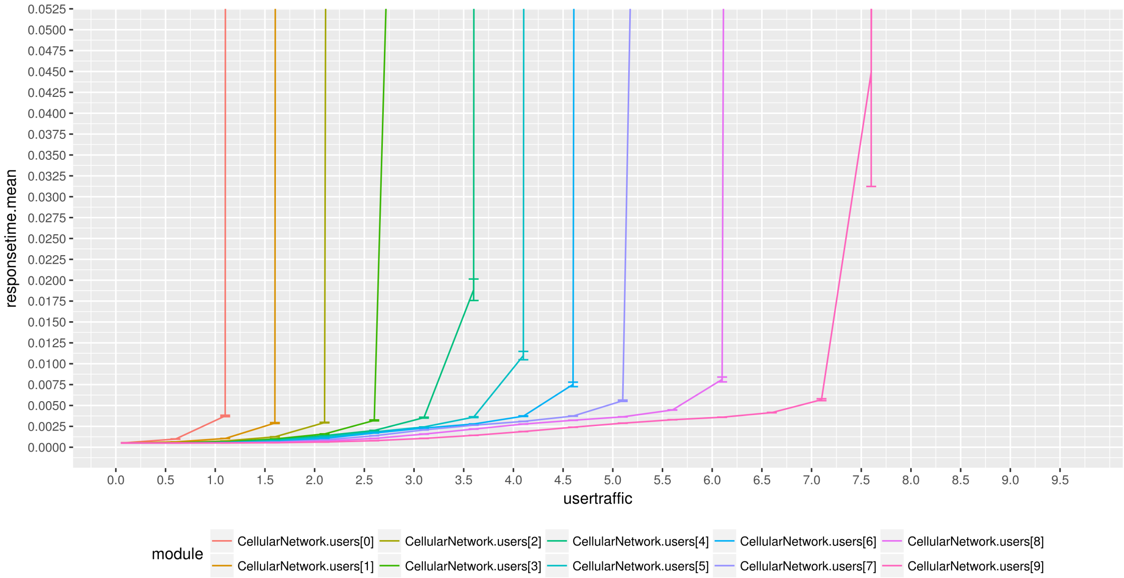

Figure 5.10: Binomial scenario - BestCQI: response time zoom

As explained before this is due the extra RBs that users with higher mean CQI can use for the frame filling. The most noticeable change is for `user[9]`. In the Binomial Fair scenario its response time has a spike around $\lambda = 6.1$, now it is bounded and well defined until $\lambda = 7.1$

In the following table there is a summary of throughput in λ_{sat} per each user.

user	λ_{sat}	**throghput [bps]**
0	1.1	342852 ± 1028
1	1.6	499580 ± 1092
2	2.6	749257 ± 2282
3	3.1	961496 ± 2240
4	3.6	1123105 ± 1315
5	4.1	1277408 ± 2667
6	4.6	1435511 ± 2868
7	5.6	1727047 ± 4506
8	6.1	1904025 ± 2349
9	7.6	2370866 ± 1914

Let's now compute the antenna total throughput:

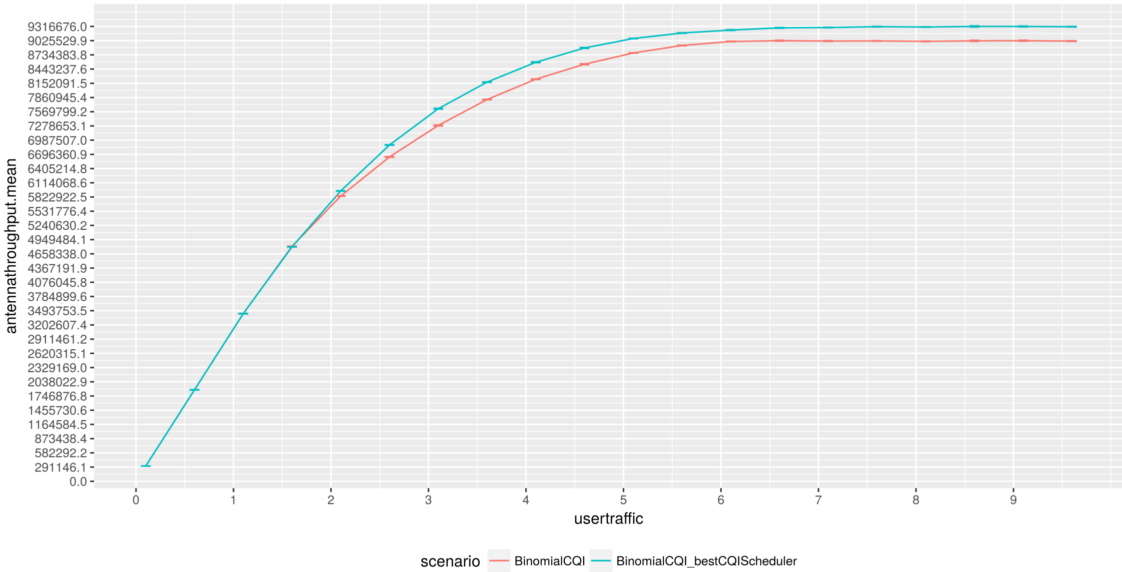

Figure 5.11: Binomial Fair/Binomial BestCQI comparison - Total Antenna Throughput

Binomial BestCQI seems the best choice to maximize this throughput. We must note that the BestCQI algorithm basically reallocates residual frame RBs to the users with the best CQI, but the total number of RBs remains the same: however better CQI correspond to higher RB sizes, which will increase the total antenna throughput.

So, is Best CQI policy scheduler the better choice for Binomial CQIs scenarios? It depends. As we saw, giving priority to users with higher CQI leads to a decreased unfairness. However channel usage efficiency will go up as the throughput for users with higher CQI, but the gained throughput will be shared only among higher CQI users. Response time just follows the throughput trend.