# OMNeT++ Fuzzy IP Scheduler

Overview, implementation, and analysis

Tomas-Adrian Boboi
Universitatea Politehnica Timișoara

2021
March

# Contents

# 1 Original assignment requirements

## The simulation model

The simulation model consists of the following OMNeT++ modules:

1. A number of mobile users. In the first stages of the model you can implement two identical users, then you can consider a number of K users, organized as an array of users. Each user generates IP packets according to a certain pattern: e.g. an IP packet at certain (random) time intervals, or a user generates files, a file consisting on a (random) number of IP packets. At the beginning the IP packets can be considered of fixed length, i.e. one IP packet = 1500 bytes. A user has a certain priority. There can be for example 3 or 4 priority levels. For 3 priority levels, they are (in increasing order): LP (low priority), medium priority (MP) and HP (high priority). For 4 levels, they are: non real-time (nrt) low priority and nrt HP, real-time (rt) LP and rt HP.

2. A scheduler. The scheduler is situated in the same Omnet module as the queues. The queues are not per user, but per priority class. This means that the packets arriving from LP user will be stored in the LP queue, the packets from MP and respectively HP users will be stored in the MP and respectively HP queue. The scheduler reads the lengths of the queues and implements a scheduling algorithm that determines which queue will send data. The sending of a packet takes a time equal to its length divided by the line rate (i.e. 1500 bytes / 1 Mega bit per second). The scheduler cannot send another packet during this time interval.

3. A sink. The sink models the destination of the data. When the data (i.e. IP) packets created by an user arrive to the sink module, the sink simply deletes the OMNeT++ messages representing the data packets. Also, the sink is used to collect statistics about the simulation, statistics that can be for each user, for each group of users (e.g. Low, Medium and High priority users) and/or for the entire system. These statistical information can be: the number of data packets that arrive to the sink, the mean, minimum and maximum delay of the data packets, etc.

In the OMNeT directories there is one called "samples", with different simulation models implemented in OMNeT++. From these samples, you can use as a starting point for your model the sources from the "fifo" system.

## Basic/minimal requirements

Implement the simulation model described above. The scheduling algorithm is not very important in this stage, it can be a simple round robin: each nonempty queue sends a data packet. Or it can be a priority queueing algorithm: a queue is not served until there are non-empty higher priority queues.

## For exam, there are two alternatives to improve the project

Implement one of the following scheduling algorithms:

- Priority queueing (a lower priority queue is served if and only if all higher priority queues are empty)

- A weighted round robin: an action takes place every time when there are resources available. The winner of the action will be served. The criteria for the action is, firstly, the time elapsed since the user was served last time. Then, we can improve the algorithm by introducing weights (positive numbers ¿1). The time elapsed since the user was served last time is multiplied with user's weight. Then, a user with a higher weight will be served more often. Compare the performance (i.e. average delay, minimum and maximum delay) of the implemented algorithms.

### New requirements (addition of the *fuzzy logic controller*)

Use a fuzzy logic controller in order to adjust the weight of HP users (W_HP) such that their average delay will be most of the time smaller than a certain value.

If the scheduling algorithm is priority queueing (PQ), then we cannot apply fuzzy weight adaptation like above, since PQ does not use weights!

## 2 Network structure

The network is described by the `IpScheduler.ned` file, which contains the users, which generate the IP packets, the packet handler, which handles the packets received from the users, and the sink, which receives the packets sent from the packet handler.

### 2.1 IpScheduler

`IpScheduler` is the main network, where all the sub-components reside.

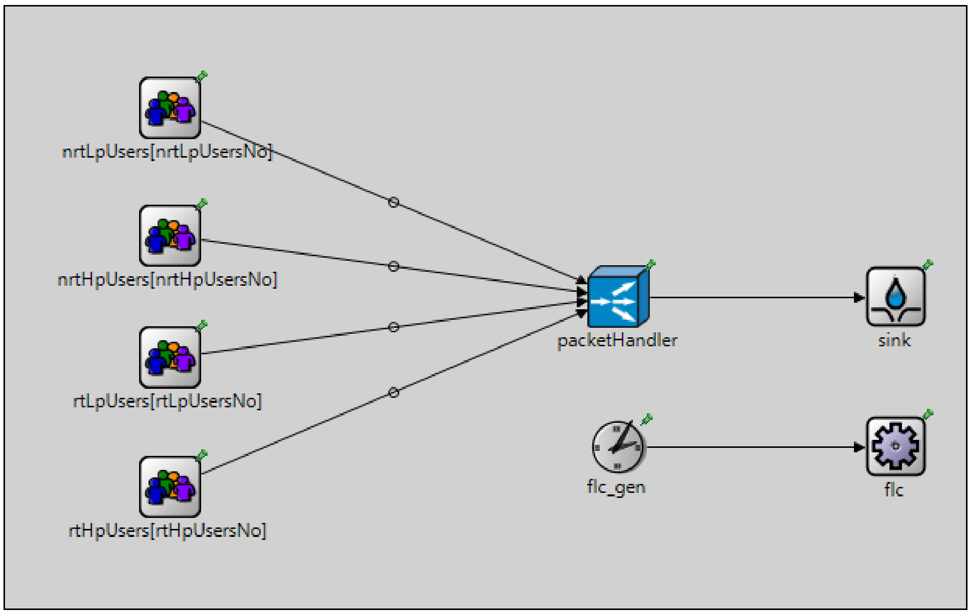

Figure 1: The main network structure

### 2.2 PacketHandler

`PacketHandler` is the component responsible with receiving the packets from the users, storing them in the corresponding queue, depending on the priority level of the users, scheduling the sending of the packets, and sending the packets to the sink.

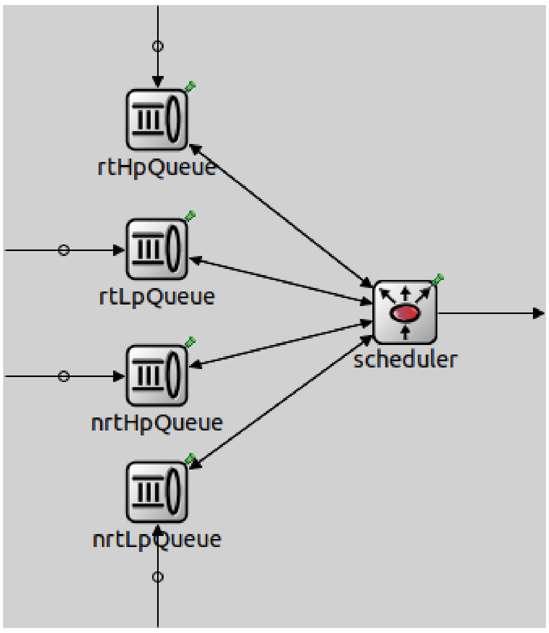

Figure 2: The packet handler structure

## 2.3 User

The `User` is the component which generates IP packets at a certain rate, and sends them to be handled bu the `PacketHandler`. The users are grouped in priority classes:

- non-real-time, low priority (nrtLp)
- non-real-time, low priority (nrtHp)
- real-time, high priority (rtLp)
- real-time, low priority (rtHp)

The size of the generated IP packets, the packet generation rate, and the number of users in each priority class can be configured in the `ipsched/config/users.ini` and `ipsched/config/user_packets.ini` files.

## 2.4 Queue

The `Queue` is the module which stores the IP packets generated by the users, and sends them towards the sink, whenever the scheduler allows it. There are four queues, one for each priority class, and they all communicate with the scheduler bidirectionally: they receive control messages from the scheduler, and send packets to the scheduler, when asked to. The *Queue* class contains a `cPacketQueue` attribute, where the IP packets are stored. The `Queue` module is also responsible with logging each queue's length, so we can collect data about the length of each queue at any given time.

### 2.5 Sink

The `Sink` is a module whose only purpose is to receive the IP packets, delete them, and record statistics about them (packet lifetime, total number of packets, etc.).

### 2.6 Scheduler

The `Scheduler` can be considered the most complex module of the system, and the most important, because it is responsible for deciding which queue sends a packet towards the sink. This job can be accomplished in multiple ways, and the current project provides two scheduler implementations: Weighted Round Robin, with and without the addition of the *fuzzy logic controller*.

## 3 The fuzzy logic controller

### 3.1 The ***FLC*** module

Asdf

### 3.2 The ***FLC_gen*** module

Asdf

## 4 Experiments

### 4.1 Simulation parameters

Asdf

### 4.2 Standalone experiments

Asdf

### 4.3 Comparative experiments

Asdf

## 5 References

(1) `http://staff.cs.upt.ro/~todinca/cad/IP_scheduling.html`

(2) `http://staff.cs.upt.ro/~todinca/TPAC/tpac.html`

(3) `https://doc.omnetpp.org/omnetpp/manual/`

(4) `https://docs.omnetpp.org/tutorials/tictoc/`

(5) `https://doc.omnetpp.org/omnetpp/api/`