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

Modelling

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 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 rimulations 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:

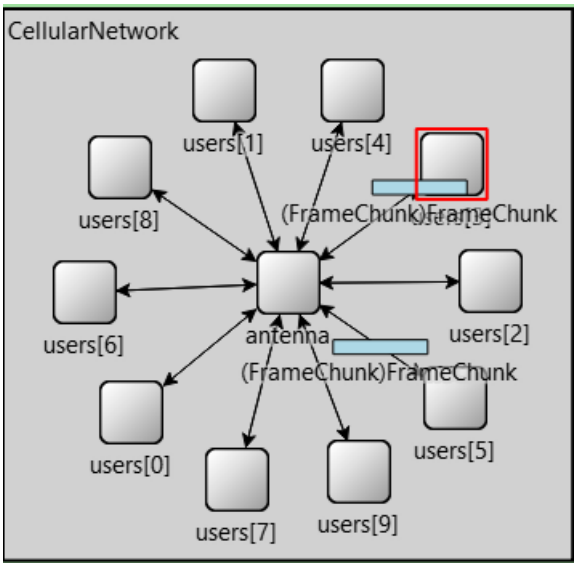

Figure 1.1: Simulated Network (omnet++)

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 $\texttt{FIFOQueue}_{\text{i}}$. 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 $\texttt{FIFOQueues}_{\text{j}}$ $j \in \{(i+1)\text{mod } n, (i+2)\text{mod } n, \ldots\}$
- **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.

1.4 Constants

In the following chapters, if not different specified, we will consider this constants.

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

Chapter 2

Validation

2.1 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.

2.2 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

$$th_{\text{in}} = \frac{packetsize \times 8}{1/(1000\lambda)}[bps] \tag{2.1}$$

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.

$$th_{\max} = \frac{\#RB \times RBsize_{\max} \times 8}{T_{\text{slot}}} = \frac{25 \times 93 \times 8}{0.001} = 18.6 \text{ Mbps} \tag{2.2}$$

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

$$\lambda_{\max} = \frac{\#RB \times RBsize_{\max}}{1000 \times T_{\text{slot}} \times packetsize_{\max}} = \frac{25 \times 93}{1000 \times 0.001 \times 75} = 31 \text{ ms}^{-1} \tag{2.3}$$

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.

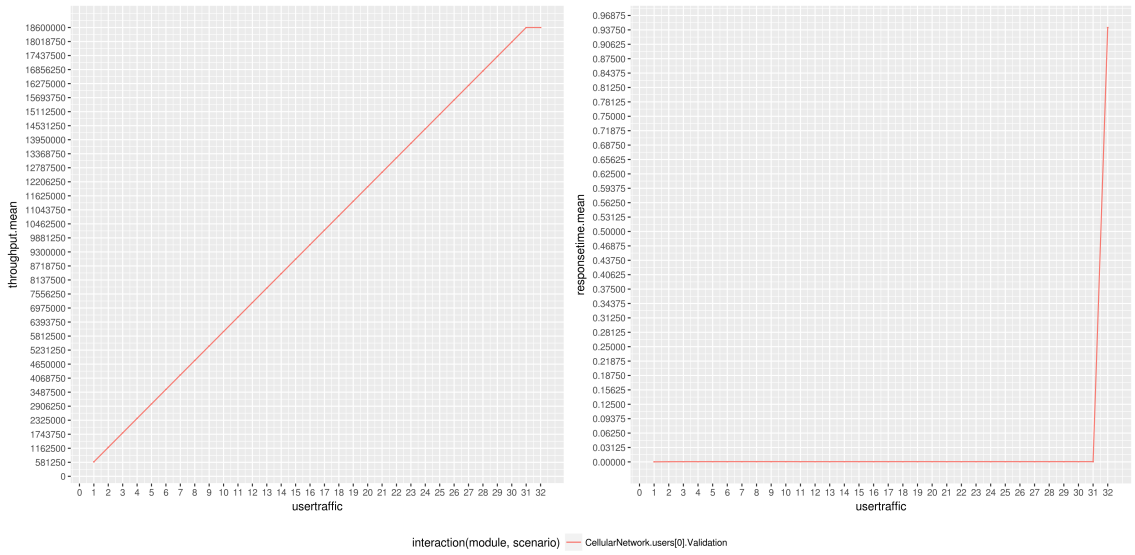

Figure 2.1: 1st validation scenario: throughput, response time

2.3 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

$$th_{\text{in}} = \frac{packetsize_0 \times 8}{1/(1000\lambda_0)} + \frac{packetsize_1 \times 8}{1/(1000\lambda_1)} \tag{2.4}$$

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.

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

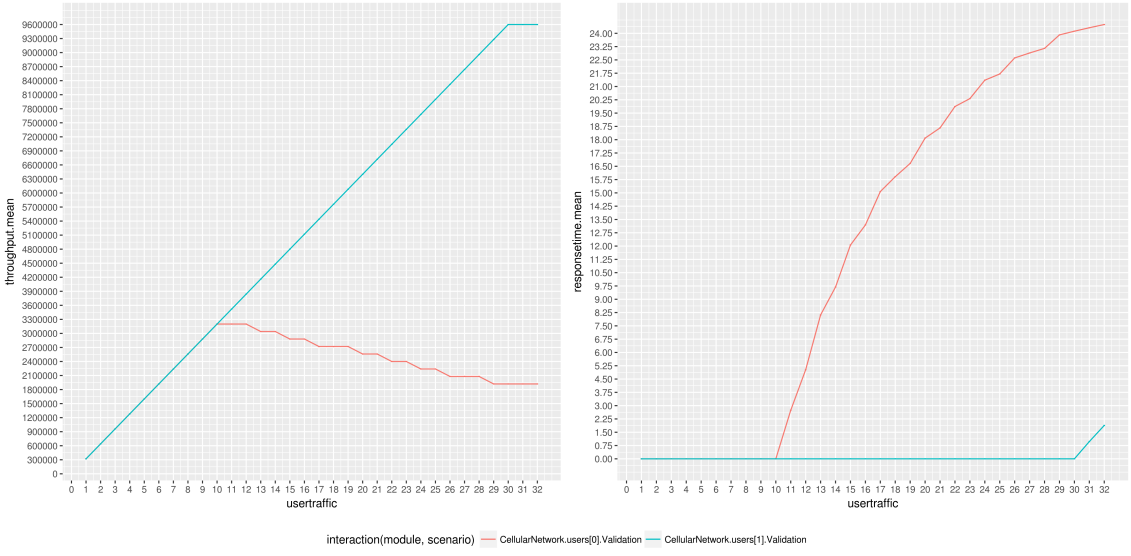

Figure 2.2: 2nd validation scenario: throughput, response time

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.

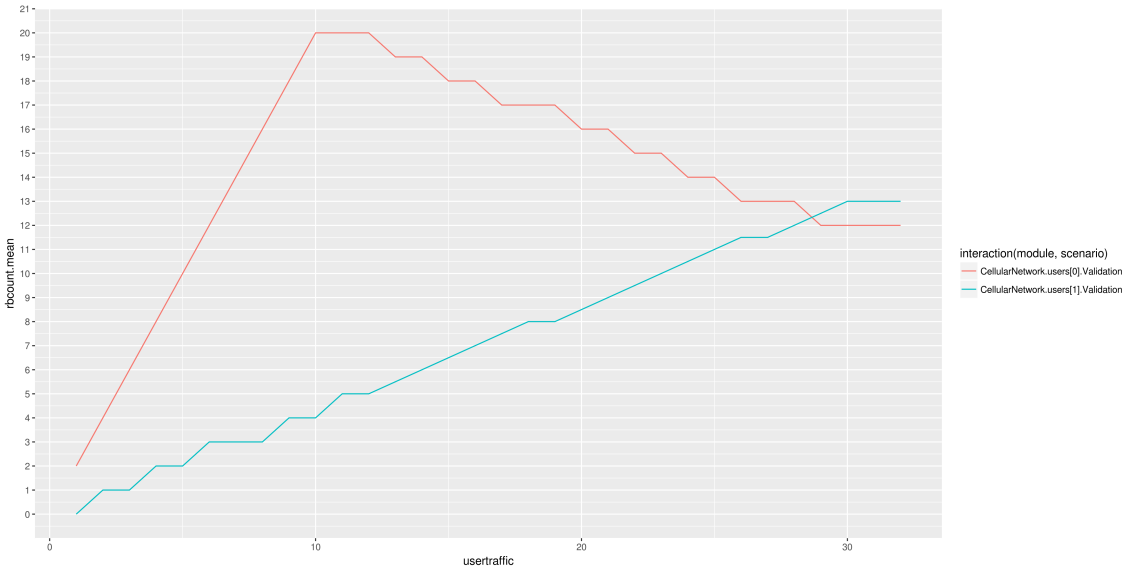

Figure 2.3: 2nd validation scenario: mean RB count

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 $lambda = 40$.

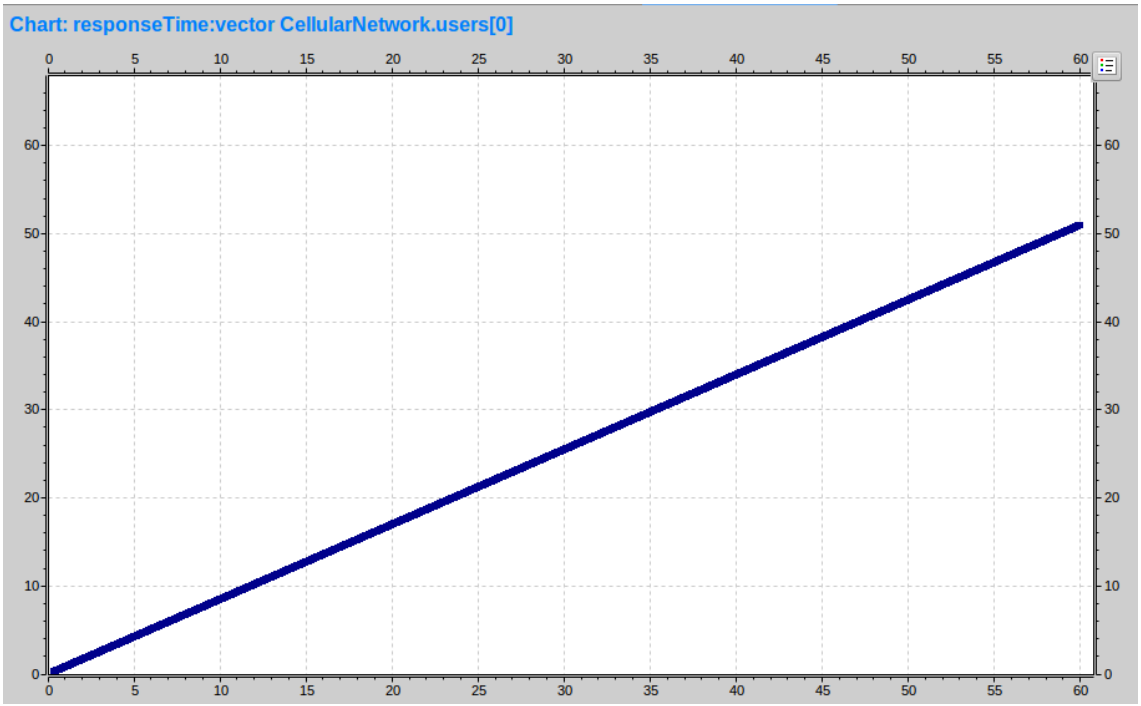

Figure 2.4: 2nd validation scenario: mean RB count

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:

$$RT(t) \approx \frac{\max(RT)}{ST} t, \quad 0 \leq t \leq ST \tag{2.5}$$

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.*

$$\begin{aligned} E[RT] = \frac{\sum_{t=0}^{ST} RT(t)}{ST} = \frac{\sum_{t=0}^{ST} \frac{\max(RT)}{ST} t}{ST} = \frac{\max(RT)}{ST^2} \sum_{t=0}^{ST} t = \\ = \frac{\max(RT)}{ST^2} \frac{ST(ST+1)}{2} = \frac{\max(RT)}{2} \frac{ST+1}{ST} \approx \frac{\max(RT)}{2} \end{aligned} \tag{2.6}$$

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