“POLITEHNICA” UNIVERSITY TIMIȘOARA
FACULTY OF AUTOMATION AND COMPUTERS
FUZZY LOGIC AND APPLICATIONS (LFA)

Project #3

Andrei-Szilárd DOBAI

4th Year CTI-RO

1. Requirements

1.1. Initial Network

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.

A scheduler. The scheduler reads the lengths of users' queues and implements a scheduling algorithm that allocates a number of radio channels to each user. Then it informs each users how many radio blocks to send during the current scheduling cycle. You can consider that the total number of radio channels is B, for example B=30 in LTE. The scheduler is activated every TB ms (a scheduling cycle). The value of the scheduling cycle is 1 ms in LTE.

A sink. The sink models the destination of the data. When the data 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 and for the entire system. This 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.

A user consists of a source module (or generator) and a buffer module, implemented as a queue. The generator generates a number of data packets at certain time intervals. The data packets are stored in the buffer. When the scheduler informs the user that it can send a number of, e.g. m data packets, then the firsts m data packets from the buffer are send to the destination (the sink module).

We can consider that all data packets have a fixed length.

1.2. Fuzzy Logic Upgrade

Consider a radio system (a cell) where there can be at most K users (eg K = 10). The resource allocation algorithm is of the WRR (Weighted Round Robin) type. We consider that of the K users, some have W_HP (high priority) weight, others W_MP (medium priority), and others W_LP (low priority). For example W_HP = 4, W_MP = 2 and W_LP = 1. The total capacity of the network is B data packets per ms Tc (milliseconds) (eg B = 8 and Tc = 20 at GPRS, or B = 20 and Tc = 1 at LTE). All data packets are considered to have a fixed length equal to 1. Adjust the share of HP users so that the average delay of their packets is not (as far as possible) greater than a given value. Fuzzy inference will be used for this purpose.

2. Implementation

2.1. Network topology

The network consists of a given number K of users, a scheduler, a fuzzy logic controller and a sink. Each user consists of a generator and a queue.

In the omnetpp.ini file, you can specify the desired scheduling algorithm to be used inside the network, the number of users, the number of communication channels and the network's load.

2.1.1. User

Seen as a black box, a user is a subnetwork that generates messages and waits for the scheduler to tell it how many messages it can send. There are three types of users: low priority (weight = 1), medium priority (weight = 2) and high priority (weight = 4).

In the omnetpp.ini file, you can specify each user's type, as well as a starting message number (used to easily visualize which user's message arrives at the sink).

2.1.2. Generator

The generator is an OMNeT++ source module that creates messages at a fixed interval. This interval depends on the number of users in the network, the number of communication channels and the selected network load.

2.1.3. Queue

The queue is an OMNeT++ module that holds the messages created by the generator in a buffer until told by the scheduler to pop them out.

2.1.4. Scheduler

The scheduler is an OMNeT++ module that calculates, using a certain algorithm, how many messaged can each user send to the sink.

2.1.5. Fuzzy Logic Controller

The FLC is an OMNeT++ module that uses fuzzy logic to recalculate the high priority users' weights in order to keep the latest user's wait time to a minimum. For example, if a user has a large wait time, the FLC will decide to increase that user's weight in an attempt of lowering it.

2.1.6. Sink

The sink is an OMNeT++ module that receives the messages created by the users and disposes of them.

2.2. How it works

Upon starting the simulation, each user will calculate its own interval to generate messages and begin filling up their queues. Each 1ms (configured in omnetpp.ini), the scheduler sends a message to each user asking for their current queue's lengths. After receiving the answers, it will calculate how many messages each user can send and communicate it to them via an OMNeT++

message. The user receives that information and pops however many messages it is allowed to out of the queue and sends them towards the sink.

In the fuzzy logic upgrade, when the scheduler asks each user for their queues' length, it also tells the FLC to start. The FLC will then calculate each high priority user's new weight and set it via OMNeT++'s parameters.

3. Results

In order to test whether the fuzzy logic upgrade help the network overall or not, I created 3 experiments and ran each of them once with the FLC upgrade and once without.

Common configurations between experiments:

- Scheduling algorithm: weighted round robin
- Users count: 10
- Communication channels: 20
- Scheduler run interval: 1ms
- User types: [HP, HP, HP, HP, HP, HP, HP, LP, MP, LP]

3.1. Experiment 1

For the first experiment, I decided to test the network during normal conditions (having the network at 90% capacity).

As expected, the scheduling algorithm alone can easily manage the workload. Having the FLC run alongside it brought virtually no change whatsoever to the results.

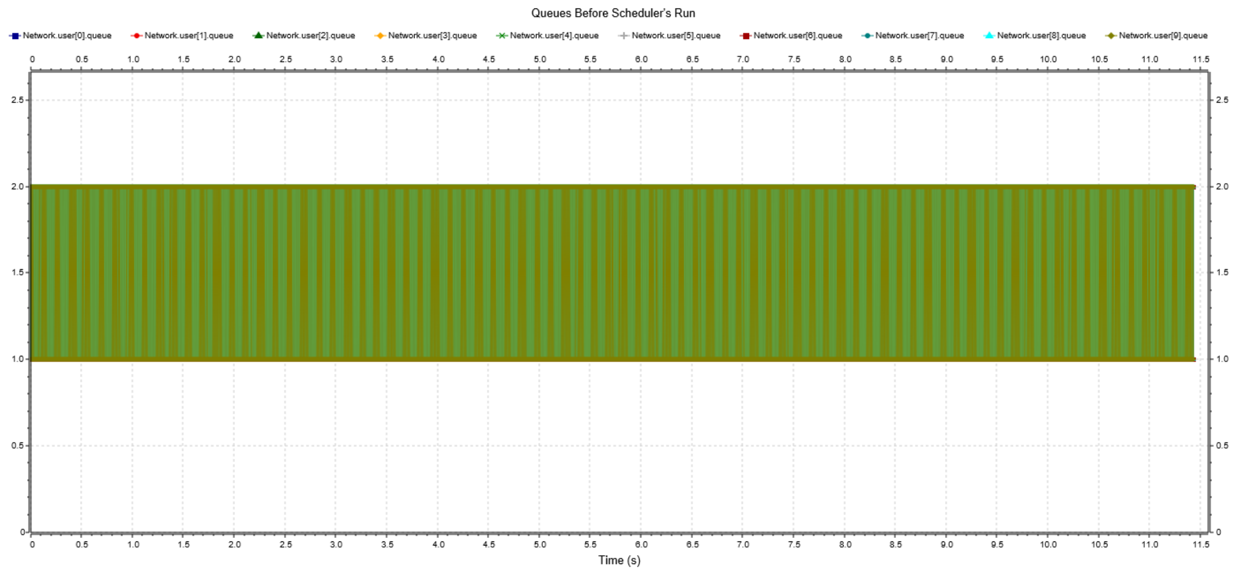

Fig. 1: Queue lengths before running the scheduler.

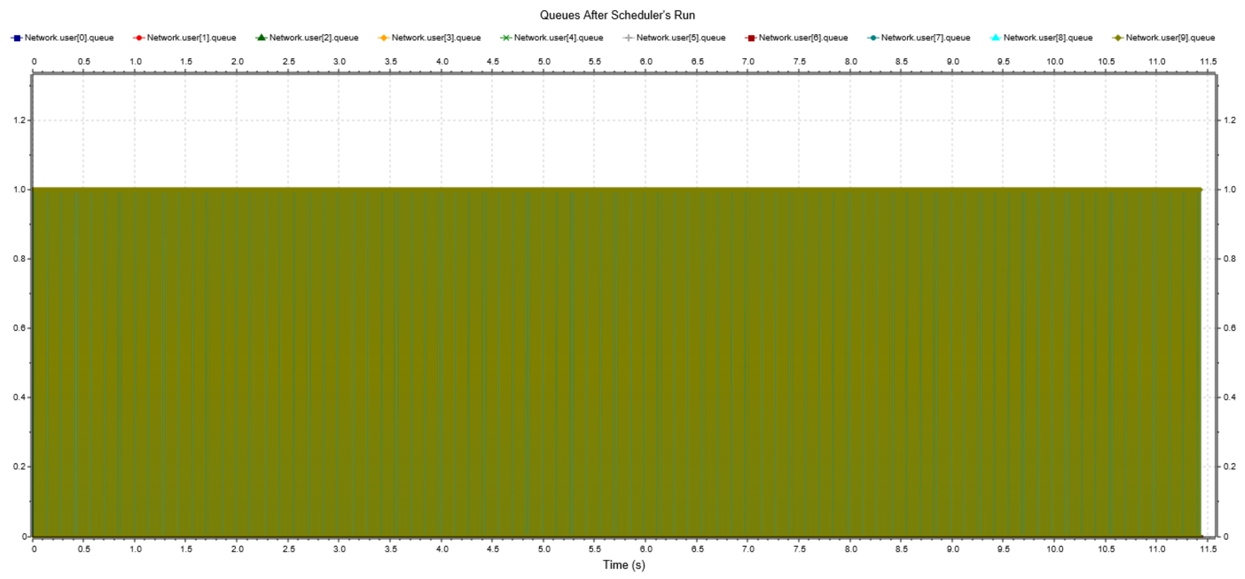

Fig. 2: Queue lengths after running the scheduler.

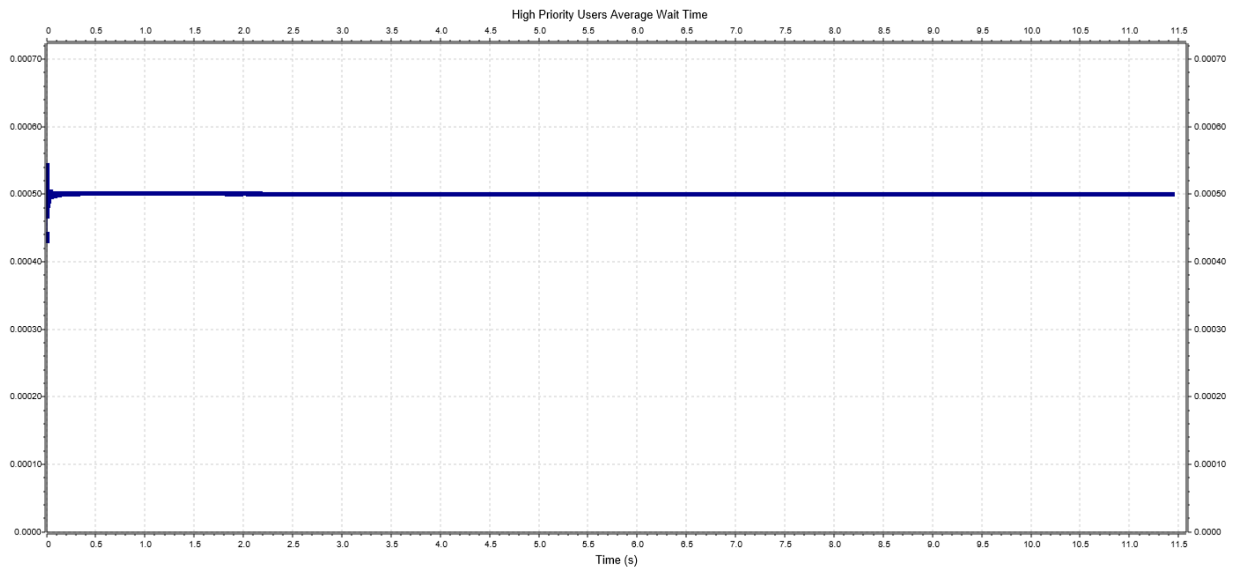

Fig. 3: High priority users' average wait time.

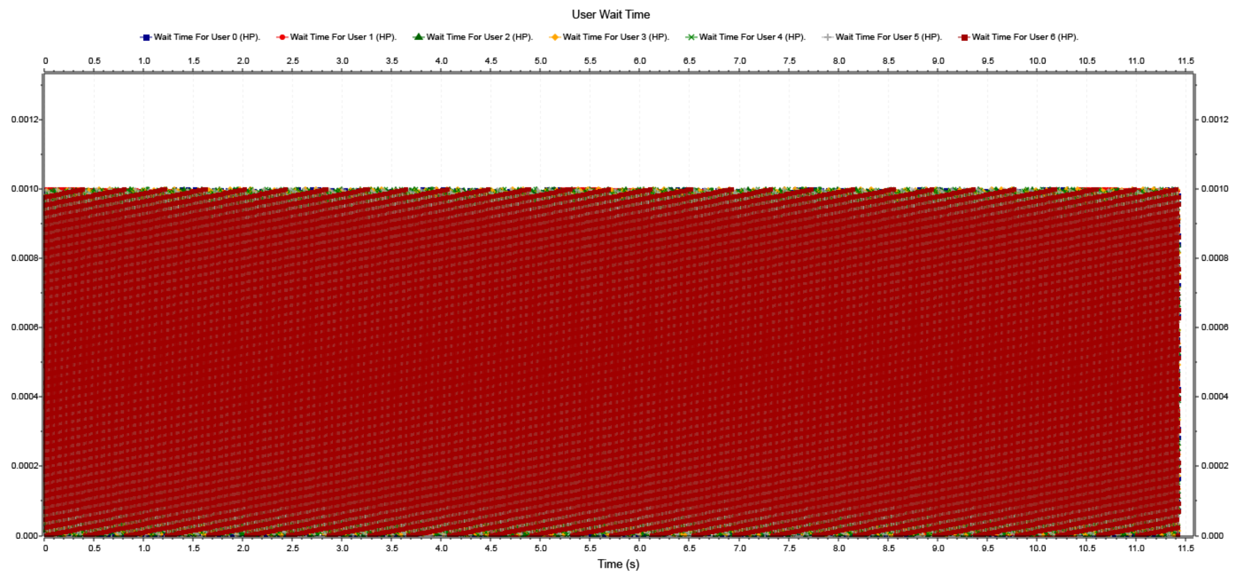

Fig. 4: High priority user's wait time.

3.2. Experiment 2

For the second experiment, I decided to push the network a bit, having it run at 130% capacity.

3.2.1. Without the FLC

Here, having it rely only on the scheduling algorithm proved it to be quite ineffective for all users, even the High Priority ones.

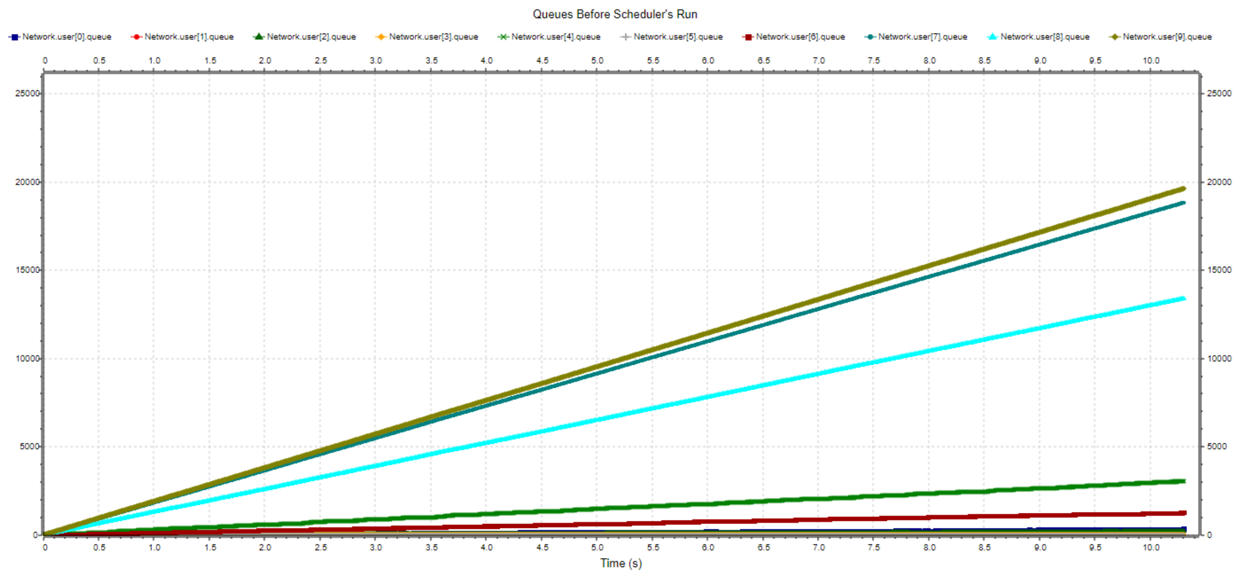

Fig. 5: Queue lengths before running the scheduler.

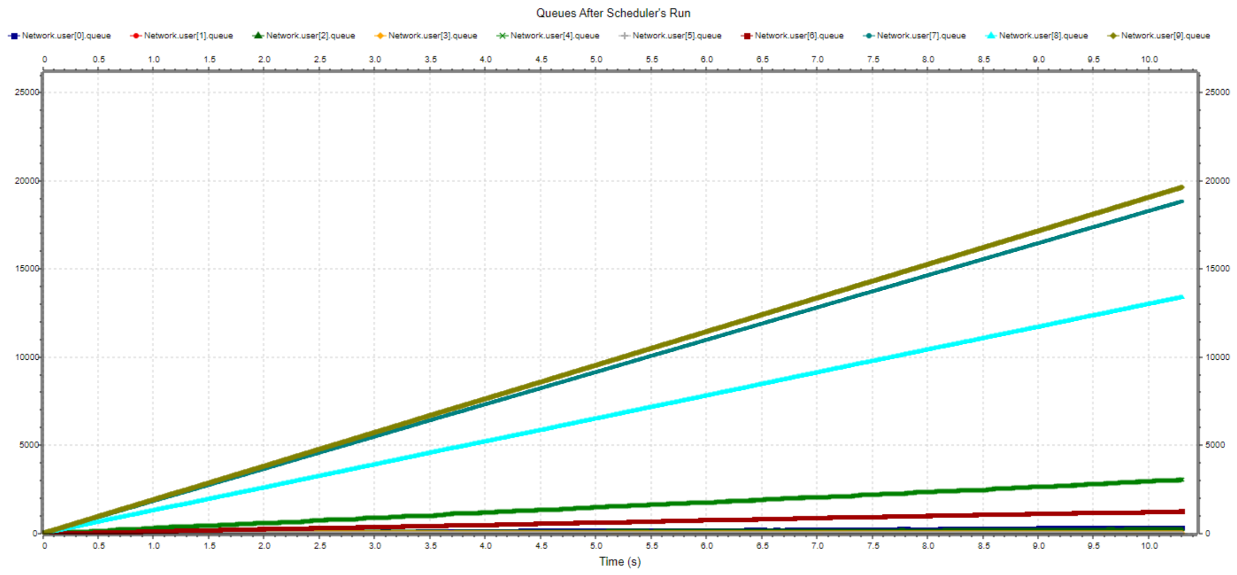

Fig. 6: Queue lengths after running the scheduler.

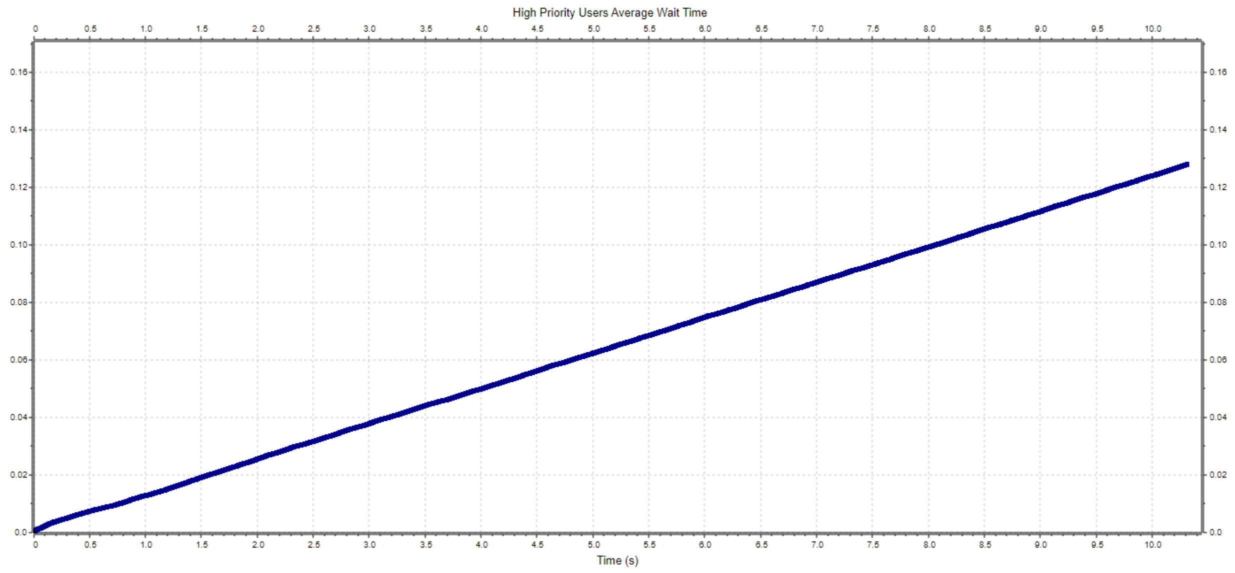

Fig. 7: High priority users' average wait time.

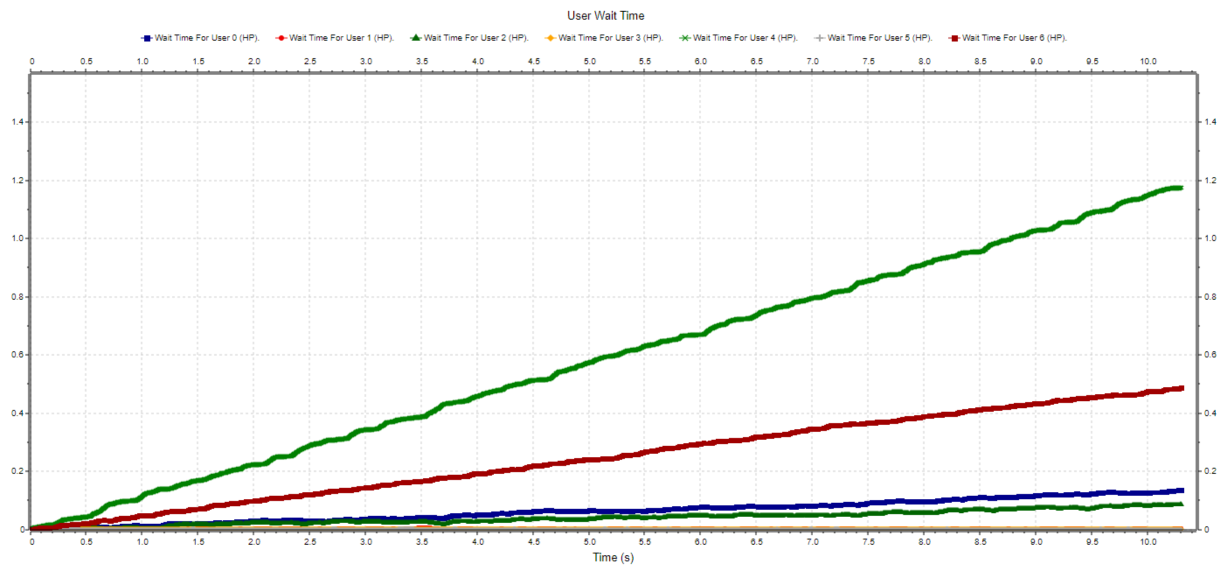

Fig. 8: High priority user's wait time.

Some HP users would be served regularly, but others would get forgotten over time by the scheduler, amassing even 3000 messages in the queue during the simulation period (+/- 10s). Even though the overall HP average wait time does not seem that bad right now, it is increasing linearly. Also, for some users it almost got to a wait time of 1.2 seconds between getting the 'ok' from the scheduler to send messages to the sink.

3.2.2. With the FLC

With the fuzzy logic upgrade in place, though, things changed drastically for the high priority users.

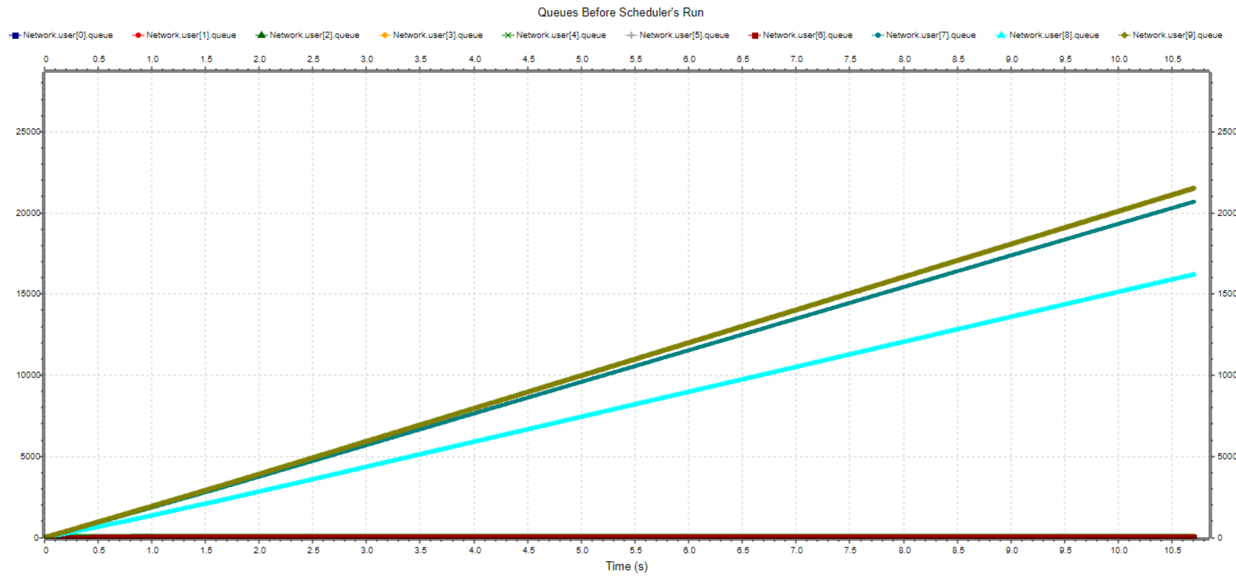

Fig. 9: Queue lengths before running the scheduler.

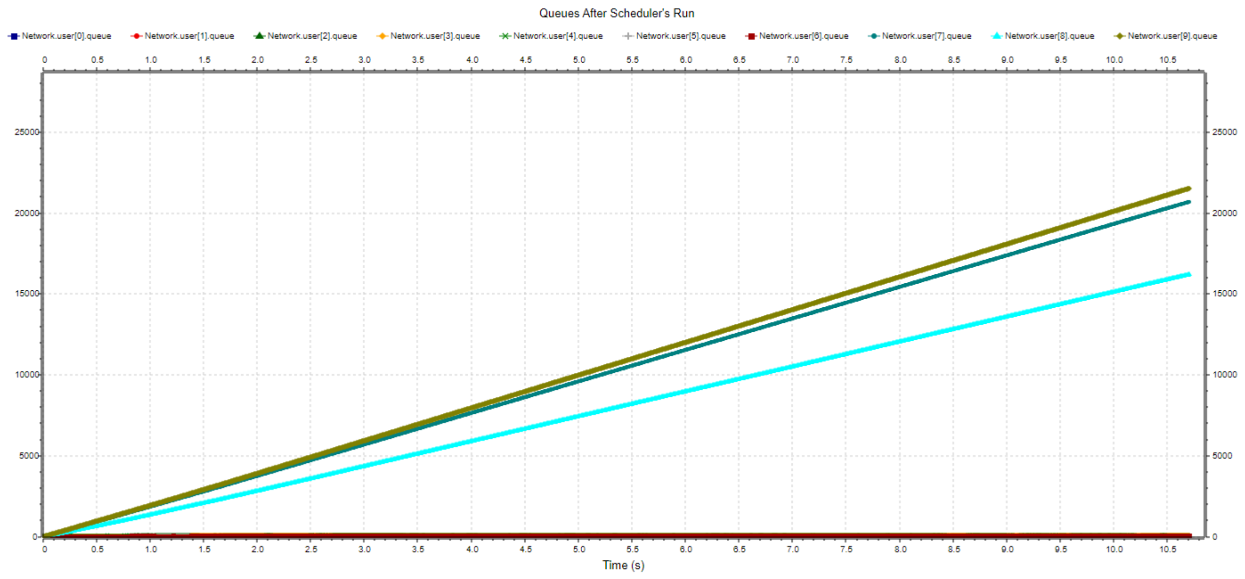

Fig. 10: Queue lengths after running the scheduler.

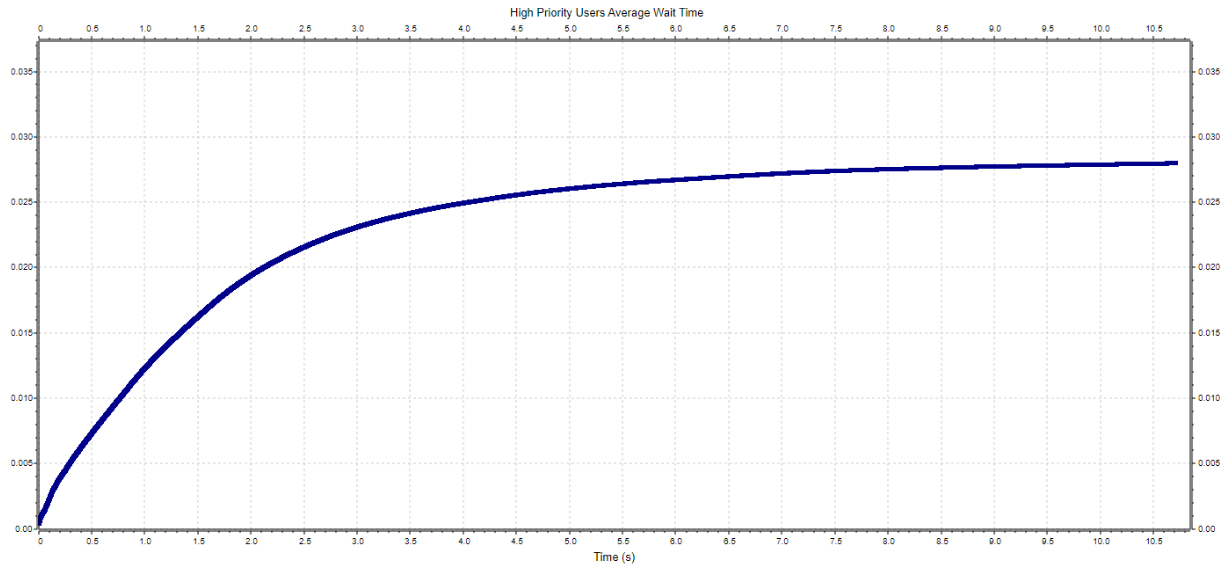

Fig. 11: High priority users' average wait time.

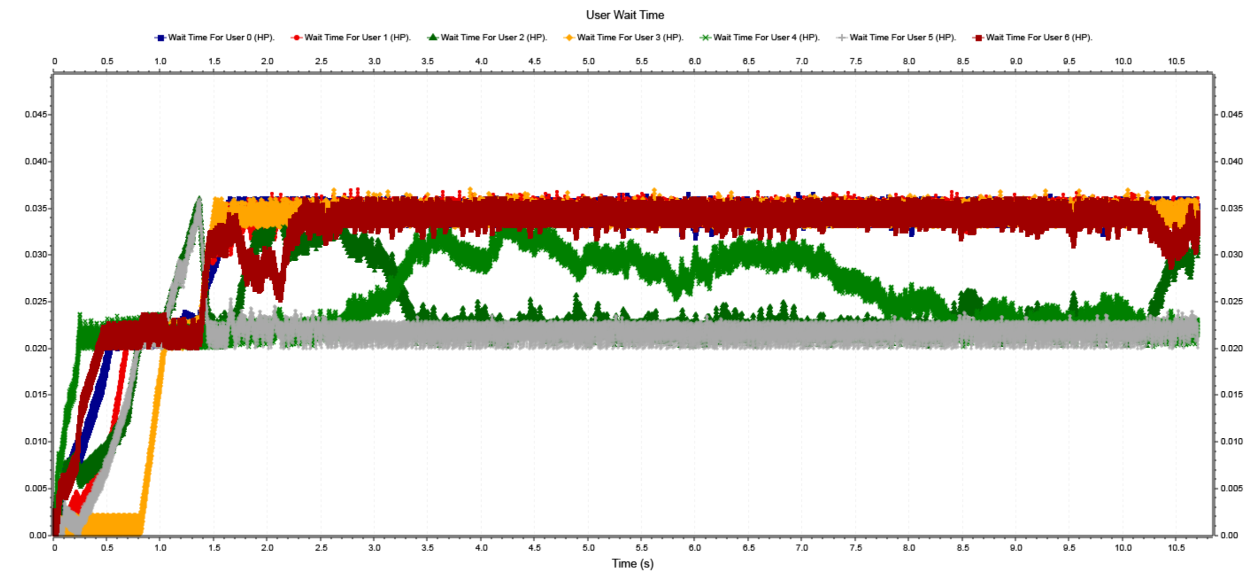

Fig. 12: High priority user's wait time.

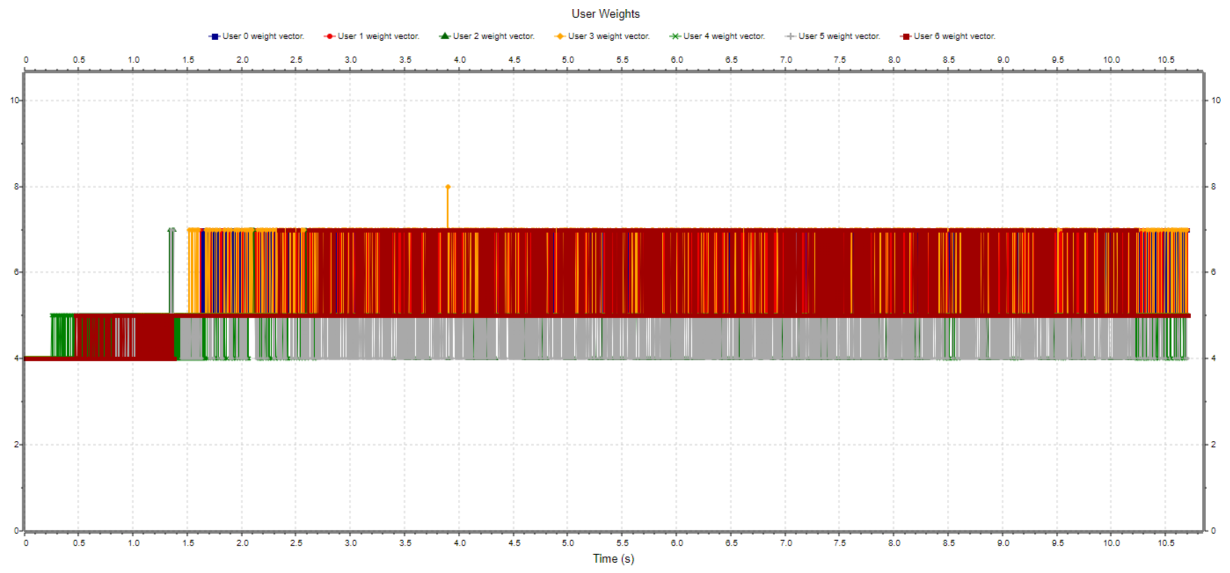

Fig. 13: High priority users' weight in time.

Strictly speaking for HP users, the network handled the workload very good, some users amassing only 80 messages in their queues (compared to the non FLC 3000).

As for the average wait time, even though the value is increasing over time, it does seem to be capping at less than 0.03s. Individually, users' wait times rarely go above 0.035s.

In fig. 13 we can see how the fuzzy logic controller varies each user's weight depending on their needs.

3.3. Experiment 3

In this experiment I decided to push the network even more by having it run at 200% capacity.

3.3.1. Without the FLC

As expected, without fuzzy logic modifications, the network is severely overwhelmed.

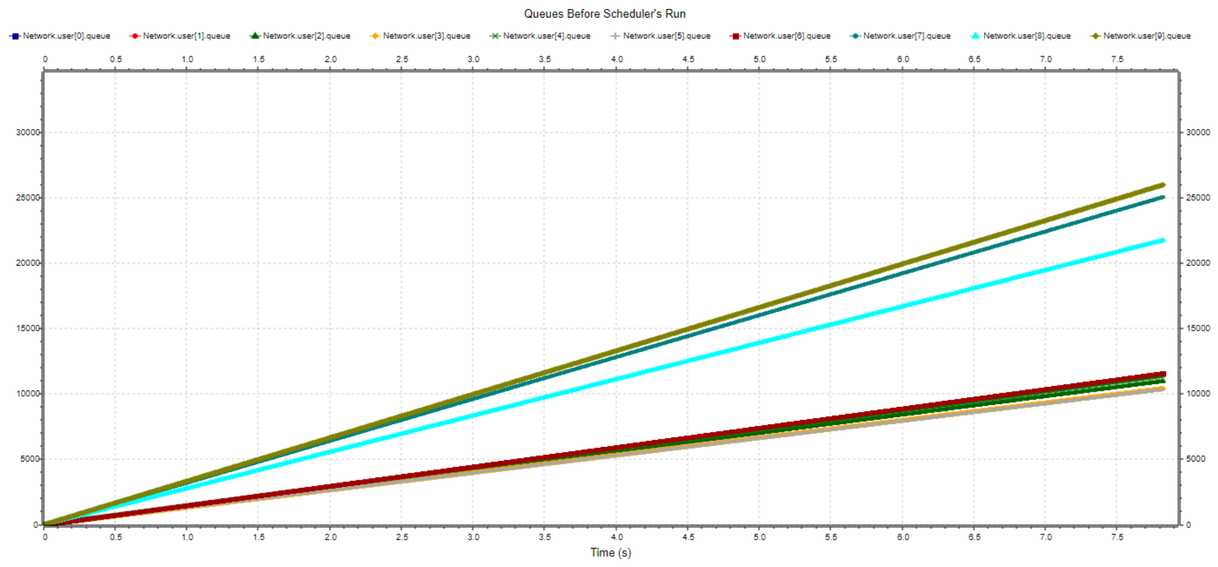

Fig. 14: Queue lengths before running the scheduler.

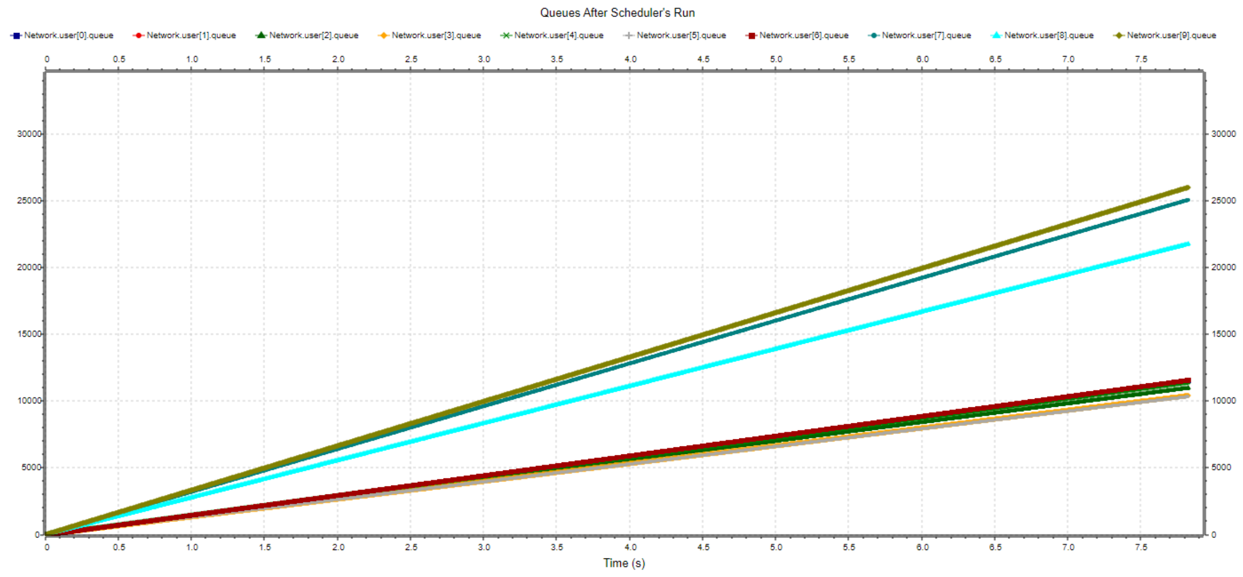

Fig. 15: Queue lengths after running the scheduler.

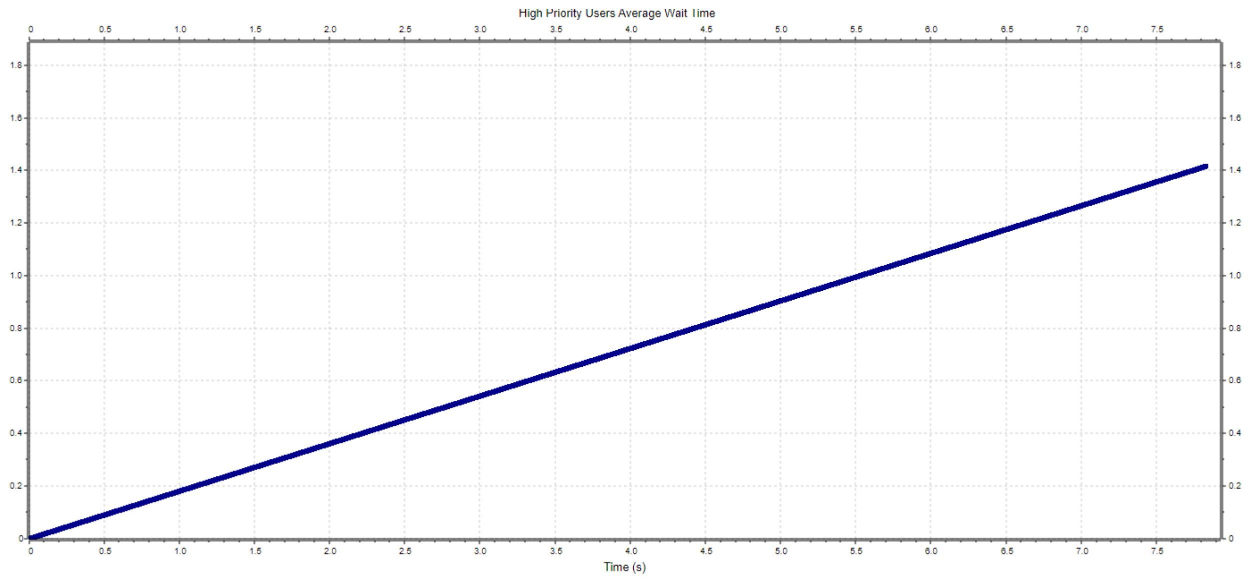

Fig. 16: High priority users' average wait time.

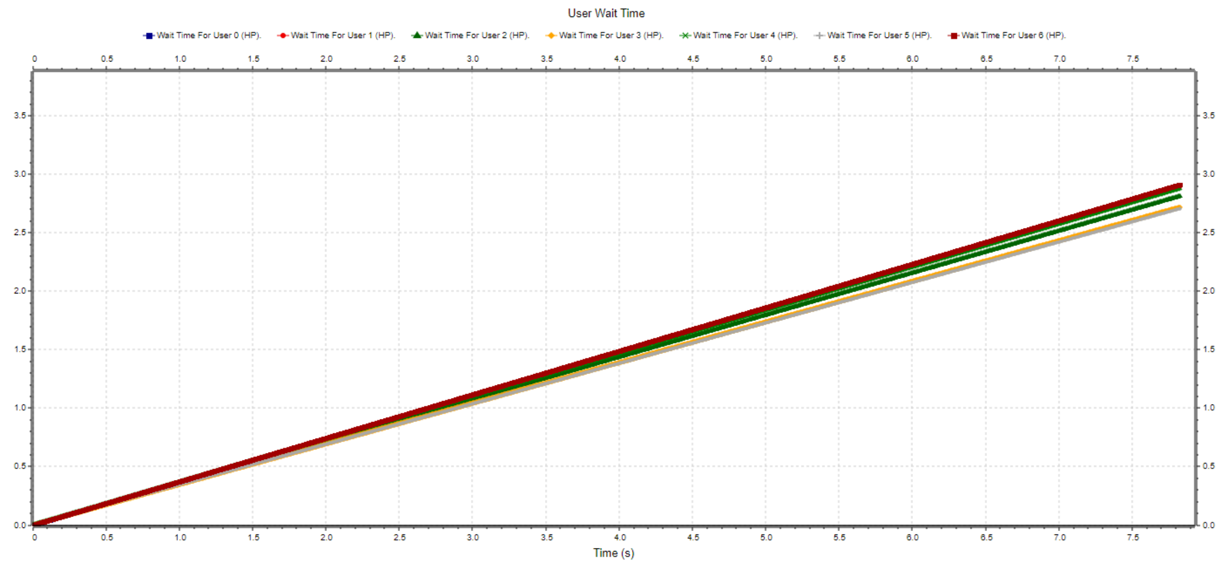

Fig. 17: High priority user's wait time.

All users are amassing more and more messages in their respective queues. Both average HP wait time and individual times are increasing linearly over time at a large rate.

3.3.2. With the FLC

Things are much greater with the FLC in place either. Even though HP users are being served more, it still isn't enough to make the network usable.

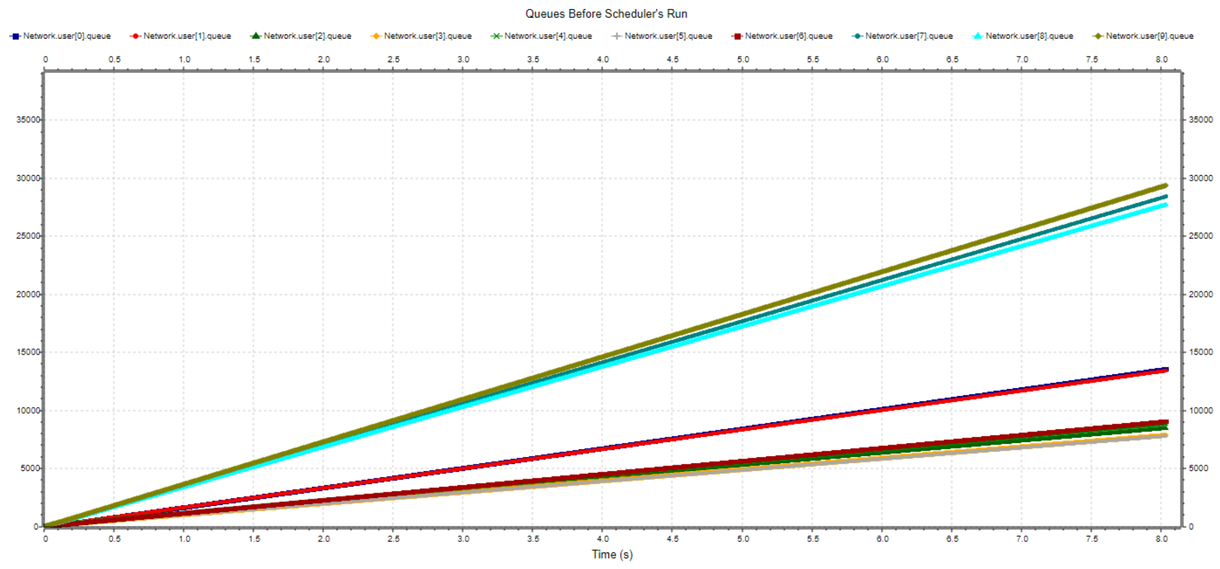

Fig. 18: Queue lengths before running the scheduler.

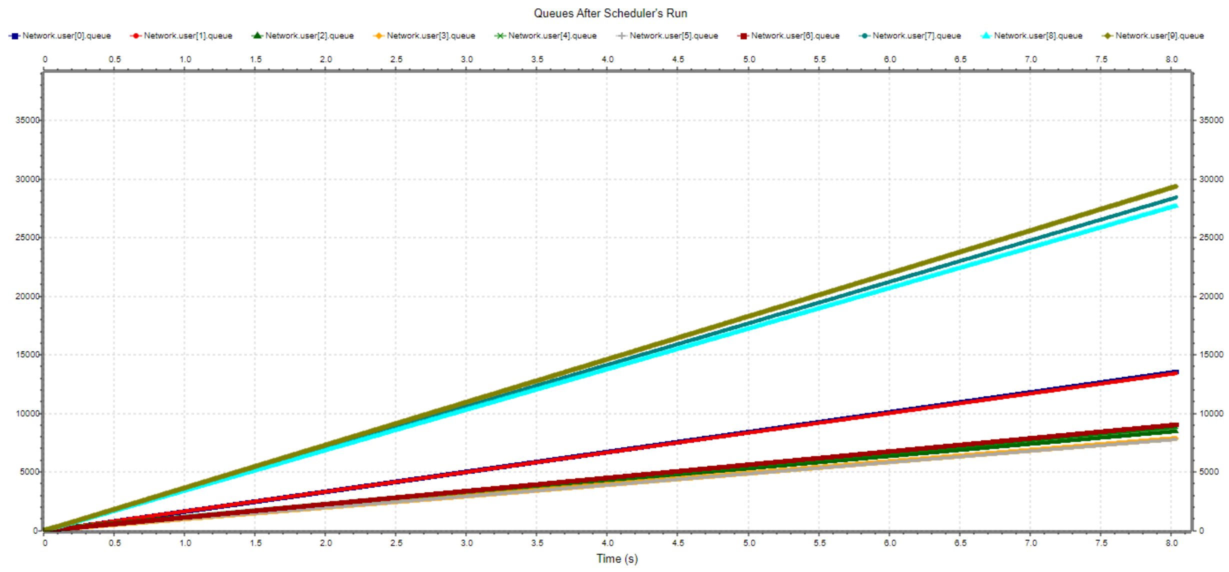

Fig. 19: Queue lengths after running the scheduler.

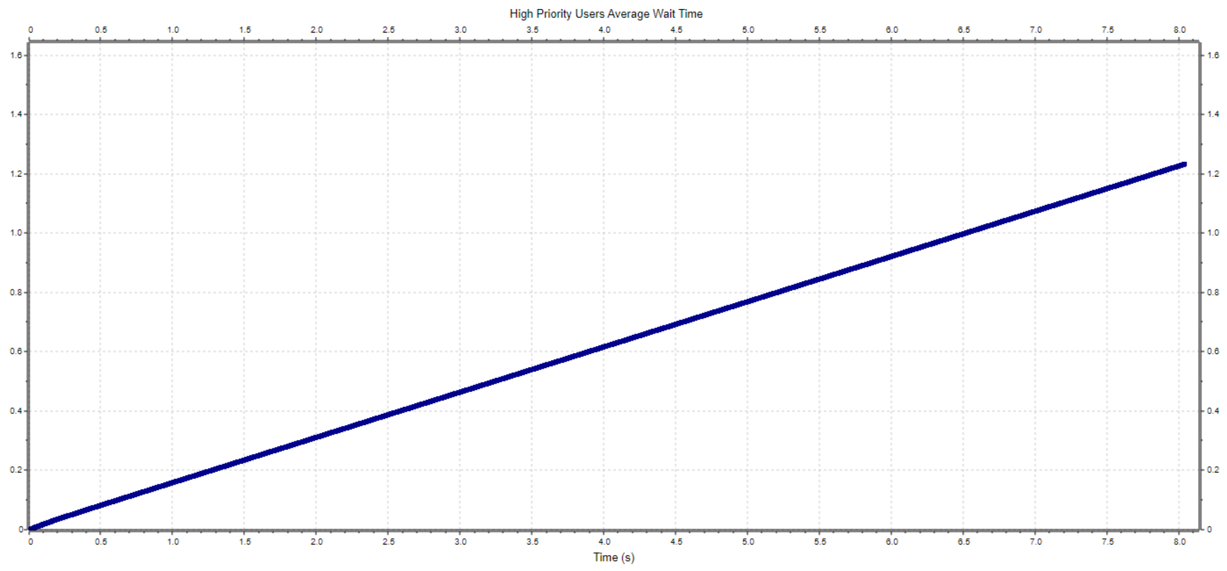

Fig. 20: High priority users' average wait time.

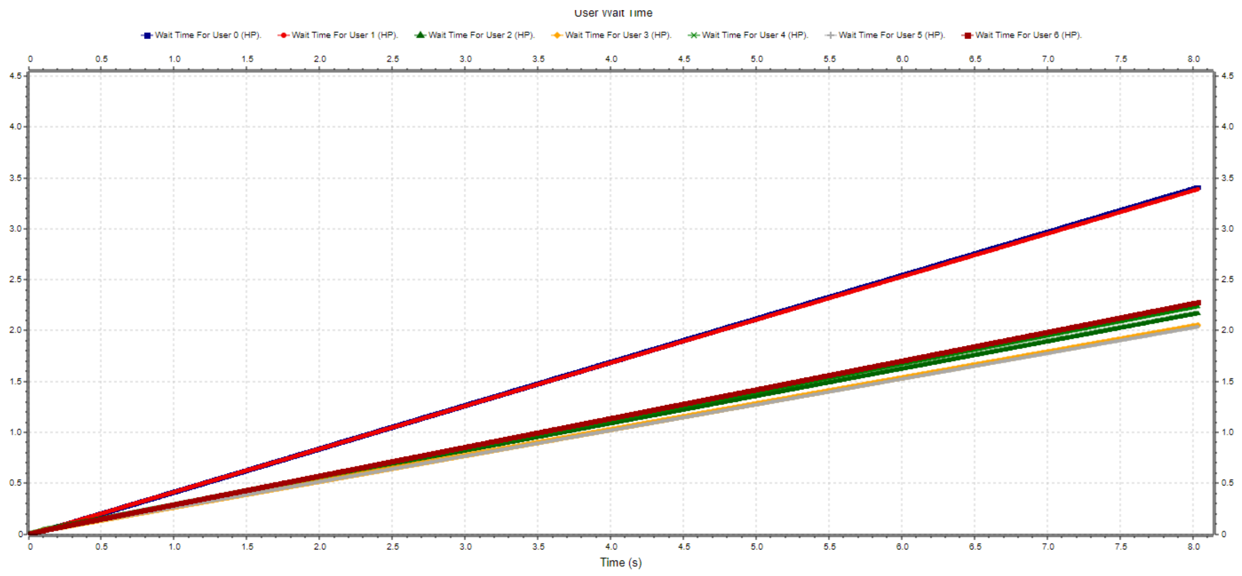

Fig. 21: High priority user's wait time.

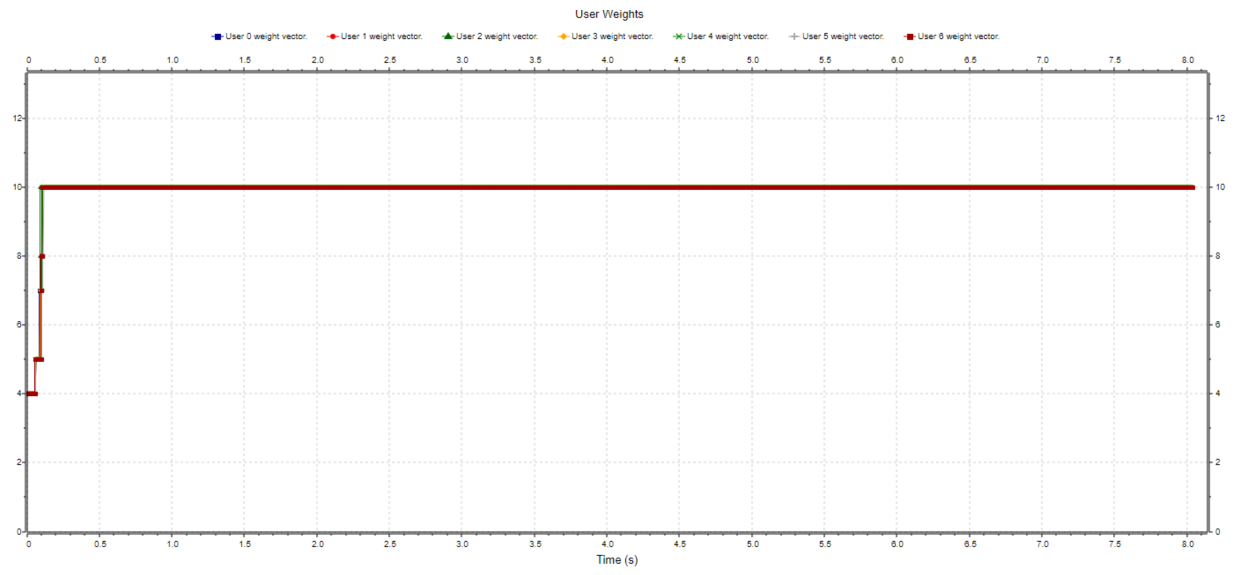

Fig. 22: High priority users' weight in time.

Every user's queue sits at around 10.000 messages after approximately 10s of simulation. All wait times increase linearly over time, even though the FLC put all HP users in "overdrive" (assigning them a constant large weight).

4. Conclusions

As seen in the above experiments, the fuzzy logic upgrade does, indeed, prove to be useful but only as long as the network is being pushed above its limits, but not too much, at the same time.

Having the FLC run during normal network capacity (0-100%) is completely redundant.

Having it run while the network is under severe stress (200% +) does help the high priority users, but not enough to have it be usable from a client's perspective.

Table of Contents