UNIVERSITÄT HAMBURG

UNIVERSITÀ DEGLI STUDI
DELL'AQUILA

ERASMUS MUNDUS CONSORTIUM "MATHMODS"

Joint Degree of Master of Science in
Mathematical Modelling in Engineering: Theory, Numerics, Applications

In the framework of the
Consortium Agreement and Award of a Joint/Multiple Degree 2013-2019

MASTER'S THESIS

Validation of a Model Predictive Control Algorithm for Priority-Based Routing

Supervisor:
Prof. Alessandro D'Innocenzo

Candidate:
Amr Ibrahim
Matricola: 238924

2015/2016

Laurea Magistrale in Ingegneria Matematica
Dipartimento di Ingegneria e Scienze dell'Informazione e Matematica
Universit degli Studi dell'Aquila

Contents

List of Figures

Chapter 1

Introduction

Data communication networks are highly complex systems, thus modeling and analyzing their behavior is quite challenging. The problem aggravates as networks become larger and more complex. The most accurate network models are packet-level models [1] that keep track of individual packets as they travel across the network. These are used in network simulators such as ns-2. Packet-level models have two main drawbacks: the large computational requirements (both in processing and storage) for large-scale simulations and the difficulty in understanding how network parameters affect the overall system performance. Aggregate fluid-like models [2], [3] overcome these problems by simply keeping track of the average quantities that are relevant for network design and provisioning (such as queue sizes, transmission rates, drop rates, etc).The main limitation of these aggregate models is that they mostly capture steady state behavior because the averaging is typically done over large time scales. For instance, detailed transient behavior during congestion control cannot be captured. Consequently, these models are unsuitable for a number of scenarios, including capturing the dynamics of short-lived flows.

Our approach to modeling computer networks and its protocols is to use hybrid systems which combine both continuous time dynamics and discrete-time logic. Our hybrid systems framework fills the gap between packet-level and aggregate models by averaging discrete variables over a very short time scale (on the order of a round-trip time). This means that the model is able to capture the dynamics of transient phenomena fairly accurately, as long as their time constants are larger than a couple of round-trip times.This time scale is quite appropriate for the analysis and design of network protocols including policy-based queuing and congestion avoidance algorithms, as these generally last for periods no shorter than one round-trip time. These models permit complexity reduction through continuous approximation of variables like queue and congestion window size, without compromising the expressiveness of logic-based models [4]. The hybridness of the model comes from the fact that, by using averaging, many variables that are essentially discrete (such as queue and window sizes) are allowed to take continuous values. However, because averaging occurs over short time intervals, one still models discrete events such as the occurrence of a drop and the consequent reaction (e.g.,congestion control) [5].

Here, we have extended the framework in [4], [6] to take into account Priority-Based Networking. Priority-Based Networking is the ability to treat packets differently as they transit a network device, based on the packet contents. Without Priority-Based policies each service is given equal access to resources. In order for a company to efficiently utilize its network resources, it must identify which network traffic is critical and allocate appropriate resources to support those traffic streams. Priority Queueing is one of the ways with which a congested network can be managed, and it is a part of the so called Quality of Service management(QoS) [15], [16]. QoS is the network centric performance indicator, while QoE (Quality of Experience) is a user centric performance indicator of the network. It is the most important factor for customers while choosing a network provider.

To ensure superior QoE, it is important to understand the relationship between QoE and QoS. A high QoE normally translates into a high QoS, and it is possible to derive some mathematical relationships between QoE and QoS (see for example [7], [8], [9]). Given this fact, we can only focus on QoS features [10].

MDRR (Modified Deficit Round Robin) is the priority based Networking algorithm implemented in Telecom Italia. In MDRR, flows are classified according to their priority. High priority flows are given absolute preferential treatment among the ones with lower priority. High priority packets are given the whole link bandwidth if necessary and they always have the permission to be sent even there would be drops in the lower priority flows. The remaining bandwidth after sending Higher priority packets is distributed between Medium and Normal flows. MDRR algorithm states that Medium flows are given 80% of the remaining bandwidth and Normal flows are given 20%. Giving a fixed portion of the available bandwidth to each flow is not the best solution to have better QoS. For example if the incoming rates of the Normal flows are too small and for the Medium flows are very high , assigning 20% to Normal flows in this case is a waste of resources and it is better to increase the 80% of the Medium queue to something like 95%. By this way Medium-priority packets (which have higher priority than Normal-priority packets) will be saved from droppings. But of course Normal packets still need to be sent. So the assigned percentage to each flow should be dynamically changed according to different parameters.

In order to improve the behavior of the MDRR scheme to get better QoS features, a hybrid model of queue dynamics for MDRR scheme has been developed. Our hybrid model has 41 states, each state describes a specific behavior of the 3 different queues (High, Medium, Normal). These behaviors are filling queue,empty queue or full queue. Also sending and dropping rate for each flow is calculated. In order to be able to improve the QoS features, we need to change some of the model parameters.

A controller was designed which can help us understand how to dynamically change the parameters to best tune them. The controller being used in this thesis, is a modern controller, named Model Predictive Controller or MPC [11], [12], [14], [13]. MPC enables us to impose other constraints in the control procedure to take into account QoS(and thus QoE) limitations. This provides us with more flexibilities than the traditional controllers to attain the desired QoS. MPC controller is based on repeatedly solving an on-line constrained optimization problem. The aim is to minimize a predefined objective function over a finite prediction horizon. The result from this minimization problem is a sequence of future control inputs over the predicted horizon. Only the first input will be applied to the plant.

A comparison between the controlled and uncontrolled MDRR algorithm was done. The uncontrolled MDRR is the implemented algorithm in Telecom-Italia routers which is distributing the bandwidth between Medium and Normal flows with a fixed rate. The controlled MDRR algorithm means that MPC controller is implemented and the allocated value of the bandwidth is changing depending on the state and other constraints. Metrics such as Drop Rates, Sending Rates, Queue Sizes were compared between the controlled and uncontrolled algorithms. We noticed improvements in those metrics in the controlled MDRR algorithm over the uncontrolled one. The hybrid model was tested with real data from Telecom-Italia.

The report is organized as follows: Chapter 2 covers Routing Problem, Priority based networking concepts, and Hybrid model of Priority Queuing (a particular Priority based Networking method), the MDRR scheme (Priority based Networking implemented in Telecom Italia); Chapter 3 covers an introduction to Model predictive controllers, mathematical formulation, and different tools and solvers; Chapter 4 covers the description of the optimization problem and the control problem; Chapter 5 covers the MATLAB Simulation

of the MDRR scheme, Hybrid MPC and comparison between the flow behaviors when no controller is used and when MPC controller is implemented.

Chapter 2

Quality of Service Networking. Hybrid Models for Queue Dynamics, Priority Queuing and MDRR.

2.1 Routing Problem and QoS

Quality of Service (QoS) [15] refers to the capability of a network to provide better service to selected network traffic over various technologies. The primary goal of QoS is to provide priority including dedicated bandwidth, controlled jitter and latency (required by some real-time and interactive traffic), and improved loss characteristics. Also important is making sure that providing priority for one or more flows does not make other flows fail. QoS technologies provide the elemental building blocks that will be used for future business applications in campus, WAN, and service provider networks. Almost any network can take advantage of QoS for optimum efficiency, whether it is a small corporate network, an Internet service provider, or an enterprise network. The QoS software provides these benefits:

- **Control over resources** - To have control over which resources (bandwidth, equipment, wide-area facilities, and so on) are being used. For example, to limit the bandwidth consumed over a backbone link by FTP transfers or give priority to an important database access.
- **More efficient use of network resources** - To know what your network is being used for and that you are servicing the most important traffic to your business.
- **Tailored services** - The control and visibility provided by QoS enables Internet service providers to offer carefully tailored grades of service differentiation to their customers.
- **Coexistence of mission-critical applications** - To make certain that your WAN is used efficiently by mission-critical applications that are most important to your business, that bandwidth and minimum delays required by time-sensitive multimedia and voice applications are available, and that other applications using the link get their fair service without interfering with mission-critical traffic.
- **Foundation for a fully integrated network in the future** - It is a good first step toward the fully integrated multimedia network needed in the near future.

2.1.1 Why IP QoS

- Application X is slow(not enough bandwidth) lack of bandwidth is caused by multiple flows are contesting for a limited amount of bandwidth.

- Video broadcast occasionally stalls(delay temporarily increases (jitter) variable delay is caused because sometimes there is a lot of other traffic, which results in more delay.
- Phone calls over IP are no better than over satellite(too much delay) Too much delay is caused because Packets have to traverse many network devices and links.
- Phone calls can have very bad voice quality(too many phone calls (admission control)
- ATMs (the money-dispensing type) are nonresponsive(too many drops) drops are caused when a link is congested.

To Increase Available Bandwidth: Take bandwidth from some less important applications.

To Reduce delay: Forward the important packets first.

To prevent packet loss: Guarantee enough bandwidth to sensitive packets. Prevent congestion by randomly dropping less important packets before congestion occurs.

2.1.2 How Can QoS Be Applied?

- **Best effort** - no QoS is applied to packets (default behavior)
- **Integrated Services model** - applications signal to the network that they require special QoS
- **Differentiated Services model** - the network recognizes classes that require special QoS

Here we will be discussing Differentiated Services model of the QoS.

2.1.3 Basic QoS Architecture

The basic architecture [15] introduces the three fundamental pieces for QoS implementation:

- **QoS identification and marking techniques** for coordinating QoS from end to end between network elements.Identification and marking is accomplished through classification and reservation.
- **QoS within a single network element** (for example, queuing, scheduling, and traffic-shaping tools). Congestion management, queue management, link efficiency, and shaping/policing tools provide QoS within a single network element.
- **QoS policy, management, and accounting functions** to control and administer end-to-end traffic across a network.QoS management helps to set and evaluate QoS policies and goals.

In this thesis, only the case of congestion management will be discussed. For further reading refer to [15].

Congestion Management Tools: One way network elements handle an overflow of arriving traffic is to use a queuing algorithm to sort the traffic, and then determine some method of prioritizing it onto an output link. This includes the following queuing tools:

- First-in, first-out (FIFO) queuing
- Priority queuing (PQ)
- Custom queuing (CQ)

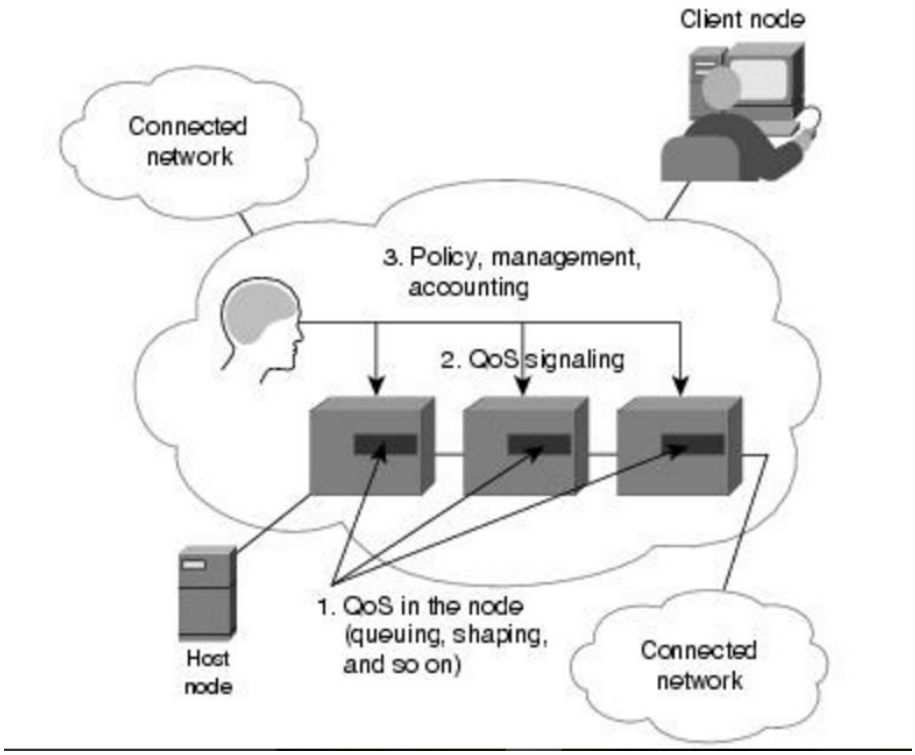

Figure 2.1: QoS Architecture

- Flow-based weighted fair queuing (WFQ)
- Class-based weighted fair queuing (CBWFQ)

Each queuing algorithm was designed to solve a specific network traffic problem and has a particular effect on network performance, as described in the following sections.

In this thesis, only modeling the queue dynamics in PQ has been considered.

2.1.4 PQ: Prioritizing Traffic

PQ ensures that important traffic gets the fastest handling at each point where it is used. It was designed to give strict priority to important traffic. Priority queuing can flexibly prioritize according to network protocol, incoming interface, packet size, source/destination address, and so on. In PQ, each packet is placed in one of four queues-high, medium, normal, or low-based on an assigned priority. Packets that are not classified by this priority list mechanism fall into the normal queue (see Figure 2.3). During transmission, the algorithm gives higher-priority queues absolute preferential treatment over low-priority queues.
PQ is useful for making sure that mission-critical traffic traversing various WAN links gets priority treatment. PQ currently uses static configuration and thus does not automatically adapt to changing network requirements.
The queuing is explained even better with Figure 2.2.

2.1.5 MDRR (Modified Deficit Round Robin)

Here we will describe the congestion management mechanism implemented in Telecom Italia.

The input flows are classified based on IP Precedence and served through three queues-High,Medium and Default. The High Queue is given strict high priority and is pre-emptied with Medium and Default queues.When the High Queue is empty the medium and default queues are served in round robin fashion and each queue is reserved with a percentage of link bandwidth. Each queue has a maximum number of bytes it can hold.

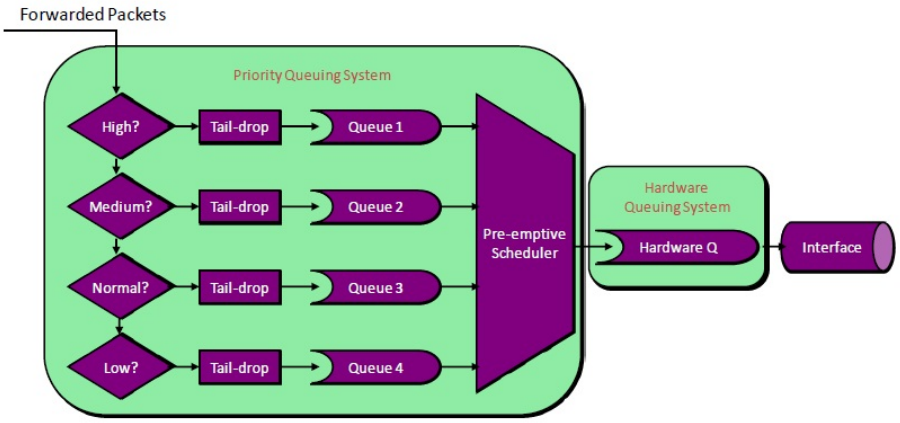

Figure 2.2: Priority Queuing in QoS

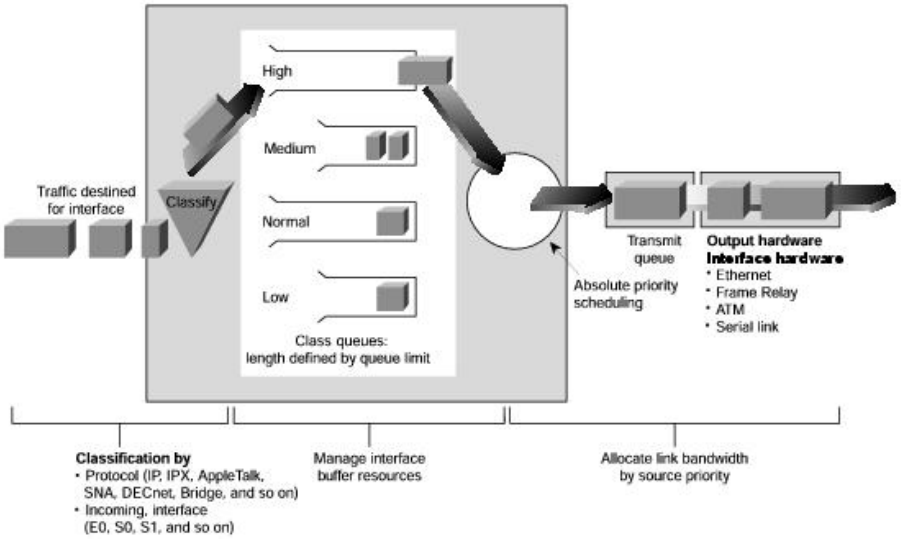

Figure 2.3: Priority Queuing Places Data into Four Levels of Queues: High, Medium, Normal, and Low

From Figure 2.4 it is seen that medium queue is given 80% of the available link bandwidth when the high queue is empty and the default queue is given 20% of the available link bandwidth.

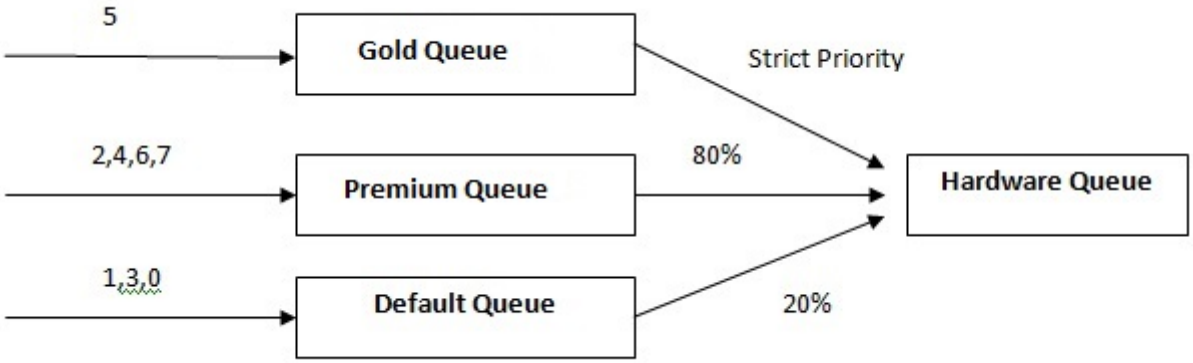

Figure 2.4: Modified Deficit Round Robin

2.1.6 Introduction to QoE

Quality-of-Experience (QoE) is a human centric notion that produces the blue print of human perception, feelings, needs and intentions while Quality-of-Service (QoS) is a technology centric metric used to assess the performance of a multimedia application and/or network. QoE is usually expressed using a Mean Opinion Score (MOS) of 1 (Bad) to 5 (Excellent).

2.2 Hybrid Modeling Framework for Queue Dynamics

Consider a communication network consisting of just 2 nodes connected by a link in which QoS is implemented. This link is characterized by a finite Bandwidth B. We assume that the network is being loaded by a set $\mathcal{F}$ of end-to-end flows. Given a flow $f \in \mathcal{F}$, we denote by r_f , the flow's sending rate, i.e., the rate at which packets are generated and enter the next node. Of course, the sum of the flow sending rates must not exceed the bandwidth B, i.e. :

$$\sum_{f \in \mathcal{F}} r_f \leq B \tag{2.1}$$

In general, the flow sending rates r_f, $f \in \mathcal{F}$ are determined by congestion control mechanisms. Associated with each node, there is a *queue* that temporarily holds packets before transmission. We denote by q_f the number of bytes in this queue that belong to the f-flow. The total number of bytes in the queue is then given by:

$$q := \sum_{f \in \mathcal{F}} q_f \tag{2.2}$$

The queue can hold, at most, a finite number of bytes that we denote by q_{max}. When q reaches q_{max}, drops will occur. Also, we denote by s_f the rate at which flow packets arrive and call it the f-flow arrival rate.

2.2.1 Queue Dynamics

For the queue dynamics, we have the following two assumptions:

ASSUMPTION 1: The packets of different flows arrive at each node in their paths uniformly distributed over time.

ASSUMPTION 2: Packets of different flows are uniformly distributed in each queue.

The continuous queue dynamics for the *l-link* is given by:

$$\dot{q_f} = s_f - d_f - r_f$$

where d_f denotes the *f-flow drop rate.*

The discrete dynamics are given as follows with three discrete states:

- **Empty Queue** ($q = 0$):
  The outgoing rates r_f are equal to the arrival rates s_f and there are no drops as long as the bandwidth constraint (2.1) is not violated. When $\sum_{f \in \mathcal{F}} s_f \geq B$, the available bandwidth B is distributed among all flows proportionately to their arrival rates s_f by assumption 1. To summarize , we have:

$$d_f = 0, r_f = \begin{cases} s_f & \sum_{f \in \mathcal{F}} s_f \leq B \\ \dfrac{s_f}{\sum_{f \in \mathcal{F}} s_f} B & \text{otherwise} \end{cases} \tag{2.3}$$

- **Queue neither empty nor full** ($0 < q < q_{max}$):
  In this case the drops do not occur and because of assumption 2, the available link bandwidth B is distributed among the flows proportionally to their percentage of bytes in the queue.
$$d_f = 0, r_f = \frac{q_f}{\sum_{f \in \mathcal{F}} q_f} B \tag{2.4}$$

- **Queue full and still filling** ($q = q_{max}$ and $\sum_{f \in \mathcal{F}} s_f > B$):
  In this case there will be drops, and the drop rate is given by the difference between the total arrival rate and the total available bandwidth.
$$d = \sum_{f \in \mathcal{F}} s_f - B > 0. \tag{2.5}$$

From assumption 1, the drop rate d should be distributed among all flows proportionally to their arrival rates s_f. Also from assumption 2, the available link bandwidth is distributed among the flows proportionally to their percentage of bytes in the queue. To summarize, we have
$$d_f = \frac{s_f(\sum_{f \in \mathcal{F}} s_f - B)}{\sum_{f \in \mathcal{F}} s_f}, r_f = \frac{q_f}{\sum_{f \in \mathcal{F}} q_f} B \tag{2.6}$$

2.3 Hybrid Modeling Framework for Queue Dynamics in Priority Queuing

Consider a communication network consisting of just 2 nodes connected by a link in which QoS is implemented. It is assumed that the flows have already been classified and IP Precedence has been set. The congestion management mechanism used here is Priority Queuing. In Priority Queuing the link consists of 4 software queues named High, Medium, Normal, Low for different flows based on their IP-Precedence setting. The dynamics for medium, normal and low queue are similar compared to the dynamics of high queue. The assumptions made before for queue dynamics holds here also for each queue.

The continuous queue dynamics as mentioned before holds here also.
$$\dot{q}_{i_f} = s_{i_f} - d_{i_f} - r_{i_f} \text{ where } i \in h, m, n, lw$$
(h denotes high queue, m denotes medium queue, n denotes normal queue and lw denotes low queue).

where:
high queue denotes queue containing high priority flows.
medium queue denotes queue containing medium priority flows.
normal queue denotes queue containing normal priority flows.
low queue denotes queue containing low priority flows.

The following relations between the quantities are equivalent and these are used interchangeably in the forthcoming sections:
$$\sum_{f \in \mathcal{F}_i} s_{i_f} = s_i,\ \sum_{f \in \mathcal{F}_i} r_{i_f} = r_i,\ \sum_{f \in \mathcal{F}_i} q_{i_f} = q_i,\ \sum_{f \in \mathcal{F}_i} d_{i_f} = d_i$$
where $i \in h, m, n, lw$

2.3.1 High Queue Dynamics

As described before in Priority Queuing section,the High queue has the highest priority over all the other queues. The packet from the other queues will be transmitted only if there are no packets in the high queue. This is well explained in Figure 2.5.

The different states of High Queue dynamics are as follows:

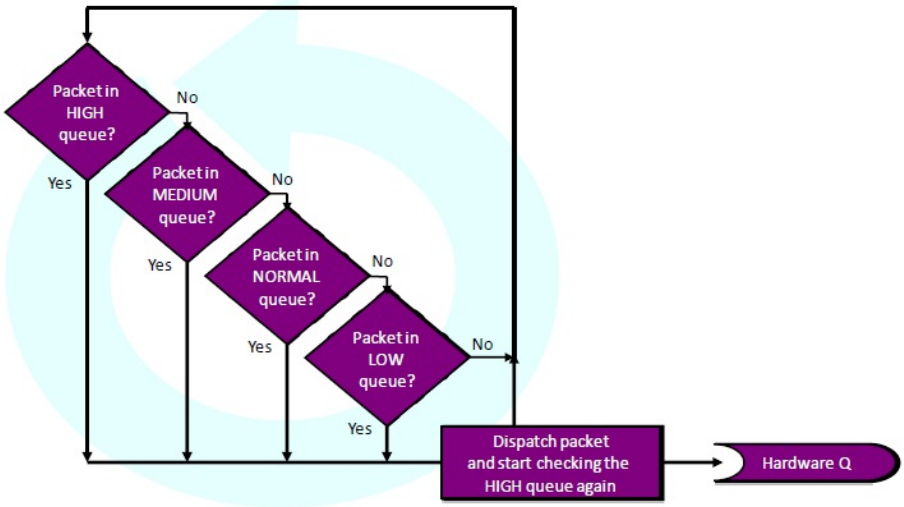

Figure 2.5: Scheduling in Priority Queuing

- **High 1** ($q_h \leq 0$ & $s_h \leq B$)**:**
  The High queue is empty and the bandwidth constraint is not violated. So, the outgoing rates r_{h_f} are equal to the arrival rates s_{h_f}.

High1

$$q_h \leq 0 \quad \& \quad s_h = \sum_{f \in \mathcal{F}_h} s_{h_f} \leq B$$
$$r_{h_f} = s_{h_f} \quad ; \quad \dot{q}_{h_f} = s_{h_f} - r_{h_f} = 0 \quad ; \quad d_{h_f} = 0$$

Figure 2.6: Dynamics of High Queue: state High 1

- **High 2** ($q_h \leq 0$ & $s_h > B$)**:**
  The High queue is empty, but the bandwidth constraint is violated. So the available bandwidth B is distributed among all flows proportionately to their arrival rates s_{h_f} by assumption 1. In other words one has $r_{h_f} = \frac{s_{h_f}}{s_h} B$.

High2

$$q_h \leq 0 \quad \& \quad s_h > B$$
$$r_{h_f} = \frac{s_{h_f}}{s_h} B \quad ; \quad \dot{q}_{h_f} = s_{h_f} - r_{h_f} \quad ; \quad d_{h_f} = 0$$

Figure 2.7: Dynamics of High Queue: state High 2

- **High 3** ($0 < q_h \leq q_{h,max}$ & $s_h \leq B$)**:**
  The High queue has some packets, and the bandwidth constraint is not violated. So there will be no drops even though the length of the queue would be maximum. The available bandwidth (in this case the whole bandwidth)is distributed among the flows proportionally to their percentage of bytes in the queue, in other words one has $r_{h_f} = \frac{q_{h_f}}{q_h} B$.

High3

$$0 < q_h \leq q_{h,max} \quad \& \quad s_h \leq B$$
$$r_{h_f} = \frac{q_{h_f}}{\sum_{f \in \mathcal{F}_h} q_{h_f}} B \quad ; \quad \dot{q}_{h_f} = s_{h_f} - r_{h_f} \quad ; \quad d_{h_f} = 0$$

Figure 2.8: Dynamics of High Queue: state High 3

- **High 4** ($0 < q_h < q_{h,max}$ & $s_h > B$):
  In this case, the bandwidth constraint is violated, but there will be no drops as the queue is not full yet. Instead, queue is filling up. The sending rate is still the same as the previous state $r_{h_f} = \frac{q_{h_f}}{q_h} B$.

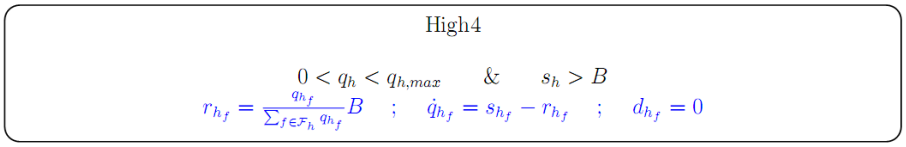

Figure 2.9: Dynamics of High Queue: state High 4

- **High 5** ($q_h = q_{h,max}$ & $s_h > B$):
  The High queue is full and the bandwidth constraint is violated. So there will be drops of packets. From assumption 1, the drop rate d_h should be distributed among all flows proportionally to their arrival rates s_{h_f}. Also, the sending rate is the same as the previous state. So $r_{h_f} = \frac{q_{h_f}}{q_h} B$ & $d_{h_f} = \frac{s_{h_f}}{s_h}(s_h - B)$.

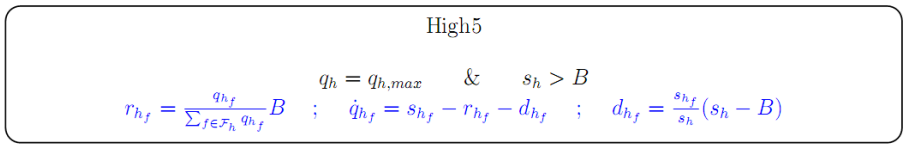

Figure 2.10: Dynamics of High Queue: state High 5

2.3.2 Dynamics of Medium Queue

The packets in the medium queue will be transmitted only if the high queue is empty. Also, there should be some bandwidth left ($B - r_h > 0$) even if the permission for transmission is given ($qh \leq 0$). It is important to mention that the necessary conditions for the Medium states only corresponds to the state High 1, and this means that $r_h = s_h$.

- **Medium 1** ($q_m = 0$ & $q_h \leq 0$ & $B - r_h > 0$ & $s_m \leq (B - s_h)$):
  The Medium queue is empty, and the bandwidth constraint is not violated $s_m \leq (B - s_h)$. The outgoing rates r_{m_f} are equal to the arrival rates s_{m_f}.

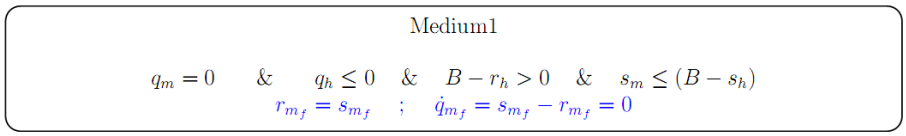

Figure 2.11: Dynamics of Medium Queue: state Medium 1

- **Medium 2** ($q_m = 0$ & $q_h \leq 0$ & $B - r_h > 0$ & $s_m > (B - s_h)$):
  The Medium queue is empty, and the bandwidth constraint is violated $s_m > (B - s_h)$. So the available bandwidth $B - r_h$ is distributed among all flows portionately to their arrival rates s_{m_f} by assumption 1. In other words one has $r_{m_f} = \frac{s_{m_f}}{s_m}(B - s_h)$.

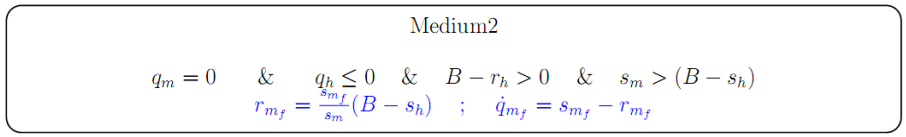

Figure 2.12: Dynamics of Medium Queue: state Medium 2

- **Medium 3** ($0 < q_m \leq q_{m,max}$ & $q_h \leq 0$ & $B - r_h > 0$ & $s_m \leq (B - s_h)$)**:**
The Medium queue has some packets, and bandwidth constraint is not violated. So there will be no drops even though the length of the queue would be maximum. The available bandwidth (in this case $B-r_h$) is distributed among the flows proportionally to their percentage of bytes in the queue, in other words one has $r_{m_f} = \frac{q_{m_f}}{q_m}(B - s_h)$.

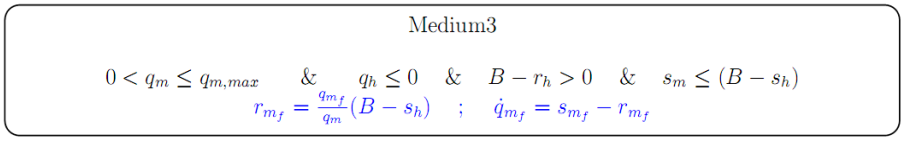

Figure 2.13: Dynamics of Medium Queue: state Medium 3

- **Medium 4** ($0 < q_m < q_{m,max}$ & $q_h \leq 0$ & $B - r_h > 0$ & $s_m > (B - s_h)$)**:**
In this case, the bandwidth constraint is violated, but there will be no drops as the queue is not full yet. Instead, queue is filling up. The sending rate is still the same as the previous state $r_{m_f} = \frac{q_{m_f}}{q_m}(B - s_h)$.

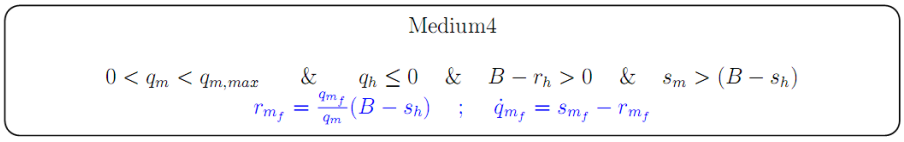

Figure 2.14: Dynamics of Medium Queue: state Medium 4

- **Medium 5** ($q_m = q_{m,max}$ & $q_h \leq 0$ & $B - r_h > 0$ & $s_m > (B - s_h)$)**:**
The Medium queue is full and the bandwidth constraint is violated. So there will be drops of packets. From assumption 1, the drop rate d_m should be distributed among all flows proportionally to their arrival rates s_{m_f} . Also, the sending rate is the same as the previous state. So $r_{m_f} = \frac{q_{m_f}}{q_m}(B - s_h)$ & $d_{m_f} = \frac{s_{m_f}}{s_m}(s_m + s_h - B)$

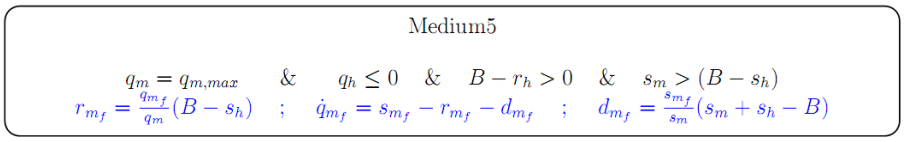

Figure 2.15: Dynamics of Medium Queue: state Medium 5

- **Medium 6** ($0 \leq q_m < q_{m,max}$ & $q_h \leq 0$ & $B - r_h = 0$)**:**
In this state, the available bandwidth is zero. In spite of the given permission for transmission, the sending rate is zero because there is no available bandwidth to use $r_{m_f} = 0$. But there is no dropping as the queue is not full yet.

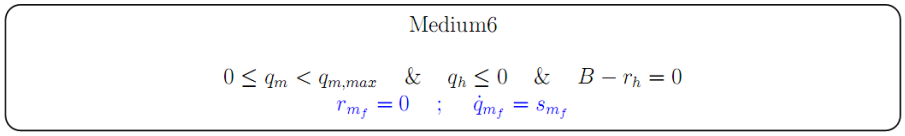

Figure 2.16: Dynamics of Medium Queue: state Medium 6

- **Medium 7** ($q_m = q_{m,max}$ & $q_h \leq 0$ & $B - r_h = 0$)**:**
Dropping occurs $d_{m_f} = \frac{s_{m_f}}{s_m}(s_m + s_h - B) = \frac{s_{m_f}}{s_m}s_m = s_{m_f}$. Pay attention to the simplification.

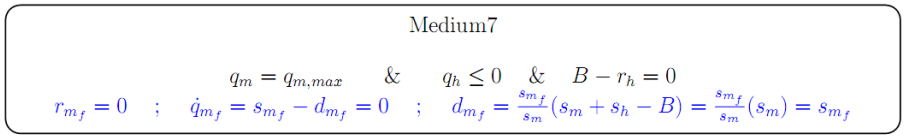

Figure 2.17: Dynamics of Medium Queue: state Medium 7

- **Medium 8** ($0 \leq q_m < q_{m,max}$ & $q_h > 0$)**:**
  The permission for the transmission is not given disregarding the availability of the bandwidth. $r_{m_f} = 0$. Also there is no dropping.

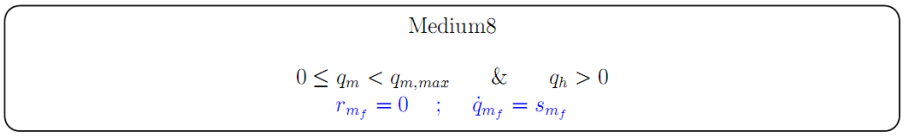

Figure 2.18: Dynamics of Medium Queue: state Medium 8

- **Medium 9** ($q_m = q_{m,max}$ & $q_h > 0$)**:**
  The permission for the transmission is not given disregarding the availability of the bandwidth $r_{m_f} = 0$. Dropping occurs $d_{m_f} = \frac{s_{m_f}}{s_m}(s_m + s_h - B) = \frac{s_{m_f}}{s_m} s_m = s_{m_f}$. Here, the simplification occurs because in all the High states where $q_h > 0$, one has $r_h = B$.

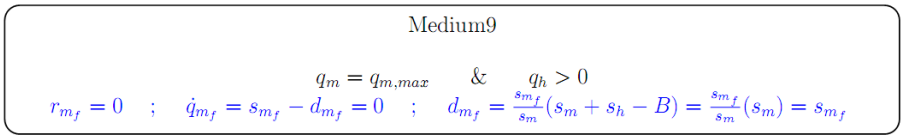

Figure 2.19: Dynamics of Medium Queue: state Medium 9

2.3.3 Dynamics of Normal Queue

The flows in the normal queue will be transmitted only if the high queue and medium queue are empty. Also, there should be some bandwidth left $(B - rh - rm) > 0$ even if the permission for transmission is given ($qh \leq 0$ & $qm \leq 0$). The explanations are pretty much the same. It is important to mention that the necessary conditions for the Normal states only corresponds to the state High 1, and Medium 1. This means that $r_h = s_h$ and $r_m = s_m$.

- **Normal 1** ($q_n = 0$ & $q_h \leq 0$ & $q_m \leq 0$ & $B - r_h - r_m > 0$ & $s_n \leq (B - s_h - s_m)$)**:**
  The Normal queue is empty, and the bandwidth constraint is not violated $s_n \leq (B - s_h - s_m)$. The outgoing rates r_{n_f} are equal to the arrival rates s_{n_f}.

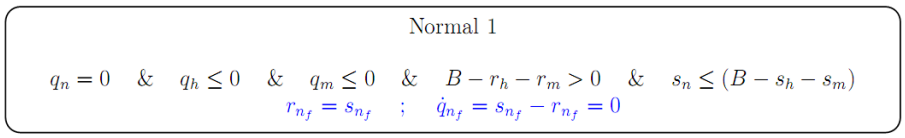

Figure 2.20: Dynamics of Normal Queue: state Normal 1

- **Normal 2** ($q_n = 0$ & $q_h \leq 0$ & $q_m \leq 0$ & $B - r_h - r_m > 0$ & $s_n \leq (B - s_h - s_m)$)**:**
  The Normal queue is empty, and the bandwidth constraint is violated $s_n > (B - s_h - s_m)$. So the available bandwidth $B - r_h - s_m$ is distributed among all flows proportionately to their arrival rates s_{n_f} by assumption 1. In other words one has $r_{n_f} = \frac{s_{n_f}}{s_n}(B - s_h - s_m)$.

Normal 2

$$q_n = 0 \quad \& \quad q_h \leq 0 \quad \& \quad q_m \leq 0 \quad \& \quad B - r_h - r_m > 0 \quad \& \quad s_n > (B - s_h - s_m)$$
$$r_{n_f} = \frac{s_{n_f}}{s_n}(B - s_h - s_m) \quad ; \quad \dot{q}_{n_f} = s_{n_f} - r_{n_f}$$

Figure 2.21: Dynamics of Normal Queue: state Normal 2

- **Normal 3** $(0 \leq q_n < q_{n,max}$ & $q_h \leq 0$ & $q_m \leq 0$ & $B - r_h - r_m > 0$ & $s_n \leq (B - s_h - s_m))$**:** The Normal queue has some packets, and bandwidth constraint is not violated. So there will be no drops even though the length of the queue could be maximum. The available bandwidth (in this case $B - s_h - s_m$is distributed among the flows proportionally to their percentage of bytes in the queue, in other words one has $r_{n_f} = \frac{q_{n_f}}{q_n}(B - s_h - s_m)$.

Normal 3

$$0 < q_n \leq q_{n,max} \quad \& \quad q_h \leq 0 \quad \& \quad q_m \leq 0 \quad \& \quad B - r_h - r_m > 0 \quad \& \quad s_n \leq (B - s_h - s_m)$$
$$r_{n_f} = \frac{q_{n_f}}{q_n}(B - s_h - s_m) \quad ; \quad \dot{q}_{n_f} = s_{n_f} - r_{n_f}$$

Figure 2.22: Dynamics of Normal Queue: state Normal 3

- **Normal 4** $(0 < q_n < q_{n,max}$ & $q_h \leq 0$ & $q_m \leq 0$ & $B - r_h - r_m > 0$ & $s_n > (B - s_h - s_m))$**:** In this case, the bandwidth constraint is violated, but there will be no drops as the queue is not full yet. Instead, queue is filling up. The sending rate is still the same as the previous state $r_{n_f} = \frac{q_{n_f}}{q_n}(B - s_h - s_m)$.

Normal 4

$$0 < q_n < q_{n,max} \quad \& \quad q_h \leq 0 \quad \& \quad q_m \leq 0 \quad \& \quad B - r_h - r_m > 0 \quad \& \quad s_n > (B - s_h - s_m)$$
$$r_{n_f} = \frac{q_{n_f}}{q_n}(B - s_h - s_m) \quad ; \quad \dot{q}_{n_f} = s_{n_f} - r_{n_f}$$

Figure 2.23: Dynamics of Normal Queue: state Normal 4

- **Normal 5** $(q_n = q_{n,max}$ & $q_h \leq 0$ & $q_m \leq 0$ & $B - r_h - r_m > 0$ & $s_n > (B - s_h - s_m))$**:** The Normal queue is full and the bandwidth constraint is violated. So there will be drops of packets. From assumption 1, the drop rate d_n should be distributed among all flows proportionally to their arrival rates s_{n_f} . Also, the sending rate is the same as the previous state. So $r_{n_f} = \frac{q_{n_f}}{q_n}(B - s_h - s_m)$ & $d_{n_f} = \frac{s_{n_f}}{s_n}(s_n + s_m + s_h - B)$.

Normal 5

$$q_n = q_{n,max} \quad \& \quad q_h \leq 0 \quad \& \quad q_m \leq 0 \quad \& \quad B - r_h - r_m > 0 \quad \& \quad s_n > (B - s_h - s_m)$$
$$r_{n_f} = \frac{q_{n_f}}{q_n}(B - s_h - s_m) \quad ; \quad \dot{q}_{n_f} = s_{n_f} - r_{n_f} - d_{n_f} \quad ; \quad d_{n_f} = \frac{s_{n_f}}{s_n}(s_n + s_m + s_h - B)$$

Figure 2.24: Dynamics of Normal Queue: state Normal 5

- **Normal 6** $(0 \leq q_n < q_{n,max}$ & $q_h \leq 0$ & $B - r_h - r_m = 0)$**:** In this state, the available bandwidth is zero. In spite of the given permission for transmission, the sending rate is zero because there is no available bandwidth to use $r_{n_f} = 0$. But there is no dropping as the queue is not full yet.
- **Normal 7** $(q_n = q_{n,max}$ & $q_h \leq 0$ & $B - r_h - r_m = 0)$**:** Dropping occurs $d_{n_f} = \frac{s_{n_f}}{s_n}(s_n + s_m + s_h - B) = \frac{s_{n_f}}{s_n}s_n = s_{n_f}$. Pay attention to the simplification.

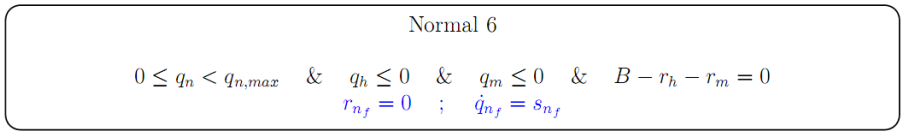

Figure 2.25: Dynamics of Normal Queue: state Normal 6

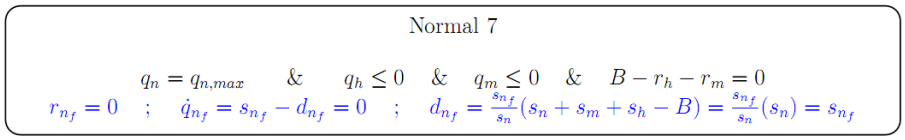

Figure 2.26: Dynamics of Normal Queue: state Normal 7

- **Normal 8** ($0 \le q_n < q_{n,max}$ & $q_h > 0$)**:**
  The permission for the transmission is not given disregarding the availability of the bandwidth. $r_{n_f} = 0$.Here notice that either $qh > 0$ or $qm > 0$is enough for prohibiting transmission. Also there is no dropping.

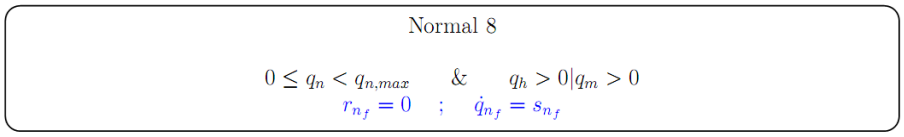

Figure 2.27: Dynamics of Normal Queue: state Normal 8

- **Normal 9** ($q_n = q_{n,max}$ & $q_h > 0$)**:**
  The permission for the transmission is not given disregarding the availability of the bandwidth $r_{n_f} = 0$. Dropping occurs $d_{n_f} = \frac{s_{n_f}}{s_n}(s_n + s_m + s_h - B) = \frac{s_{n_f}}{s_n} s_n = s_{n_f}$.Here, the simplification occurs because in any case, either $q_h > 0$ or $q_m > 0$ or both, one has $r_m + r_h = B$.

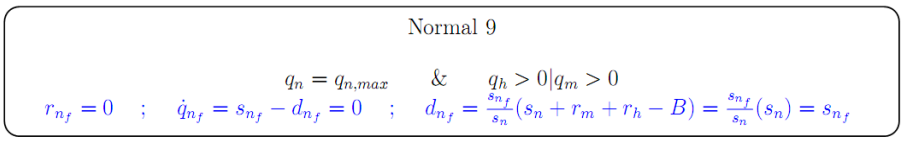

Figure 2.28: Dynamics of Normal Queue: state Normal 9

2.3.4 Dynamics of Low Queue

The flows in the low queue will be transmitted only if the high,medium and normal queues are empty. Also, there should be some bandwidth left $(B - r_h - r_m - r_n) > 0$ even if the permission for transmission is given ($q_h \le 0$ & $q_m \le 0$ & $q_n \le 0$). The explanations are pretty much the same. The dynamics are shown in Figure 2.29.

2.4 Hybrid Modeling Framework for Queue Dynamics in Modified Deficit Round Robin

Consider a communication network consisting of just 2 nodes connected by a link in which QoS is implemented. It is assumed that the flows have already been classified and IP Precedence has been set. The congestion management mechanism used here is Modified Deficit Round Robin. The hybrid model of MDRR is just the extension of the Queue dynamics of priority queuing described in the previous section. In MDRR the link consists

Figure 2.29: Dynamics of Low Queue

of 3 software queues named Gold, Premium and Default for different flows based on their IP-Precedence setting (or High, Medium and Normal queues). The assumptions made in chapter 2 for queue dynamics holds here also for each queue. The continuous queue dynamics as mentioned before holds here also.

$$\dot{q_{i_f}} = s_{i_f} - d_{i_f} - r_{i_f} \text{ where } i \in h, m, n, lw$$

(h denotes High queue, m denotes Medium queue, n denotes Normal queue).

where:

High queue denotes queue containing high priority flows(IP Precedence 5).

Medium queue denotes queue containing medium priority flows(IP Precedence 2,4,6 and 7).
Normal queue denotes queue containing default priority flows(IP Precedence 1,3 and 0).

The following relations between the quantities are equivalent and these are used interchangeably in the forthcoming sections:

$$\sum_{f\in\mathcal{F}_i} s_{i_f} = s_i,\ \sum_{f\in\mathcal{F}_i} r_{i_f} = r_i,\ \sum_{f\in\mathcal{F}_i} q_{i_f} = q_i,\ \sum_{f\in\mathcal{F}_i} d_{i_f} = d_i$$

where $i \in h, m, n$

2.4.1 High Queue Dynamics

High queue has strict priority over other two queues. It means that as long as the buffer for this queue has some packets, we are not allowed to send the packets from other queues. But if the High buffer is empty, we are allowed to dedicate the left bandwidth (the whole Bandwidth subtracted by the sending rate of High queue) to other two queues based on MDRR policy as we will see. Nevertheless, there could be a case that we have the permission to send, but the sending rate of Medium and Normal queues will become zero because the left bandwidth is zero. (The high queue is consuming all the Bandwidth).

- **state h1(The queue is empty)**
  The length of High queue (or High buffer) is zero, and there is no stress on the bandwidth. Therefore all the queues send packets with the same rate as they receive.

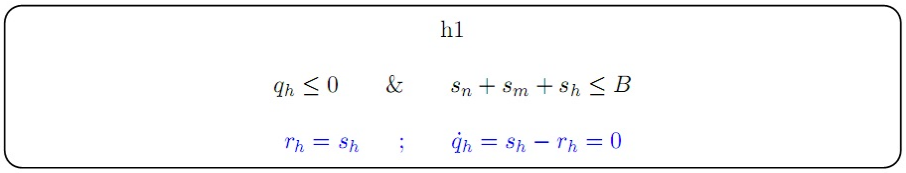

Figure 2.30: state h1(The queue is empty)

- **state h2(The queue is empty)**
  In this state, the sending rate is equal to the incoming rate like before, and there is some left bandwidth which can be allocated to other two queues.

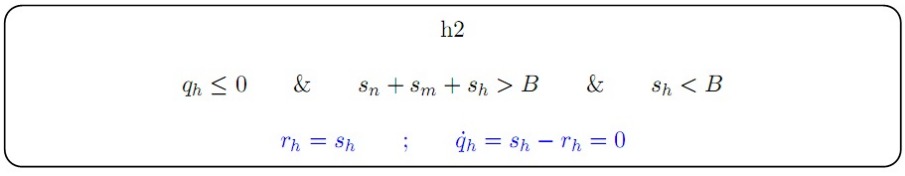

Figure 2.31: state h2(The queue is empty)

- **state h3(The queue is empty)**
  Here we have $s_h > B$. Even though other two queues are allowed to send packets, but their sending rates are zero because the left bandwidth is zero.

- **state h4(The queue is filling)**
  The High buffer has some packets. Therefore, other two queues are not allowed to send anything. Since the incoming rate is less than the bandwidth, packet dropping will not occur, even though the length of the buffer could be equal to the maximum length. However the sending rate is not equal to the incoming rate. This is different from the previous cases. The idea is that when the buffer is not empty, the sending

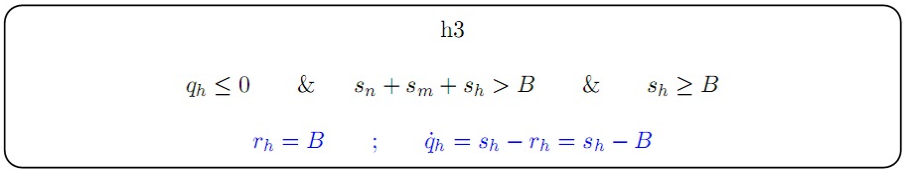

Figure 2.32: state h3(The queue is empty)

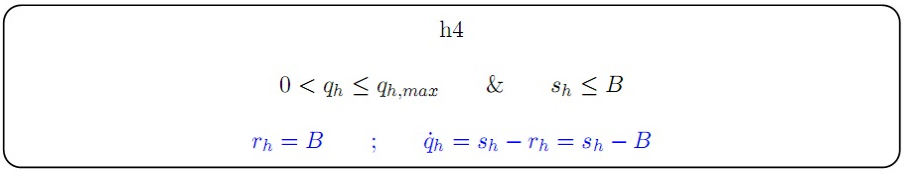

Figure 2.33: state h4(The queue is filling)

rate can be even more than the incoming rate. In other words, one would use all the bandwidth in order to make the queue empty as quickly as possible.

- **state h5(The queue is filling)**
  Since the incoming rate s_h is bigger than B, the equality on q_h should be removed, otherwise dropping will occur. But it will be considered as a different state.

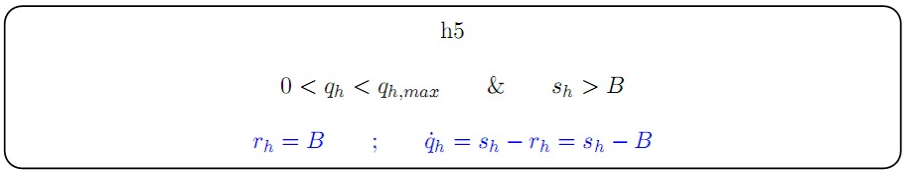

Figure 2.34: state h5(The queue is filling)

- **state h6(Dropping occurs)**
  When the queue is full, and the incoming rate is bigger than the bandwidth, the incoming flows are dropped. The total dropping rate is equal to the difference between the incoming rate and the bandwidth, and it has to be distributed among the flows based on their participation in the incoming rate ($d_h = (s_h - B)$ & $d_{h_f} = \frac{s_{h_f}}{s_h} d_h$) but here the High queue has only one type of flow.

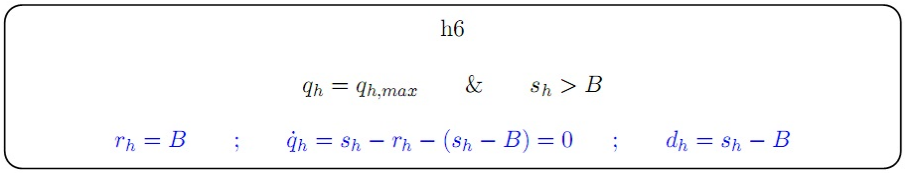

Figure 2.35: state h6(Dropping occurs)

2.4.2 Dynamics of Medium Queue

The packets in the Medium queue will be transmitted only if the high queue is empty, and there is also some bandwidth left.

- **state M1(The queue is empty)**
  In this state the queue is empty. The inequality $B - r_h > 0$ implies that there is some bandwidth left with which the Medium packets can be sent (It is called the available bandwidth). When $s_n + s_m \leq (B - r_h)$, there is enough bandwidth to send both Medium and Normal packets, without the need for implementing MDRR policy, and

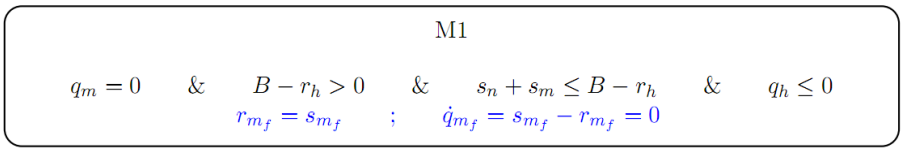

Figure 2.36: state M1(The queue is empty)

therefore the sending rates of both Medium and Normal queues are the same as their incoming rates. And of course $q_h \leq 0$ gives us the permission to dispatch packets from Medium queue.

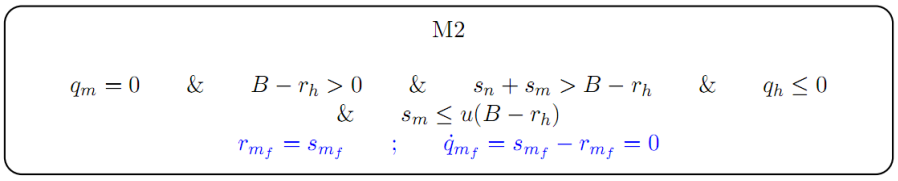

Figure 2.37: state M2(The queue is empty)

- **state M2(The queue is empty)**
  The difference with the previous state is that the sum of incoming rates is bigger than the left bandwidth. In order to know how the left bandwidth should be distributed among two queues, the space will be partitioned more by adding another constraint $s_m < u(B - r_h)$. This constraint tells us that the receiving rate s_m will be equal the sending rate r_m. Here, u is a number between one and zero.

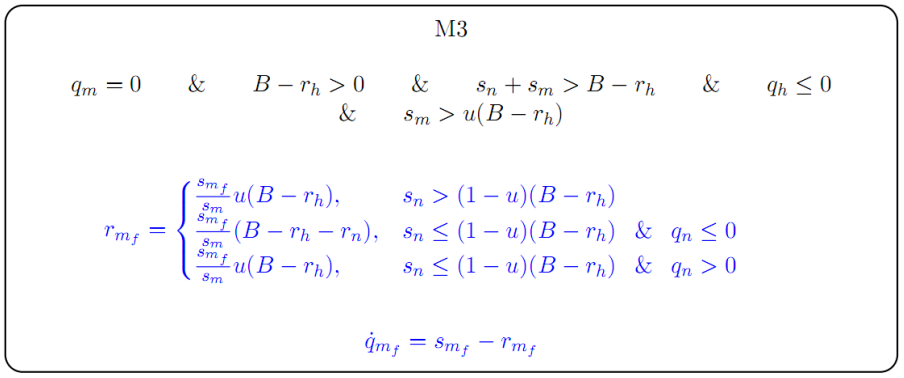

Figure 2.38: state M3(The queue is empty)

- **state M3(The queue is empty)**
  In this case, the sending rate is dependent on s_n and q_n. The idea is that when $s_n \leq (1-u)(B - r_h)$ & $q_n \leq 0$, the Normal queue dispatches the packets with the same rate as it receives, and it dedicates the leftover to the Medium queue, which is something greater than $u(B - r_h)$ because the Medium queue gets emptied sooner this way. When $s_n \leq (1-u)(B - r_h)$ & $q_n > 0$ only $u(B - r_h)$ of the bandwidth is dedicated to the Medium queue because the same idea stands here which was saying one has to empty the Normal queue as quickly as possible if it has some packets. So, Based on MDRR policy, $(1-u)(B - r_h)$ of the available bandwidth should be dedicated to the Normal queue and the rest to the Medium one. When $s_n > (1-u)(B - r_h)$, MDRR is used because we already have $s_m > u(B - r_h)$

- **state M4(The queue is filling)**
  In this case, the buffer is not empty. So the sending rate, could be larger than the incoming rate. When $q_n \leq 0$, the Normal packets are sent with the same incoming

M4

$$0 < q_m \leq q_{m,max} \quad \& \quad B - r_h > 0 \quad \& \quad s_n + s_m \leq B - r_h \quad \& \quad q_h \leq 0$$

$$r_{m_f} = \begin{cases} \frac{q_{m_f}}{q_m}(B - r_h - r_n), & q_n \leq 0 \\ \frac{q_{m_f}}{q_m}(s_m + u(B - r_h - s_n - s_m)), & q_n > 0 \end{cases}$$

$$\dot{q}_{m_f} = s_{m_f} - r_{m_f}$$

Figure 2.39: state M4(The queue is filling)

rate, but since the Medium buffer is not empty, the remaining part of the available bandwidth $B - r_h - s_n$ is dedicated to the Medium queue. But when $q_n > 0$, the sending rate of both Medium and Normal queues can be larger than their incoming rates. Note that, $s_n + s_m \leq (B - r_h)$ means that there is enough bandwidth to send both Medium and Normal packets. Therefore the remaining part of the available bandwidth (whatever is left after subtracting s_m and s_n, i,e $(B - r_h - s_n - s_m)$ is distributed between two queues in an MDRR way.

M5

$$0 < q_m \leq q_{m,max} \quad \& \quad B - r_h > 0 \quad \& \quad s_n + s_m > B - r_h \quad \& \quad q_h \leq 0$$
$$\& \quad s_m \leq u(B - r_h)$$
$$r_{m_f} = \frac{q_{m_f}}{q_m} u(B - r_h)$$
$$\dot{q}_{m_f} = s_{m_f} - r_{m_f}$$

Figure 2.40: state M5(The queue is filling)

- **state M5(The queue is filling)**
  Since $s_n + s_m > B - r_h$ and $s_m \leq u(B - r_h)$, then s_n has to be greater than $(1-u)(B-r_h)$. At first, it may come to mind that the sending rate should be equal to the coming rate ($r_m = s_m$), but smarter way is to dedicate $(1-u)(B-r_h)$ of the left bandwidth to the Normal queue (According to MDRR policy) and dedicate $u(B-r_h)$ of the left bandwidth to the Medium queue (more than s_m) because Medium buffer has some packets and we would like to empty the buffer as quickly as possible.

M6

$$0 < q_m < q_{m,max} \quad \& \quad B - r_h > 0 \quad \& \quad s_n + s_m > B - r_h \quad \& \quad q_h \leq 0$$
$$\& \quad s_m > u(B - r_h)$$

$$r_{m_f} = \begin{cases} \frac{q_{m_f}}{q_m}(B - r_h - r_n), & s_n \leq (1-u)(B - r_h) \quad \& \quad q_n \leq 0 \\ \frac{q_{m_f}}{q_m} u(B - r_h), & s_n > (1-u)(B - r_h) \\ \frac{q_{m_f}}{q_m} u(B - r_h), & s_n \leq (1-u)(B - r_h) \quad \& \quad q_n > 0 \end{cases}$$

$$\dot{q}_{m_f} = s_{m_f} - r_{m_f}$$

Figure 2.41: state M6(The queue is filling)

- **state M6(The queue is filling)**
  In this case, the sending rate is dependent on s_n.The same explanation of the state M3 holds here.

M7

$$q_m = q_{m,max} \quad \& \quad B - r_h > 0 \quad \& \quad s_n + s_m > B - r_h \quad \& \quad q_h \leq 0$$
$$\& \quad s_m > u(B - r_h)$$

$$r_{m_f} = \begin{cases} \frac{q_{m_f}}{q_m}(B - r_h - r_n), & s_n \leq (1-u)(B - r_h) \ \& \ q_n \leq 0 \ \Rightarrow \ d_m = s_m - r_m \\ \frac{q_{m_f}}{q_m}u(B - r_h), & s_n \leq (1-u)(B - r_h) \ \& \ q_n > 0 \ \Rightarrow \ d_m = s_m - r_m \\ \frac{q_{m_f}}{q_m}u(B - r_h), & s_n > (1-u)(B - r_h) \ \Rightarrow \ d_m = s_m - r_m \end{cases}$$

$$\dot{q}_{m_f} = s_{m_f} - r_{m_f} - d_{m_f} \quad ; \quad d_{m_f} = \frac{s_{m_f}}{s_m} d_m$$

Figure 2.42: state M7(The queue is full, dropping occurs)

- **state M7(The queue is full, dropping occurs)**
  The same explanations as for the state M6 are true here with the difference that here also dropping occurs.

M8

$$0 \leq q_m < q_{m,max} \quad \& \quad B - r_h = 0 \quad \& \quad q_h \leq 0$$
$$\dot{q}_{m_f} = s_{m_f}$$

Figure 2.43: The queue is filling, the available bandwidth is zero, no packet is sent

- **state M8(The queue is filling, no packet is sent because the available bandwidth is zero)**
  Since the available bandwidth is zero, the sending rate is zero practically even though sending packets are allowed $(qh \leq 0)$.

M9

$$q_m = q_{m,max} \quad \& \quad B - r_h = 0 \quad \& \quad q_h \leq 0$$
$$d_{m_f} = s_{m_f} \quad \Rightarrow \quad \dot{q}_{m_f} = s_{m_f} - d_{m_f} = 0$$

Figure 2.44: state M9(The queue is full, the available bandwidth is zero, dropping occurs

- **state M9(The queue is full, the available bandwidth is zero, dropping occurs)**
  The difference with the previous state is that since the buffer is full, the incoming packets should be dropped.

M10

$$0 \leq q_m < q_{m,max} \quad \& \quad q_h > 0$$
$$\dot{q}_{m_f} = s_{m_f}$$

Figure 2.45: state M10(The queue is filling)

- **state M10(The queue is filling, but dispatching is not allowed)**
  Since the High buffer is not empty, sending packets is not allowed. So, they get filled up.

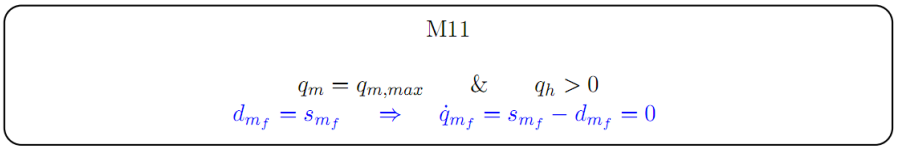

Figure 2.46: state M11(The queue is full,dropping occurs)

- **state M11(The queue is full, but dispatching is not allowed and dropping occurs)**
  The buffer is full, so the packets are dropped.

2.4.3 Dynamics of Normal Queue

The hybrid model of Normal queue is very similar to that of Medium queue.

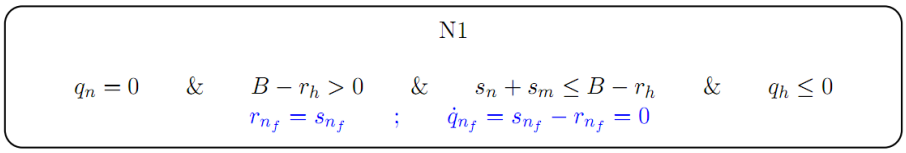

Figure 2.47: state N1(The queue is empty)

- **state N1(The queue is empty)**
  This state is exactly like its correspondent state from Medium queue. There is no need for implementing MDRR policy, and sending rate is equal to the incoming rate.

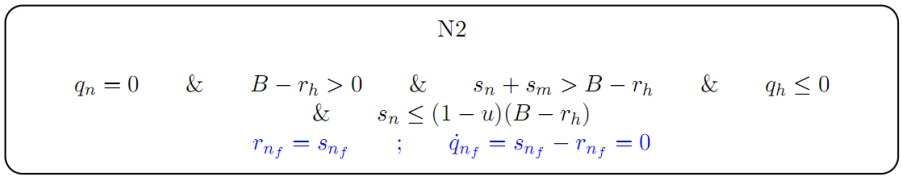

Figure 2.48: state N2(The queue is empty)

- **state N2(The queue is empty)**
  This state is also like the state M2.

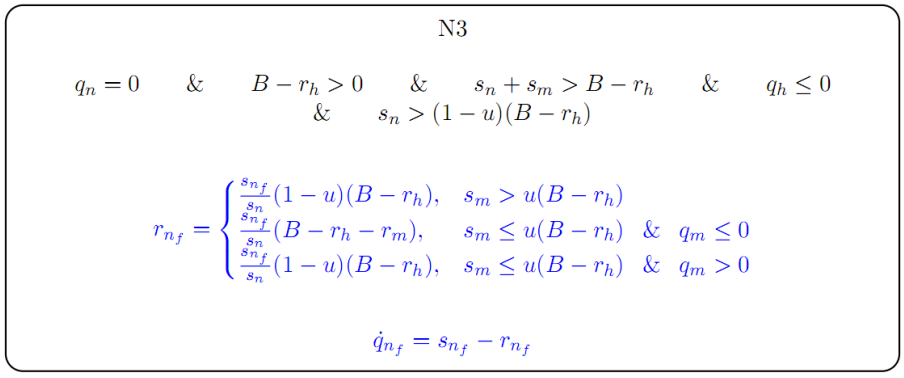

Figure 2.49: state N3(The queue is empty)

- **state N3(The queue is empty)**
  In this case, the sending rate is dependent on s_m. The idea is that when $s_m \leq u(B - r_h)$ & $q_m \leq 0$, the Medium queue dispatches the packets with the same rate as it receives, and it dedicates the leftover to the Normal queue, which is something

greater than $(1-u)(B-r_h)$ because the Normal queue gets emptied sooner this way. When $s_m \leq u(B-r_h)$ & $q_m > 0$ only $(1-u)(B-r_h)$ of the bandwidth is dedicated to the Normal queue because the same idea stands here which was saying one has to empty the Medium queue as quickly as possible if it has some packets. So, Based on MDRR policy, $u(B-r_h)$ of the available bandwidth should be dedicated to the Medium queue and the rest to the Normal one. When $s_m > u(B-r_h)$, MDRR is used because we already have $s_n > (1-u)(B-r_h)$

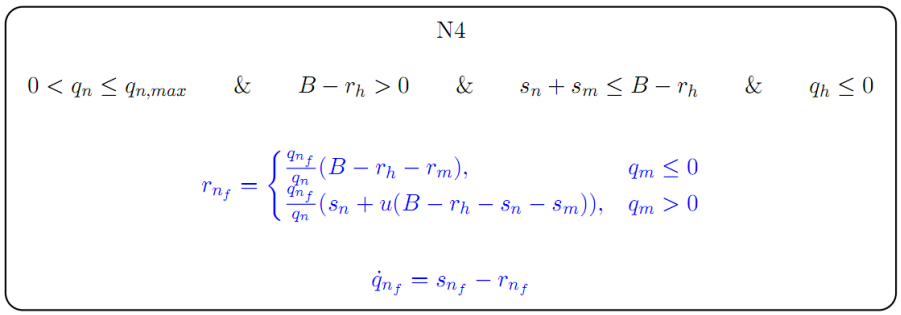

Figure 2.50: state N4(The queue is filling)

- **state N4(The queue is filling)**
  In this case, the buffer is not empty. So the sending rate, could be larger than the incoming rate. When $q_m \leq 0$, the Medium packets are sent with the same incoming rate, but since the Normal buffer is not empty, the remaining part of the available bandwidth $(B - r_h - s_m)$ is dedicated to the Normal queue. But when $q_m > 0$, the sending rate of both Normal and Medium queues can be larger than their incoming rates. Therefore the remaining part of the available bandwidth (whatever is left after subtracting $s_m$and s_n), i,e $B - r_h - s_n - s_m$ is distributed between two queues in an MDRR way.

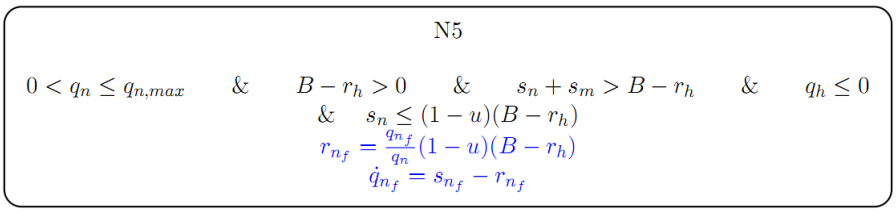

Figure 2.51: state N5(The queue is filling)

- **state N5(The queue is filling)**
  Since $s_n + s_m > B - r_h$ and $s_n \leq (1-u)(B-r_h)$, then s_m has to be greater than $u(B-r_h)$. At first, it may come to mind that the sending rate should be equal to the incoming rate ($r_n = s_n$), but the smarter way is to dedicate $u(B-r_h)$ of the available bandwidth to the Medium queue (According to MDRR policy) and dedicate $(1-u)(B-r_h)$ of the available bandwidth to the Normal queue (more than s_n) because Normal buffer has some packets and we would like to empty the buffer as quickly as possible.

- **state N6(The queue is filling)**
  In this case, the sending rate is dependent on s_m. The same explanation of the state N3 holds here.

- **state N7(The queue is full, dropping occurs)**
  The same explanations as for the state N6 are true here with the difference that here also dropping occurs.

N6

$$0 < q_n < q_{n,max} \quad \& \quad B - r_h > 0 \quad \& \quad s_n + s_m > B - r_h \quad \& \quad q_h \leq 0$$
$$\& \quad s_n > (1-u)(B - r_h)$$

$$r_{m_f} = \begin{cases} \frac{q_{n_f}}{q_n}(B - r_h - r_m), & s_m \leq u(B - r_h) \ \& \ q_m \leq 0 \\ \frac{q_{n_f}}{q_n}(1-u)(B - r_h), & s_m > u(B - r_h) \\ \frac{q_{n_f}}{q_n}(1-u)(B - r_h), & s_m \leq u(B - r_h) \ \& \ q_m > 0 \end{cases}$$

$$\dot{q}_{n_f} = s_{n_f} - r_{n_f}$$

Figure 2.52: state N6(The queue is filling)

N7

$$q_n = q_{n,max} \quad \& \quad B - r_h > 0 \quad \& \quad s_n + s_m > B - r_h \quad \& \quad q_h \leq 0$$
$$\& \quad s_n > (1-u)(B - r_h)$$

$$r_{n_f} = \begin{cases} \frac{q_{n_f}}{q_n}(B - r_h - r_m), & s_m \leq u(B - r_h) \ \& \ q_m \leq 0 \ \Rightarrow \ d_n = s_n - r_n \\ \frac{q_{n_f}}{q_n}(1-u)(B - r_h), & s_m \leq u(B - r_h) \ \& \ q_m > 0 \ \Rightarrow \ d_n = s_n - r_n \\ \frac{q_{n_f}}{q_n}(1-u)(B - r_h), & s_m > u(B - r_h) \ \Rightarrow \ d_n = s_n - r_n \end{cases}$$

$$\dot{q}_{n_f} = s_{n_f} - r_{n_f} - d_{n_f} \ ; \ d_{n_f} = \frac{s_{n_f}}{s_n} d_n$$

Figure 2.53: state N7(The queue is full, dropping occurs)

N8

$$0 \leq q_n < q_{n,max} \quad \& \quad B - r_h = 0 \quad \& \quad q_h \leq 0$$
$$\dot{q}_{n_f} = s_{n_f}$$

Figure 2.54: The queue is filling, the available bandwidth is zero, no packet is sent

- **state N8 (The queue is filling , no packet is sent because the available bandwidth is zero)**
  Since the available bandwidth is zero, the sending rate is practically zero, even though sending packets are allowed ($q_h \leq 0$).

N9

$$q_n = q_{n,max} \quad \& \quad B - r_h = 0 \quad \& \quad q_h \leq 0$$
$$d_{n_f} = s_{n_f} \quad \Rightarrow \quad \dot{q}_{n_f} = s_{n_f} - d_{n_f} = 0$$

Figure 2.55: state N9(The queue is full, the available bandwidth is zero, dropping occurs)

- **state N9(The queue is full, the available bandwidth is zero, dropping occurs)**
  The difference with the previous state is that since the buffer is full, the incoming packets should be dropped.
- **state N10(The queue is filling, but dispatching is not allowed)**
  Since the High buffer is not empty, sending packets is not allowed. So, they get filled up.

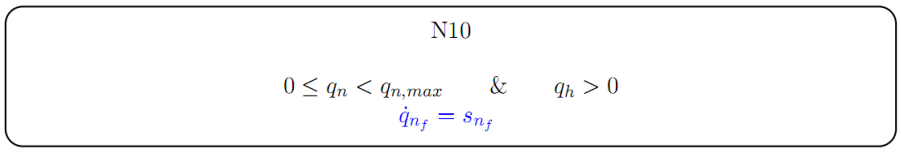

Figure 2.56: state N10(The queue is filling)

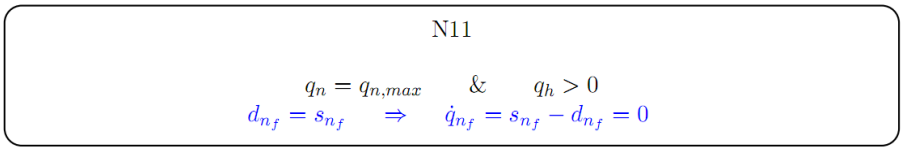

Figure 2.57: state N11(The queue is full, dropping occurs)

- **state N11(The queue is full, but dispatching is not allowed and dropping occurs)**
  The buffer is full, so the packets are dropped.

2.5 Final Hybrid Model

In this section, the final hybrid model is presented which is a combination of three queues (High, Medium and Normal). Every state in any of the queues can be matched to a state from another queue, but not all the combinations are possible. The reason is that some of the constraints in a state of a queue might contradict other constraints in a state of another queue. For example, the state h1 cannot be combined with the state M2 because in h1, one has $s_m + s_n + s_h \leq B$ but in M2, one has $s_m + s_n > B - r_h$ which obviously contradict each other. However the combination of h1 with M1 is possible, and this twosome can be combined with N1 as well such that h1M1N1 is a state of the final hybrid model. Calculating all the possible combinations is a trivial task, and there is no need to do it in this thesis. So, the writer only brings the results:

Possible combinations with h1:
h1,M1,N1 — h1,M1,N4 — h1,M4,N1 — h1,M4,N4

Possible combinations, with h2:
h2,M2,N3 — h2,M2,N6 — h2,M2,N7
h2,M3,N2 — h2,M3,N3 — h2,M3,N5 — h2,M3,N6 — h2,M3,N7
h2,M5,N3 — h2,M5,N6 — h2,M5,N7
h2,M6,N2 — h2,M6,N3 — h2,M6,N5 — h2,M6,N6 — h2,M6,N7
h2,M7,N2 — h2,M7,N3 — h2,M7,N5 — h2,M7,N6 — h2,M7,N7
In general, there are 21 states with h2.

In the state of h3, the available bandwidth will become zero $B - r_h = 0$. Therefore it cannot be combined with many states of the medium and normal queues. We only have:
h3,M8,N8 — h3,M8,N9 — h3,M9,N8 — h3,M9,N9

Possible combinations, with hi: (i= 4, 5, 6)
hi,M10,N10 — hi,M10,N11 — hi,M11,N10 — hi,M11,N11

So the final hybrid model has 4+21+4+4+4+4= 41 states. Like before, in each state of this final hybrid model, there are some constraints which specify the state (the set of constraints comes from the combination of the constraints of the merged states)and there is a dynamic correspondent to that state. Here, a different name for each state is adopted. For example, the state h1,M1,N1 is called $\boldsymbol{\Theta}1$ and so on. State $\boldsymbol{\Theta}4$= h1,M4,N4 and state

$\boldsymbol{\Theta}$25= h2,M7,N7 are presented in section 4.1.1.1.36

Chapter 3

Model Predictive Control

3.1 Introduction

Optimal operation and control of dynamic systems and processes has been a subject of significant research for many years. Important early results on optimal control of dynamic systems include optimal control based on the Hamilton-Jacobi-Bellman equation, Pontryagin's maximum principle, and the linear quadratic regulator . One methodology for improving process performance is to employ the solution of optimal control problems (OCPs) on-line. In other words, control actions for the manipulated inputs of a process are computed by formulating and solving a dynamic optimization problem on-line that takes advantage of a dynamic process model while accounting for process constraints. With the available computing power of modern computers, solving complex dynamic optimization problems (e.g., large-scale, nonlinear, and non-convex optimization problems) on-line is becoming an increasingly viable option to use as a control scheme to improve the steady-state and dynamic performance of process operations [11].

Model Predictive Control or MPC is a natural control framework to deal with the design of coordinated, distributed control systems because of its ability to handle input and state constraints and predict the evolution of a system with time while accounting for the effect of asynchronous and delayed sampling, as well as because it can account for the actions of other actuators in computing the control action of a given set of control actuators in real-time [12].

3.2 Mathematical Formulation

Model predictive control (MPC) is widely adopted in industry as an effective approach to deal with large multi-variable constrained control problems. The main idea of MPC is to choose control actions by repeatedly solving an on-line constrained optimization problem, which aims at minimizing a performance index(objective/cost function) over a finite prediction horizon based on predictions obtained by a system model.

3.2.1 The general design of MPC

In general, an MPC design is composed of three components:

A model of the system-This model is used to predict the future evolution of the system in open-loop and the efficiency of the calculated control actions of an MPC depends highly on the accuracy of the model.

A performance index over a finite horizon- This index will be minimized subject to constraints imposed by the system model, restrictions on control inputs and system state and other considerations at each sampling time to obtain a trajectory of future control inputs.

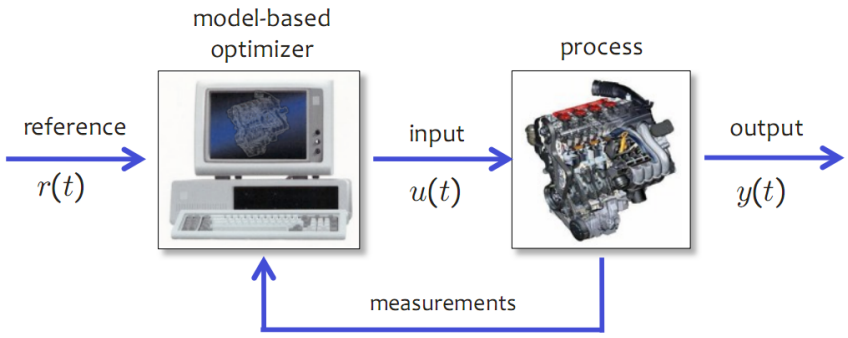

Figure 3.1: Model Predictive Control.

A receding horizon scheme- This scheme introduces feedback into the control law to compensate for disturbances and modeling errors [12]

3.2.2 Standard MPC formulation

The standard MPC is formulated as follows:

$$U_t^*(x(t)) := \underset{U_t}{\operatorname{argmin}} \sum_{k=0}^{N-1} q(x_{t+k}, u_{t+k}) \qquad \text{objectivefunction} \qquad (3.1)$$

subj.to

$$x_t = x(t) \qquad \text{measurements} \qquad (3.2)$$

$$x_{t+k+1} = Ax_{t+k} + Bu_{t+k} \qquad \text{system model} \qquad (3.3)$$

$$x_{t+k} \in \mathcal{X} \qquad \text{state constraints} \qquad (3.4)$$

$$u_{t+k} \in \mathcal{U} \qquad \text{input constraints} \qquad (3.5)$$

$$U_t = \{u_t, u_{t+1}, ..., u_{t+N-1}\} \qquad \text{optimization variables} \qquad (3.6)$$

Where:

- Equation (3.1) represents the objective that is minimized , e.g., distance from origin, sum of squared/absolute errors, economic,etc.
- Equation (3.2) represents the state measured/estimated that is fed back to the model as shown in figure 3.1
- Equation (3.3) represents the internal system model to predict system behavior e.g., linear, nonlinear, single-/multi-variable, etc
- Equations (3.4) & (3.5)represent state and input Constraints that have to be satisfied.

The idea of MPC is explained even better with figure 4.1 and Figure 3.2.

At each sampling time:

- Measure / estimate current state x(t)
- Find the optimal input sequence for the entire future horizon N: $U_t^* = \{u_t^*, u_{t+1}^*, ..., u_{t+N1}^*\}$
- Implement only the first control action u_t^*.

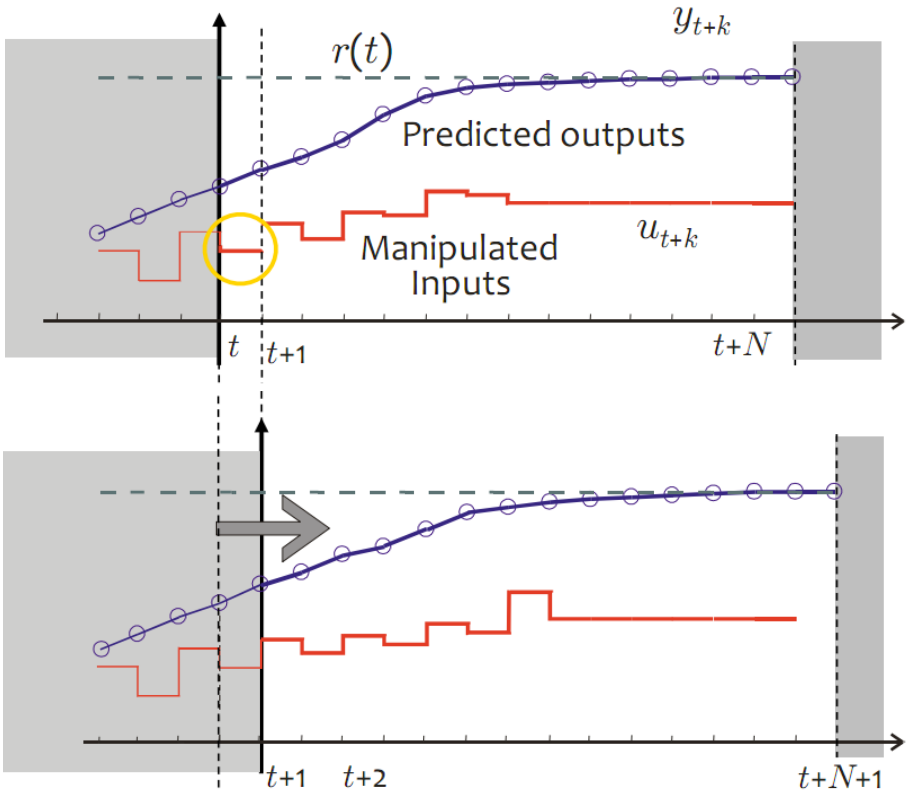

Figure 3.2: on-line Receding Horizon Control

3.3 Tools

3.3.1 Software for modeling optimization problems:

The following tools are used to formulate the optimization problem in mathematical language and then it should be passed to solver:

YALMIP(`users.isy.liu.se/johanl/yalmip`)**:** is a modelling language for advanced modeling and solution of convex and nonconvex optimization problems. It is implemented as a free (as in no charge) toolbox for MATLAB. The main motivation for using YALMIP is rapid algorithm development.

CVX(`cvxr.com/cvx`)**:** is a Matlab-based modeling system for convex optimization. CVX turns Matlab into a modeling language, allowing constraints and objectives to be specified using standard Matlab expression syntax.

AMPL (`www.ampl.com`)**:** industry standard, proprietary software. Supports basically all solvers.

GAMS (`www.gams.com`)**:** commercial high-level modeling system for large-scale optimization. Supports many different types of problems (LPs, QCQPs, MILPs, MINLPs, etc.) and solvers.

JuMP: is an open-source modeling language that allows users to express a wide range of optimization problems (linear, mixed-integer, quadratic, conic-quadratic, semidefinite, and nonlinear) in a high-level, algebraic syntax.

AIMMS(`main.aimms.com/aimms/overview/`)**:** is a development environment for building optimization (operations research) based solutions to support business decisions. AIMMS distinguishes itself from other optimization software through its advanced modeling concepts, graphical user interface for developers and end-users, and the variety of deployment options.

MPL(www.maximalsoftware.com/mpl/): is an advanced modeling system that allows the model developer to formulate complicated optimization models in a clear, concise, and efficient way. MPL offers a natural algebraic notation with outstanding expressive power, readability and user friendliness. MPL is among the fastest and most scalable optimization modeling software on the market today.

3.3.2 Software for Solving Convex Problems on Desktop PCs

Standalone solvers (tedious to use without modeling language)

SeDuMi (sedumi.ie.lehigh.edu): widely used free solver, with Matlab interface.

SDPT3 (www.math.nus.edu.sg/œmattohkc/sdpt3): Matlab software, free(GPL).

CVXOPT (abel.ee.ucla.edu/cvxopt): free Python solver, allows customization of linear system solvers.

IBM CPLEX: industry standard for (MI)LPs and (MI)QCQPs (commercial).

Gurobi (www.gurobi.com): commercial (MI)SOCP solver, by creators of CPLEX, strong Python support

MOSEK (www.mosek.com): fastest commercial solver for second-order cone programs.

3.3.3 Convex Optimization Solvers for Embedded Platforms

3.3.3.1 open source solvers:

qpOASES (www.kuleuven.be/optec/software/qpOASES): active set solver, free (LGPL).

OOQP (pages.cs.wisc.edu/œswright/ooqp): object-oriented QP solver (needs LAPACK/BLAS).

ECOS (github.com/ifa-ethz/ecos): Sparse SOCP solver, 800 lines of library free C code, Python & Matlab interface.

3.3.3.2 Solvers with C-code generation:

CVXGEN (cvxgen.com): code generation for small QPs, extremely fast, code can get large.

FiOrdOs (fiordos.ethz.ch): is a Matlab toolbox for automated C-code generation for gradient methods. Supports certication, automatic preconditioning.

FORCES PRO (www.embotech.com): code generation for interior point methods. Super-fast for MPC-like optimization problems. FORCES stands for Fast Optimization for Real-time Control on Embedded Systems and is a numerical optimization code generation framework for convex multistage problems [14].

Chapter 4

Optimization Problem

This chapter presents the way of setting up an optimization problem needed during the implementation of the MPC controller. The goal is to optimize different objective functions and to implement MPC controller in order to improve quality of experience (dropping rate, sending rate, queue sizes, etc.) based on the mathematical models for the hybrid system of priority queuing.

The router has input flows from four different links and its output is split and given to two other links. These inputs from 4 different links are summed up. The input to this model are the real time data from Telecom Italia for flows with IP- Precedence 0 to 7. The real time data from Telecom Italia was recorded for 2 days with interval time of 5 minutes.

Flow with IP Precedence 5 is given as input to the High/Gold Queue, flows with IP Precedence 2, 4, 6, 7 are given as input to the medium/Premium queue and flows with IP Precedence 1, 3, 0 are given as input to the normal/Default queue. When both the input and output link bandwidths are the same, there is no bottleneck and analysis of flow behavior makes no sense. So, we simulated to create a bottleneck scenario, where one of the output link fails. Therefore all the input flows has to be sent through this output link.

4.1 Control Problem

In section 2.1.5, it was mentioned that 80% of the available bandwidth is given to the medium queue and only 20% is given to the normal queue. Now instead of using a fixed value 80% of the bandwidth, we consider a variable u which is a number between 0 and one. We want to know which u gives better QoS.

For the hybrid model that we have, we need to design a hybrid MPC controller. The implementation of this closed loop controller includes:
1- setting up an optimization problem.
2- obtaining a model of the physical plant.
3- fed back the measured/estimated states to the controller.
These steps can be understood better with the Figure 4.1.

4.1.1 Setting up the optimization problem

To set up an optimization problem:

1- Define an objective function.

2- Define inputs/states/outputs constraints.

3- Obtain a model of the system-this model to be used to predict the future evolution of the system in open-loop.

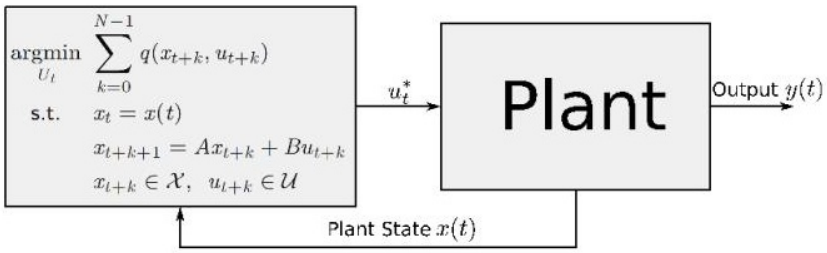

Figure 4.1: Model Predictive Control

4- Solve the problem to find the optimal input sequence for the entire future horizon N $U_t^* = \{u_t^*, u_{t+1}^*, ..., u_{t+N1}^*\}$.

4.1.1.1 Defining an objective function

To solve an optimization problem we should define first the objective function which will be minimized to obtain better QoS. Decreasing queue sizes or decreasing drop rates or combination between them could be selected as an objective function. The first objective function chosen was:

$$w_m * (q_{m1} + q_{m2} + q_{m3} + q_{m4}) + w_n * (q_{n1} + q_{n2} + q_{n3}).$$

It means that at each time instant, the controller tried to find the best input u to minimize the length of the Medium queue and the Normal queue. The priority could be controlled by tuning the weights $w_n \& w_m$. We give medium queue higher weights i.e. higher priority. Therefore we expect to observe a decrease in the number of bits in the Medium Queue, and also since the length of Medium Queue is shrinking, we expect the hybrid model to switch to other states, where there is no dropping.

It was observed during the simulation that defining a different weight for each flow would give better results. This means dealing differently with each flow, maybe one of the medium flows is more important than the others and it should be given higher priority. So the second approach to define an objective function was to put it in the following form:

$$w_{m1} * q_{m1} + w_{m2} * q_{m2} + w_{m3} * q_{m3} + w_{m4} * q_{m4} + w_{n1} * q_{n1} + w_{n2} * q_{n2} + w_{n3} * q_{n3}.$$

where w_{mi}, w_{ni} are the weights for medium and normal flows respectively. By this way we can control the priority of each flow by changing its corresponding weight.

Unfortunately our model is a sophisticated one. It contains 41 states, each state defines a specific case such as: high queue is filling, medium queue is empty and normal queue is full. So it is important to consider different objective function for each state, this will clarified in the following two examples.

Example 1: State Θ4= h1,M4,N4

This state is combination of state h1(high queue is empty), state M4(medium queue is filling) and state N4(normal queue is filling). Constraints and dynamics of h1,M4 and N4 are combined together to give new constraints and dynamics to state Θ4(h1,M4,N4) as shown in Figure 4.2.

The objective function is defined as:

$$objective = 0.8 * q_m + 0.2 * q_n$$

where q_m: medium queue size — q_n: normal queue size.

Both medium and normal queues are filling, so that the objective function is designed to minimize both queues. But medium queue has higher priority than normal queue and it should be emptied faster, so it is given larger weight. Bandwidth will be distributed between medium and normal flows in a MDRR way and optimal u will be found. No drops in this state so no need to consider decreasing the drop rates.

Example 2: state Θ25= h2,M7,N7

It is combination of states h2(high queue is empty), M7(Medium queue is full, dropping occurs), N7(normal queue is full, dropping occurs).

The objective function is defined as:

$$objective = 0 * q_m + 0 * q_n + 0.8 * d_m + 0.2 * d_n$$

where:
q_m: medium queue size.– q_n: normal queue size.
d_m: dropping rate for medium queue.– d_n: dropping rate for normal queue.

Medium and Normal queues are full so dropping occurs. The objective now is to decrease dropping rate. The queues are already full, so the size of medium and normal queues can not be decreased. If the dropping decreases, queue sizes will also decrease. Therefore zeros were put in front of q_m&q_n. Of course Weights in front of d_m are larger than d_n because Medium queue has more priority.

The parameters(weights)could be tuned if required.

It should be noted that there are states which there is no need to solve the optimization problem to find the optimal input u to split the bandwidth between Medium or Normal flows. For example, if all queues are empty and the flows receiving rate are less than the available bandwidth, all packets will be sent with the same rate as they are received. Another case, if the high queue has some packets, medium or normal queues dont have permission to send at all.

4.1.1.2 Defining inputs/states/outputs constraints

In this project we need only to put constraint on the inputs:

$$0 \leq u \leq 1.$$

4.1.1.3 Obtaining a model of the system

We define a state matrix which holds the current size of the high/medium/normal queues. It is defined as:

$$X(k) = [q_h(k) \quad q_{m1}(k) \quad q_{m2}(k) \quad q_{m3}(k) \quad q_{m4}(k) \quad q_{n1}(k) \quad q_{n2}(k) \quad q_{n3}(k)]^T$$

where:

$q_h(k)$ denotes the number of bits in the *high* queue which correspond to flow with IP Precedence 5.

$q_{m1}(k)$,$q_{m2}(k)$,$q_{m3}(k)$,$q_{m3}(k)$ denote the number of bits in the *medium* queue corresponding to flow with IP Precedence 2,4,6,7 respectively.

$q_{n1}(k)$,$q_{n2}(k)$,$q_{n3}(k)$ denote the number of bits in the *normal* queue which correspond to flow with IP Precedence 1,3,0 respectively.

h1,M4,N4= Θ_4

$$q_h \leq 0 \quad \& \quad s_n + s_m + s_h \leq B \quad \& \quad 0 < q_m \leq q_{m,max} \quad \& \quad B - r_h > 0$$
$$\& \quad 0 < q_n \leq q_{n,max}$$
$$r_h = s_h \quad ; \quad \dot{q}_h = s_h - r_h = 0$$
$$r_{m_f} = \frac{q_{m_f}}{q_m}(s_m + u(B - s_h - s_n - s_m)) \quad ; \quad \dot{q}_{m_f} = s_{m_f} - r_{m_f}$$
$$r_{n_f} = \frac{q_{n_f}}{q_n}(s_n + (1-u)(B - s_h - s_n - s_m)) \quad ; \quad \dot{q}_{n_f} = s_{n_f} - r_{n_f}$$

$$\begin{pmatrix} q_h(k+1) \\ q_{m_1}(k+1) \\ q_{m_2}(k+1) \\ q_{m_3}(k+1) \\ q_{m_4}(k+1) \\ q_{n_1}(k+1) \\ q_{n_2}(k+1) \\ q_{n_3}(k+1) \end{pmatrix} = \begin{pmatrix} q_h(k) \\ q_{m_1}(k) + H[s_{m_1} - \frac{q_{m_1}}{q_m}(s_m + u(k)(B - s_h - s_n - s_m))] \\ q_{m_2}(k) + H[s_{m_2} - \frac{q_{m_2}}{q_m}(s_m + u(k)(B - s_h - s_n - s_m))] \\ q_{m_3}(k) + H[s_{m_3} - \frac{q_{m_3}}{q_m}(s_m + u(k)(B - s_h - s_n - s_m))] \\ q_{m_4}(k) + H[s_{m_4} - \frac{q_{m_4}}{q_m}(s_m + u(k)(B - s_h - s_n - s_m))] \\ q_{n_1}(k) + H[s_{n_1} - \frac{q_{n_1}}{q_n}(s_n + (1-u(k))(B - s_h - s_n - s_m))] \\ q_{n_2}(k) + H[s_{n_2} - \frac{q_{n_2}}{q_n}(s_n + (1-u(k))(B - s_h - s_n - s_m))] \\ q_{n_3}(k) + H[s_{n_3} - \frac{q_{n_3}}{q_n}(s_n + (1-u(k))(B - s_h - s_n - s_m))] \end{pmatrix}$$

$$X(k+1) = X(k) + \begin{pmatrix} 0 \\ -H\frac{X_2}{X_2+X_3+X_4+X_5}s_m(k) \\ -H\frac{X_3}{X_2+X_3+X_4+X_5}s_m(k) \\ -H\frac{X_4}{X_2+X_3+X_4+X_5}s_m(k) \\ -H\frac{X_5}{X_2+X_3+X_4+X_5}s_m(k) \\ -H\frac{X_6}{X_6+X_7+X_8}(B - s_h(k) - s_m(k)) \\ -H\frac{X_7}{X_6+X_7+X_8}(B - s_h(k) - s_m(k)) \\ -H\frac{X_8}{X_6+X_7+X_8}(B - s_h(k) - s_m(k)) \end{pmatrix} + \begin{pmatrix} 0 \\ Hs_{m_1}(k) \\ Hs_{m_2}(k) \\ Hs_{m_3}(k) \\ Hs_{m_4}(k) \\ Hs_{n_1}(k) \\ Hs_{n_2}(k) \\ Hs_{n_3}(k) \end{pmatrix}$$
$$+ \begin{pmatrix} 0 \\ -H\frac{X_2}{X_2+X_3+X_4+X_5}(B - s_h(k) - s_m(k) - s_n(k)) \\ -H\frac{X_3}{X_2+X_3+X_4+X_5}(B - s_h(k) - s_m(k) - s_n(k)) \\ -H\frac{X_4}{X_2+X_3+X_4+X_5}(B - s_h(k) - s_m(k) - s_n(k)) \\ -H\frac{X_5}{X_2+X_3+X_4+X_5}(B - s_h(k) - s_m(k) - s_n(k)) \\ H\frac{X_6}{X_6+X_7+X_8}(B - s_h(k) - s_m(k) - s_n(k)) \\ H\frac{X_7}{X_6+X_7+X_8}(B - s_h(k) - s_m(k) - s_n(k)) \\ H\frac{X_8}{X_6+X_7+X_8}(B - s_h(k) - s_m(k) - s_n(k)) \end{pmatrix} u(k)$$

Figure 4.2: State $\boldsymbol{\Theta}4$: high queue is empty, medium queue is filling, normal queue is filling

The continuous dynamics are on the form:

$$\dot{X} = A * X + B * U$$

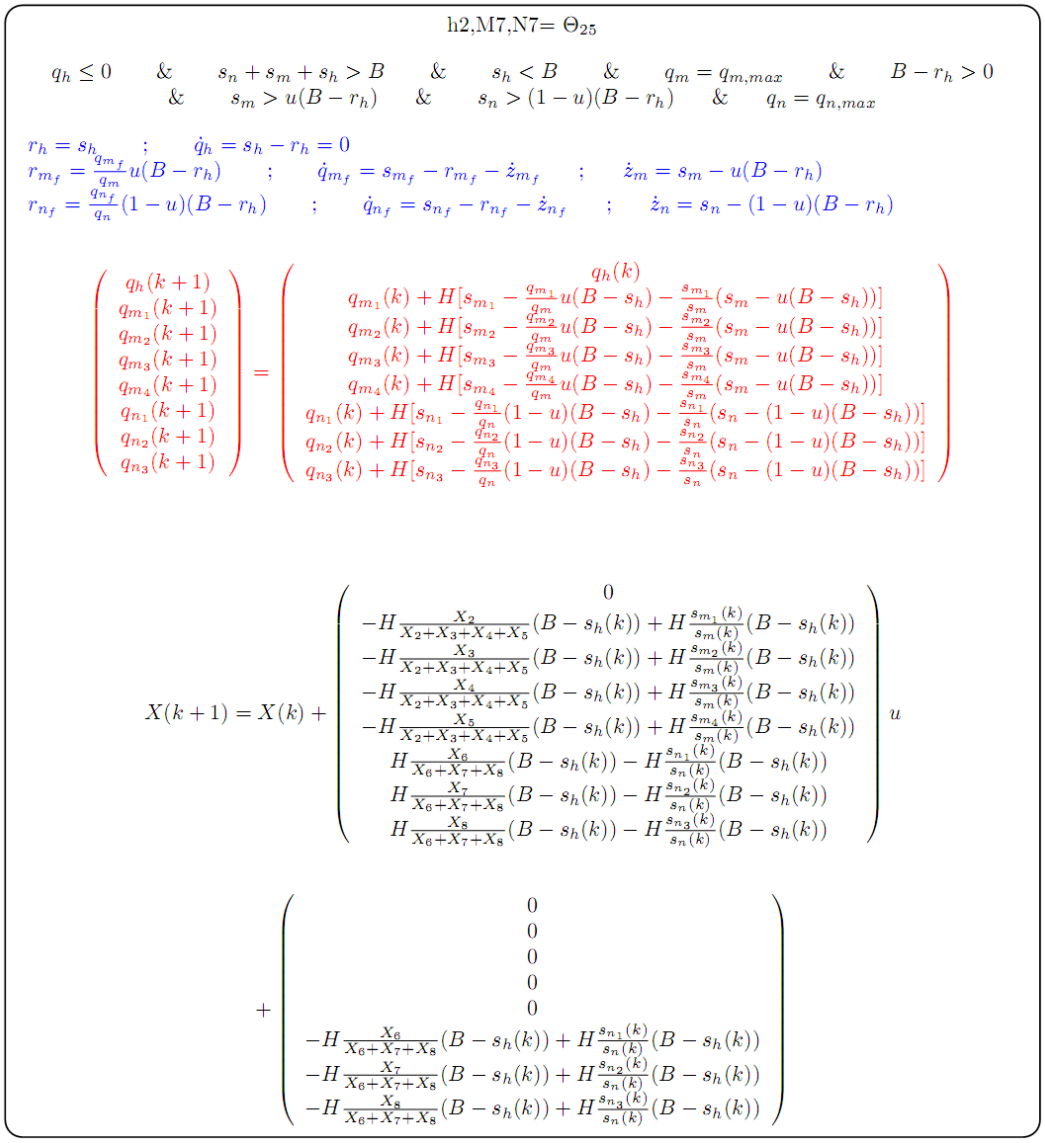

Figure 4.3: State Θ25: high queue is empty, medium queue is full, normal queue is full

After discretization:

$$X(k+1) = X(k) + H * (A * X(k) + B * U(k))$$

where H is the discretization step.

This hybrid model will used twice (see Figure 4.1), once for calculating(predicting) the future evolution of the system and once as the mathematical model for the physical plant.

4.2 Solver

YALMIP is a modeling language for advanced modeling and solution of convex and non-convex optimization problems. Solving a Hybrid MPC problem requires both modeling and optimization tasks. YALMIP acts as a compiler for MLD model generation (which is another form of Hybrid Models). It is a MATLAB toolbox for rapid prototyping of optimization problems. The main motivation for using YALMIP is rapid algorithm development. The language is consistent with standard MATLAB syntax, thus making it extremely simple to use for anyone familiar with MATLAB. Another benefit of YALMIP is that it implements a large amount of modeling tricks, allowing the user to concentrate on the high-level model, while YALMIP takes care of the low-level modeling to obtain as

efficient and numerically sound models as possible. The writer used these solvers: IPOPT and FMINCON to solve the nonlinear optimization problem in the Hybrid MPC problem.

Chapter 5

SIMULATIONS

Our purpose in this thesis, was to first simulate the MDRR scheme and obtain the flow behaviors of different IP precedence. Then, we wanted to design a closed loop MPC controller in order to see what improvements we can achieve, based on the chosen objective function, and as a final step, to compare the two cases with each other.

This Hybrid model is developed for a Telecom Italia router in which the QoS is implemented. The router has input flows from four different links and its output is split and given to two other links. These inputs from 4 different links are summed up.

The input to this model are the real time data from Telecom Italia for flows with IP-Precedence 0 to 7. The real time data from Telecom Italia was recorded for 2 days with interval time of 5 minutes. Flow with IP Precedence 5 is given as input to the High/Gold Queue, flows with IP-Precedence 2, 4, 6, 7 are given as input to the medium/Premium queue and flows with IP-Precedence 1, 3, 0 are given as input to the normal/Default queue.

When both the input and output link bandwidths are the same, there is no bottleneck and analysis of flow behavior makes no sense. So, we simulated to create a bottleneck scenario, where one of the output link fails. Therefore all the input flows has to be sent through this output link.

5.1 Preliminary analysis of data inputs:

In this section we want to give an overview of the data we are dealing with. It is assumed the output link bandwidth is 0.25×10^8. We want to compare the incoming rates for high, medium and normal flows with the available link bandwidth.

We first start by plotting and comparing the incoming rate of high flow and the available bandwidth. Packets with high priority are given absolute preferential treatment over medium or normal packets. The remaining bandwidth after sending high queue packets is distributed among medium and normal queues. Figure 5.1 shows the incoming rate for high priority packets and the available bandwidth. It is clear from this figure that drops will never occur for the high packets because the incoming rate is always less than the available bandwidth.

Figure 5.2 shows the remaining bandwidth after sending high priority packets. This should be distributed among medium and normal flows in an MDRR way to give us better QoS. In figure 5.3, we compare between receiving rate of medium and normal flows and the remaining bandwidth after sending high priority packets. We expect from this figure that dropping will occur especially for medium flows in the time interval $[8 \times 10^4 \ , \ 10 \times 10^4]$. Droppings will occur in this period because the medium-flow incoming rate is very high compared to the available link bandwidth. It is clear from this figure also that normal-flow

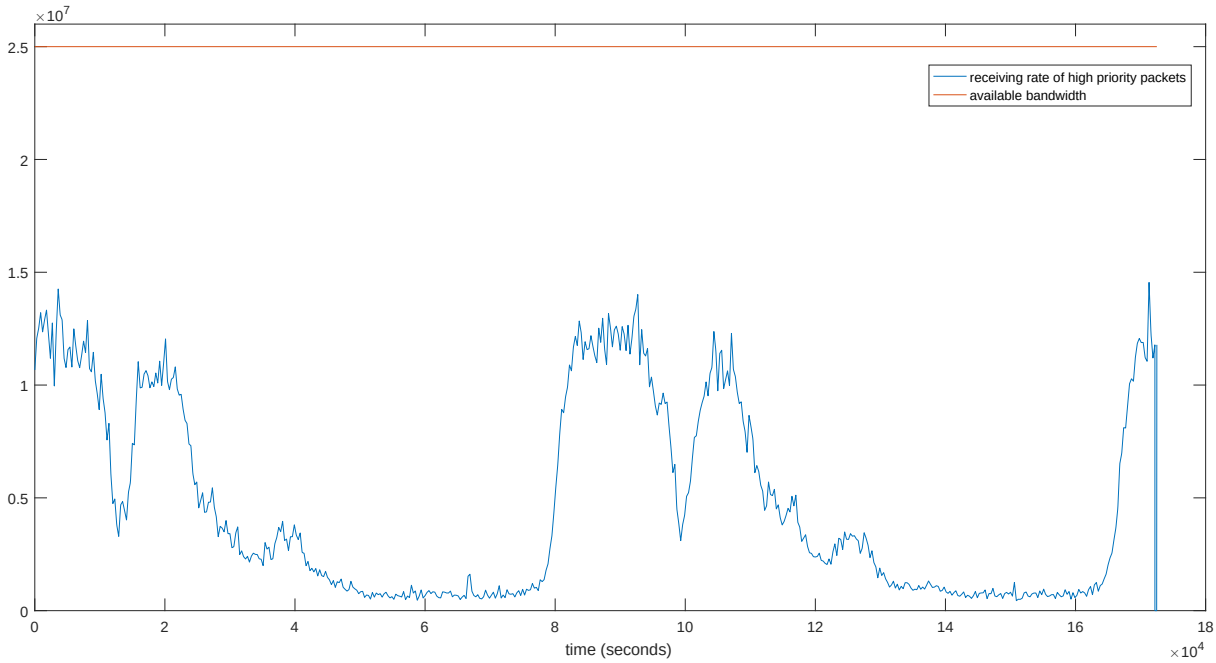

Figure 5.1: High priority packets vs available bandwidth

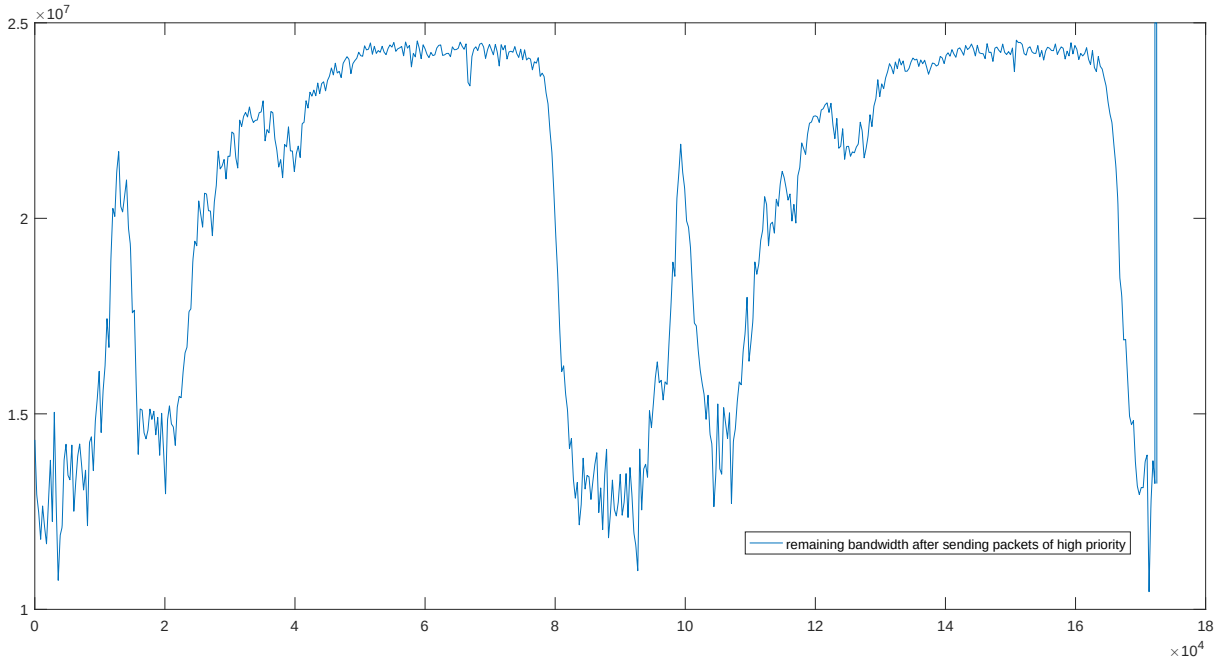

Figure 5.2: Remaining bandwidth after sending high priority packets

incoming rate is too small compared with medium-flow incoming rate and the available link bandwidth. This may give indication that normal-flow packets will be sent without suffering too much from droppings.

From Figure 5.4, it can be seen that medium-flows have different incoming rates. Medium flow (M3) with IP-Precedence 6 has the highest incoming rate in some periods so it may suffer from droppings. Medium flow (M1) with IP-Precedence 2 has a lower incoming rate. So each flow should be treated differently depending on its importance compared to the others. In the optimization problem we can give higher weights(higher

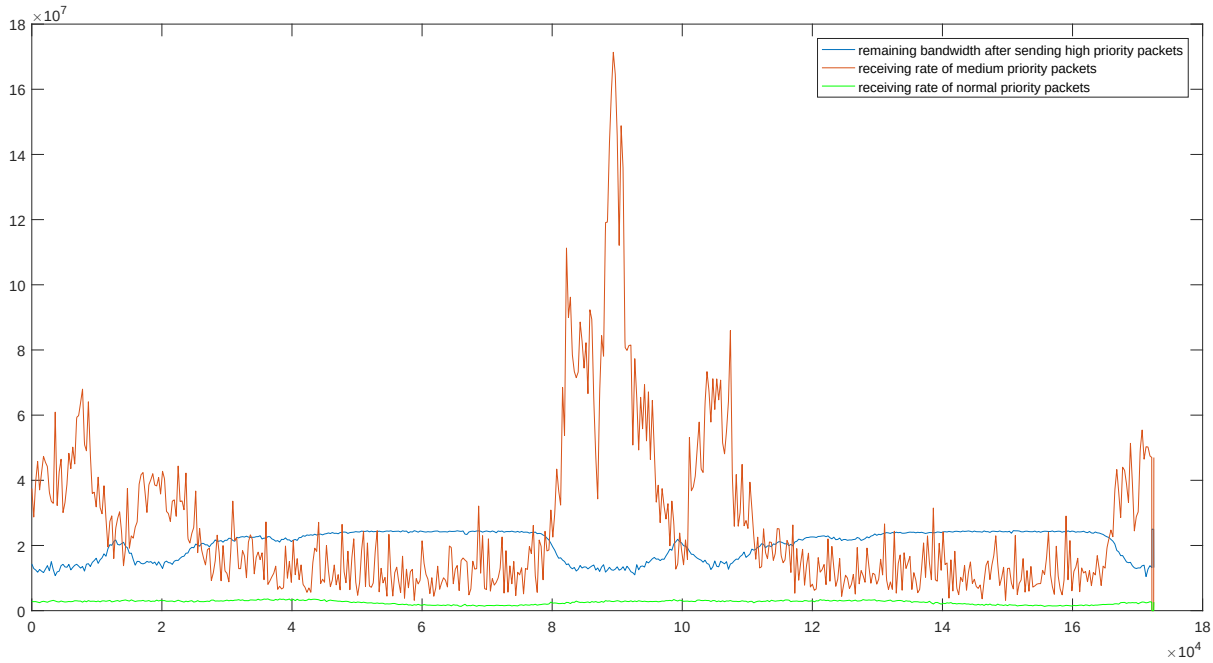

Figure 5.3: Remaining bandwidth after sending high priority packets vs receiving rates of medium and normal priority packets

priority) to some flows to avoid their droppings.

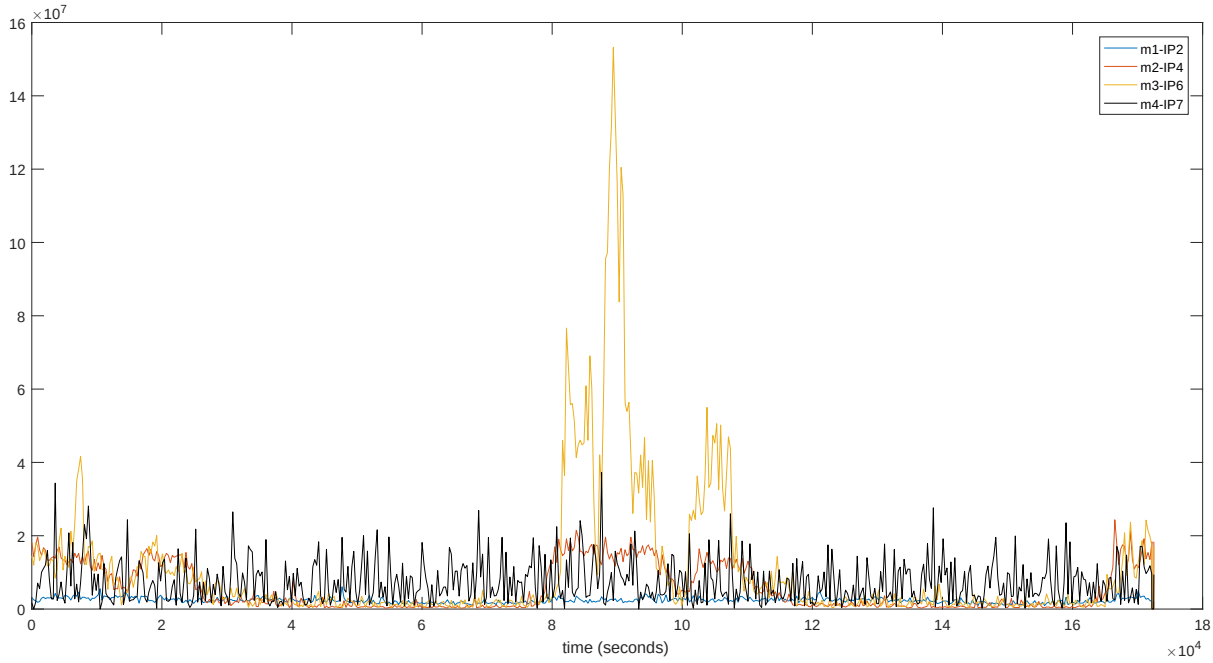

Figure 5.4: Different medium-flows incoming rates

5.2 Uncontrolled case(without using MPC controller)

In this section we show the results for the uncontrolled case(without using MPC controller). The simulation has been done with a MATLAB function. The amount of u is

fixed to 0.8 as shown in Figure 5.5. The congestion management mechanism implemented in Telecom Italia is Modified Deficit Round Robin (MDRR) which gives 80% of the link bandwidth to medium-flow packets and 20% to normal-flow packets.

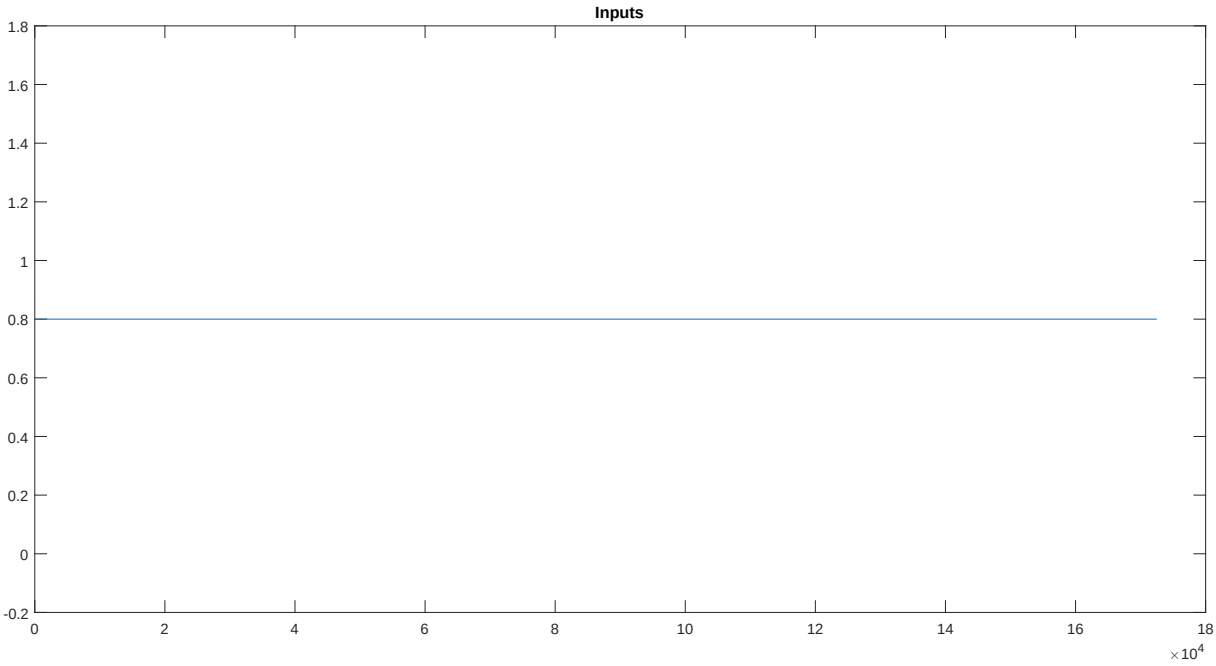

Figure 5.5: Applied inputs

The different flow behaviors including incoming, sending and dropping rates of different flows have been plotted. The incoming rates which were collected from Telecom Italia had a sampling time of five minutes. So, the data was interpolated linearly in order to predict the incoming rate between every two consecutive samples. As mentioned before, the dynamics had to be discretized. Since discretization always introduces some error in the model, the time step was chosen small enough H = 1s. And finally the incoming rate was calculated at each second based on the interpolated data.

In the sequel, we bring and discuss the flow behavior of different flows:

High Flow: We can see from Figure 5.6 that the incoming rate of the high flow is exactly the same as its sending rate, no dropping occurs.

Medium Flows: Figures 5.7 — 5.10 show the flow behavior for Medium flows.Dropping occurs for medium flows, so the 80% of the allocated bandwidth to the Medium flows should be increased to prevent droppings. This will be discussed in the next section when an MPC controller is implemented.

Normal Flows: Figures 5.11 — 5.13 show that the different normal flows do not suffer from droppings at all and the their incoming rates are almost the same as their sending rates. This means that the 20% of the available bandwidth allocated to normal queue is more than enough and it should be decreased to save Medium queue packets from droppings which are more important than Normal queue packets. But this does not mean to ignore the Normal flows at all, they are also important and need to be given access to the output link. This approach is considered also in the next section.

Number of Bits Figure 5.14 shows the number of bits in high, medium and normal queues. High queue is always empty. If it has some packets the whole bandwidth will be dedicated to it to prevent its dropping. Because of the high incoming rate

of the Medium flows, the length of medium queue reaches its full capacity 8×10^4. Normal queue never reaches its full capacity8×10^4 because of the low incoming rate of the normal flows.

MDRR States Figure 5.15 shows the switching behavior between the different MDRR states.

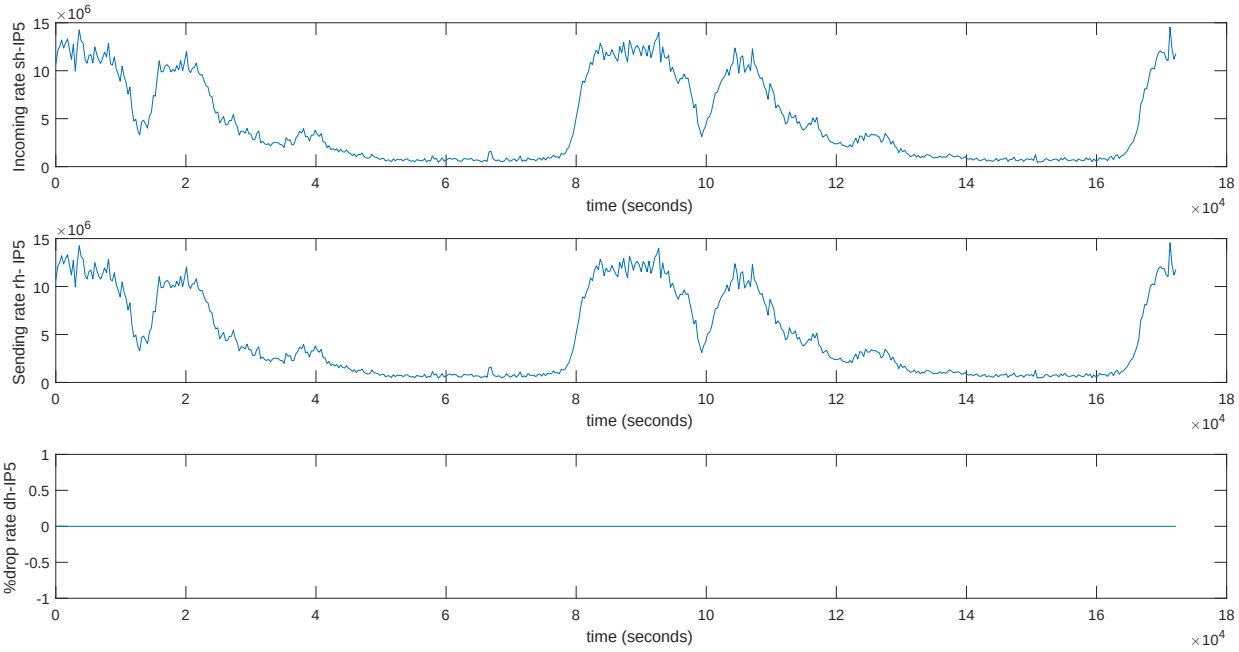

Figure 5.6: Receiving rate & Sending rate & drop rate% for high-flow(h) or IP-Precedence 5

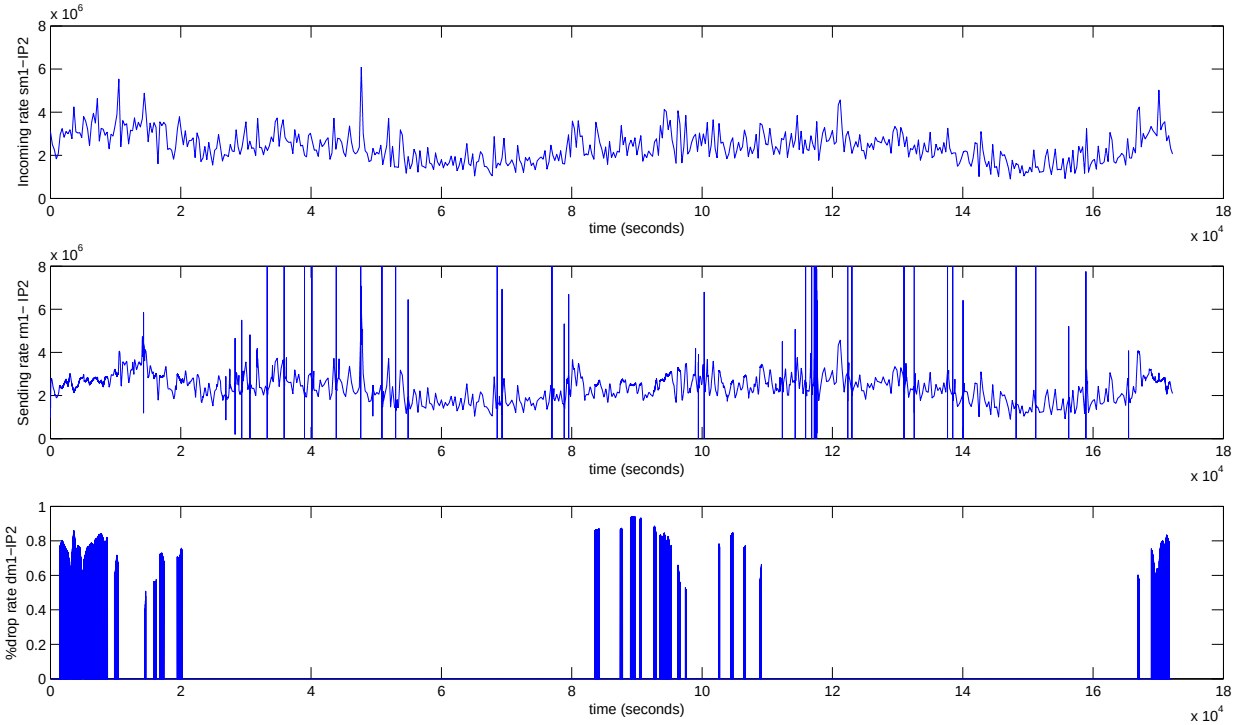

Figure 5.7: Receiving rate & Sending rate & drop rate% for medium-flow(M1) or IP-Precedence 2

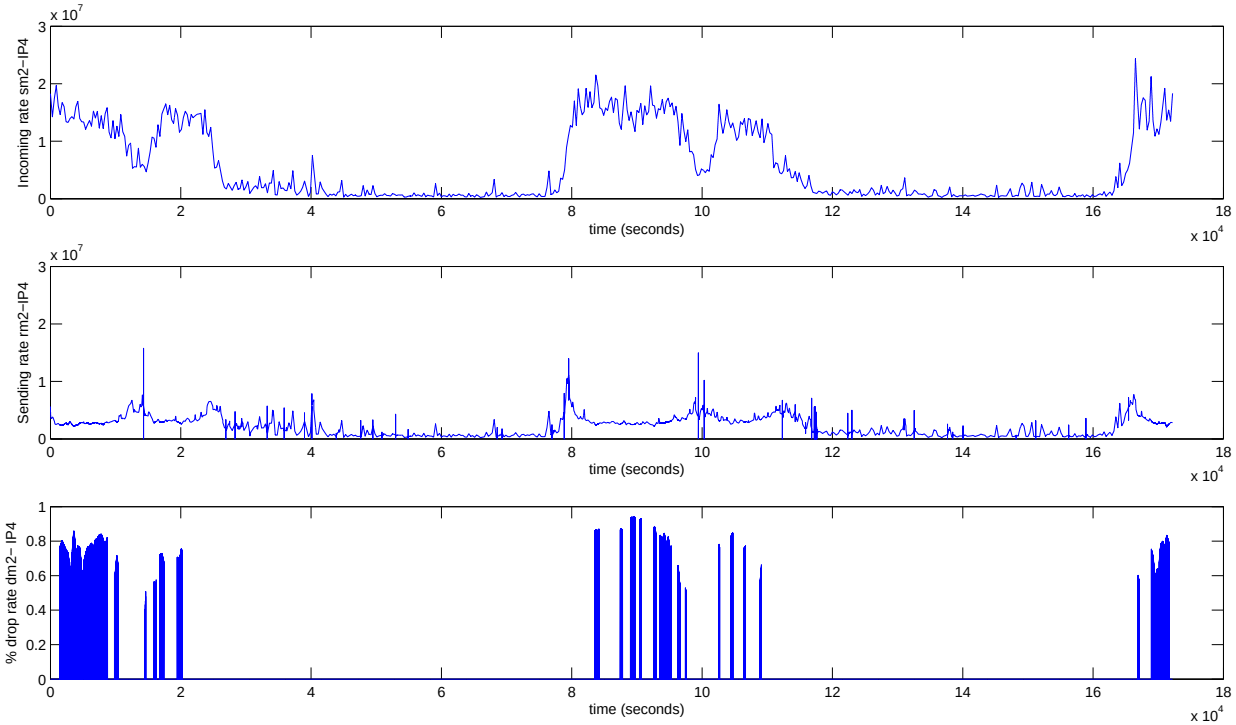

Figure 5.8: Receiving rate & Sending rate & drop rate% for medium-flow(M2) or IP-Precedence 4

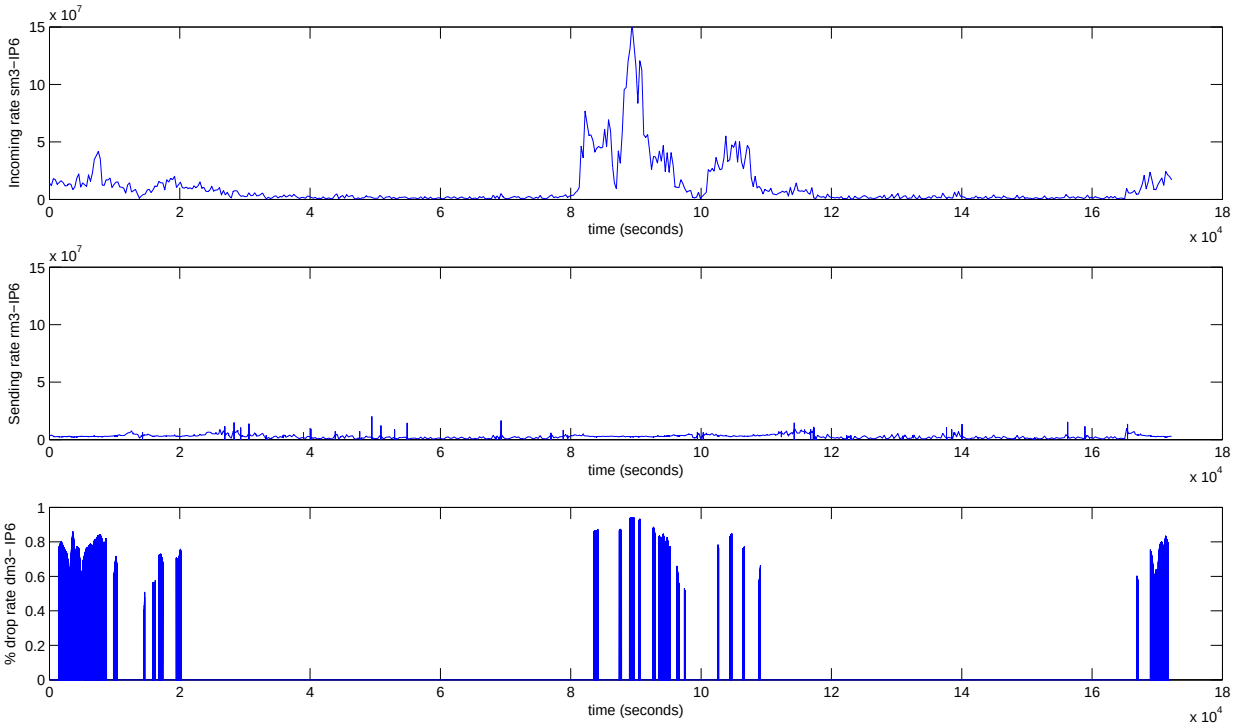

Figure 5.9: Receiving rate & Sending rate & drop rate% for medium-flow(M3) or IP-Precedence 6

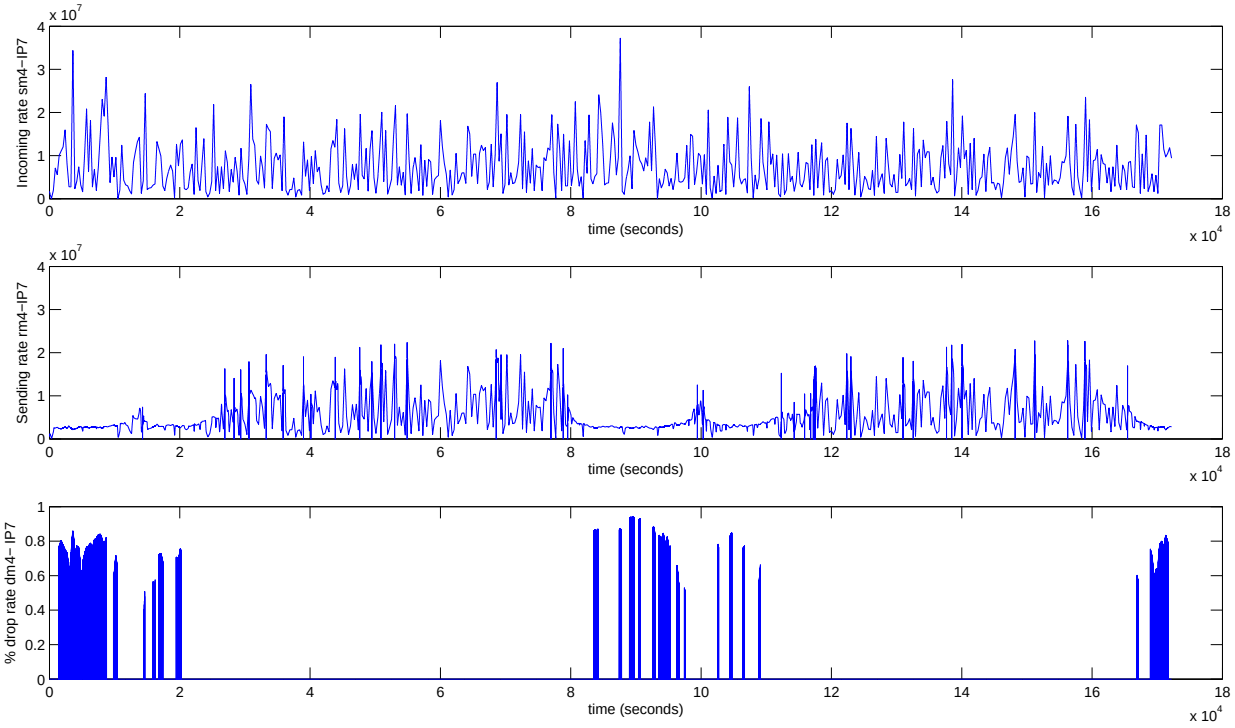

Figure 5.10: Receiving rate & Sending rate & drop rate% for medium-flow(M4) or IP-Precedence 7

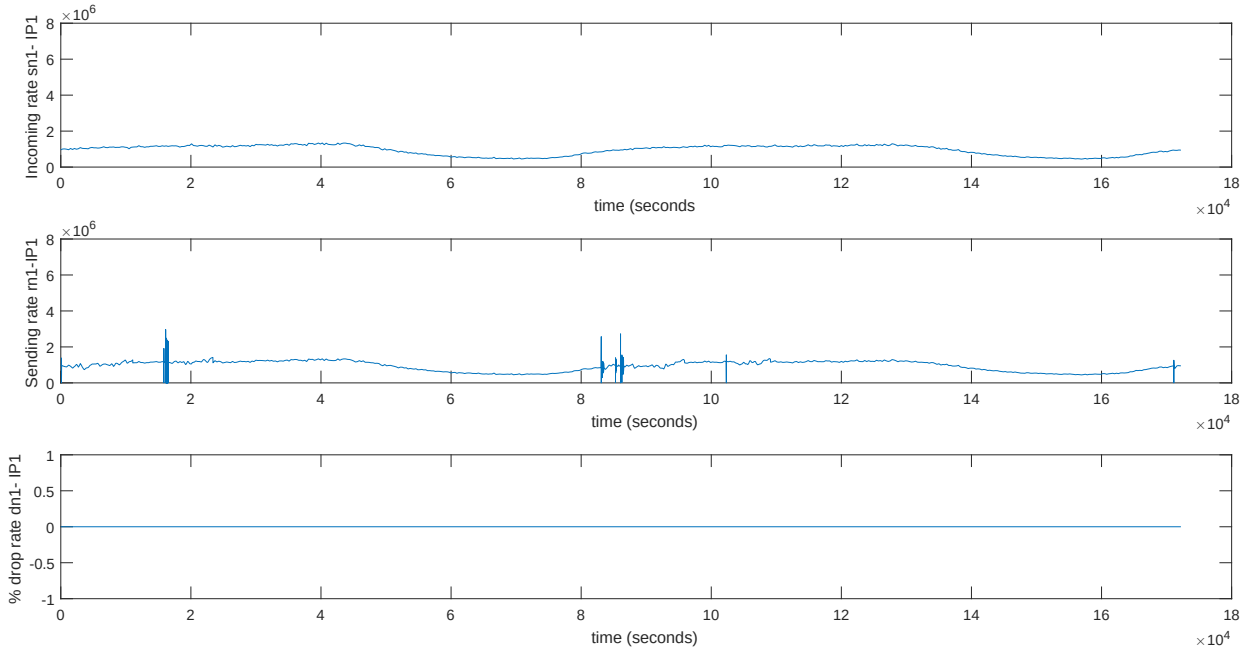

Figure 5.11: Receiving rate & Sending rate & drop rate% for normal-flow(N1) or IP-Precedence 1

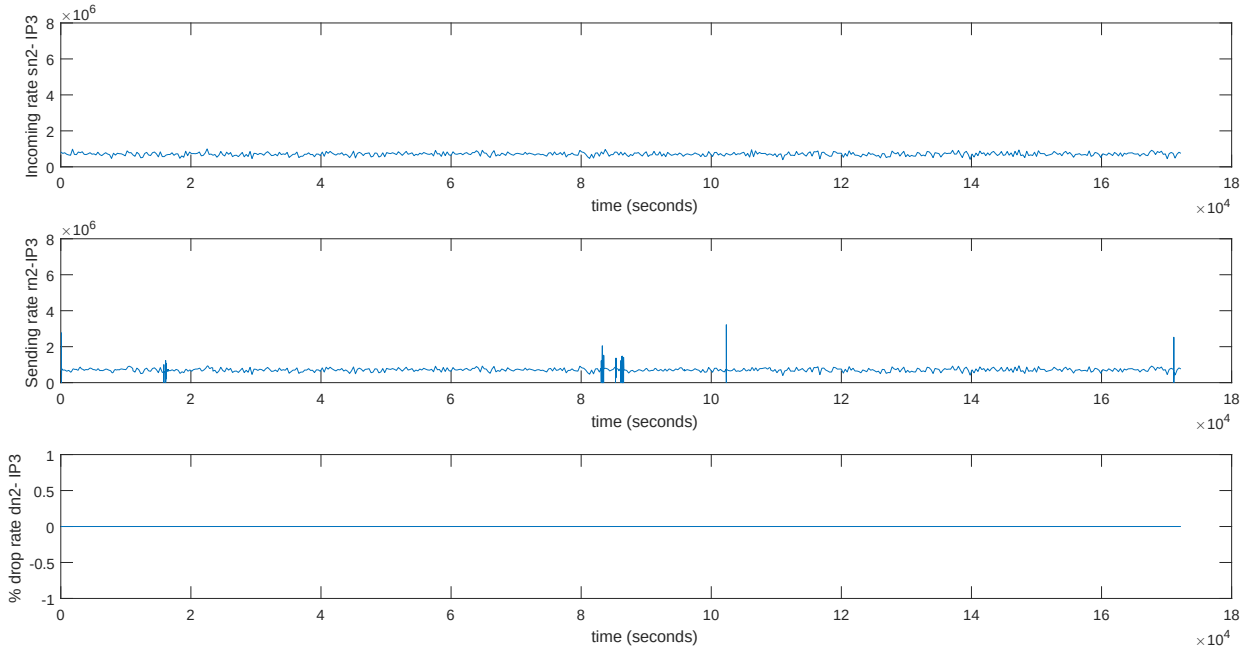

Figure 5.12: Receiving rate & Sending rate & drop rate% for normal-flow(N2) or IP-Precedence 3

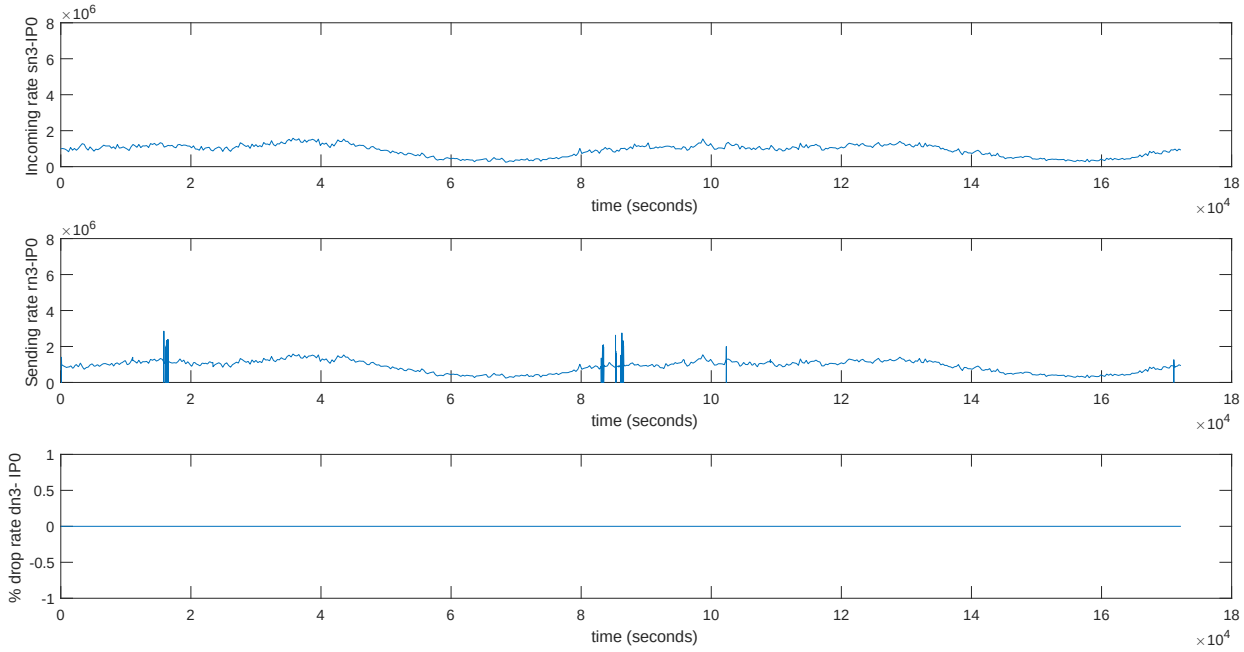

Figure 5.13: Receiving rate & Sending rate & drop rate% for normal-flow(N3) or IP-Precedence 0

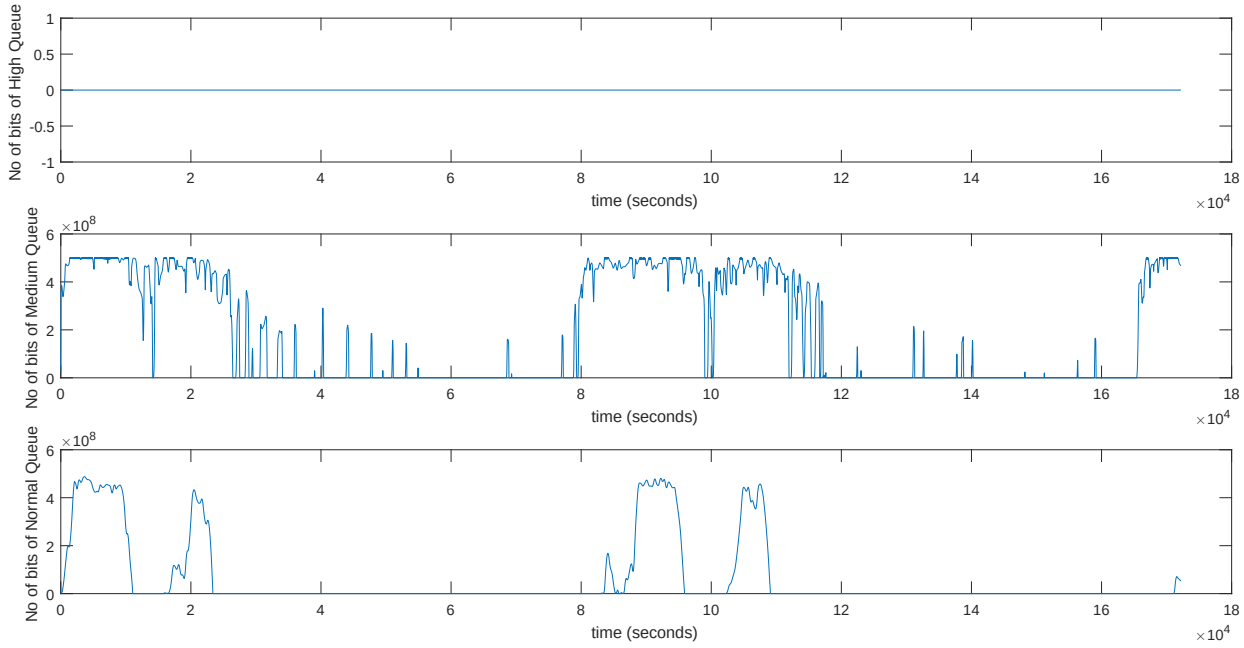

Figure 5.14: Number of bits in each queue

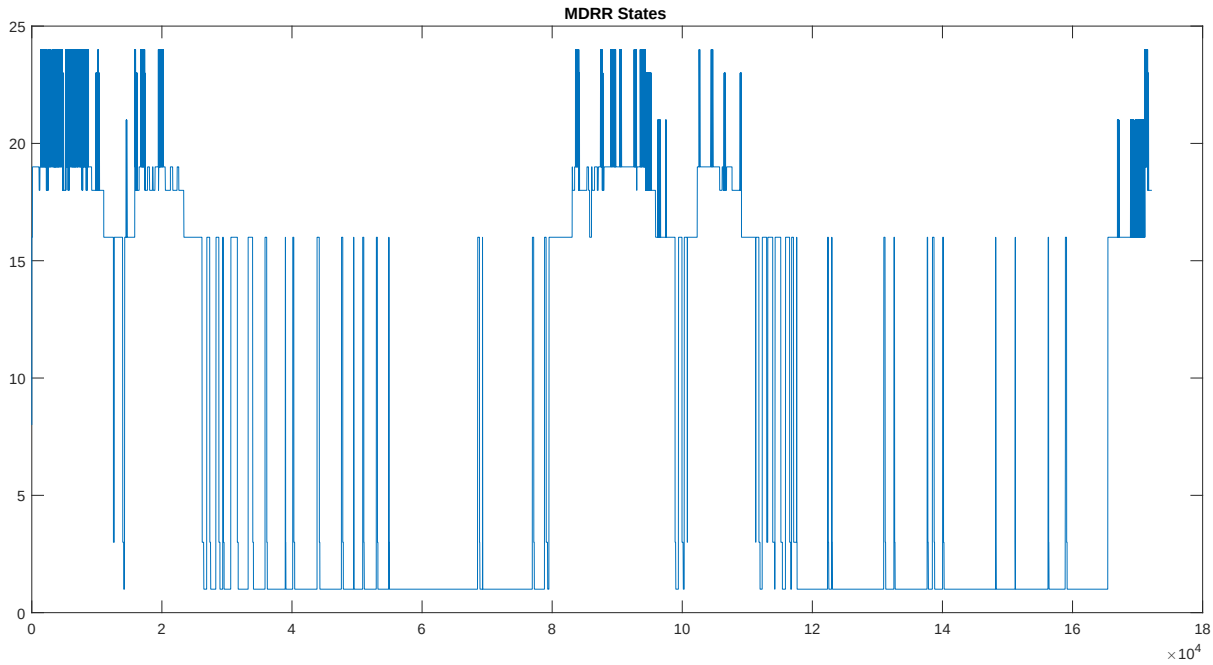

Figure 5.15: Switching between different states

Since the dynamics were discretized, the length of queues can become negative and also exceed the maximum length of the buffers both of which were impossible in practice. So we had to introduce some nonlinear function like "abs" to keep the dynamics always positive and force the maximum length not to exceed its maximum length. As a result, we see some nonlinearities in the flow behaviors. However, by choosing a small time-step (for the discretization), we may minimize the introduced error. Here we chose H = 1s.

5.3 Controlled case(using MPC controller)

In this section we discuss the results when the MPC controller is used. Instead of using a fixed value of the input u as discussed in the previous section, the input will be dynamically calculated and changed by solving an on-line constrained optimization problem.

5.3.1 Hybrid MPC controller

Since the problem had many states, and the dynamics were non-linear, the available solvers could not solve the problem. To the best of our knowledge, most of Hybrid MPC problems, do not have as many states with so many invariant constraints (the constraints which specify the states). And that is why they can normally solve these problems without difficulty. We decided to implement switching MPC, which imposes less computational burden, and the results are more trustable.

5.3.2 Switching MPC Controller

Hybrid non-linear MPC can be solved using Mixed Integer Non-linear Programming which is very hard and in particular it is well known to be NP-Hard. Motivated by this fact and the fact that our model was very sophisticated with non-linear dynamics, we proposed a relaxation of the above problem by considering a Switching MPC strategy. In particular, it is based on the following principle: if at instant k we are in a discrete state of the hybrid system, we solve a nonlinear MPC based on the nonlinear dynamics of such a discrete state, namely we suppose that during the prediction horizon a switching between the discrete states of the hybrid model cannot occur. Hence, the switching MPC is a

suboptimal controller. In the following section we provide simulations to compare the MPC controlled MDRR scheme and MDRR scheme with no controller.

5.3.3 Behavior of different flows

The horizon for the prediction part of MPC controlling was chosen to be N = 2, and the simulation was run for a total time of 172200 seconds.

Figure 5.16 shows the output of the optimization problem which is the optimal input u. This value u is calculated to minimize the objective function to obtain better QoS. From Figure 5.17 we can see that u is switching between 0.5 and 0.8. In different time intervals Figure 5.16 has some regions which are in dark blue color. This means that u is switching between 0 and 1 in these regions (see Figure 5.18). This proves the improvements after applying the MPC controller over the uncontrolled case. Previously it was said that fixing 80% of the bandwidth to the medium queue maybe not the optimal solution to have better QoS and it should be changed if necessary. Now we see that the MPC controller is giving full access for the medium queue to use the whole available link bandwidth and zero access to the normal queue. Sometimes the Normal queue is given full access but length of intervals where the full access is given to the medium queue is much larger than for the normal queue. This means that we want to save medium queue packets from droppings but at the same time allow the normal queue to send. This switching between 1 and zero occurs in the intervals when the incoming rates of the medium flows are very high and needs the full bandwidth to save packets from dropping.
We compare between the controlled and uncontrolled case in the following section.

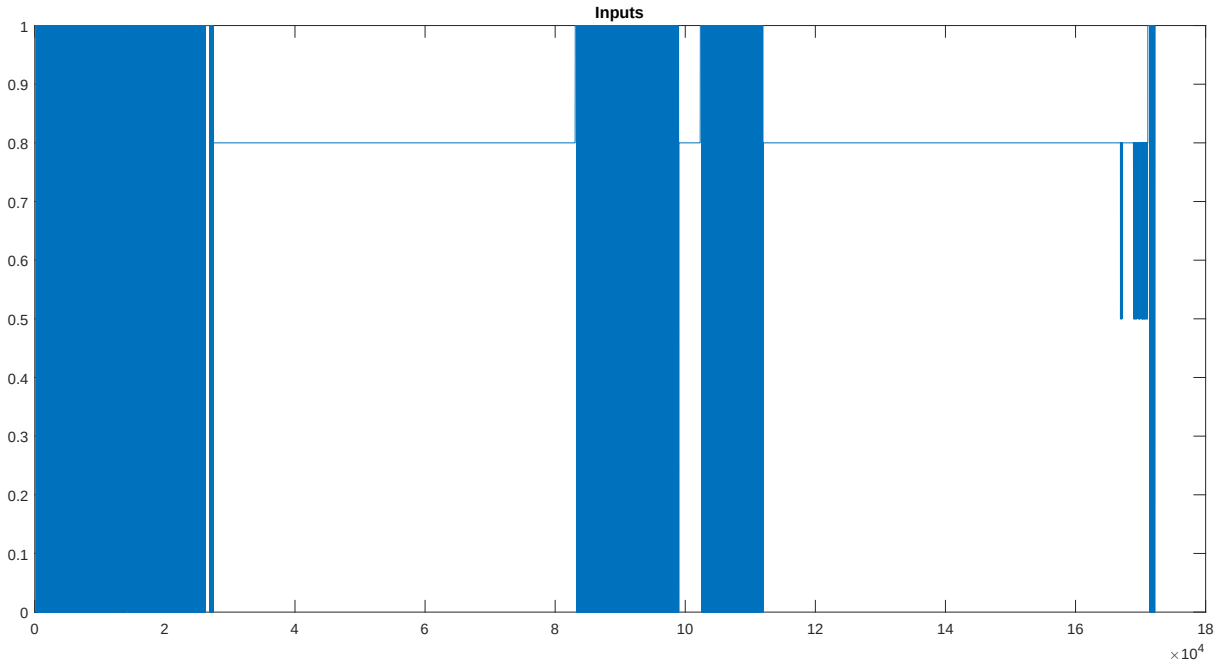

Figure 5.16: Inputs

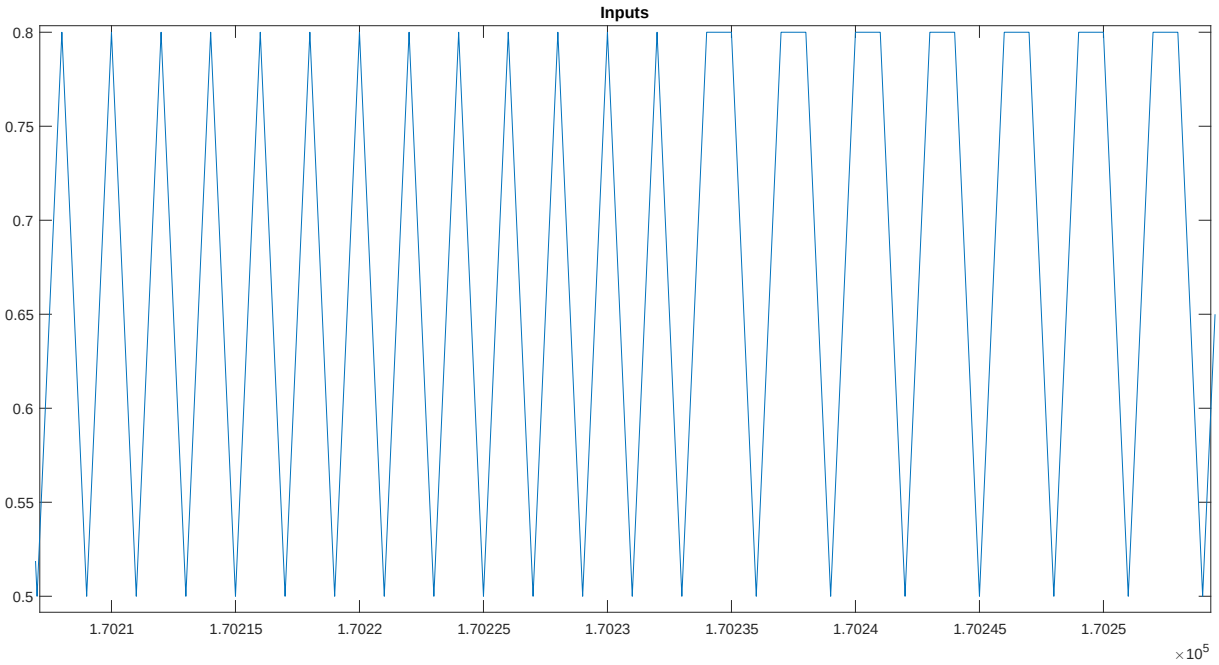

Figure 5.17: Inputs(zooming in (1/2))

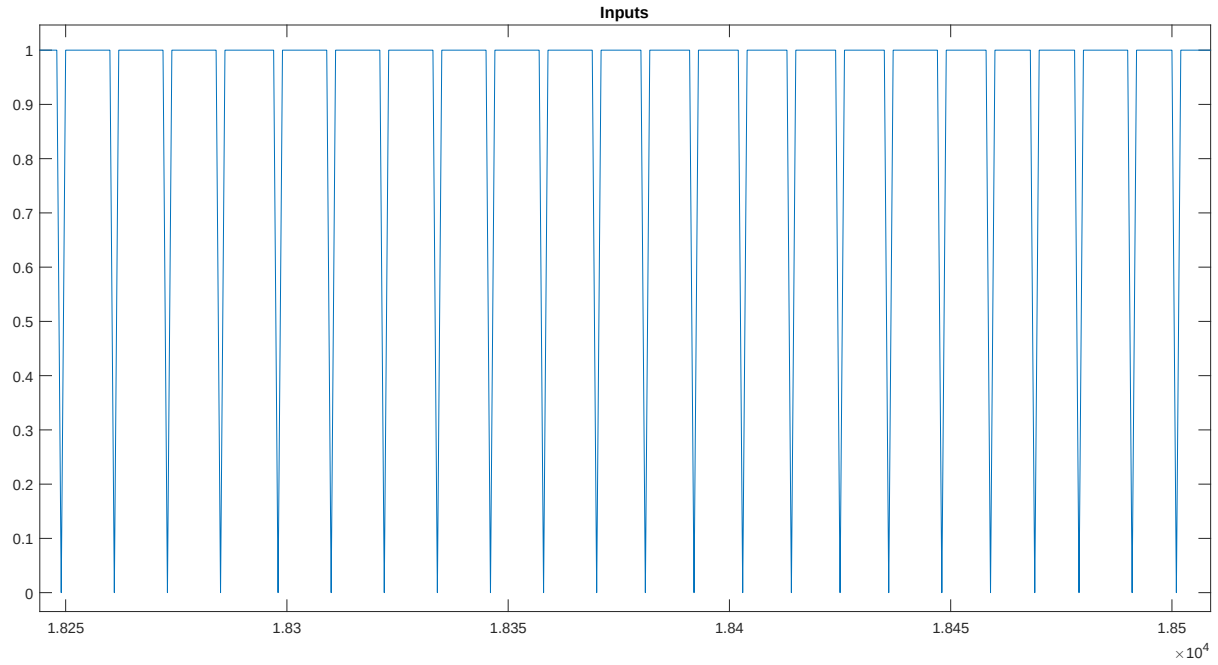

Figure 5.18: Inputs(zooming in (2/2))

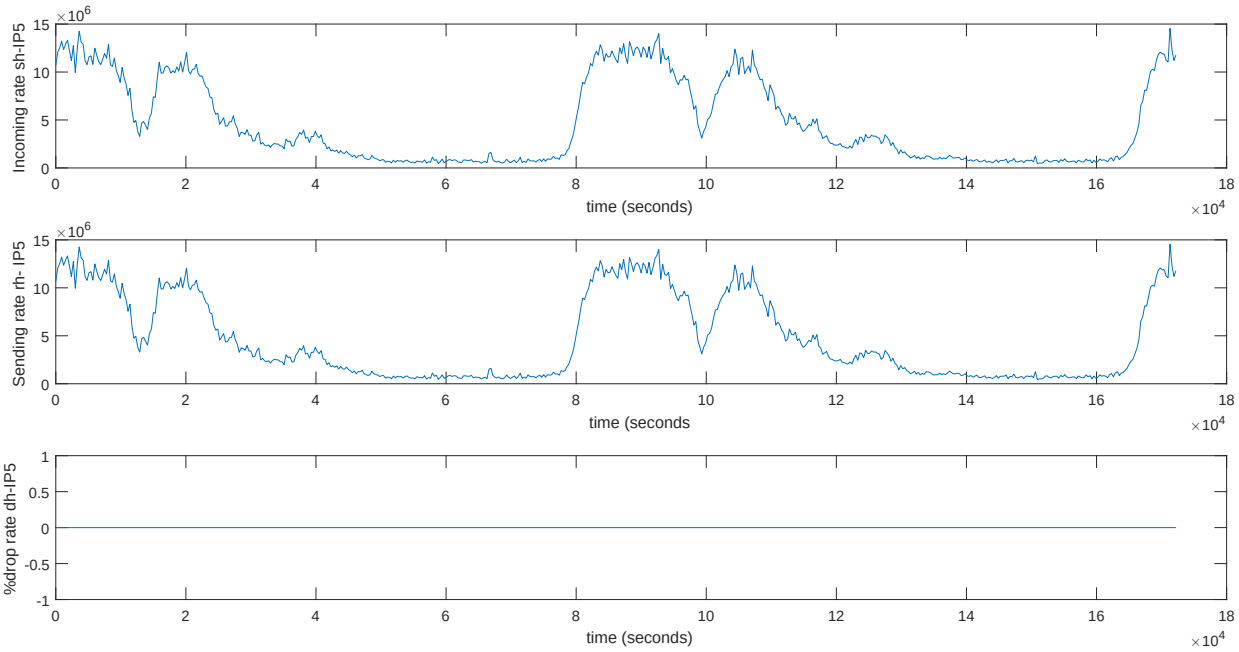

Figure 5.19: Receiving rate & Sending rate & drop rate% for high-flow(h) or IP-Precedence 5

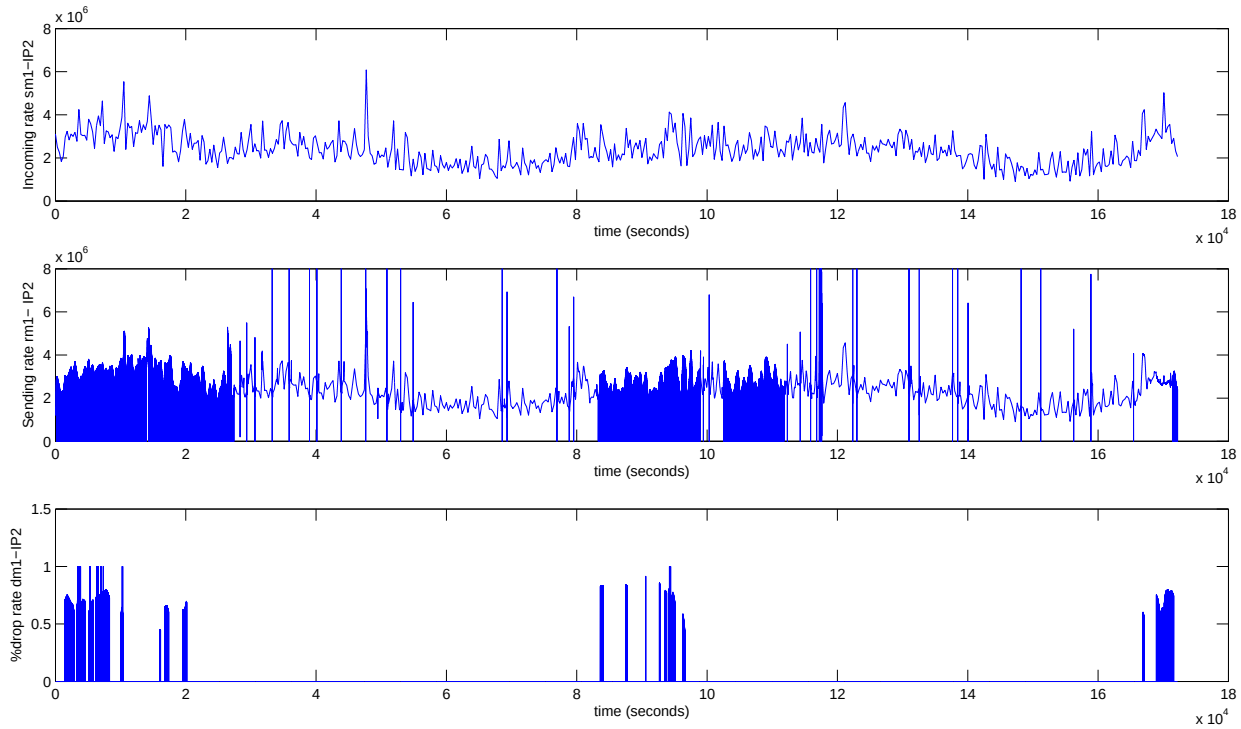

Figure 5.20: Receiving rate & Sending rate & drop rate% for medium-flow(M1) or IP-Precedence 2

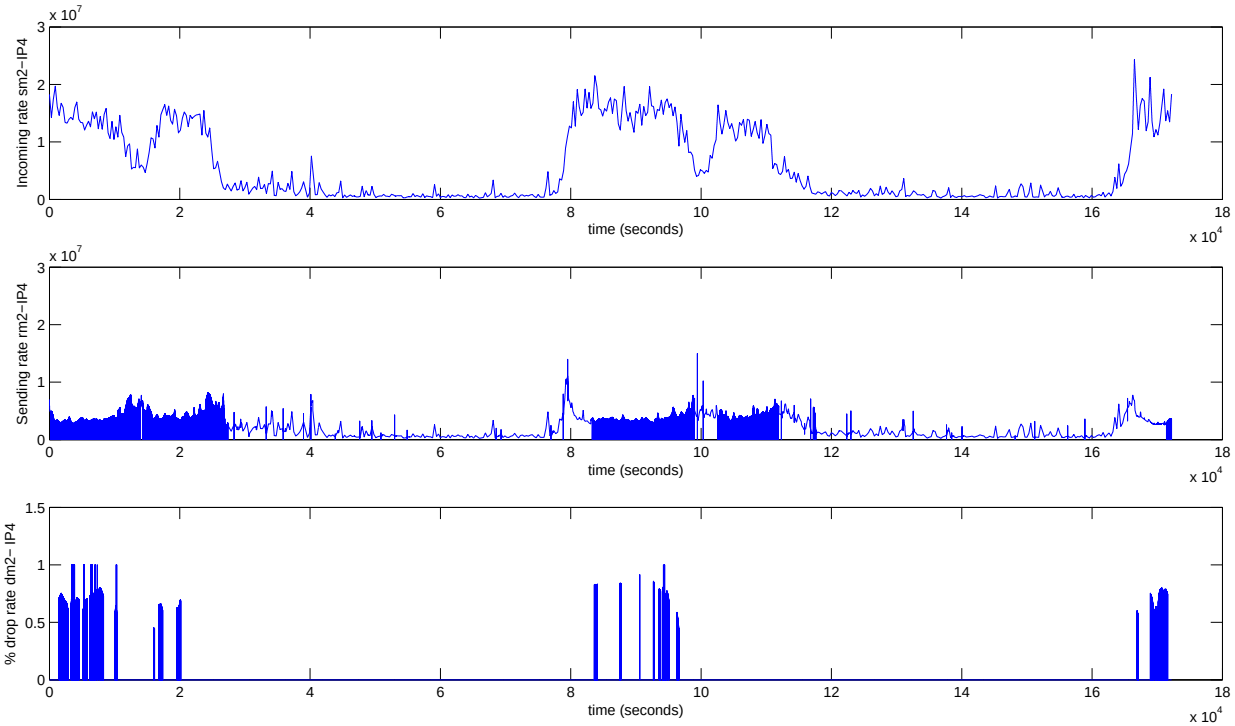

Figure 5.21: Receiving rate & Sending rate & drop rate% for medium-flow(M2) or IP-Precedence 4

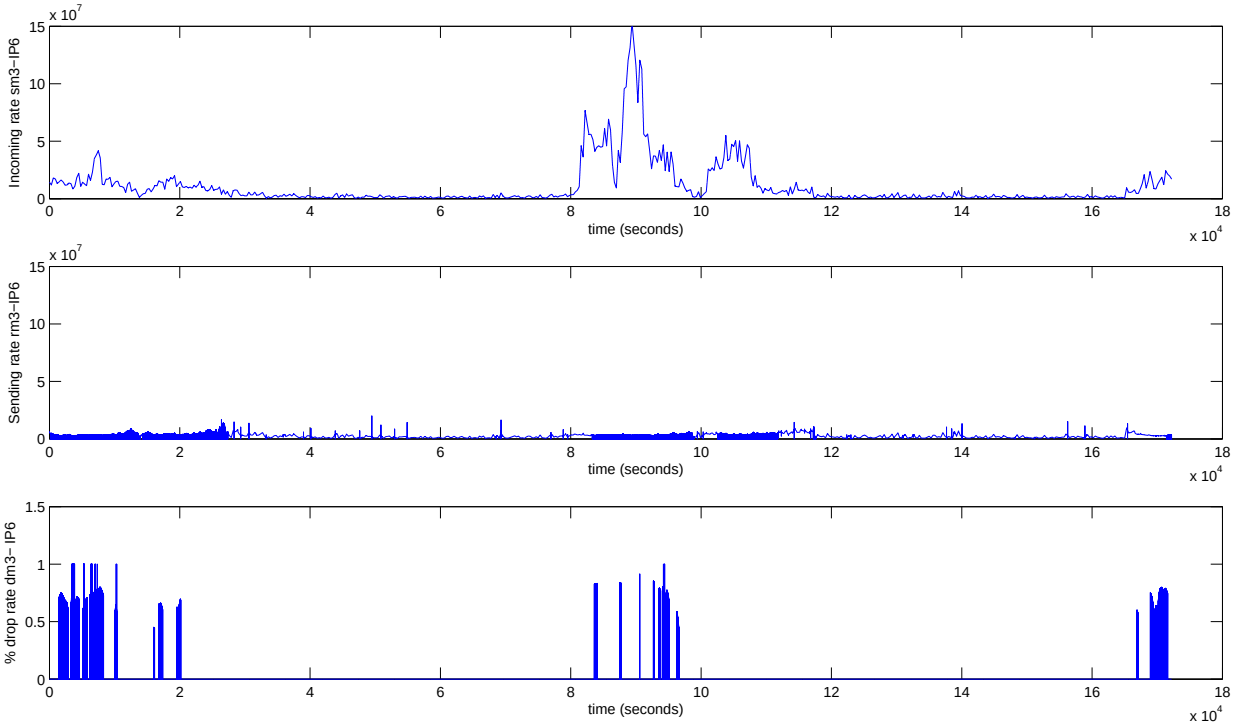

Figure 5.22: Receiving rate & Sending rate & drop rate% for medium-flow(M3) or IP-Precedence 6

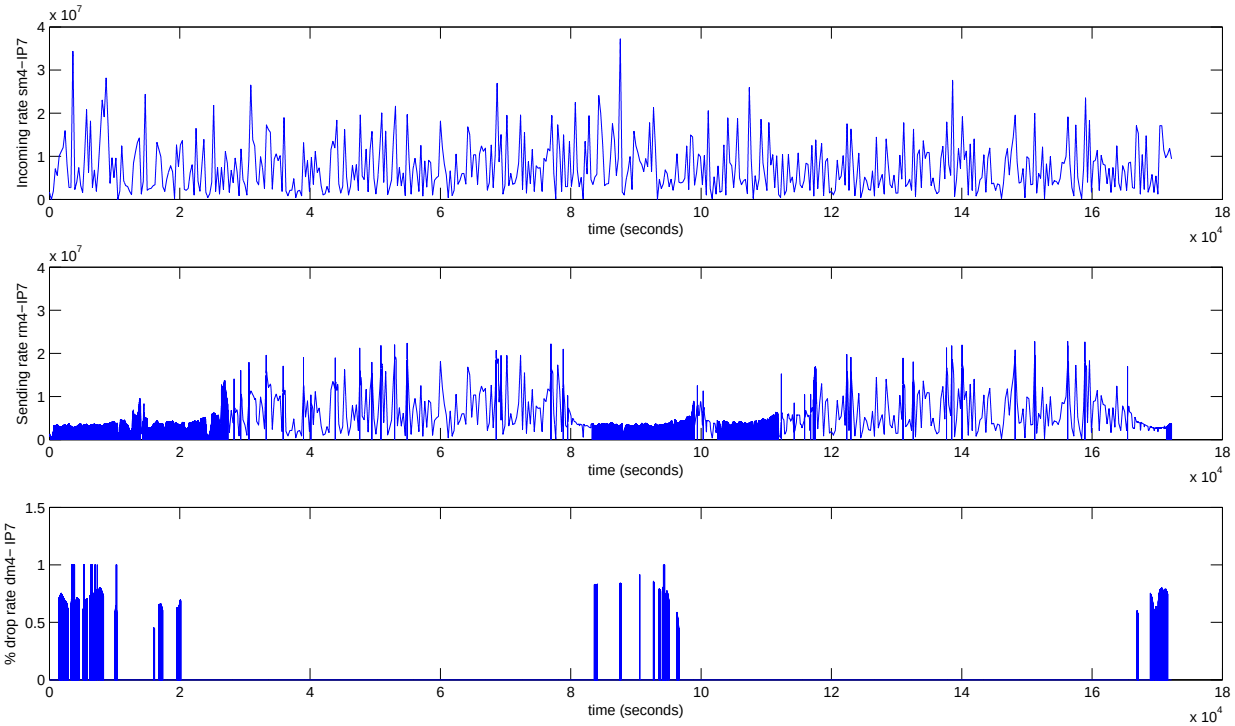

Figure 5.23: Receiving rate & Sending rate & drop rate% for medium-flow(M4) or IP-Precedence 7

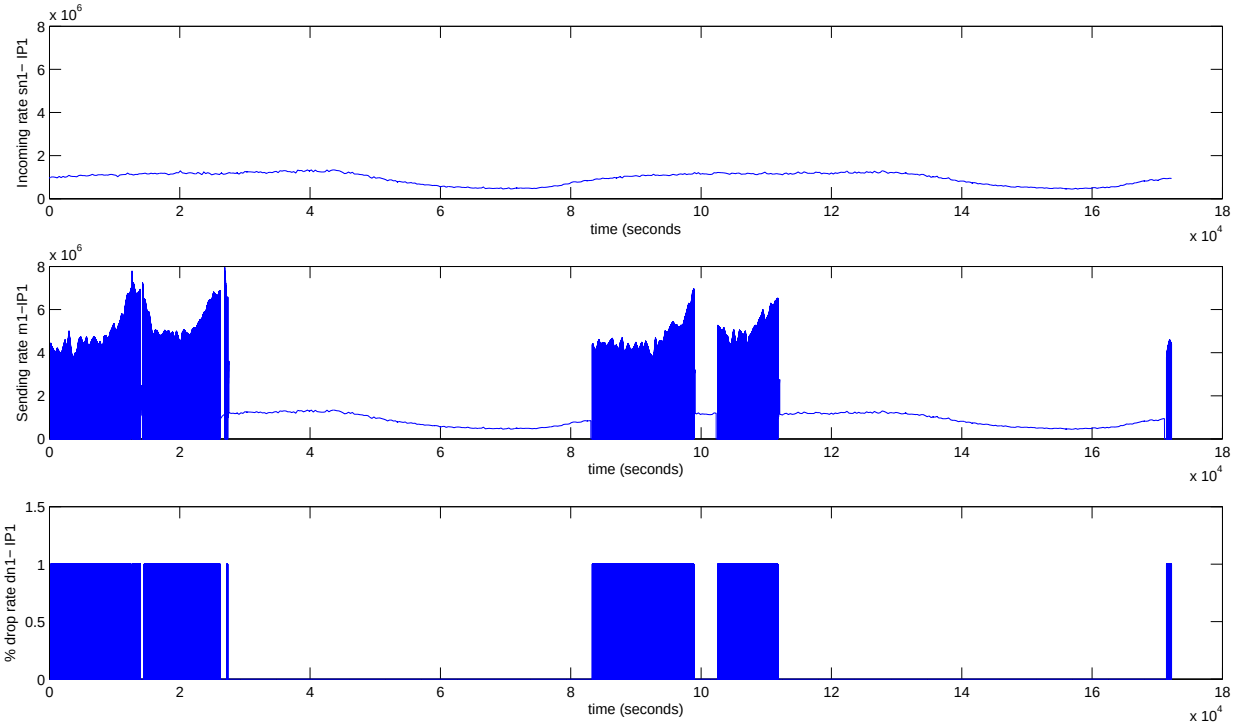

Figure 5.24: Receiving rate & Sending rate & drop rate% for normal-flow(N1) or IP-Precedence 1

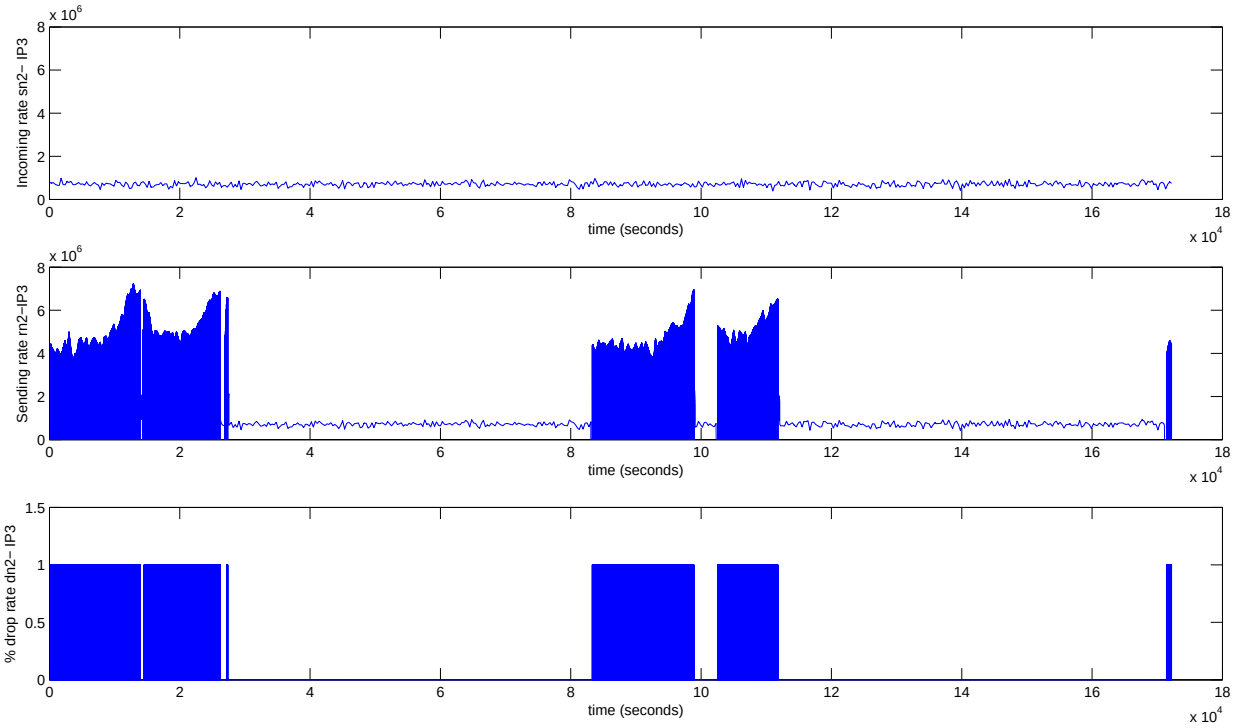

Figure 5.25: Receiving rate & Sending rate & drop rate% for normal-flow(N2) or IP-Precedence 3

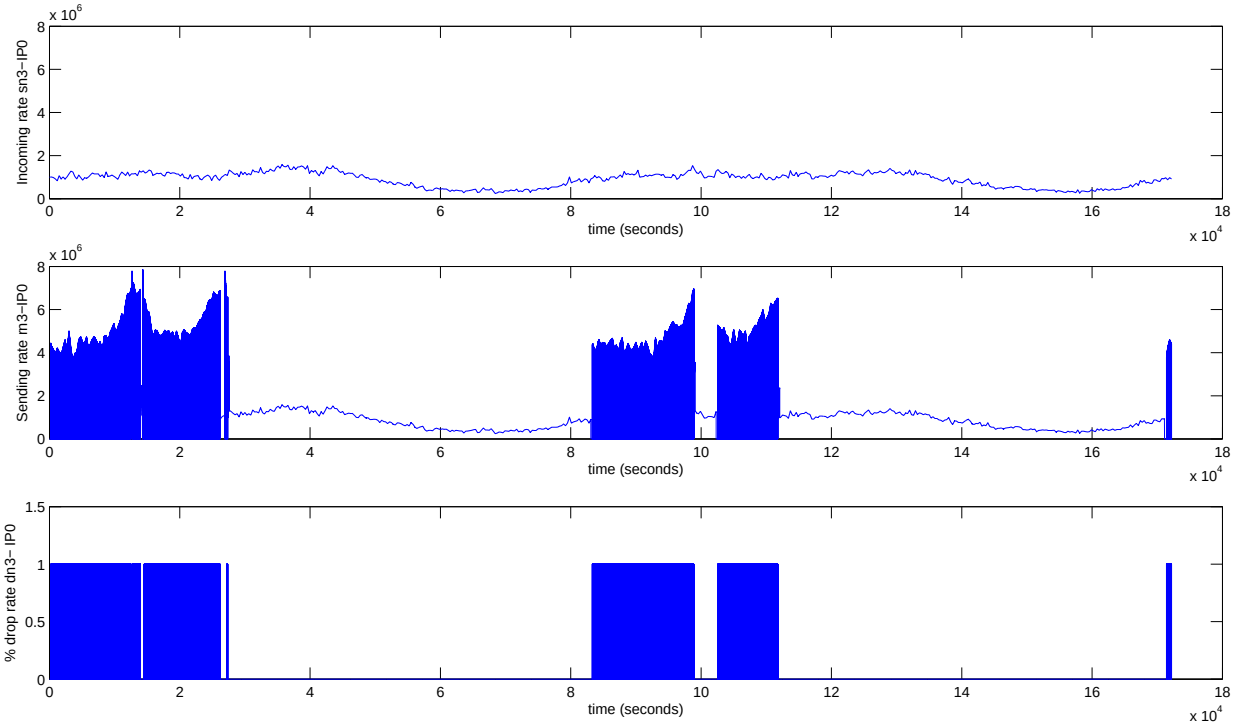

Figure 5.26: Receiving rate & Sending rate & drop rate% for normal-flow(N3) or IP-Precedence 0

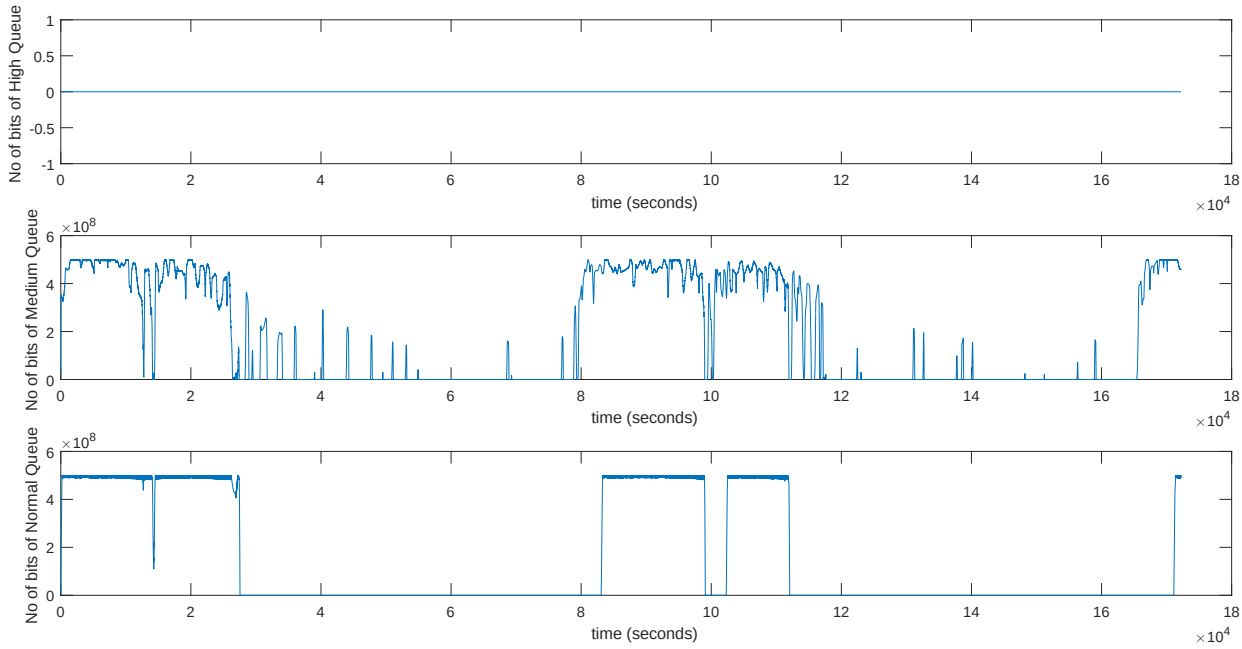

Figure 5.27: Number of bits in each queue

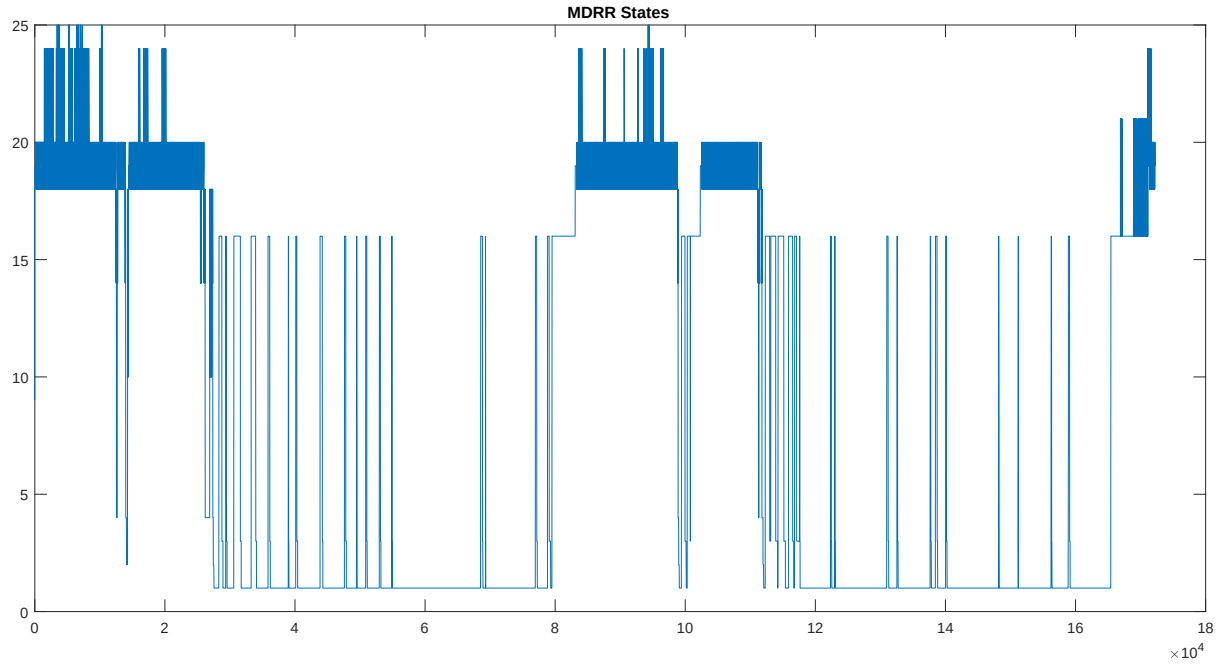

Figure 5.28: Switching between different states

5.4 Comparison between controlled and uncontrolled case

The comparison is based on the following parameters:

Sending Rate: When the results were closely observed, the sending rate increases after running the controller. This was expected because in many states, the sending rate depends on u and length of queues, and not only on the available bandwidth. And it increases in average due to the fact that less packets will be dropped. This can be seen in Figures 5.29 — 5.42.

Dropping Rate %: Dropping rate for Normal queue increases when MPC controller is used. In the uncontrolled case dropping rate is zero, when MPC controller is used it reaches 3×10^{-3} . The reason is that the controller chooses u in such a way to empty the the higher priority queues faster by giving more bandwidth to medium queue. As a result the state switches to a dropping state for the normal packets and this increases their dropping rate. This can be seen in Figures 5.43 — 5.45.

Queue Sizes: It is a direct consequence of optimized objective that the queue sizes should decline for the Medium buffer as the controller runs. So, it was predictable, but on the other hand, we observe that those of Normal buffer grow and reach its maximum size as the controller tries to allocate more bandwidth to the Medium queues by changing u. Figures 5.46 — 5.50 explain this even better.

Switching States The length of the Medium queue is decreased due to using the MPC controller, so in Figure 5.51 we observe that the controller is no longer switching to states where Medium queue is full. Instead it is switching more to states where Medium queue is filling and Normal queue is full.

5.4.1 Sending Rate

5.4.1.1 Medium flow(M1) or IP-Precedence 2

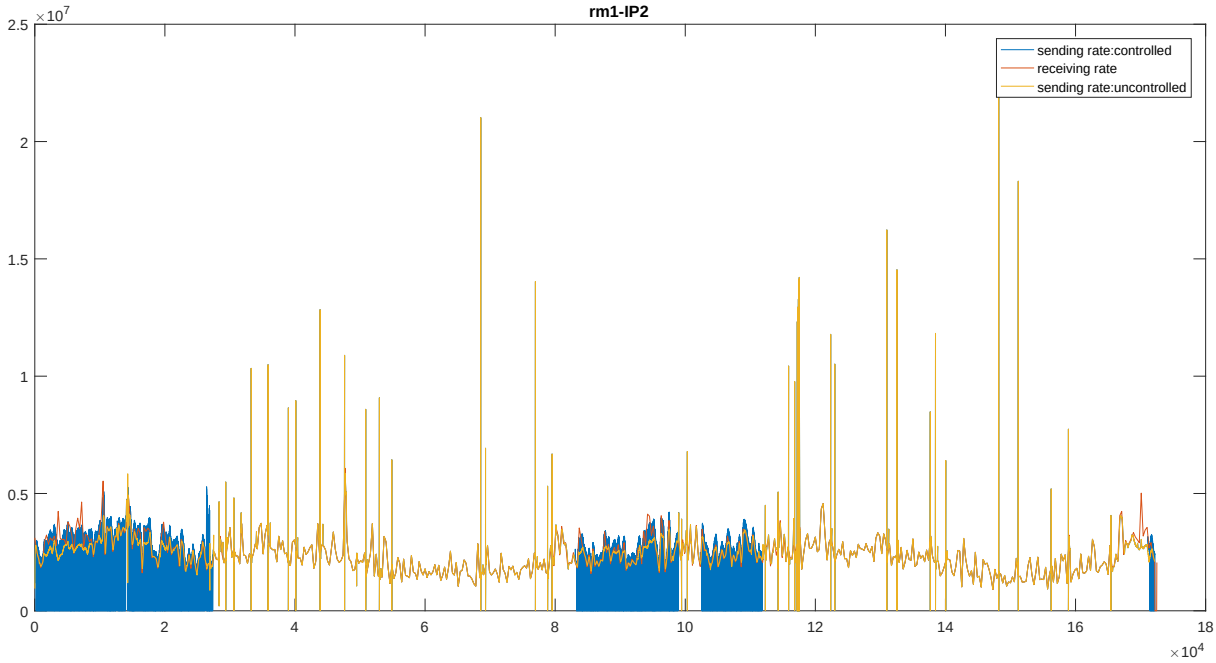

Figure 5.29: Comparison between sending and receiving rates for medium flow(M1):controlled/uncontrolled

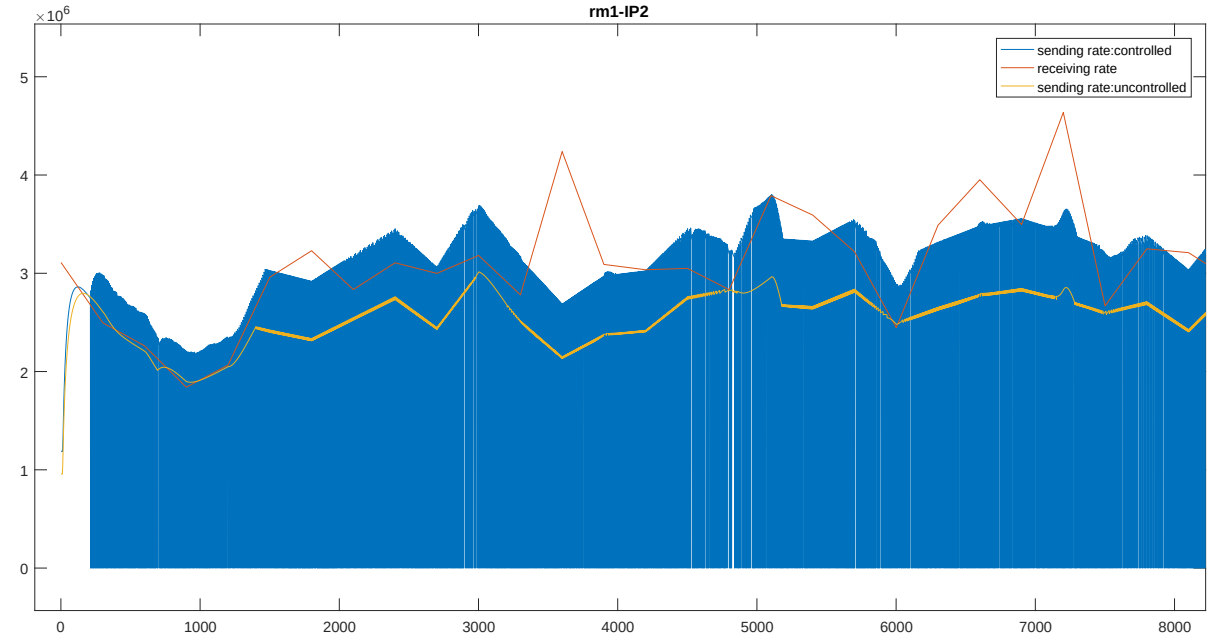

Figure 5.30: Comparison between sending and receiving rates for medium flow(M1):controlled/uncontrolled (zooming in 1/2)

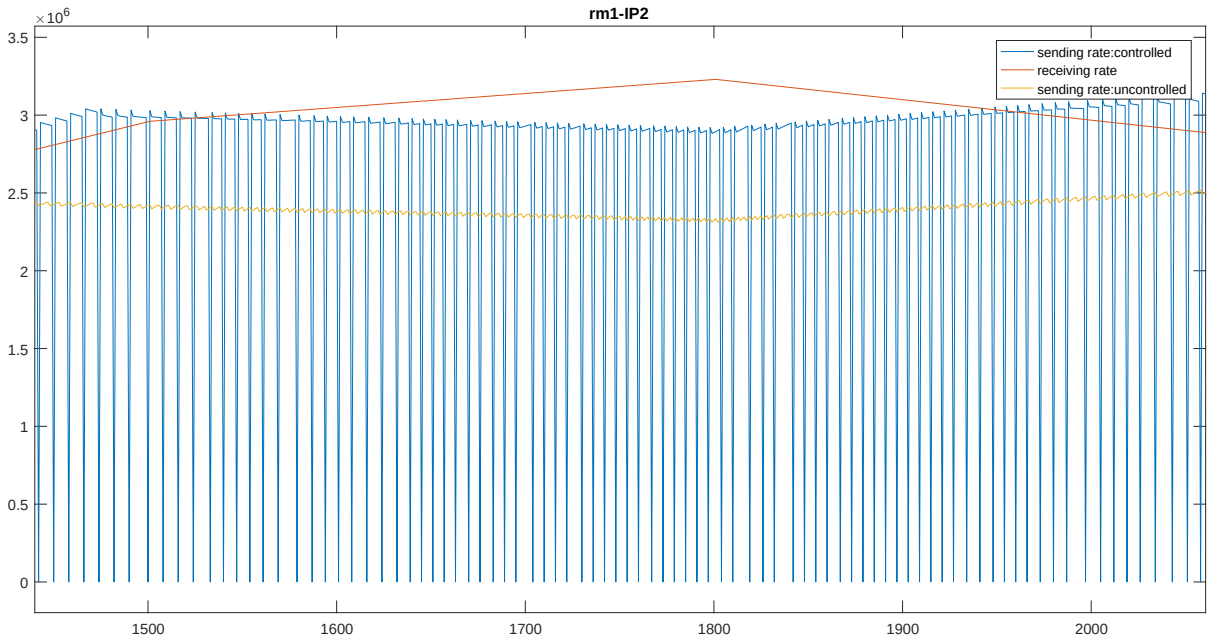

Figure 5.31: Comparison between sending and receiving rates for medium flow(M1):controlled/uncontrolled (zooming in 2/2)

5.4.1.2 Medium flow(M2) or IP-Precedence 4

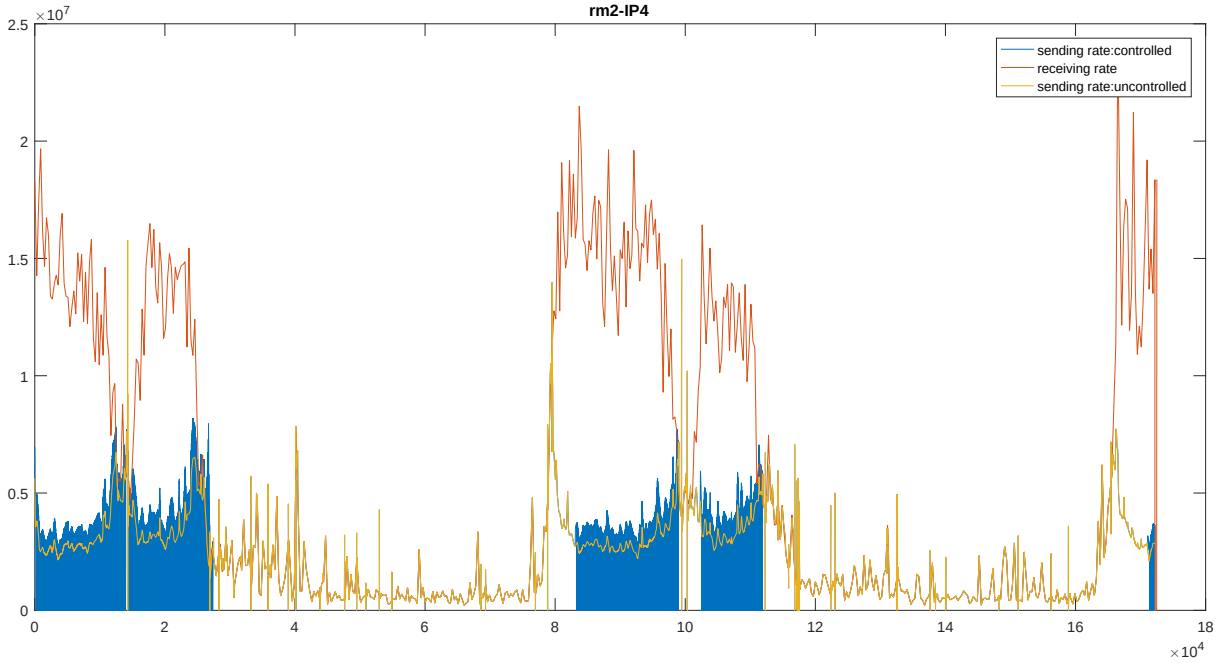

Figure 5.32: Comparison between sending and receiving rates for medium flow(M2):controlled/uncontrolled

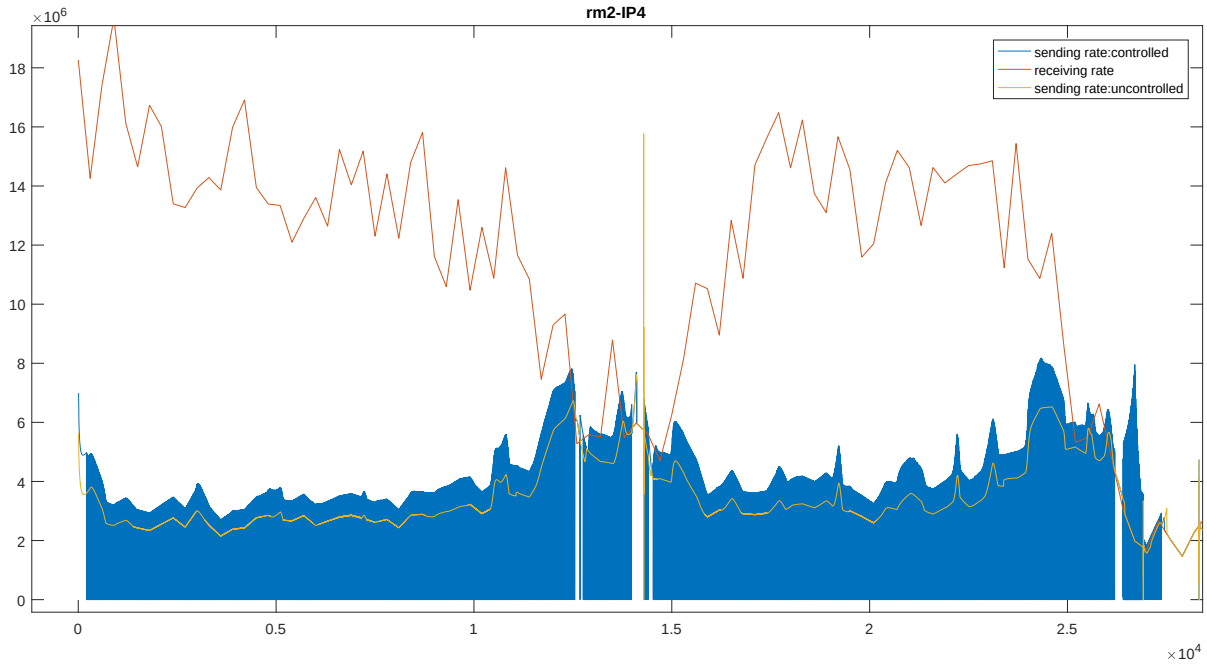

Figure 5.33: Comparison between sending and receiving rates for medium flow(M2):controlled/uncontrolled (zooming in 1/2)

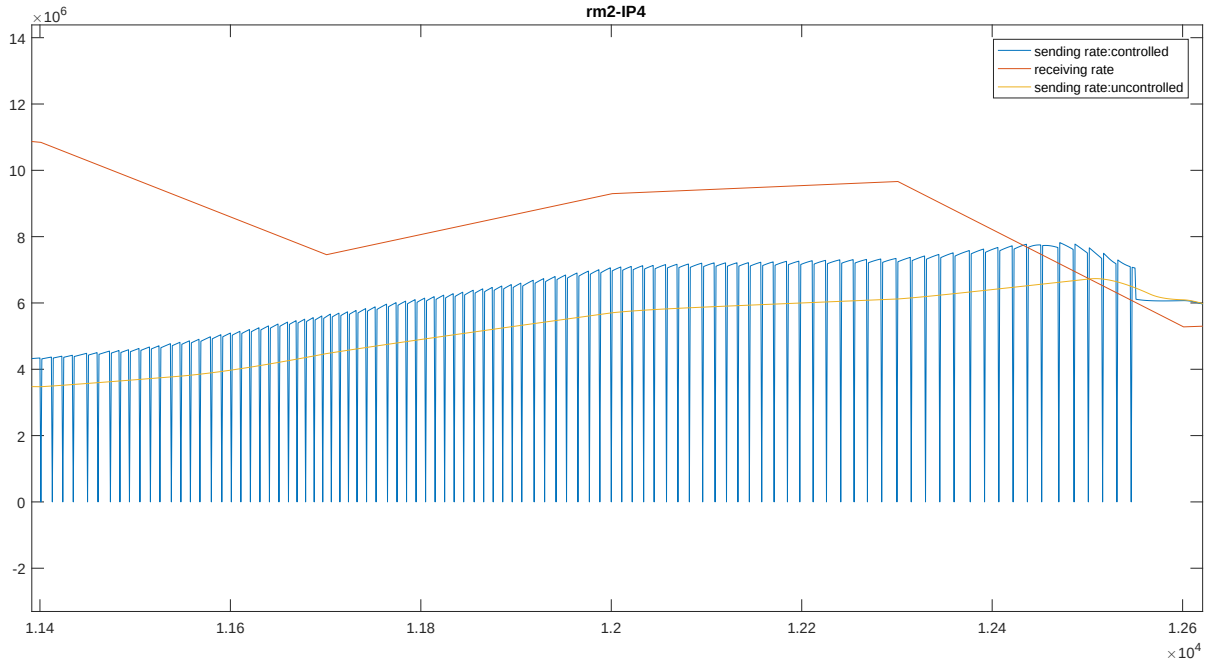

Figure 5.34: Comparison between sending and receiving rates for medium flow(M2):controlled/uncontrolled (zooming in 2/2)

5.4.1.3 Medium flow(M3) or IP-Precedence 6

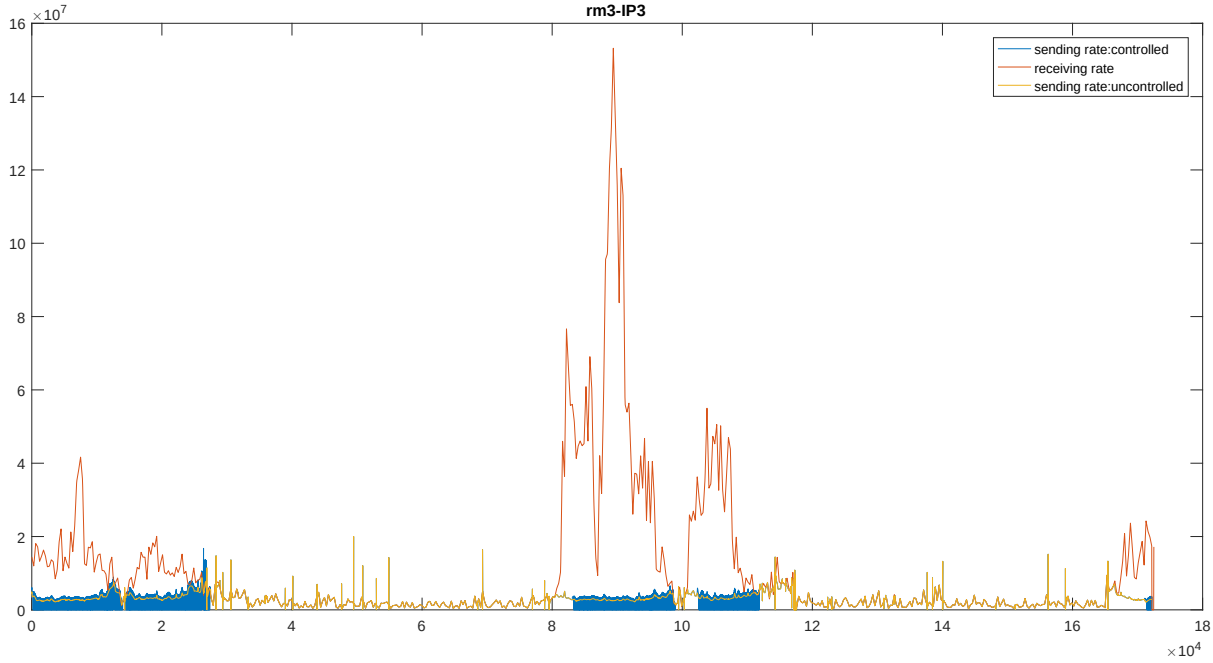

Figure 5.35: Comparison between sending and receiving rates for medium flow(M3):controlled/uncontrolled

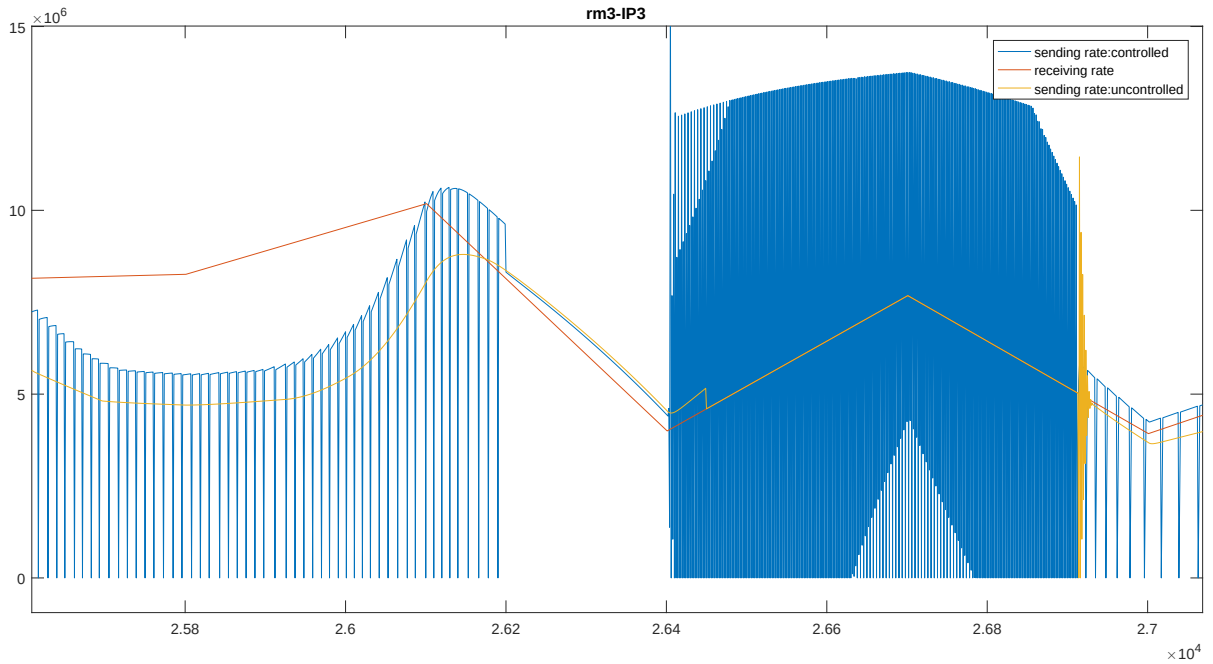

Figure 5.36: Comparison between sending and receiving rates for medium flow(M3):controlled/uncontrolled (zooming in)

5.4.1.4 Medium flow(M4) or IP-Precedence 7

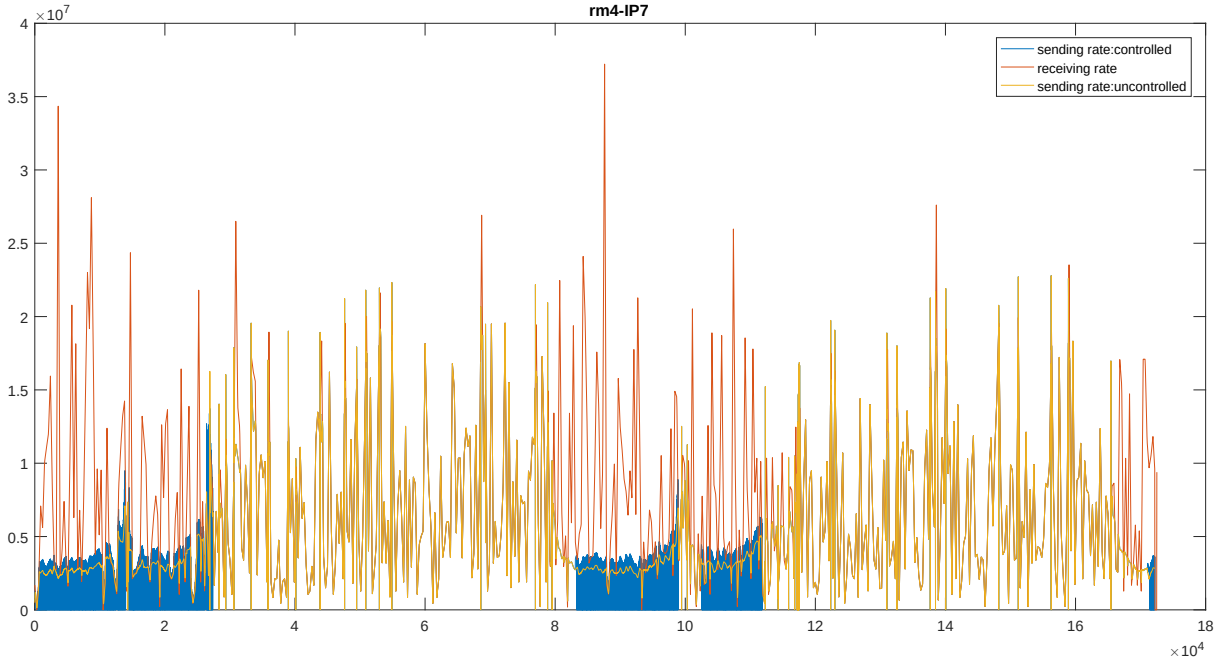

Figure 5.37: Comparison between sending and receiving rates for medium flow(M4):controlled/uncontrolled

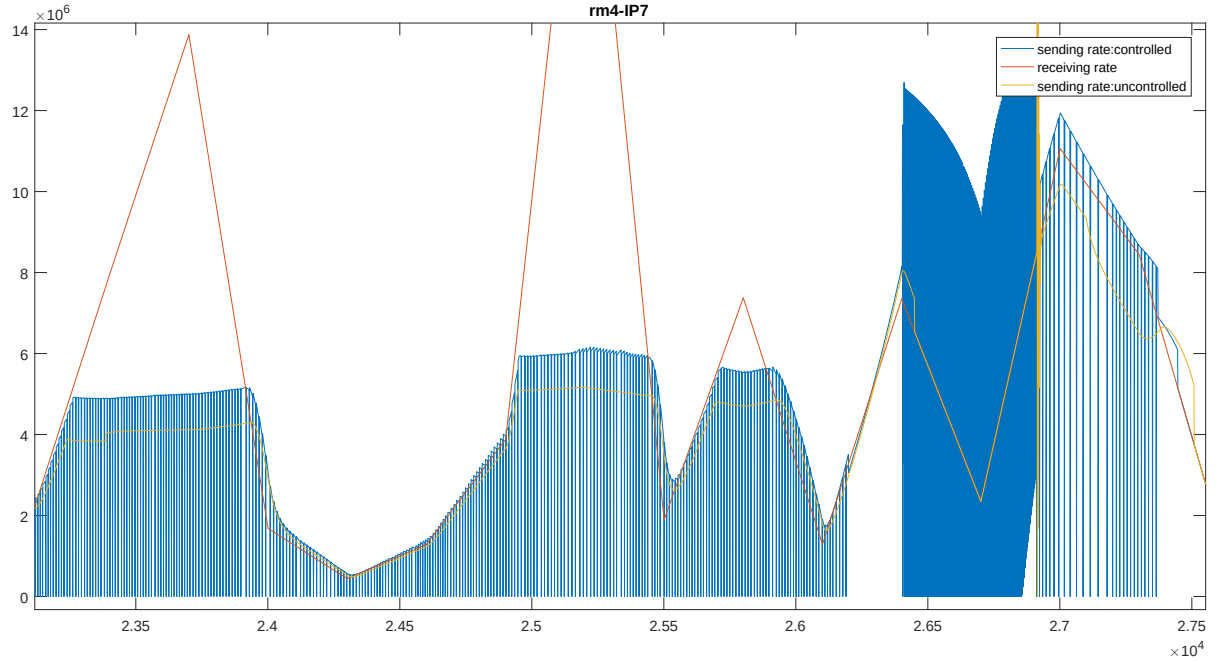

Figure 5.38: Comparison between sending and receiving rates for medium flow(M4):controlled/uncontrolled (zooming in)

5.4.1.5 Normal flow(N1) or IP-Precedence 1

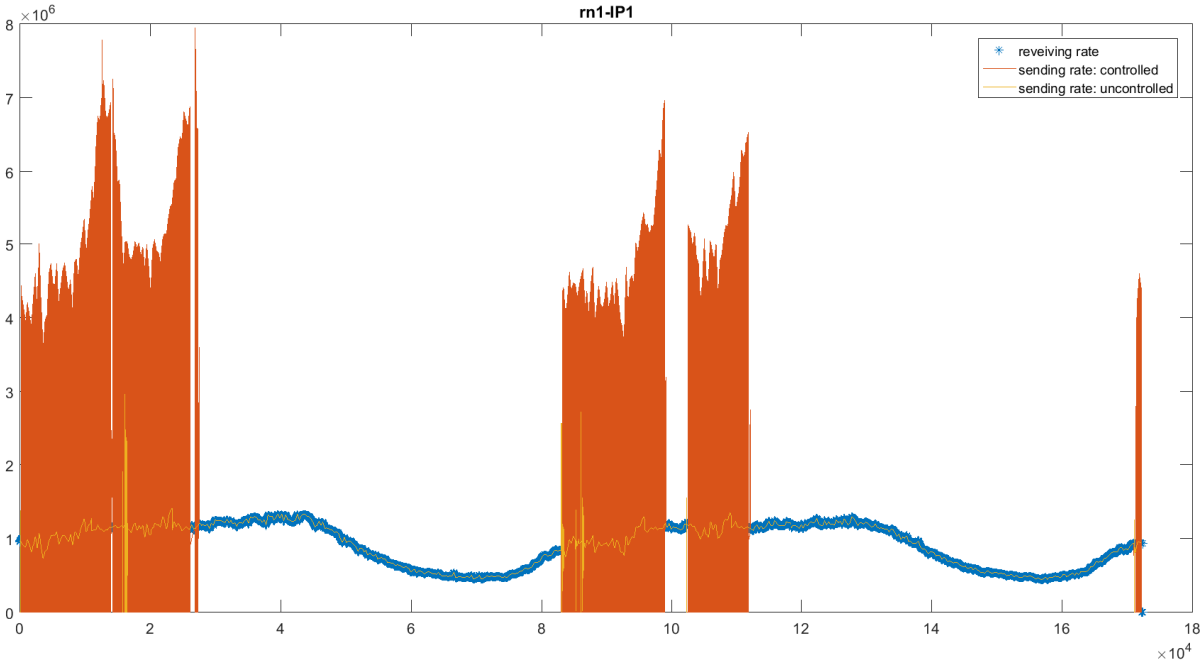

Figure 5.39: Comparison between sending and receiving rates for normal flow (N1):controlled/uncontrolled

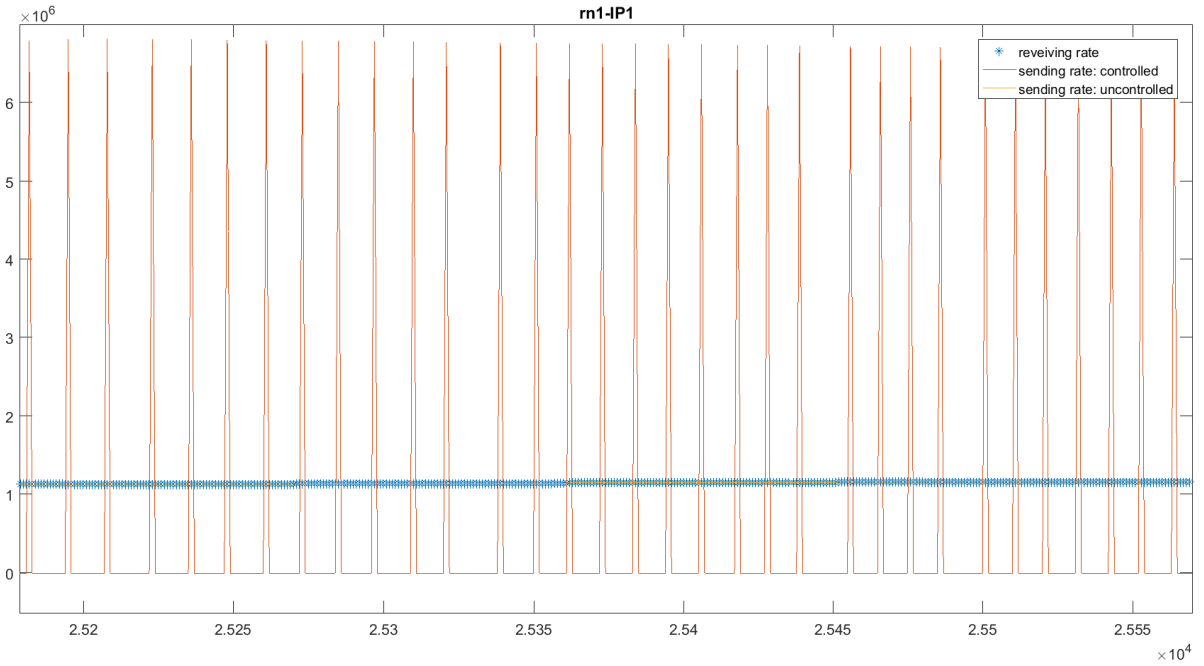

Figure 5.40: Comparison between sending and receiving rates for normal flow (N1):controlled/uncontrolled (zooming in)

5.4.1.6 Normal flow(N2) or IP-Precedence 3

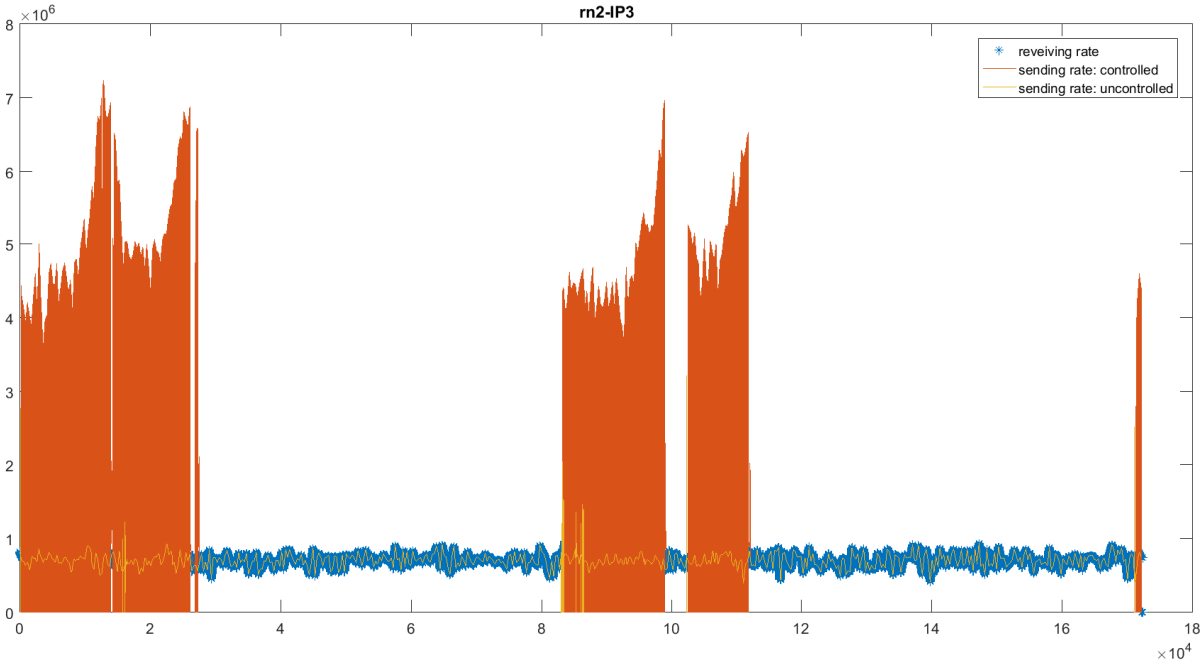

Figure 5.41: Comparison between sending and receiving rates for normal flow (N2):controlled/uncontrolled

5.4.1.7 Normal flow (N3) or IP-Precedence 0

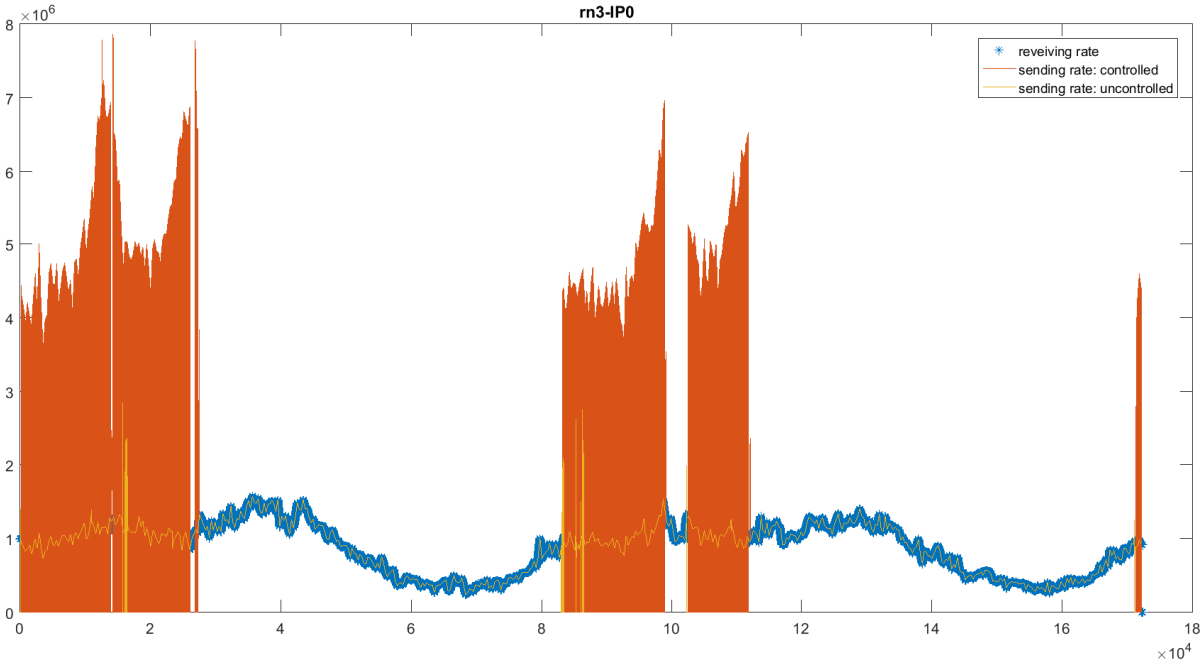

Figure 5.42: Comparison between sending and receiving rates for normal flow (N3):controlled/uncontrolled

5.4.2 Dropping Rate %

5.4.2.1 Normal flow(N1) or IP-Precedence 1

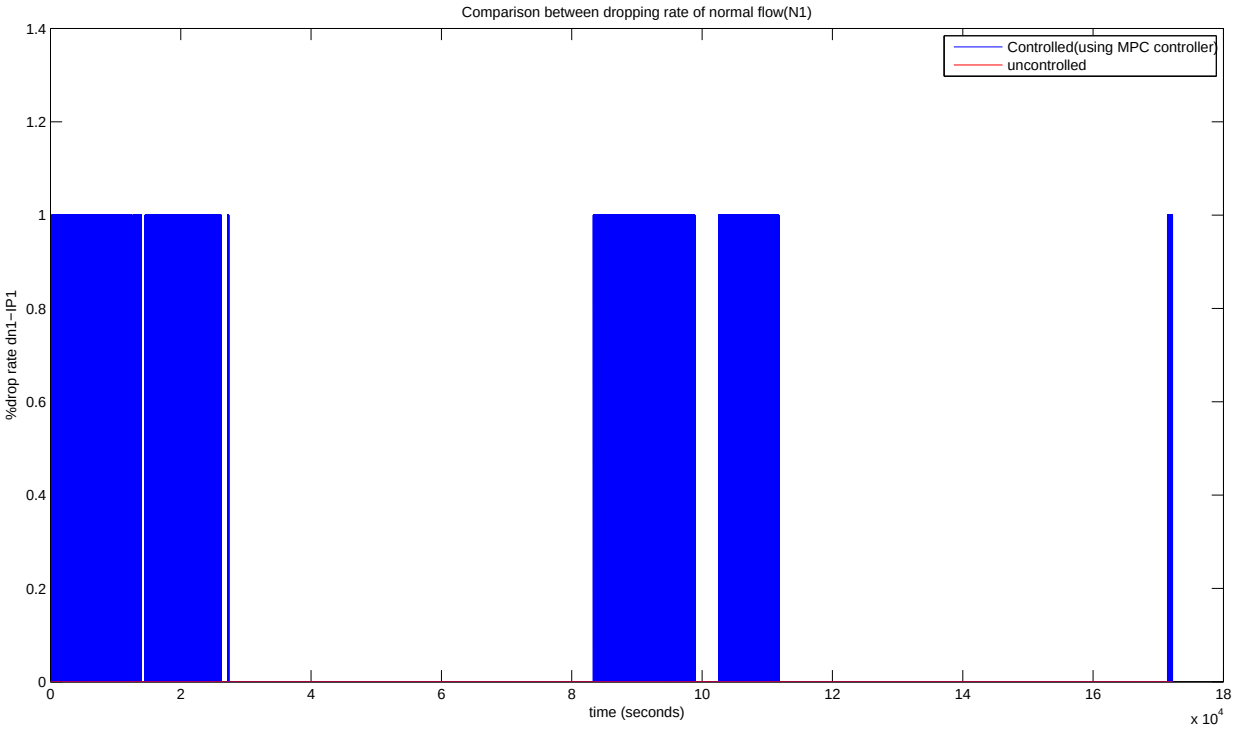

Figure 5.43: Comparison between Dropping rate for normal flow (N1): controlled / uncontrolled

5.4.2.2 Normal flow(N2) or IP-Precedence 3

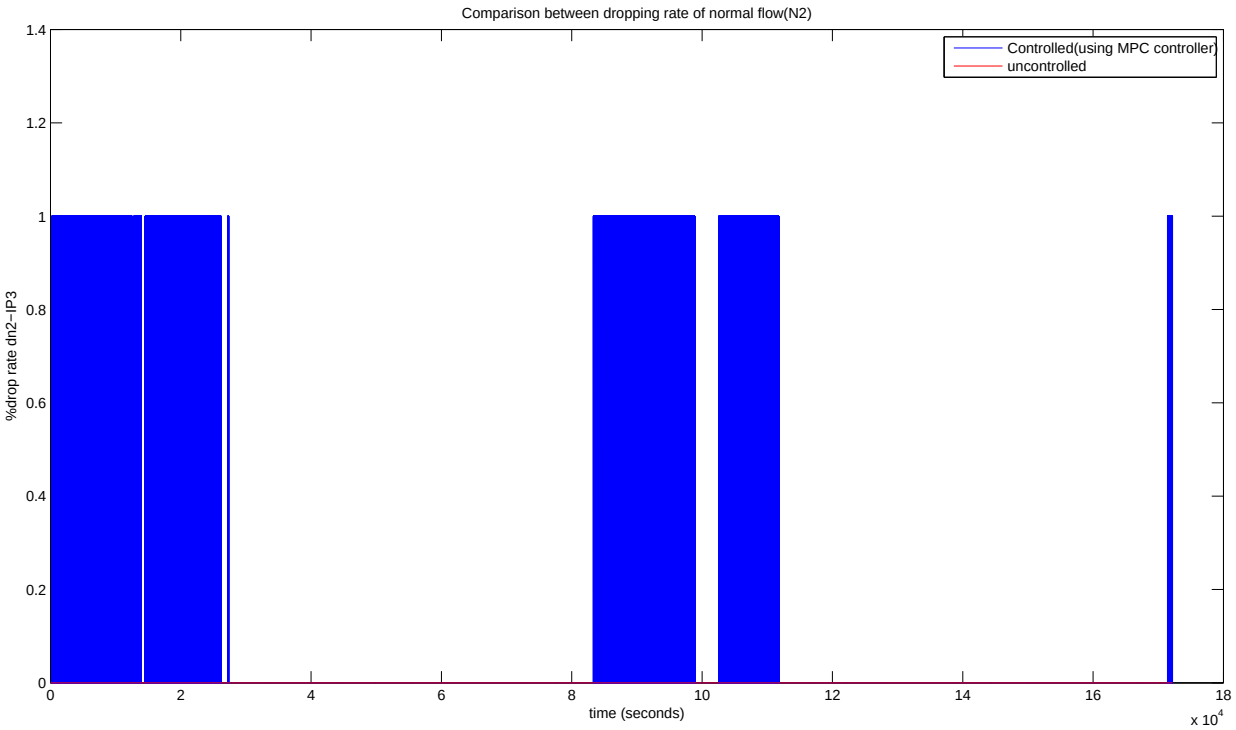

Figure 5.44: Comparison between Dropping rate for normal flow (N2): controlled / uncontrolled

5.4.2.3 Normal flow (N3) or IP-Precedence 0

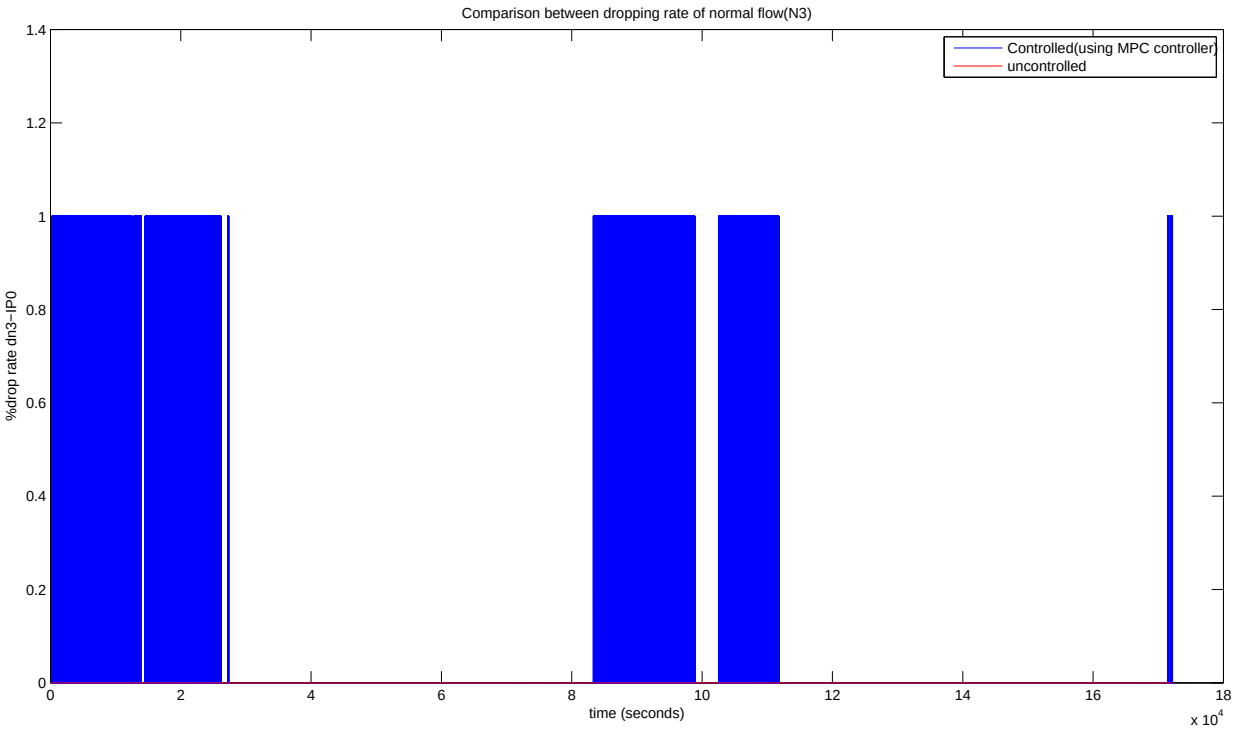

Figure 5.45: Comparison between Dropping rate for normal flow (N3): controlled / uncontrolled

5.4.3 Number of bits in each queue

5.4.3.1 Medium Queue

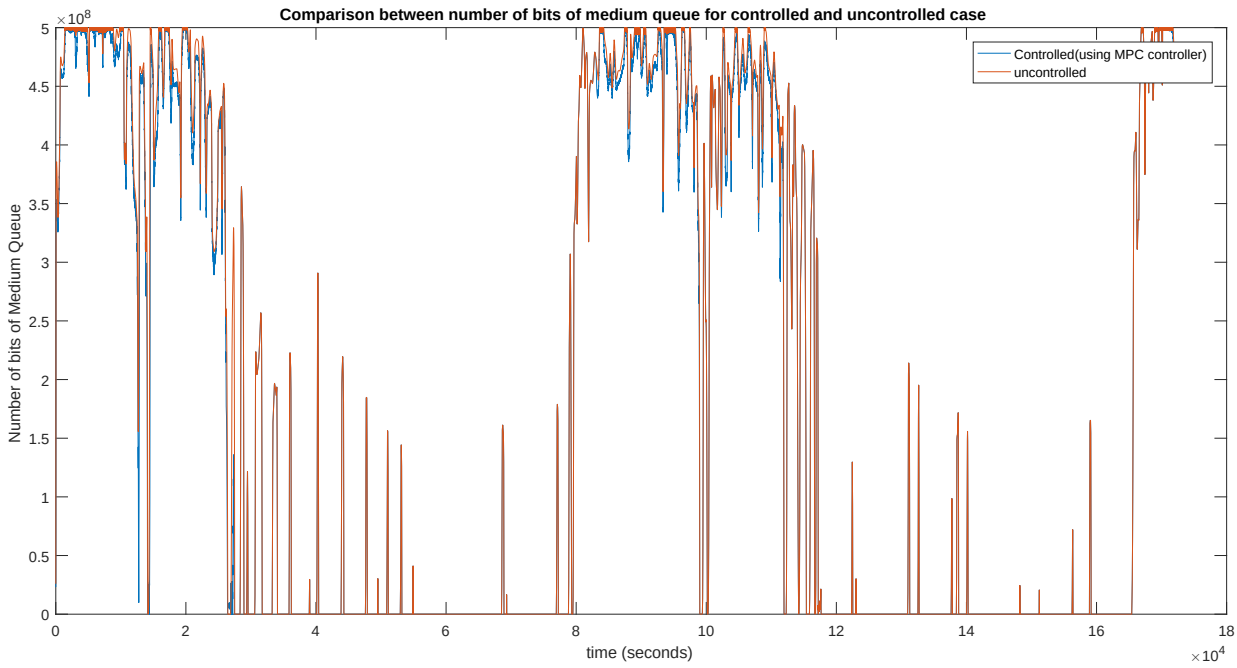

Figure 5.46: Comparison between number of bits in the Medium queue for controlled & uncontrolled case

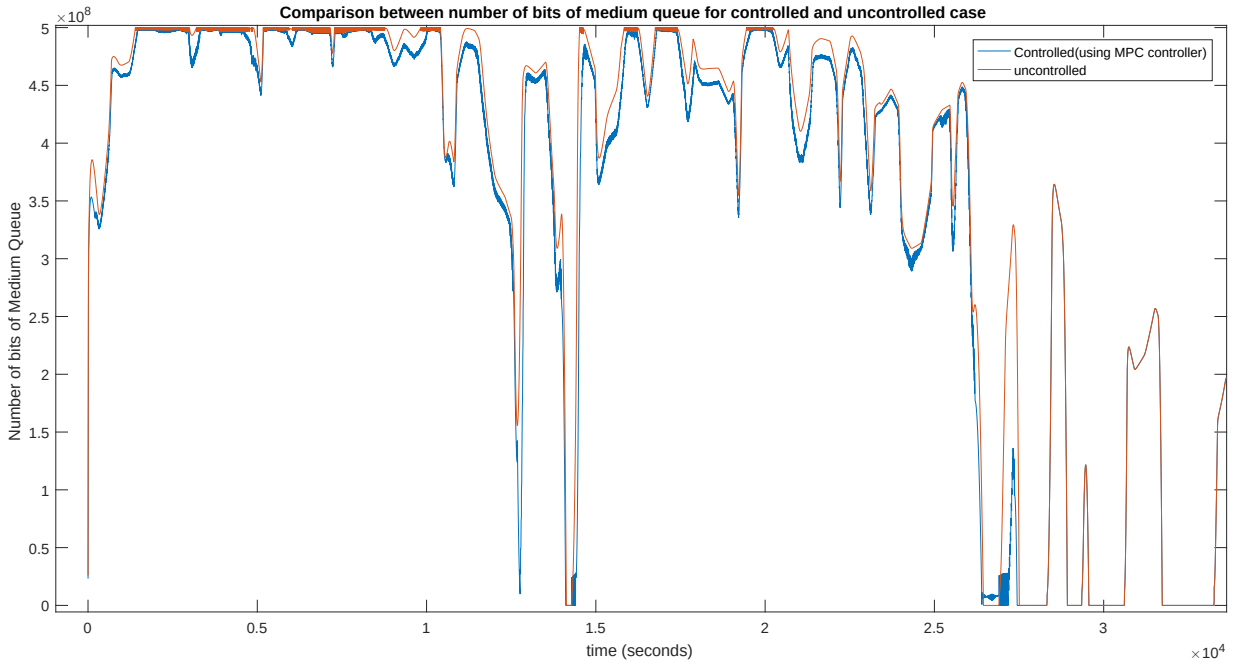

Figure 5.47: Comparison between number of bits in the Medium queue for controlled & uncontrolled case (zooming in 1/2)

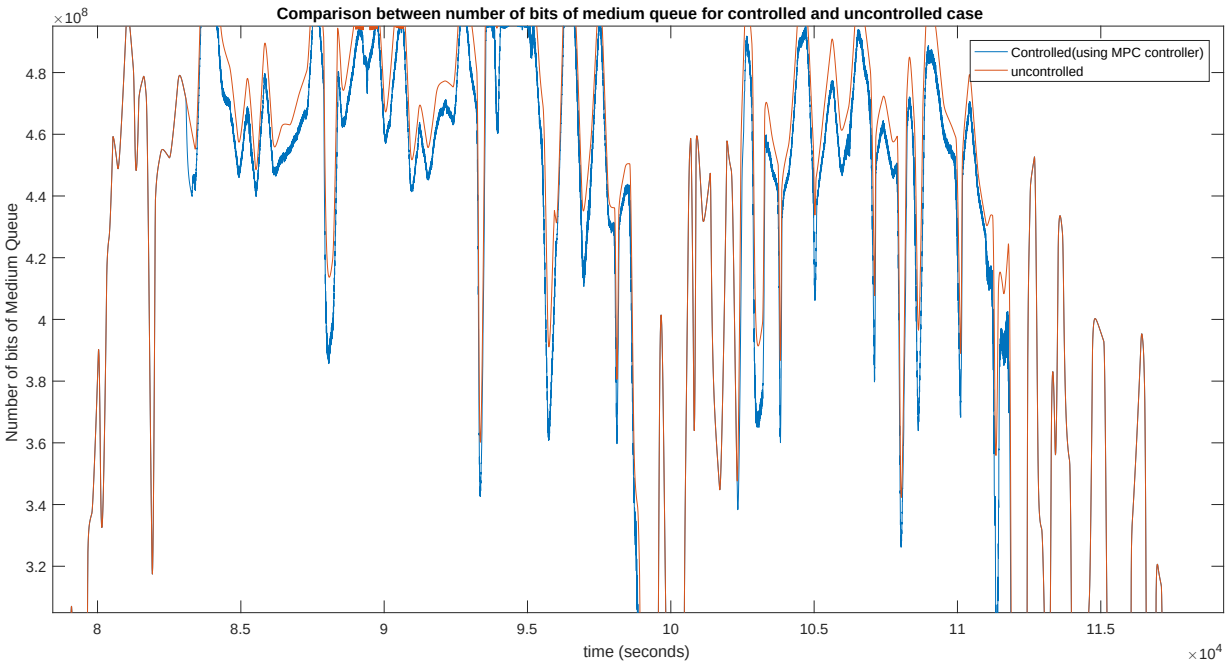

Figure 5.48: Comparison between number of bits in the Medium queue for controlled & uncontrolled case (zooming in 2/2)

5.4.3.2 Normal Queue

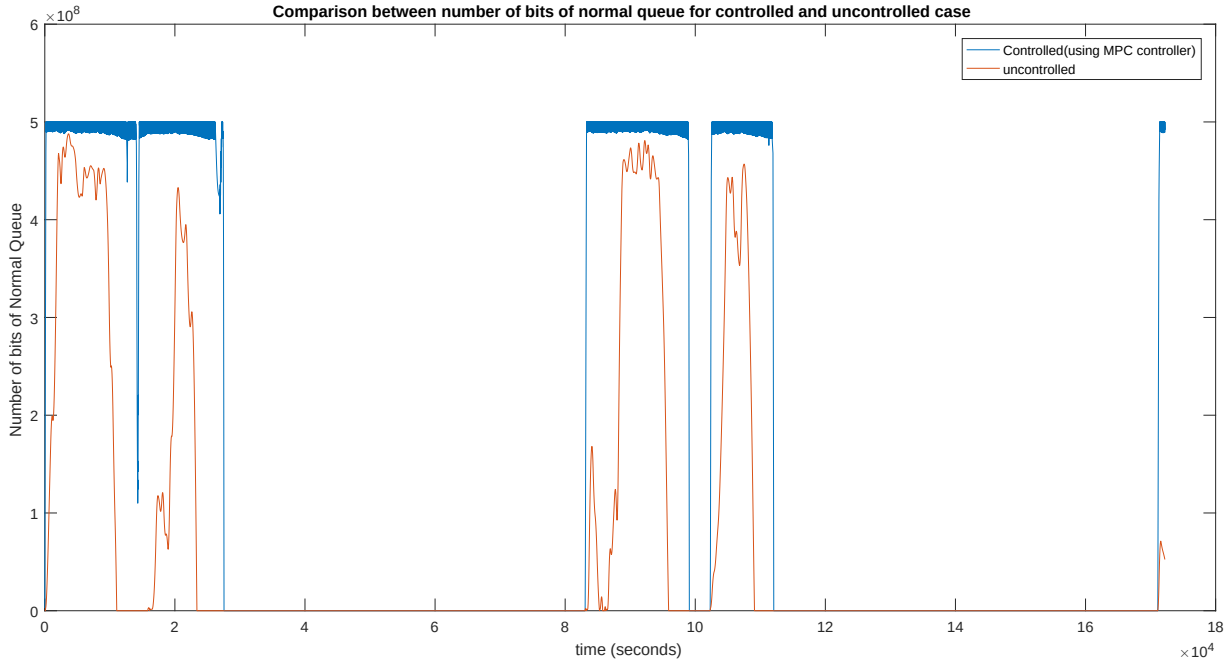

Figure 5.49: Comparison between number of bits in the Normal queue for controlled & uncontrolled case

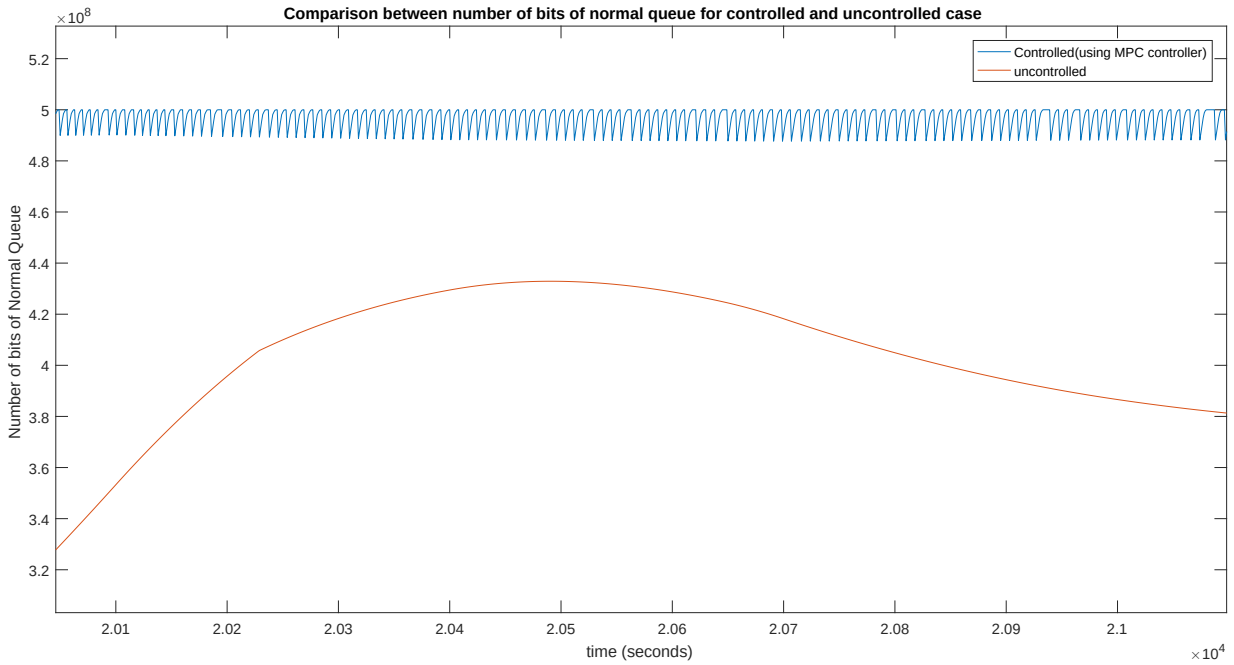

Figure 5.50: Comparison between number of bits in the Normal queue for controlled & uncontrolled case (zooming in)

5.4.4 Switching States

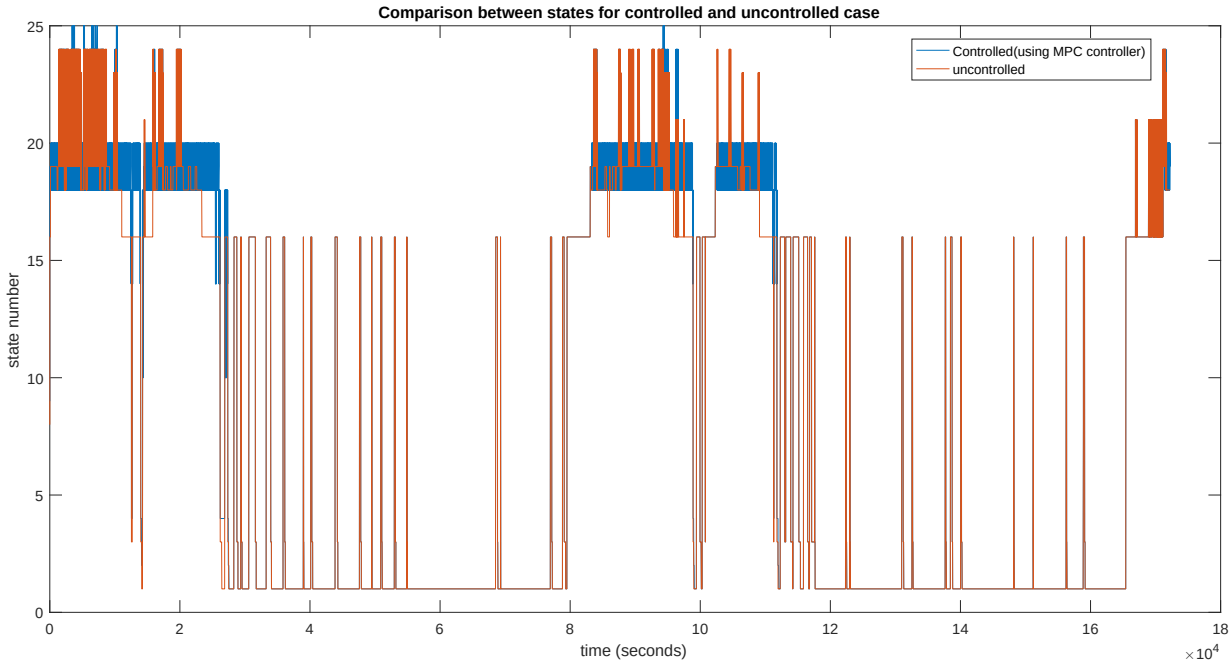

Figure 5.51: Comparison between switching states for controlled & uncontrolled case

5.4.5 Inputs

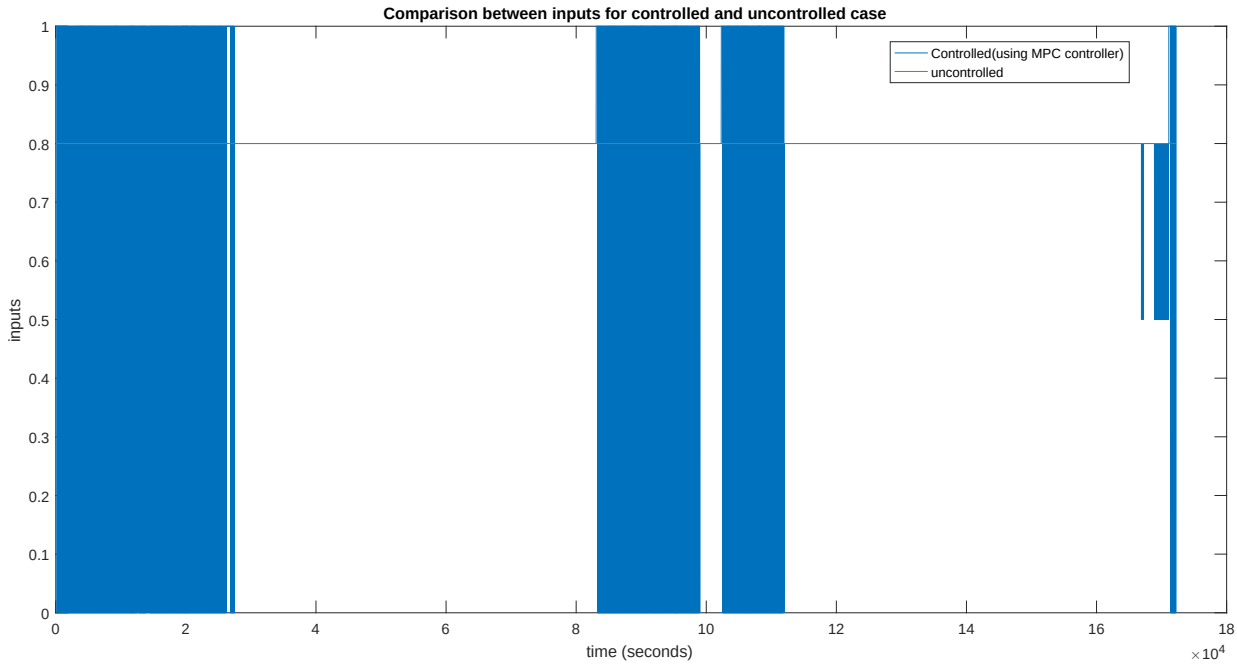

Figure 5.52: Comparison between Input u for controlled & uncontrolled case

Chapter 6

Conclusion

Concluding Remarks and Future Work:

This is a first step towards setting up mathematical models for Congestion Management Tools(priority queueing and Modified Round Robin Deficit (MDRR) algorithm used in Telecom Italia) for obtaining better QoS. Implementing MPC controller is an important step towards achieving better QoS. Model Predictive Control gives us the flexibility to optimize different objective functions and to define constraints on the inputs, outputs and states. By choosing different weights for the different flows we were able to control the priority for each flow by tuning these weights. By choosing an appropriate objective function and constraints, we can decide to either enhance the sending rates, reduce queue sizes or reduce the dropping rates for a certain flow. Of course the bigger N (time horizon), the better predictions will be. The network administrator may choose the parameters and the objective at different times to respond more optimally to different needs of the network.

Future work aims to find a suitable solver for the Hybrid Nonlinear MPC instead of solving a switching-MPC. Also to consider increasing the sending rate for each flow as an objective function and to find a suitable solver to solve this problem.

Bibliography

[1] IP Precedence for Video Streaming, http://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoS-SRND-Book/QoSIntro.html

[2] Misra, Vishal, Wei-Bo Gong, and Don Towsley. "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED." ACM SIGCOMM Computer Communication Review. Vol. 30. No. 4. ACM, 2000.

[3] Misra, Vishal, Wei-Bo Gong, and Don Towsley. "Stochastic differential equation modeling and analysis of TCP-windowsize behavior." Proceedings of PERFORMANCE. Vol. 99. 1999.

[4] Di Benedetto, M.D., Di Loreto, A., D'Innocenzo, A. and Ionta, T., 2014, June. Modeling of traffic congestion and re-routing in a service provider network. In 2014 IEEE International Conference on Communications Workshops (ICC) (pp. 557-562). IEEE.

[5] Amirthalakshmi Veeraraghavan. Modeling of traffic congestion in a QoS Based service provider network, Master's thesis, The university of L'Aquila.

[6] Bohacek, S., Hespanha, J. P., Lee, J., & Obraczka, K. (2003, June). A hybrid systems modeling framework for fast and accurate simulation of data communication networks. In ACM SIGMETRICS performance evaluation review (Vol. 31, No. 1, pp. 58-69). ACM.

[7] Issa, O., Speranza, F. and Falk, T.H., 2012, December. Quality-of-experience perception for video streaming services: Preliminary subjective and objective results. In Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC), 2012 Asia-Pacific (pp. 1-9). IEEE.

[8] Eckert, Marcus, Thomas Martin Knoll, and Florian Schlegel. "Advanced MOS calculation for network based QoE Estimation of TCP streamed Video Services." In Signal Processing and Communication Systems (ICSPCS), 2013 7th International Conference on, pp. 1-9. IEEE, 2013.

[9] Mok, Ricky KP, Edmond WW Chan, and Rocky KC Chang. "Measuring the quality of experience of HTTP video streaming." In 12th IFIP/IEEE International Symposium on Integrated Network Management (IM 2011) and Workshops, pp. 485-492. IEEE, 2011.

[10] Babak Mehrabi. QoE based optimal control of priority queuing in a service provider network, Master's thesis, The university of L'Aquila.

[11] Ellis, M., Durand, H.,& Christofides, P. D. (2014). A tutorial review of economic model predictive control methods. Journal of Process Control,24(8), 1156-1178.

[12] Christofides, Panagiotis D., et al. "Distributed model predictive control: A tutorial review and future research directions." Computers & Chemical Engineering 51 (2013): 21-41.

[13] MPC course at IMT LUCCA http://cse.lab.imtlucca.it/~bemporad/mpc_course.html

[14] Mpc lab @ uc-berkeley, http://www.mpc.berkeley.edu/home.

[15] Ip quality of service, Srinivas Vegesna , Cisco Press, 2000.

[16] http://docwiki.cisco.com/wiki/Quality_of_Service_Networking