CHALMERS

Modeling and analysis of satellite terminal with an AQM (Active Queue Management)

Master of Science Thesis

ROMAIN DELPOUX

Department of Signals and Systems
Division of Automatic Control, Automation and Mechatronics
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2009
Report No. EX066/2009

Year 2008-2009

MASTER SYSTEMS, CONTROL, AND MECHATRONICS

Automatic Control

Modeling and analysis of satellite terminal with an AQM (Active Queue Management)

Romain DELPOUX

Chalmers University

Groups : MAC and OLC

Supervisors: P. Berthou, F. Gouaisbaut and Y. Labit

Examiner: B. Egardt

Thanks

The work present in this report has been carried out at the Laboratoire d'Analyse et d'Architecture des Systèmes *(LAAS)* of the Centre National de la Recherche Scientifique *(CNRS)* of Toulouse (France), within the framework of the Master program Systems, Control and Mechatronics of Chalmers University of Göteborg (Sweden).

I would like to thank initially Mister Raja Chatila to have received me in the laboratory LAAS, François Vernadat for the group OLC and Isabelle Queinnec for the group MAC to have allowed me to realize my Master's thesis.

I thank in particular my three supervisors Pascal Berthou, Frédéric Gouaisbaut, and Yann Labit who supported me throughout my Master's thesis. I hold to emphasize their availability along this few months. It was a pleasure to work with them. I learned a lot during this period. They allowed me to progress considerably.

I thank also my examiner Bo Egardt, from Chalmers University, who has accepted to support my project in France, and make the necessary for that this Master's thesis happens well between France and Sweden.

And finally I thank all the members of the groups MAC and OLC for their greeting and help. I would like to thanks also all the PhD students and other students of the group. They brought a sympathetic and friendly environment to the laboratory life. And a special thanks to Yassine Ariba who has always answered to my innumerable questions. Without forgetting Mohamad El Masri and Baptiste Jacquemin.

Abstract

The standardization of a Return Channel via Satellite (DVB-RCS) and the satellite community efforts in term of interoperability over the last few years are expected to play, in a near future, a decisive role in Next Generation Networks (NGNs) through the integration of Satellite networks as an alternative to terrestrial networks like DSL (Digital Subscriber Line) in low terrestrial infrastructure areas. Furthermore, the advance of distributed multimedia applications like voice over IP and videoconference implies some new requirements to guarantee the Quality of Service (QoS). It concerns a limited transmission delay, a weak jitter, a minimal loss rate and a guaranteed bandwidth.

The transport protocols considered in this work are TCP (Transmission Control Protocol) and UDP (User Datagram Protocol). TCP is used for the NRT (Non-Real Time) applications (Peer to Peer, FTP...), and UDP for time-sensitive applications (VoIP ...). The data applications are stored in the BE and AF queue while multimedia applications in the EF and AF queue. Consequently, AF queue receives data and multimedia streams. The goal of this work is to regulate a priori the Diffserv AF queue and avoid the overflow. Considering the AF queue shared between UDP based multimedia applications and TCP based data transfer, the main idea consists in controlling TCP streams to guarantee transmission capacity of UDP packets. The most constraint application can then enjoy low buffers time delay and very few losses.

In order to solve this problem, we design a congestion control mechanism based on Active Queue Management (AQM) techniques by using control theory. To this end a fluid model of TCP connection originally designed for wire-networks is proposed for satellite networks. Then, the design of a robust proportional integral (PI) and a robust dead time based controller are investigated. To avoid AF queue over flooding, TCP packets are voluntary dropped in the ST according to regulation rules. TCP connection throughput is then controlled and limited to protect UDP streams against unnecessary drops. The different methods are then simulated on matlab and NS2-Simulator and compared to a classical DropTail mechanism.

Contents

List of Figures

List of Tables

Acronyms

AF	Assured Forwarding
$AIMD$	Additive Increase Multiplicative Decrease
AQM	Active Queue Management
BE	Best Effort
$DAMA$	Demand Assignment Multiple Access
DVB	Digital Video Broadcast
$DVB-RCS$	Digital Video Broadcast - Return Channel Satellite
$DVB-S$	Digital Video Broadcast - Satellite
EDF	Earliest Deadline First
EF	Expedite Forwarding
IP	Internet Protocol
LFT	Linear Fractional Transformation
MAC	Medium Access Control
NCC	Network Control Center
NRT	Non Real Time
PI	Proportional Integral
PQ	Priority Queueing
QoS	Quality of Service
RED	Random Early Detection
RT	Real Time
RTT	Round Trip Time
ST	Satellite Termianl
TCP	Transmission Control Protocol
TO	Time Out
UDP	User Datagram Protocol

Introduction

Several commercial DVB-RCS (Digital Broadcast - Return Channel Satellite) based networks are already deployed and many efforts are done in order to enhance interoperability. Most recent commercial deployments provide either Internet access or mesh connectivity over a transparent geostationary satellite. Fixed bandwidth contracts are generally offered to consumers thanks to a simple resource management scheme. It simplifies admission control, reduces cost and gains experience while waiting for the standardization of finer resource management strategies and equipment. A lot of work on IP (Internet Protocol) over satellite remains particularly in the Quality of Service (QoS) field and the next step is, obviously, to take benefits from DVB-RCS dynamic allocation schemes and IP QoS architectures to cope with the satellite delay and the scarce uplink resources.

This work deals with an algorithm of the ST (Satellite Terminal) , more precisely, it contributes to the QoS management of the return link, a central problem in satellite network (compared to wire network) due to the satellite delay and the scarce uplink resources. Instead of over-sizing the connection, which could be very expensive, we aim at reaching an optimal exploitation of uplink resources [8].

In order to solve this problem, we design a congestion control mechanism (Active Queue Management (AQM)). This mechanism is equivalent to a controller designed by using automatic control methods. Two AQM are described in this article, a robust proportional integral (PI) controller and a robust dead time based controller. These mechanisms are applied to an existing model of the TCP (Transmission Control Protocol) for wire network described in [12] that we adapt to the satellite network. The different methods are simulated on NS2-Simulator and compared to the DropTail mechanism.

This report is composed of three parts. The first part presents the context of the Satellite networks and the model. The second part deals with the model of TCP and the design of both controllers. Finally, the third part presents the simulations and the results in order to show the behavior of this method. The last part concludes on the model evaluation and the future work.

Chapter 1

Background in Satellite Network and Modeling

In this first chapter, we introduce the context of the application. It is important to understand the position of the problem regarding the satellite network as well as the TCP. We will define the system used during the work and its modeling. It is from this point that we will be able to apply the automatic control theory.

1.1 Satellite Network

Geostationary satellite access networks are expected to play, in a near future, a decisive role in Next Generation Networks (NGNs) as they are intended to provide broadband access to interactive multimedia services in low infrastructure areas. Known as a real complementary technology in geographical locations beyond reach of terrestrial means, satellite networks still suffer, in comparison to terrestrial networks, from long delays, scarce bandwidth resources and high equipment costs.
The first DVB norm described a transmission scheme based on MPEG-2 (Motion Picture Expert Group) video compression and transmission schemes, using MPEG-TS (MPEG Transport Stream). This latter was adapted for satellite systems through DVB-S (DVB transmission via Satellite) that defines series of options to send MPEG-TS packets over satellite links and that is nowadays commonly used for Digital TV. The User Satellite Terminals could therefore only receive DVB-S frames from the satellite, but did not have the ability to send any traffic towards the satellite. In 1999, the ETSI (European Telecommunications Standards Institute) proposed a standard for a return channel via satellite, the DVB-RCS, which supplements the STs with the ability to transmit traffic towards the satellite.

1.1.1 Satellite Access Scenario

The satellite access scenario, shown in Figure (1.1) is a typical Satellite Networks Architecture. It consists of a geostationary satellite interconnected to terrestrial stations (ST) network. STs provide single PC or Local Area Network (LAN) with access to network, while Gateways (GTs) allows connection with Internet core network.The satellite network resources are managed by a Network Control Center (NCC). The uplink access from each ST is managed through DVB-RCS interface, whereas transmissions form GTs are implemented through DVB-S (DVB-Satellite) interfaces.

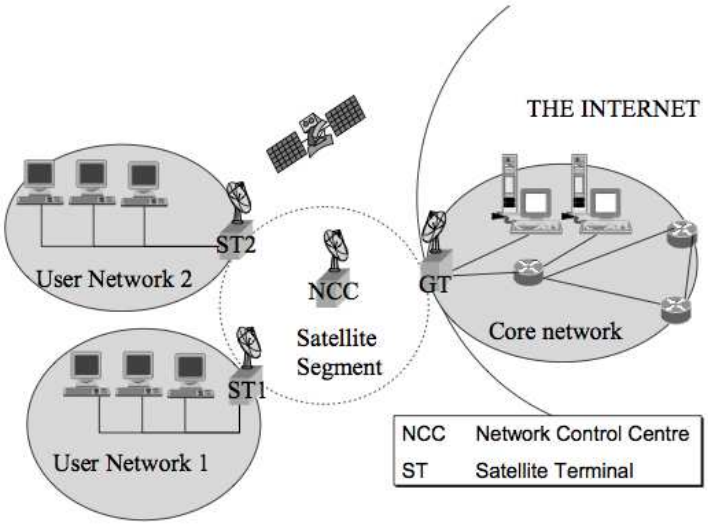


Figure 1.1: Satellite Access Scenario

1.1.2 QoS Architecture

This work deals with a part of the ST, more precisely, it contributes to the QoS management of the return link. The QoS architecture is shown Figure (1.2). This structure is a basic QoS structure, however, in classical networks, the QoS problem is not fundamental thanks to the high link capacity. In Satellite Networks QoS is a central problem, due to the satellite delay and the scarcee uplink resources. Instead of over-sizing the connection, which will be very expensive, we try to reach an optimal exploitation of uplink resuources [8].
The QoS is managed in the satellite systems [1] and especially in the ST at two levels : MAC[1] (Medium Access Control) and IP(Internet Protocol).

MAC Layer

The MAC layer controls the access to the medium (i.e. air in a satellite network). The mechanisms at the MAC layer have as objective to maximize the resources utilization between different STs while offering a good QoS using a Bandwidth on Demand (DAMA) protocol.
Two classes of services are implemented at the MAC layer in the ST:

- *DVB-RT (Real-Time)*: dedicated to applications with high temporal constraints (VoIP)
- *DVB-NRT (Non Real-Time)*: dedicated to more tolerant applications, or even not affected by delay. (Peer to Peer, FTP...)

IP Layer

The QoS architecture proposed at the IP layer divided the traffic in three classes of service

[1]MAC protocol provides addressing and channel access control mechanisms that make it possible for several terminals or network nodes to communicate within a multipoint network, typically a local area network (LAN).

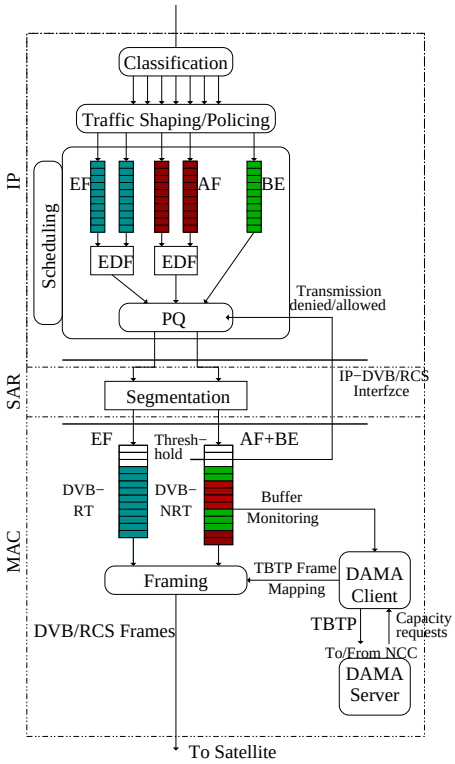


Figure 1.2: ST QoS architecture

- *BE (Best-Effort)*: guaranteeing nothing
- *AF (Assured Forwarding)*: ensuring a relative QoS
- *EF (Expedite Forwarding)*: guaranteeing a total QoS

The EF and AF traffics are controlled by EDF (Earliest Deadline First) scheduler, while BE traffic is stored in simple FIFO (First In First Out) queue. The three categories are then served by PQ (Priority Queueing) Scheduler. The EF traffic is mapped on MAC DVB-RT queue, instead of the AF and BF traffic shared the MAC DVB-NRT queue. IP packets are extracted towards MAC layer as follows :

- EF packets tidy up through EDF scheduler are served as long as the EDF queue is full.
- AF packets are then served and oriented to the MAC DVB-NRT queue.
- Lastly BE packets are directly introduced in the MAC DVB-NRT queue when AF and EF queue are entirely empty.

Remark 1.1 *This is a simplified view of the scheduler, and it could vary according to the system architecture.*

1.2 Objectives

In this work we will consider two kinds of traffic protocols : the TCP (Transmission Control Protocol) that we will introduce in the next part, this protocol is used for the NRT

applications (data streams), and the UDP (User Datagram Protocol) which uses a simple transmission model without explicit hand-shaking dialogues for guaranteeing reliability, ordering, or data integrity. Thus, UDP provides an unreliable service and datagrams may arrive out of order, appear duplicated, or go missing without notice. UDP assumes that error checking and correction is either not necessary or performed in the application, avoiding the overhead of such processing at the network interface level. Time-sensitive applications often use UDP because dropping packets is preferable to use delayed packets. Generally, the data applications are stored in the BE and AF queue while multimedia applications in the EF and AF queue. Consequently, AF queue receives data and multimedia applications which present a problem. In other words, TCP and UDP flows come in the same queue. When congestion occurs packets are lost and impact the TCP and UDP connections. TCP connections reduce their throughput and UDP (multimedia) connections experience a lower quality.
The goal of this work is to regulate a priori the AF queue and avoid queue over-flooding. Therefore, it is necessary to :

- keep place in the buffer for UDP packets in priority, as TCP packets can be retransmit contrary to UDP.
- reduce the buffer size, while keeping reasonable transmission capacity, i.e. having as few losses as possible (and then protect UDP packets).

Hence we have to model the AF queue, and regulate it in order to respect the conditions cited above.

1.3 TCP

TCP means "Transmission Control Protocol". It is an "end-to-end" communication protocol, which means that a direct link between the source and the destination is establised. The main characteristic of this protocol is to verify the data reception by the receiver, using mechanism based on acknowledgement. If a packet is lost, the sender should send it again. Thus TCP assures the transmission of the entire information.
TCP is a general purpose protocol, and does not make assumption on the network used. To find the maximum transmission throughput, TCP probes the network until reaching the limit. This is the role of slow start and congestion avoidance mechanisms.

1.3.1 Definitions

For a better understanding of this report, some definitions are necessary.

- The Acknowledgement is a message sent by the receiver to inform the sender that the packets sent are achieved. If the sender did not receive the acknowledgement, this means that the packet is lost, then the sender has to send the packet again (see Figure (1.3)).
- The Timeout corresponds to the waiting time of an acknowledgement. The timer starts when the packet is sent. When the time is elapsed, the sender should send the same packet back. One supposes that the packet is lost (see Figure (1.3)).
- The Buffer is a random access memory area used to store temporarily the data. When the packets reach the receiver, they are stored in the buffer to be treated.

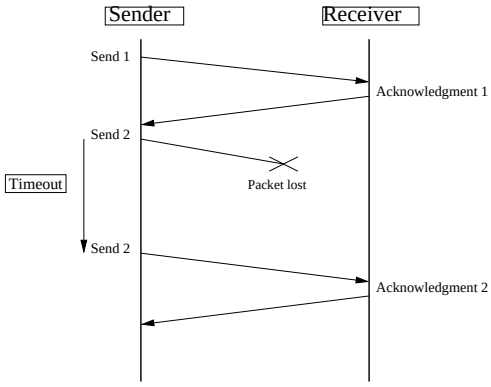


Figure 1.3: Timeout occurrence

- Congestion is a saturation phenomena. It occurs when the traffic is too heavy: for example, when the buffer receives more packets than its capacity, or when the data transit from a high capacity network to a lower capacity network. Congestion leads to data losses.
- Congestion Window is a TCP state variable that limits the amount of data that a TCP can send.

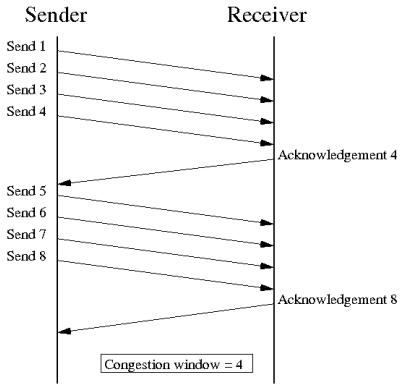


Figure 1.4: Congestion window principle

- The Round Trip Time (RTT) is the time needed by a packet to reach the destination and the come back time of the acknowledegment of this packet. It corresponds to the exchange time of a segment between the sender and the receiver.
- Three duplicate acknowledgement occurs, when a packet is lost during the transmission, the following packets will trigger at the receiver side the sending of an acknowledgement with the number of the correct last packet received. Then the packet sequence number of the just before packet lost, will be sent until it is retransmitted and received. When the same acknowlegement has been received three times by the source, it resends the lost packet (see Figure (1.5)).

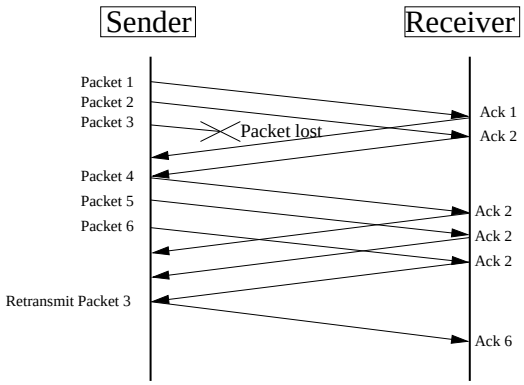


Figure 1.5: Three duplicate acknowlegement principle

1.3.2 Congestion avoidance algorithm

Congestion avoidance algorithm has been developed to regulate the flow rate as close as possible to the "transmission limits" in order to transmit a maximum of information and avoid network congestion. The basic hypothesis of this algorithm is to consider that a packet lost is equivalent to congestion. The principle of the algorithm is to control the rate of each source as a function of traffic state. The principle is simple, each source increases progressively their output flow. This increase takes place until a packet loss occurs. This means that congestion is detected somewhere in the network. Thus the flow is decreased enough in order to go out of the congestion state. The following sub-section gives an outline of the Additive Increase Multiplicative Decrease (AIMD) algorithm implemented in TCP.

Algorithm

- The source sends W packets.
- The receiver acquits the received segment and the source acts in consequence:
  - If the flux is transmitted with success, the source increases its size: $W \leftarrow W+1$
  - If there is a loss, the source should retransmit its data and reduce its congestion window. There is principally two kinds of loss identification, indications by Timeout (TO) and indications by duplicate acknowledgement (3DupAck)
    - If the source did not receive the acknowledgement, TO: $W \leftarrow 1$.
    - If the source receives three duplicate acknowledgement, 3DubAck: $W \leftarrow W/2$.

The time of one exchange corresponds to a way return, i.e. one RTT (Round Trip Time).

Remark 1.2 *Improvements:*
To improve the efficiency of the protocol, some algorithms have been added to create different versions of TCP (Tahoe, Reno, Vegas, New Reno, Santa Cruz...), New Reno being the most used currently.
Some improvements (see Figure (1.7)):

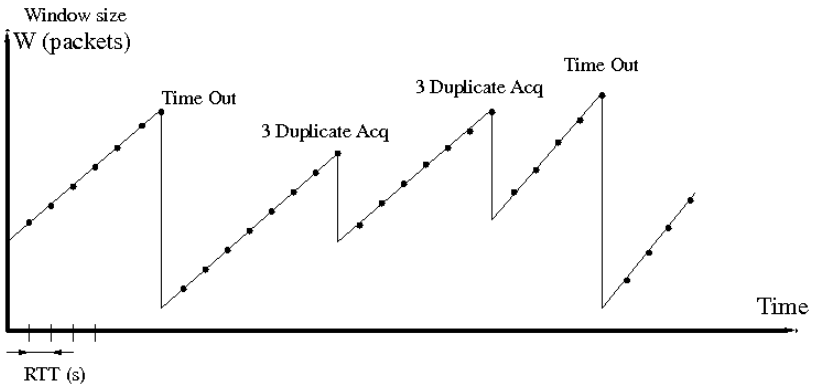


Figure 1.6: Example of congestion window W

- *Slow-Start Algorithm [10]: The goal is to obtain a threshold congestion estimation. Therefore the congestion window size grows exponentially until a loss is detected (initialization). On detecting a loss, the source sets the slow-start threshold (SSThresh) to half of the current window size, retransmits the lost packet, and re-enters slow-start by reseting its windows to one.*
- *Fast Retransmit Algorithm [10]: The goal is to recover from loss more efficiently. The receiver indicates with an acknowledgement its position. The packets being able to arrive in a mess, it is normal that the source received duplicate acknowledgement. We consider that on reception of three duplicate acknowledgement, the link is congested. Instead of waiting the Timeout, which is long, the source reduces is flow, and retransmits the missing informations.*
- *Fast Recovery Algorithm [10]: If a loss due to duplicate acknowledgement occurs, the slow-start set in motion, we use the fast recovery. The window is reduced to half of the threshold plus three time the segment size, and reach directly the congestion avoidance phase (linear phase).*

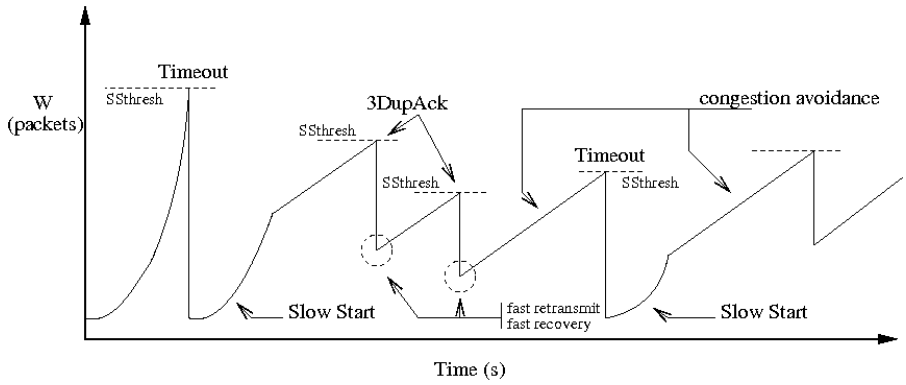


Figure 1.7: Slow-start, fast retransmit and fast recovery illustration

1.3.3 AQM (Active queue management)

The main principle is to drop intentionally TCP packets before the router queue becomes full so that the source can prevent the congestion, by reacting to the losses with the con-

gestion avoidance mechanism. The goal is to optimize the data transmission maintaining a high stream at the buffer level. AQM detects network congestion, packet losses of incipient congestion, and informs traffic sources. Sources react and decrease the congestion window to avoid buffer saturation. This way to prevent router congestion is an active research subject, see for example [3] and references therein. We propose in this work an AQM to enhance IP QoS on the return channel.
AQMs have also the objectives to insure equity between each source.
In the state of art, we can cite the following AQMs which have been developed : The RED (Random Early Detection) [7] is one of the first AQM. It introduces probabilistic early packet dropping to avoid the full queue phenomena. However, RED tuning is hard, and it has a very high network configuration sensitivity. Following RED, many variants such as SRED (Stabilized-RED), ARED (Adaptive-RED) and DRED (Dynamic-RED) have been proposed. They are auto-tuned, and adapted to networks [18]. It exists likewise the ECN (Explicit Congestion Notification) [16] using the same principle as RED. But instead of dropping the packet, a bit called ECN in the acknowledgement is set to "1" to explicitly inform an imminent congestion the source. The advantage is to avoid the packet loss and the retransmissions are not needed.

1.3.4 Modeling

Following the algorithm developed in the paragraph (1.3.2) we clearly recognize that the overall system is an interconnected feedback system as described by the Figure (1.8). Thus feedback control principle appears to be an appropriate tool for the analysis and design of AQM strategies.

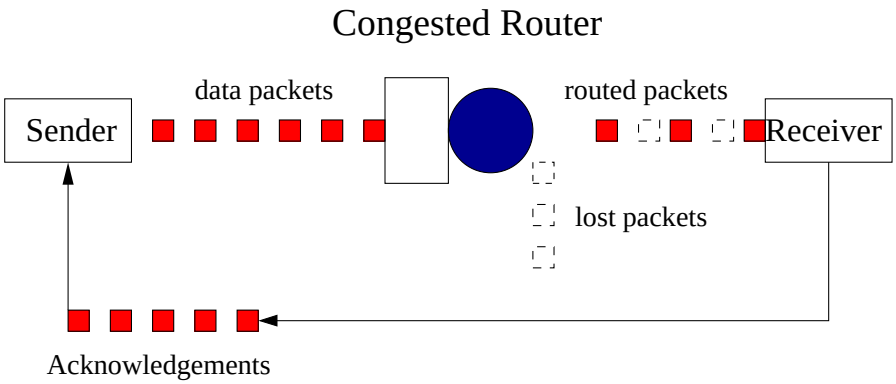


Figure 1.8: The considered system

1.3.4.1 A fluid-flow model of TCP behavior

In order to use control theory, we propose to introduce a mathematic model of the TCP behavior which has been developed in [15] and [14]. This model is based on two assumptions:

- The traffic is considered fluid-flow
- The losses are described by a Poisson Process

That means for example, when a loss occurs, all of the TCP connections react to the loss. Of course, considering a single loss, the hypothesis is improper, but considering multiple losses and because TCP has been designed to be fair, this hypothesis becomes acceptable. This is designed to guarantee the equity between TCP connections and modify the windows size of each sources simultaneously. The first assumption implies that the congestion window increases in a continuous way instead of step increase. It increases by one every RTT and hence the continuous increase is represented as dt/RTT.
The second assumption models the packets loss occurrences. We assume idealized behavior, i.e. we model the losses as Poisson streams.
Then the evolution of the congestion window size W can be described as follows:

$$dW(t) = \frac{dt}{RTT} - \frac{W(t)}{2} dN(t) \tag{1.1}$$

by noting that $dN(t)$ is defined as:

$$dN = \begin{cases} 1, \text{ if a loss occurs} \\ 0, \text{ otherwise} \end{cases} \tag{1.2}$$

This equation reflects the "Additive Increase Multiplicative Decrease" aspect of TCP. The first term corresponds to the additive increase part, which states that the windows size will increase by one every RTT. The second term corresponds to the multiplicative decrease part, which halves the window size for each arrival of a loss. Note that we use a simplified model, which ignores the TCP slow start mechanism that starts at the beginning of a connection, and timeouts. Effectively, some measures realized on Ourses-Project [17] using a DVB-S2/RCS system using a Ka-Band link, show that timeouts barely occur, and TCP works most of the time as congestion avoidance instead of a slow start only at the beginning of the connection.

1.3.4.2 The system

Using stochastic differential analysis of the equation (1.1), and considering the simplifications cited above, [15] have developed a dynamic model of the TCP behavior. In this model, we consider a system in which there is a single congested router with a transmission capacity of C. Associated with this router is an AQM that is characterized by a packet discard function p(·) that takes as its argument an estimate of the average queue length at the router, and the average congestion window size. The proposed model from [12] is then of the form:

$$\begin{cases} \dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)}{2} \frac{W(t-R(t))}{R(t-R(t))} p(t - R(t)) \\ \dot{q}(t) = -C + \frac{N(t)}{R(t)} W(t) \end{cases} \tag{1.3}$$

where $\dot{x}$ denotes the time-derivative and
$W \doteq$ average TCP window size (packets);
$q \doteq$ average queue length (packets);
$R(t) \doteq$round-trip-time $= \frac{q(t)}{C} + T_p$(secs);
$C \doteq$link capacity (packets/sec);
$T_p \doteq$ propagation delay (secs);
$N \doteq$ load factor (number of TCP sessions);
$p \doteq$ probability of packet mark, which takes values only in $[0, 1]$.

1.3.4.3 Linearization

The dynamic TCP behavior is thus modeled by a non-linear time delay systems which can be complicated to analyse from a control theory point of view. That is the reason why we are only interested in the design of an AQM around an equilibrium point (W_0, q_0, p_0). To linearize model (1.3) we first assume that the number of TCP sessions and the link capacity are constant i.e., $N(t) \equiv N$ and $C(t) \equiv C$. Taking (W, q) as the state and p as an input, the operating point (W_0, q_0, p_0) is then defined by $\dot{W} = 0$ and $\dot{q} = 0$ so that

$$\begin{cases} \dot{W} = 0 \;\Rightarrow\; W_0^2 p_0 = 2 \\ \dot{q} = 0 \;\Rightarrow\; W_0 = \frac{R_0 C}{N},\; R_0 = \frac{q_0}{C} + T_p \end{cases} \tag{1.4}$$

The important point of this work is the propagation delay. The operating point are then defined such that R_0 is as small as possible all in keeping a suitable queue size q_0. From this point, we compute W_0 and p_0. Moreover, we ignore the dependance of the time-delay argument $t - R$ on the queue-length q, and assume it fixed to $t - R_0$. The delay R_0 will be denoted by the letter h.
Given the vector of network parameters $\eta \doteq (N, C, T_p)$, we define the set of *feasible operating points* Ω_n by

$$\Omega_n = \{(W_0, q_0, p_0) : W_0 \in (0, \bar{W}), q_0 \in (0, \bar{q}), p_0 \in (0, 1) \text{ and (1.4) satisfied}\} \tag{1.5}$$

We obtain finally the linearized model (1.6) around equilibrium point defined by (1.4): (see appendix A.1):

$$\begin{cases} \delta\dot{W}(t) = -\dfrac{N}{R_0^2 C}\Big(\delta W(t) + \delta W(t-h)\Big) - \dfrac{1}{R_0^2 C}\Big(\delta q(t) - \delta q(t-h)\Big) - \dfrac{R_0 C^2}{2N^2}\delta p(t-h) \\ \delta\dot{q}(t) = \dfrac{N}{R_0}\delta W(t) - \dfrac{1}{R_0}\delta q(t) \end{cases} \tag{1.6}$$

where $\delta W \doteq W - W_0$, $\delta q \doteq q - q_0$ represent the state variables and $\delta p \doteq p - p_0$ the input. This can be rewritten as (1.7):

$$\begin{bmatrix} \delta\dot{W}(t) \\ \delta\dot{q}(t) \end{bmatrix} = \begin{bmatrix} -\frac{N}{R_0^2 C} & -\frac{1}{C R_0^2} \\ \frac{N}{R_0} & -\frac{1}{R_0} \end{bmatrix} \begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix} + \begin{bmatrix} -\frac{N}{R_0^2 C} & \frac{1}{C R_0^2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta W(t-h) \\ \delta q(t-h) \end{bmatrix} + \begin{bmatrix} -\frac{C^2 R_0}{2N^2} \\ 0 \end{bmatrix} \delta p(t-h) \tag{1.7}$$

where δW and δq are the state variables and δp the input.

Chapter 2

AQM Design for Satellite Access

This chapter treats two different AQMs control strategies to have the anticipatory congestion detection and control capability but also achieve satisfactory control performance in terms of the queue length dynamics (or equivalently delay). The first method uses the well know Proportional Integral (PI) feedback control adapted to the TCP network in [12]. The second method is a predictive controller introduced in [13] for a system with control delay that we have adapted to TCP network.

2.1 Objectives

The objective of this part is to build two robust controllers which will stabilize the delayed system. Three points are important for the AQM performance: efficient queue utilization and queueing delay, stabilization of a delayed system and robustness.

2.1.1 Efficient queue utilization

For efficient use, the queue should avoid overflow or emptiness. The former situation results in lost packets and undesired retransmissions, while an empty buffer underutilizes the link. Both of these extremes should be avoided in both transient and steady-state operation.
The time required for a data packet to be serviced by the routing queue is called the queueing delay and is equal to q/C. This time, together with the propagation delay T_p, accounts for the network's delay and it is desirable to keep small both the queueing delay and its variations. This calls for regulating to small lengths. However, doing so may result in link underutilization and this limitation presents a fundamental tradeoff to AQM design.

2.1.2 Robustness

AQM schemes need to maintain closed-loop performances in spite of varying network conditions. These conditions include variations in the number of TCP sessions N, and variations in the propagation delay T_p and link capacity C.
In this approach, we consider the delay effect as a perturbation of the nominal dynamics. The objective is to have a robust stability in relation of the delay. The method consists in dividing the system in two parts in order to get a LFT form (Linear Fractional Transformation) (see Figure (2.1)) where Δ represents the uncertainties and Σ the new considered nominal system.

The important question is Robustness, i.e., what model error Δ can be allowed without

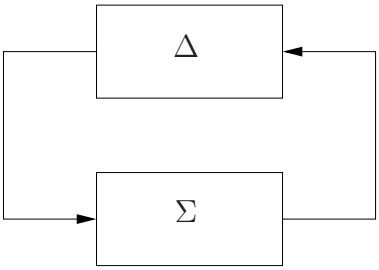


Figure 2.1: LFT form for robust approach

endangering the stability in the closed loop system.
To investigate this, we need to find the representation in Figure (2.1), i.e., the blocks Δ and Σ.

2.2 An LFT representation

In order to use the robust framework from the control theory, we recall the linearized state space model from chapter 1:

$$\begin{bmatrix} \delta\dot{W}(t) \\ \delta\dot{q}(t) \end{bmatrix} = \begin{bmatrix} -\frac{N}{R_0^2C} & -\frac{1}{CR_0^2} \\ \frac{N}{R_0} & -\frac{1}{R_0} \end{bmatrix} \begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix} + \begin{bmatrix} -\frac{N}{R_0^2C} & \frac{1}{CR_0^2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta W(t-h) \\ \delta q(t-h) \end{bmatrix} + \begin{bmatrix} -\frac{C^2R_0}{2N^2} \\ 0 \end{bmatrix} \delta p(t-h) \tag{2.1}$$

From the model (2.1), we separate the nominal dynamic and the uncertainties :

$$\begin{aligned} \begin{bmatrix} \delta\dot{W}(t) \\ \delta\dot{q}(t) \end{bmatrix} &= \begin{bmatrix} -\frac{2N}{R_0^2C} & 0 \\ \frac{N}{R_0} & -\frac{1}{R_0} \end{bmatrix} \begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix} + \begin{bmatrix} -\frac{C^2R_0}{2N^2} \\ 0 \end{bmatrix} \delta p(t-h) \\ &+ \begin{bmatrix} \frac{N}{R_0^2C} & -\frac{1}{CR_0^2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix} + \begin{bmatrix} -\frac{N}{R_0^2C} & \frac{1}{CR_0^2} \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta W(t-h) \\ \delta q(t-h) \end{bmatrix} \end{aligned} \tag{2.2}$$

Using $x(t)\hat{=}\begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix}$ and $D_h v(t) = v(t-h)$ (delay operator), this can be rewritten as

$$\dot{x}(t) = Ax(t) + Bw_1(t) + Gw_2(t) \tag{2.3}$$

with $w_1(t) = D_h\delta p(t)$,$w_2(t) = (1 - D_h)x(t)$
$A = \begin{bmatrix} -\frac{2N}{R_0^2C} & 0 \\ \frac{N}{R_0} & -\frac{1}{R_0} \end{bmatrix}$, $B = \begin{bmatrix} -\frac{C^2R_0}{2N^2} \\ 0 \end{bmatrix}$ and $G = \begin{bmatrix} \frac{N}{R_0^2C} & -\frac{1}{CR_0^2} \\ 0 & 0 \end{bmatrix}$

Figure (2.2) gives a representation of the linearized AQM control system using state-space model.

Now we switch to transfer function by Laplace transforming (2.3)

$$\begin{gathered} sX(s) = AX(s) + BW_1(s) + G\Delta_h(s)X(s); \Delta_h(s)\hat{=}1 - e^{-sR_0} \\ \delta Q(s) = [0\ 1]X(s)\hat{=}CX(s) \\ \frac{\delta Q(s)}{W_1(s)} = C[sI - A - G\Delta_h(s)]^{-1}B \end{gathered} \tag{2.4}$$

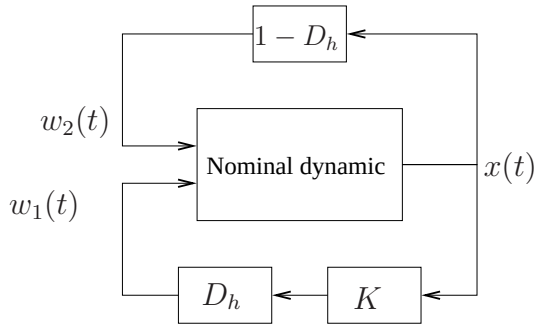


Figure 2.2: Block diagram of a linearized AQM control system

If we compute $\frac{\delta Q(s)}{W_1(s)}$, we obtain:

$$\frac{\delta Q(s)}{W_1(s)} = \frac{\frac{-N}{R_0}.\frac{C^2R_0}{2N^2}}{(s+\frac{2N}{CR_0^2})(s+\frac{1}{R_0}) - \frac{N}{CR_0^2}.s\Delta_h} = -\frac{P(s)}{1-P(s)\Delta(s)} \tag{2.5}$$

Where the transfer function $P(s)$ in (2.6), relates how the packet-marking probability dynamically affects the queue length.

$$P(s) = \frac{\frac{C^2}{2N}}{\left(s+\frac{2N}{R_0^2C}\right)\left(s+\frac{1}{R_0}\right)} \tag{2.6}$$

The transfer function $\Delta(s)$ in (2.7), relates the uncertainties.

$$\Delta(s) = \frac{2N^2s}{R_0^2C^3}(1-e^{-sR_0}) \tag{2.7}$$

Figure (2.3) shows a feedback control system depiction of AQM using transfer function.

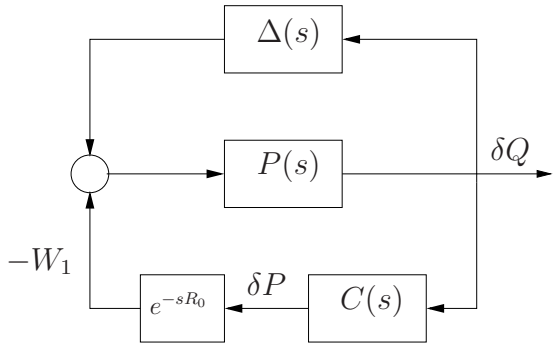


Figure 2.3: AQM as feedback control

Stabilizing AQM control law

$C(s)$ is the control law (see Figure (2.3)). Closed-Loop stability is fundamental in order to satisfy the performance objectives. Reference [12] gives the condition for stabilization which amounts to $C(s)$ stabilizing the delayed nominal plant $P(s)e^{-sR_0}$and gain-stabilizing the uncertainties $\Delta(s)$. In the following proposition, we require the transfer

function

$$V(s) = \frac{P(s)}{1 + P(s)C(s)e^{-sR_0}} \tag{2.8}$$

Proposition 2.1 *[12] Given the feasible network parameters $\eta = (N, C, T_p)$ and operating point (W_0, q_0, p_0) in Ω_η, the linearized AQM control system, illustrated in (2.3), is stable if*

i) *C(s) stabilizes the delayed nominal plant $P(s)e^{-sR_0}$;*

ii) *the uncertainties $\Delta(s)$ is gain stabilized, i.e., $|\Delta(j\omega)V(j\omega)| < 1; \forall\omega > 0$.*

Proof 2.1 *If $C(s)$ stabilizes the delayed nominal plant $P(s)e^{-sR_0}$, then V(s) is stable. Since $\Delta(s)V(s)$ is stable, the small gain condition $|\Delta(j\omega)V(j\omega)| < 1$, together with the Nyquist stability criterion implies closed loop stability.*

2.3 Control design

For the study of both controllers, we assume that the RTT is constant.

2.3.1 First case: PI controller

In this section, we recall the basic way to ensure the closed loop stability by designing a classical PI controller adapted to the system of the form (2.8). The controller $C(s)$ is defined as:

$$C(s) = K_{PI}\frac{\frac{s}{z} + 1}{s} \tag{2.9}$$

We denote $L(s)$ as the open-loop transfer function of the model (2.8)

$$L(s) = \frac{\frac{K_{PI}C^2}{2N}\left(\frac{s}{z} + 1\right)e^{-sR_0}}{s\left(s + \frac{2N}{R_0^2C}\right)\left(s + \frac{1}{R_0}\right)} \tag{2.10}$$

We define z such that the dominant pole is cancelled:

$$z = \frac{2N}{R_0^2C} \text{ or } z = \frac{1}{R_0} \tag{2.11}$$

Then, we take the loop's unity gain crossover frequency as

$$\omega_g = \frac{\beta}{R_0} \tag{2.12}$$

where β is chosen to set the phase margin. K_{PI} is chosen such that $|L(j\omega)|$=1

$$K_{PI} = \omega_g z \left| \frac{j\omega_g + \frac{1}{R_0}}{\frac{C^2}{2N}} \right| \tag{2.13}$$

We then calculate the desired phase loop by choosing β which lead to a positive phase margin.

$$\angle L(j\omega_g) = \angle\frac{e^{-j\omega_g R_0}}{j\omega_g.\left(j\omega_g + \frac{1}{R_0}\right)} = -90 - \frac{180}{\pi}\omega_g R_0 - arctan(w_g R_0) = -90 - \frac{180}{\pi}\beta - arctan(\beta) \tag{2.14}$$

Summary 2.1 *The steps to design the PI controller are:*

- *Define β such that the phase margin is positive (2.14)*
- *From β we can calculate the crossover frequency (2.12)*
- *z should cancel the dominant pole (2.11)*
- *ω_g and z being defined, we can calculate K_{PI} (2.13)*
- *Finally, we have all the parameters to calculate the controller (2.9)*

2.3.2 Second Case: Predictive Controller

Finite spectrum Assignment problem for systems with delays [13]

In the last subsection, a classical controller originally designed for finite dimensional systems is proposed to ensure the closed-loop stability of a time delay system. Nevertheless, the presence of delay could inevitably lead to performance degradation. That is the reason why a more interesting way to control such system is to introduce an infinite dimensional controller. In this subsection we propose to design a predictive controller to control the nominal system defined by the state space model (2.15):

$$\dot{x}(t) = Ax(t) + Bu(t-h) \tag{2.15}$$

where $x(t) = \begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix}$ denotes the state, $u(t) = \delta p(t)$ the input, $A = \begin{bmatrix} -\frac{2N}{R_0^2 C} & 0 \\ \frac{N}{R_0} & -\frac{1}{R_0} \end{bmatrix}$, $B = \begin{bmatrix} -\frac{C^2 R_0}{2N^2} \\ 0 \end{bmatrix}$. Finally the delay is denoted by h.
This state space is similar to the state space model (2.3), where the uncertainties are not connected to the nominal dynamic i.e., $Gw_2(t)$ is removed.
Then we look for a control law of the form:

$$u(t) = Fx(t+h). \tag{2.16}$$

which ensures the closed-loop stability of the nominal system defined by:

$$\dot{x}(t) = (A + BF)x(t) \tag{2.17}$$

This last equation shows that the problem of finding the matrix parameter F is then reduced to a classical finite spectrum assigment problem and could be easily solved using classical tools from Matlab toolboxes. Nevertheless, if the solution appears to be convenient in terms of stability, performances and calculus, it remains that, apparently, the proposed implemented controller (2.16) is not causal. The Prediction is performed by noting that:

$$x(t+h) = e^{Ah}x(t) + \int_0^h e^{A\theta}Bu(t-\theta)\,d\theta. \tag{2.18}$$

The control law is then reduced to:

$$u(t) = Fx(t+h) = F\left(e^{Ah}x(t) + \int_0^h e^{A\theta}Bu(t-\theta)\,d\theta\right). \tag{2.19}$$

which appears to be a causal function. We can rewrite the controller as follow:

$$u(t) = F_x x(t) + \int_0^h F_u(\theta) u(t-\theta)\, d\theta. \tag{2.20}$$

Where $F_x = Fe^{Ah}$ and $F_u(\theta) = Fe^{A\theta}B$.

Implementation of the controller

In order to implement the controller in C++, we have to rewrite it in a different way. The simulation on Matlab/Simulink could be easily implementable than the following way, but in order to use the same simulation with the two software, we will realize the Matlab simulations using this way.

If A-matrix has two distinct real eigenvalues λ_1 and λ_2, then the A-matrix is diagonalizable and $P^{-1}AP = D = diag(\lambda_1, \lambda_2)$

So $A = PDP^{-1}$ then $e^{A\theta} = Pe^{D\theta}P^{-1} = P \begin{bmatrix} e^{\lambda_1\theta} & 0 \\ 0 & e^{\lambda_2\theta} \end{bmatrix} P^{-}1$

Thus $F_u(\theta) = Fe^{A\theta}B = FPe^{D\theta}P^{-1}B$

with $F = \begin{bmatrix} f_1 & f_2 \end{bmatrix}$, $P = \begin{bmatrix} p_{11} & p_{12} \\ p_{21} & p_{22} \end{bmatrix}$, $P^{-1} = \begin{bmatrix} pinv_{11} & pinv_{12} \\ pinv_{21} & pinv_{22} \end{bmatrix}$ and $B = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$

Therefore

$$F_u(\theta) = C_1 e^{\lambda_1\theta} + C_2 e^{\lambda_2\theta} \tag{2.21}$$

Where $C_1 = (f_1.p_{11} + f_2.p_{21})(p_{inv11}.b_1 + p_{inv12}.b_2)$
and $C_2 = (f_1.p_{12} + f_2.p_{22})(p_{inv21}.b_1 + p_{inv22}.b_2)$
Now we can rewrite the integral of (2.20)

$$\int_0^h F_u(\theta) u(t-\theta)\, d\theta = \int_0^t F_u(t-\theta) u(\theta)\, d\theta - \int_0^{t-h} F_u(t-\theta) u(\theta)\, d\theta. \tag{2.22}$$

Using (2.20), (2.21) and (2.22), the predictive controller become:

$$\begin{array}{c} u(t) = F_x x(t) + C_1 e^{\lambda_1 t} \left[\int_0^t e^{-\lambda_1\theta} u(\theta)\, d\theta - \int_0^{t-h} e^{-\lambda_1\theta} u(\theta)\, d\theta \right] \\ + C_2 e^{\lambda_2 t} \left[\int_0^t e^{-\lambda_2\theta} u(\theta)\, d\theta - \int_0^{t-h} e^{-\lambda_2\theta} u(\theta)\, d\theta \right] \end{array} \tag{2.23}$$

This form of the controller is implementable with simulations software.

Addition of an integral action

In order to cancel the steady-state error in the predictive controller, we add an integral action to the state feedback law. The principle is to add an integrator in the direct chain as shown in Figure (2.4):

The closed-loop equations have the form:

$$\begin{cases} \dot{x}(t) = Ax(t) + Bu(t-h) \\ y(t) = Cx(t) \\ \dot{x}_i(t) = y(t) \end{cases} \tag{2.24}$$

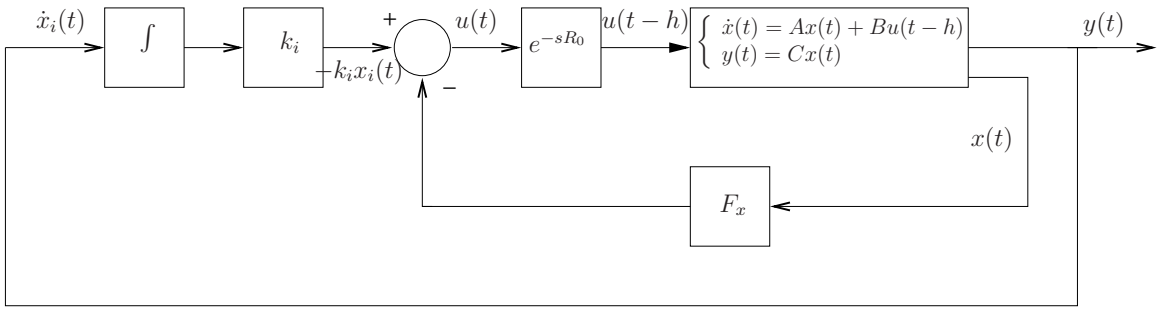


Figure 2.4: State feedback and integral action

This gives the following augmented system:

$$\begin{bmatrix} \dot{x}(t) \\ \dot{x}_i(t) \end{bmatrix} = \begin{bmatrix} A & 0 \\ C & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ x_i(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} u(t-h) \tag{2.25}$$

Or taking $x_a(t) = \begin{bmatrix} x(t) \\ x_i(t) \end{bmatrix}$, we obtain the following augmented state space model which is similar to the state space model (2.15):

$$\dot{x_a}(t) = \hat{A}x_a(t) + \hat{B}u(t-h) \tag{2.26}$$

So we can apply the same control law as previously to the augmented model:

$$u(t) = \tilde{F}x_a(t+h) = \tilde{F}_x x_a(t) + \int_0^h \tilde{F}_u(\theta)u(t-\theta)\,d\theta \tag{2.27}$$

where $\tilde{F} = \begin{bmatrix} F_x & k_i \end{bmatrix}$, $\tilde{F}_x = \tilde{F}e^{\hat{A}h}$ and $\tilde{F}_u = \tilde{F}e^{\hat{A}\theta}\hat{B}$
This lead to a system of the third order, with an input delay.
The problem of a state feedback with an integral action is equivalent to a classical state feedback on an augmented system $(\hat{A}, \hat{B}, \hat{C}, 0)$
In order to make the simulation of the model with the integral action, we make the same transformation than in sub-section (2.3.2), so the predictive control law used for the simulations is:

$$\begin{aligned} u(t) = \tilde{F}_x x_a(t) + C_1 e^{\lambda_1 t}\left[\int_0^t e^{-\lambda_1\theta}u(\theta)\,d\theta - \int_0^{t-h} e^{-\lambda_1\theta}u(\theta)\,d\theta\right] \\ + C_2 e^{\lambda_2 t}\left[\int_0^t e^{-\lambda_2\theta}u(\theta)\,d\theta - \int_0^{t-h} e^{-\lambda_2\theta}u(\theta)\,d\theta\right] + C_3 e^{\lambda_3 t}\left[\int_0^t e^{-\lambda_3\theta}u(\theta)\,d\theta - \int_0^{t-h} e^{-\lambda_3\theta}u(\theta)\,d\theta\right] \end{aligned} \tag{2.28}$$

Chapter 3

Results and Applications

The controllers developed in the previous chapter are now implementable. In this part, we will present the results of both controllers though Matlab/Simulink and NS2-Simulator, and show that the integral part is a future improvement to be implemented. These simulations will validate the behavior of the system using the satellite network that we consider.

3.1 Network topology

We consider the network topology consisting of 6 TCP sources and 1 UDP source, with the same propagation delay connected to a destination node through a router (see Figure (3.1)). All these sources together arrive to the satellite connection, which has a bigger propagation delay and a smaller link capacity, where the phenomenon of congestion collapse appears. It is then necessary at this node to control the stream with the help of an AQM.

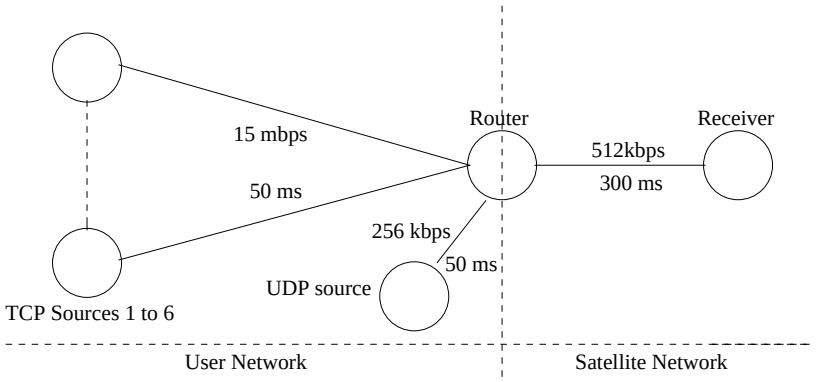


Figure 3.1: The network topology

The simulations are realized with Matlab/Simulink and NS2-simulator [2] and [6]. We have adapted the parameters to the conditions of the satellite network, i.e. the link capacity $C = 128$ packets/sec[1], the propagation delay $T_p = 0.7$ secs[2] and the number of sources $N = 6$ sources. The operating point is chosen such that the average queue length $q_0 = 35$ packets. Using the equations (1.4) we obtain a RTT $R_0 = 0.9734$ secs, an average

[1]corresponds to a 0.512 Mb/s link with an average packet size 500 bytes

[2]corresponds to a return from the source to the receiver, i.e. $(50ms + 300ms) \times 2$

TCP window size $W_0 = 20.77$ packets and a probability of packet mark $p_0 = 0.0046$. The transfer functions $P(j\omega)$ and $\Delta(j\omega)$ with these numerical values are:

$$\begin{array}{l} P(j\omega) = \frac{1365}{s^2+1.126s+0.01016} \\ \Delta(j\omega) = 3.623 \times 10^{-05} s(1 - e^{-0.9734s}) \end{array} \tag{3.1}$$

and the state-space model (3.2):

$$\begin{bmatrix} \delta\dot{W}(t) \\ \delta\dot{q}(t) \end{bmatrix} = \begin{bmatrix} -0.0989 & 0 \\ 6.1637 & -1.0273 \end{bmatrix} \begin{bmatrix} \delta W(t) \\ \delta q(t) \end{bmatrix} + \begin{bmatrix} -221.5111 \\ 0 \end{bmatrix} \delta p(t - h(t)) \tag{3.2}$$

The magnitude Bode plots for the transfer function $P(j\omega)$ and $\Delta(j\omega)$ are shown in Figure (3.2). The plot of $P(j\omega)$ reveals the low-pass nature of the TCP-queue dynamics. The frequency response of the uncertainties $|\Delta(j\omega)|$ shows its influences at higher frequencies. Hence the uncertainties will not have influences on the nominal dynamic.

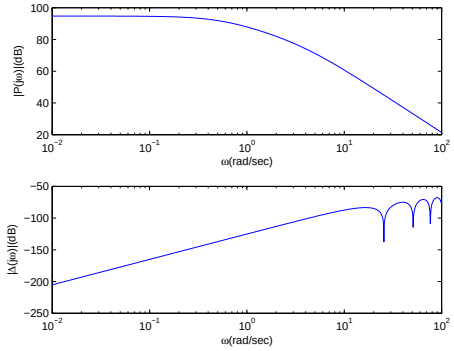


Figure 3.2: Magnitude Bode plots of $P(s)$ and $\Delta(s)$ for TCP loads

3.2 Controllability and stability

3.2.1 PI controller

A plot of $\angle L(j\omega_g)$ versus β in Figure (3.3) shows that values of $\beta \in (0, 1.39)$ give positive phase margins, with margins increasing for decreasing β.

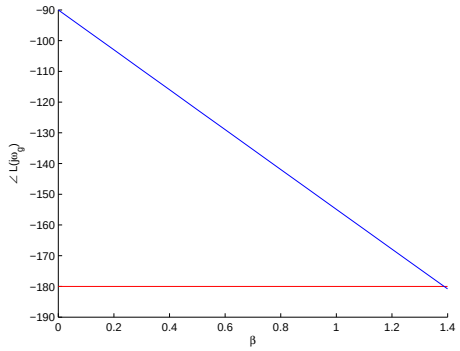
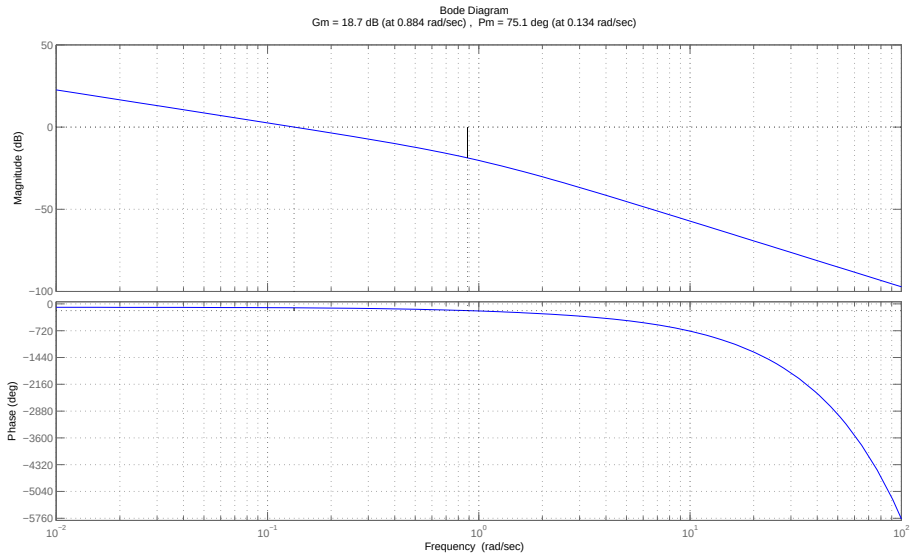


Figure 3.3: Loop phase angle $\angle L(j\omega_g)$ as a function of design parameter β. Positive phase margins occur for $\beta \in (0, 1.39)$.

For the parameters of the satellite network, we choose $\beta = 0.13$, because we have the a larger positive phase margin. This gives $\omega_g = 0.13 rad/sec$ and $K_{PI} = 4.27.10^{-5}$ and we choose z as the dominant pole i.e., $z = \frac{2N}{R_0^2 C} = 0.099$. Hence $C(s) = 4.27.10^{-5} \frac{\left(\frac{s}{0.099}+1\right)}{s}$.



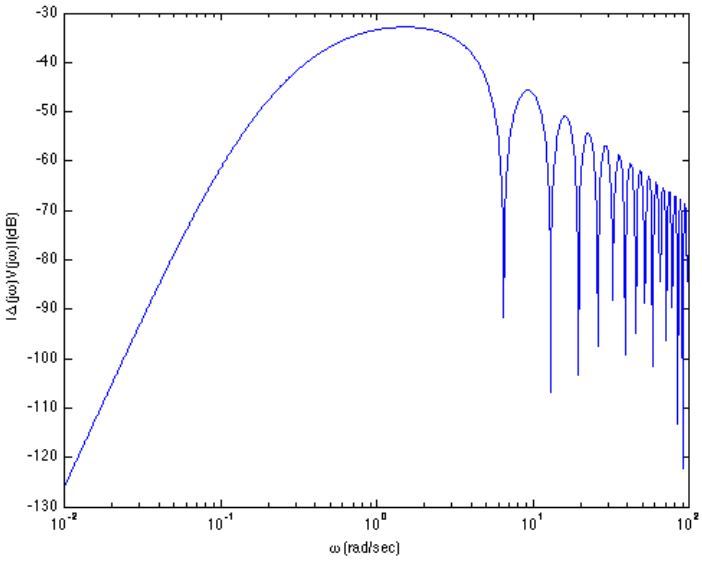


Figure 3.4: Frequency response using PI controller $C(s) = 4.27.10^{-5} \frac{\left(\frac{s}{0.099}+1\right)}{s}$. Frequency response of $L(s)$ shows positive gain and phase margin (left plot). $|\Delta(j\omega)V(j\omega)| < 1$ showing stability on face of uncertainties. (right plot)

For this pairs (z,K_{PI}) the PI compensator stabilizes the nominal plant and enjoys robust stabilization. In Figure (3.4), we give the resulting frequency response of $L(j\omega)$ and $\Delta(j\omega)V(j\omega)$. The PI design has a bandwidth of $0.13 rad/sec$, and the figure shows stability in the presence of uncertainties.

Remark 3.1 *Here is described only one case with a low bandwidth, because different simulations have been realized, with different bandwidth. This configuration gives the best results.*

Remark 3.2 *As with any application of integral control, one should be aware of integrator windup due to control signal saturation. This is a possibility in AQM where the control signal (the packet-marking probability) takes value in* $[0, 1]$

3.2.2 Predictive Controller

The system is continuous, so we have to define two negatives poles. To place the poles, we have to take care of the system speed due to control signal saturation. The further the poles are on the left-hand of the imaginary axis, the faster the system will be. However, if we choose poles to large, the control signal will be saturated. By doing some simulations, we observe that the control signal is not saturated for the pairs of poles $[-0.55 - 0.55]$. The poles have a negative real part so the system is stable. Using the Matlab function **acker**, we define the gain matrix $F = [-0.1184\ 0.1668].10^{-}3$. Then we have to check the controllability matrix of the state space model to be sure that the system is controllable. The system is controllable, if the controllability matrix $S(A, B) = [B\ AB\ A^2B\ ...\ A^{n-1}B]$ has full full rank [9].
For the model (3.2), S(A,B) has full rank (second order system and the rank of S is 2) then the model is controllable. We plot the frequency response and the stability condition (see Figure (3.5)). As shown Figure (3.5), the uncertainties will not cause instability on the nominal plant.

3.3 Matlab/Simulink Simulations

The second step was to implement the simulation on Matlab/Simulink to analyse the behavior of the mathematical model. The simulation is realized with the linearized model, and the non-linear model. We can verify if the linearization is close to the original model. We simulate for both controllers, with a perturbation of 20 packets between the 50^th and 70^th seconds, to evaluate the effect on the differents controllers. The initial conditions are defined such that the buffer is empty at the beginning of the simulation ($q_0 = 0$), the congestion window size such that the simulation is in congestion avoidance from the begininng ($W_0 = 20$) and the packet mark probability is null ($p_0 = 0$) . The Simulink model of the predictive controller is shown in appendix B.1

In obedience to the Lyapunov's linearization, if the linearized system is strictly stable, then the equilibrium point is asymptotically stable for actual nonlinear system. If we look at the Figure (3.6), we see that the behavior of the linear and non-linear systems are similar. Furthermore, we remark that when the perturbation occurs, the comportment changes between the linear and non-linear.
Concerning the settling time, both controllers are similar, and before the perturbation, the steady state error is null. When the perturbation appears, the reaction is different. The PI with the Integral part tries to cancel the steady state error, and come back to the reference, while the predictive controller behaves differently.

3.4 NS Simulations

NS2-simulator is the reference simulator in terms of network. The simulations are realized using three different algorithms (DropTail, PI and predictive control). The controllers are implemented in C++, and the simulations are implemented in TCL, which is a specific language for NS (see appendix C). For each simulation, we observe the window size, the

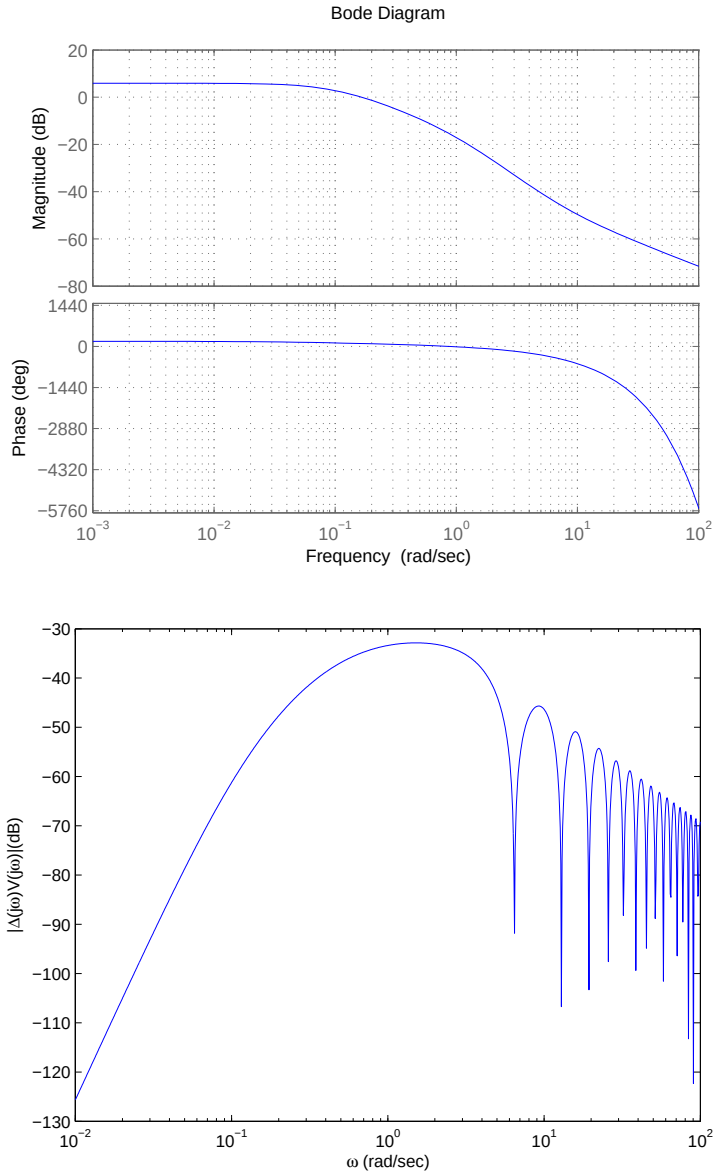


Figure 3.5: Frequency response using predictive controller $F = [-0.11840.1668].10^{-}3$. Frequency response of the open loop (left plot). $|\Delta(j\omega)V(j\omega)| < 1$ showing stability on face of uncertainties. (right plot)

queue size, and the propagation delay.
The DropTail queue management is already available in the NS library. We only need to specify the queue management type when we create the link between two sources.
The PI controller, on NS, is the controller developed by [11] which was already implemented (see appendix B.2.1).
The predictive controller was not included in the library, so it has been implemented and added to the NS source (see appendix B.2.2). The integrals are realized using the rectangle method (3.3). This method is usually very approximative. Some more efficient methods, like the Simpson's method could be used. However, as the sampling frequency is calculated between two packets arrivals, and one of the assumption of the model is that the model is fluid because of the large number of arrivals. Hence the rectangle methods is sufficient. Moreover, a comparison between the integral realized with Matlab using the

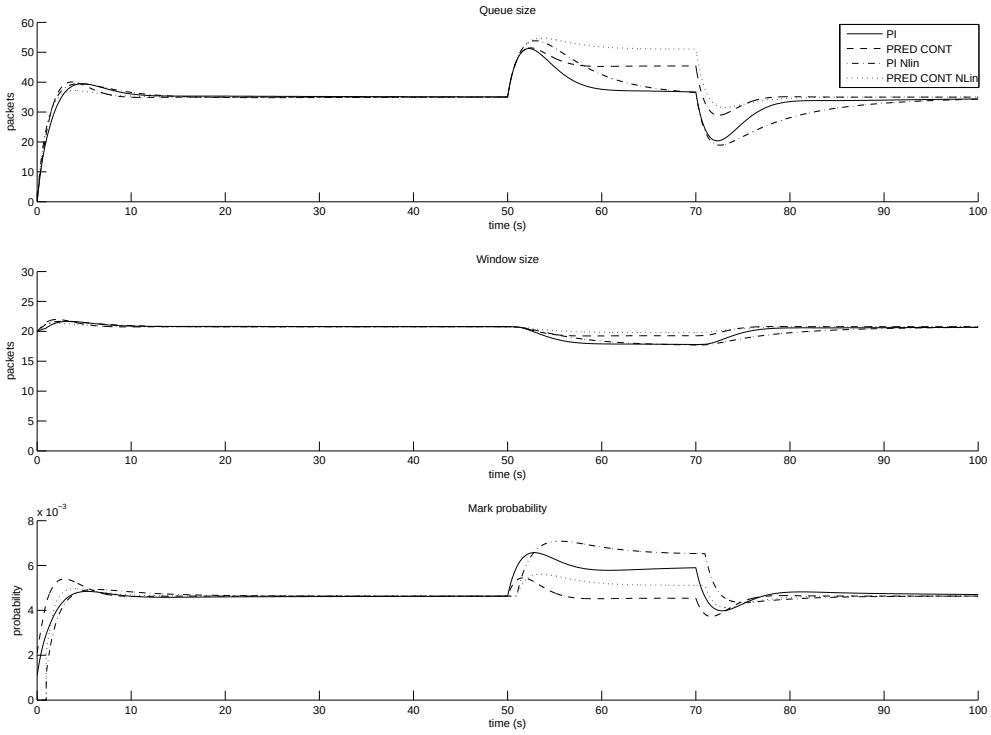


Figure 3.6: Matlab simulations of the linearized system and the non-linear system with the PI and the Predictive controller

solver ode4 (Rung-Kutta) and the integral realized using the rectangle has been observed, and the result is similar.

$$\int_t^{t+h} f(x)dx \approx hf(t) \tag{3.3}$$

For each AQM, two simulations are realized, the first one only with TCP sources, we set the UDP source to 0, in order to evaluate the TCP behavior only. During the second simulation, we add a first UDP connection between the 40^{th} and the 160^{th} seconds, a second one between the 80^{th} and 120^{th} seconds.
The probability of packet mark for the PI and the predictive controller is computed at each packet arrival, which defines the sampling frequency by calculating the time between each arrivals. The probability is a value between 0 and 1. We draw a random variable v(·). If p is smaller than v we keep the packet, if p is bigger, we drop the packet. With DropTail, when the queue is filled to its maximum capacity, the newly arriving packets are dropped until the queue has enough space to accept incoming traffic.

3.4.1 Congestion Windows

We represent in Figures (3.7), (3.8) and (3.9) the congestion windows of the 6 sources.

On the three figures, we observe the slow-start behavior at the beginning of each simulation. The phase is not include in our model, so we neglect this phase. However, right after this part, as expected we can recognize the congestion avoidance mechanism for the three methods. Indeed, each source increases linearly until a loss occurs. Then the source is halved, like congestion avoidance mechanism described part (1.3.2). On the

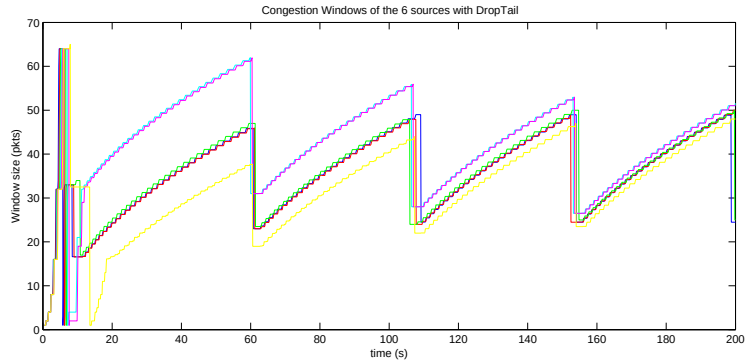


Figure 3.7: Congestion windows for the 6 souces with DropTail

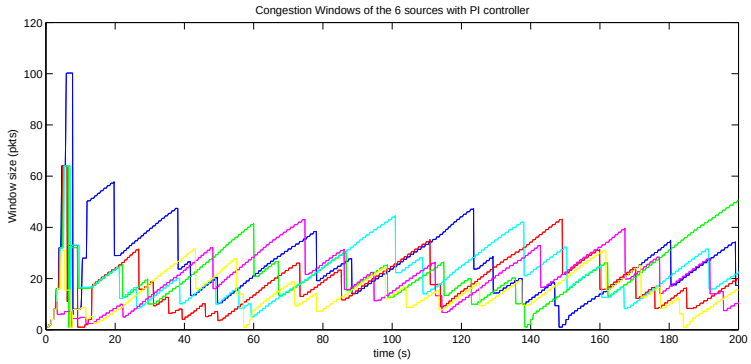


Figure 3.8: Congestion windows for the 6 souces with PI controller

other hand, for the PI and predictive controller, we see that the sources are set to one, which means that timeout occurs. Timeouts have been neglected in our model, but this choice is justified by the fact that only few timeouts occurs (four for the PI controller and only two for the predictive controller). All these points allow us to validate the choice of our model.

3.4.2 First experiment: TCP evaluation

In this part, we make a simulation with TCP connections only in order to evaluate the behavior of TCP, and verify that our model is close to the NS simulation.

AQM	DT	PI	PC
$Mean(pkts)$	136.2	36.8	30.6
$Stand.dev.(pkts)$	45.8	28.8	27.7
$Average\ queueing\ delay(secs)$	1.7	0.99	0.94

Table 3.1: Statistics on the queue length for the three AQM

With the aid of the Figures (3.15), (3.16) and (3.17) as well as Table (3.1), we observe that obviously using DropTail the average congestion window size as well as average queue size are larger than the two other controllers. This is justified by the fact that the window size of each source increases until the buffer is full, leading to a loss. Figures point that thanks to the regulation the queue size is more stable for the two automatic based AQM

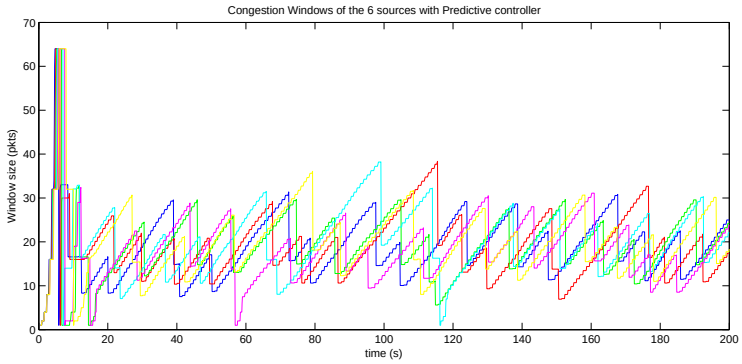


Figure 3.9: Congestion windows for the 6 souces with predictive controller

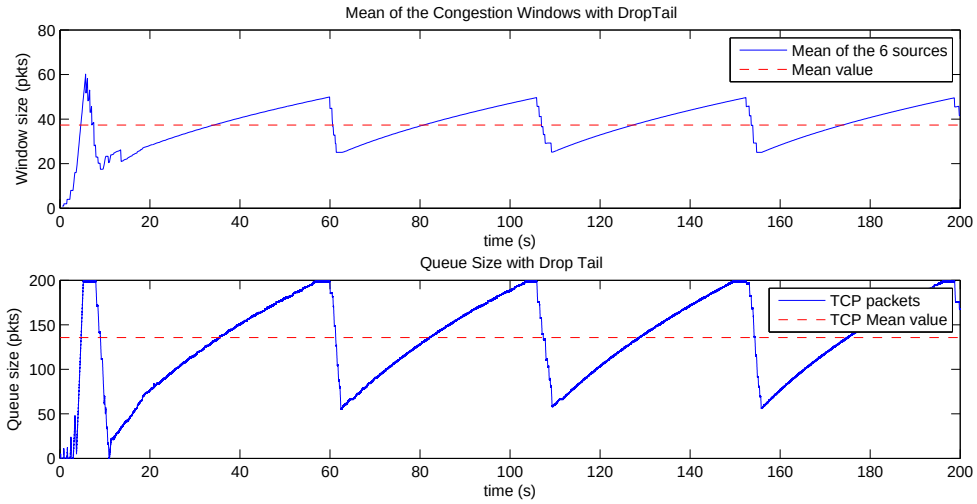


Figure 3.10: NS simulation of the System with DropTail

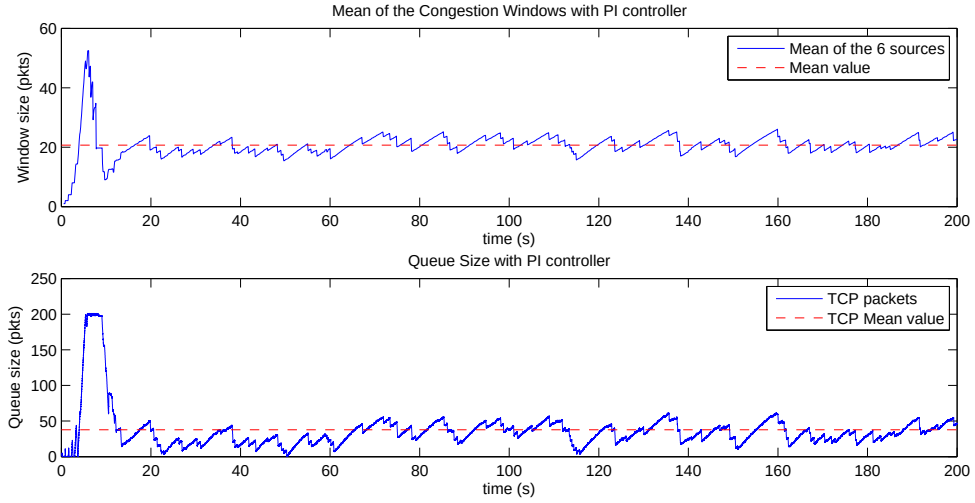


Figure 3.11: NS simulation of the System with PI controller

than for the DropTail. Thanks to the Integral part of the PI controller, which cancels the steady state error, the average queue size is close to the fixed reference (35 packets). So we have reasonable delays because the queue size does not grow. About the predictive

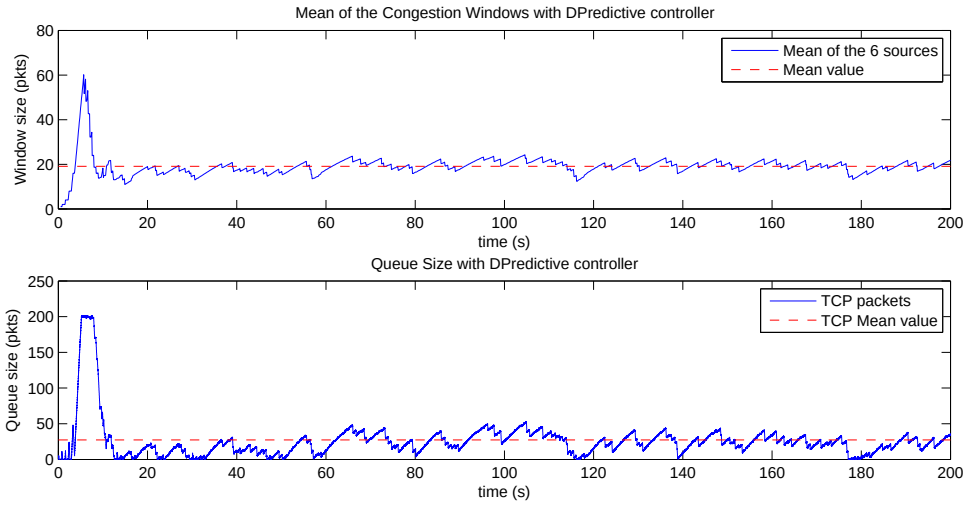


Figure 3.12: NS simulation of the System with predictive controller

controller, the average queue size is less than the reference, but the standard deviation is smaller than the PI's standard deviation.
In terms of delay, we know that the RTT is proportional to the average queue size ($R(t) = T_p + \frac{q(t)}{C}$). Looking at the figure (3.13), the delay with DropTail exceed two seconds, while they are around one second for the two others. From the losses point of view, with Figure

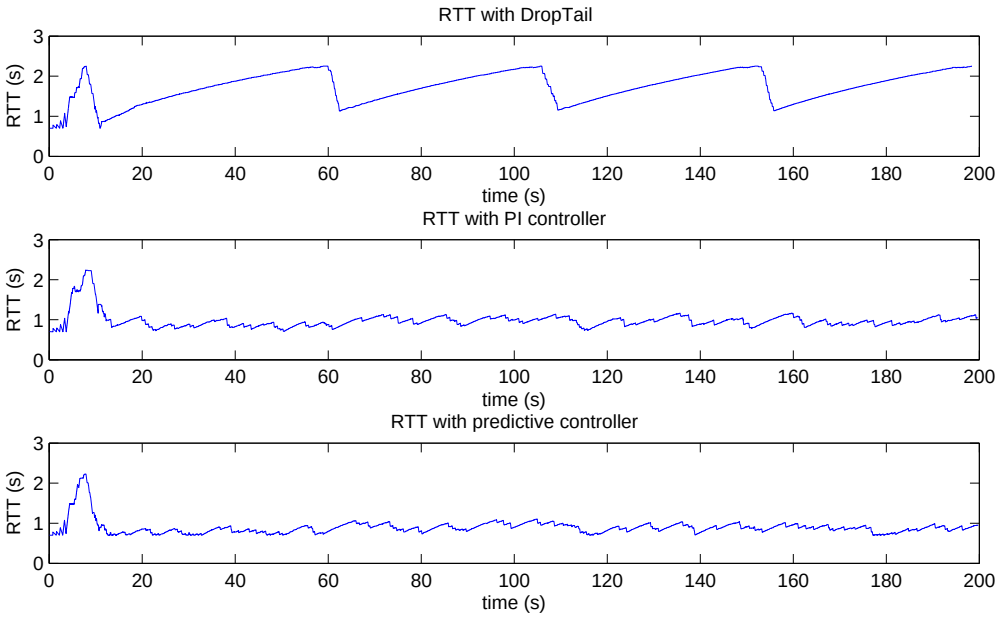


Figure 3.13: Delays using the three methods

(3.14) and Table (3.2), we observe that with DropTail there is less losses than with the two other controllers, however it is not very significative. We can also point that with DropTail, losses occur at the moment where the queue is full, then when congestion occurs, all the packets are dropped. This can pose a problem to solve the congestion problem, because the router is already congested and we will have to resend all the dropped packets. Using the two other methods, the buffer is never congested, and packets are lost continuously

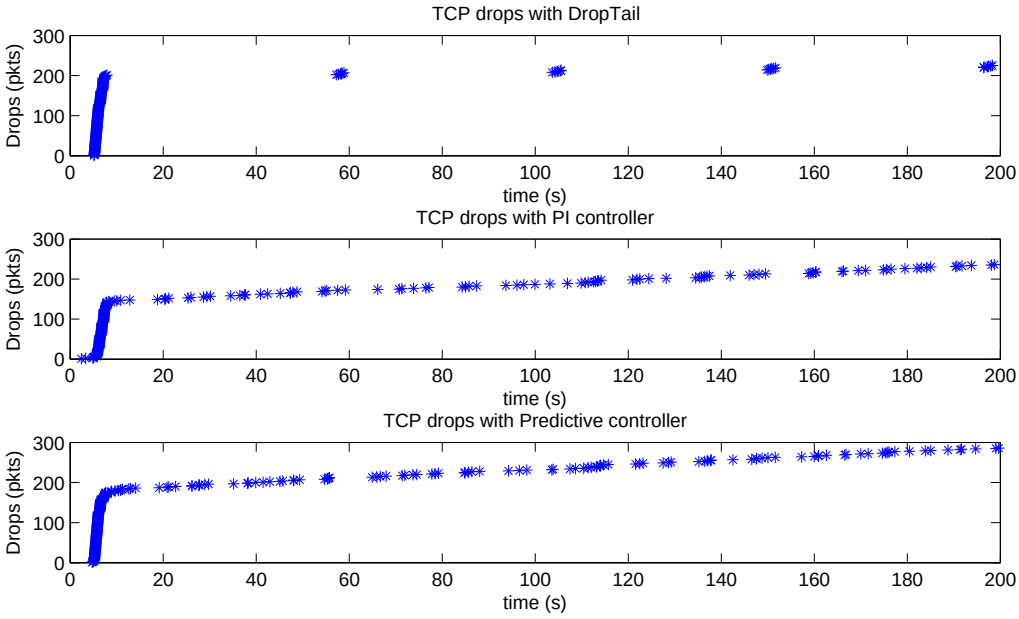


Figure 3.14: Losses using the three methods

one by one. It will be easier to resend the lost packet.

AQM	DT	PI	PC
$Transmitted(pkts)$	25702	25590	25513
$Dropped(pkts)$	225	236	286
$Percentage(\%)$	0.88	0.92	1.1

Table 3.2: Statistics on TCP packet losses for the three AQM

After the study of the performances of TCP connections only, we can conclude that the AQM using automatic control are much more efficient in terms of delay. This thanks to the fact that using regulation, the queue length stays approximatively constant around the queue reference (35 packets). Now that we have verified that we have the desired behavior with TCP connection, we will study the effect of perturbations on the connection. We will make the simulation using UDP connections in addition to TCP connections.

3.4.3 Second experiment: TCP behavior in the presence of UDP perturbation

In this simulation, UDP flux arrive with a flow rate of $128kbits$. A first UDP connection occurs between the 40^{th} and the 160^{th} seconds, a second one between the 80^{th} and 120^{th} seconds. As for the previous part, we have a slow-start phase at the beginning of each simulation, that we do not take in consideration.

In Figure (3.15) we see that we have a buffer overflow when the perturbation occurs. This will lead to higher RTT. Concerning Figures (3.16) and (3.17) we remark that the queue size increases a bit when congestion occurs, however, the queue never reaches its maximum. We also notice that with the PI controller, the integral part does not manage

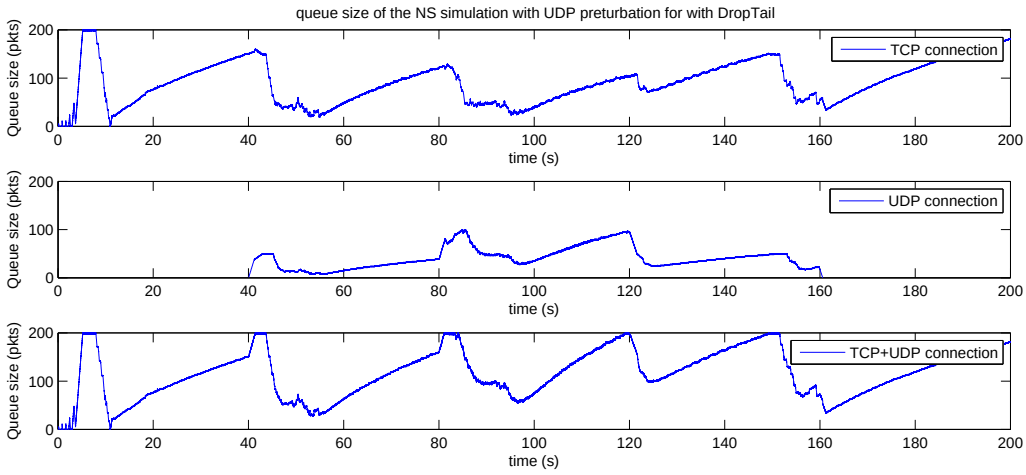


Figure 3.15: NS simulation of the System with DropTail

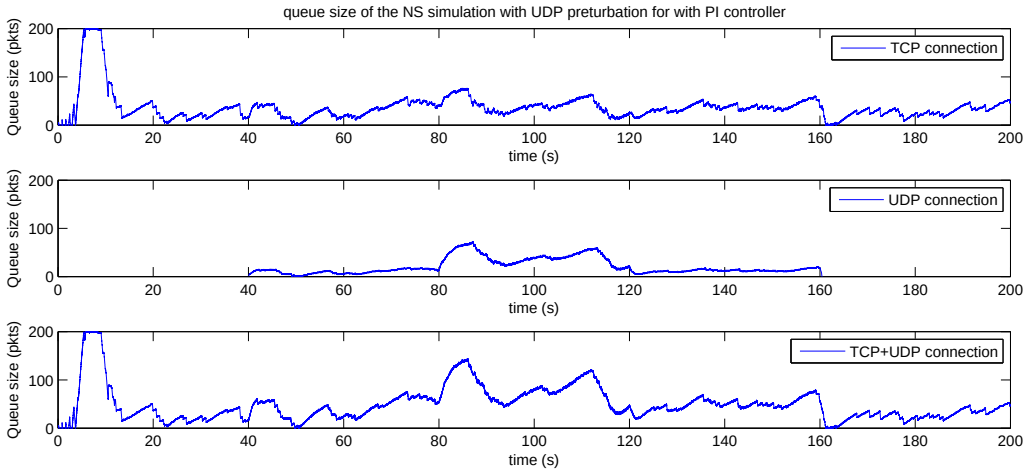


Figure 3.16: NS simulation of the System with PI controller

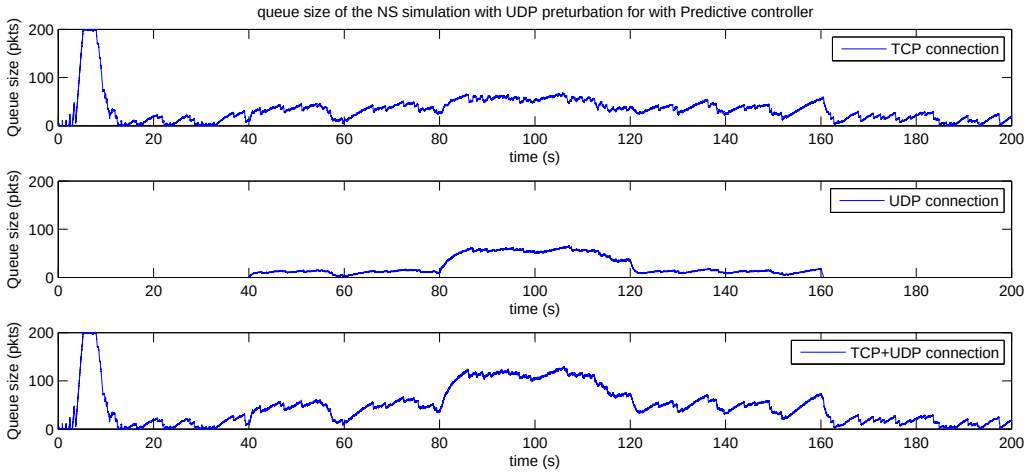


Figure 3.17: NS simulation of the System with predictive controller

to cancel the static error. Regarding the figure with the predictive controller, the response oscillates less. We clearly recognize the three different phases of the simulation.

AQM	DT	PI	PC
$Transmitted(pkts)$	5120	5120	5120
$Dropped(pkts)$	29	23	0
$Percentage(\%)$	0.57	0.45	0

Table 3.3: Statistics on UDP packet losses for the three AQM

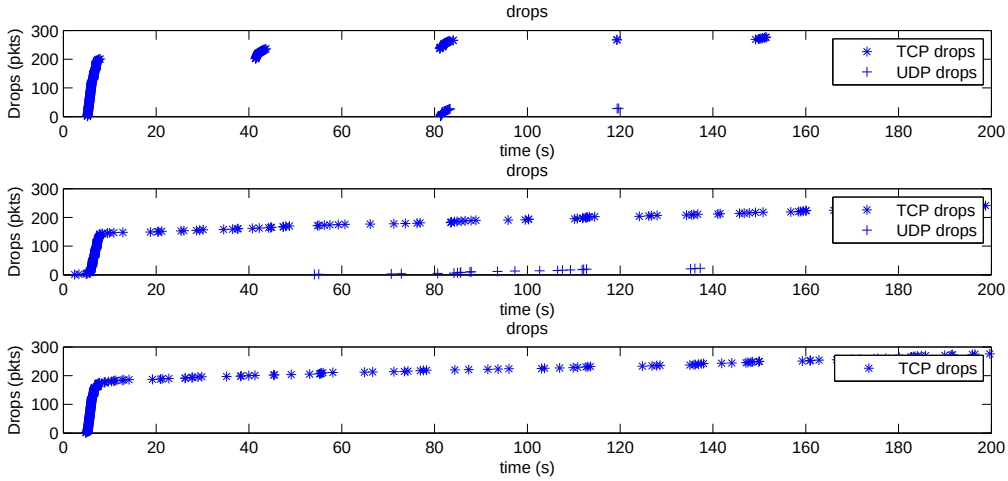


Figure 3.18: NS simulation of the System with predictive controller

In term of UDP packet loss (see Table (3.3)) we notice that with DropTail and PI the number of losses is similar. However there is no UDP losses using the predictive controller. Moreover Figure (3.18) points that using DropTail, if the perturbation occurs when congestion occurs, the buffer is full, so all the UDP packets will be lost. Using this controller, the connection will be cut during the congestion phenomena. With the PI controller some UDP packets are dropped during the connection, which will lead to a lower connection quality, but still acceptable. Otherwise using the predictive controller, none of the UDP packets are lost, then the UDP connection will be completely transmit.

3.5 Improvement

As observed in the previous section, the simulations using the predictive controller show a response with less delay and with less oscillations. Nevertheless, the static error is not canceled, so when the perturbation occurs, the stabilization is not close to the reference. In order to remove this error, we apply to the predictive controller an integral action that has been developed sub-section (2.3.2). For a question of time, the integral action has not been implemented on NS-simulator, but with Matlab only. It will have to be carried out in a forthcoming work. In this part, we will present only the Matlab results which are promising.

The simulations on Matlab are realized using the equation (2.28). This is the same technique described for the controller without integral action. The difference is that with the integral action, we have a third order equation. To evaluate the behavior, we have plot the windows size and the queue size of the model without and with the integral part. The aim of the integral part is to remove the static error on the queue size, with a goal to have a queue size as close as possible to the reference.
The C matrix of the system (2.4) is defined such that the output $y(t)$ represents the queue size ($q(t)$), i.e. $C = [0\ 1]$. Then we define the $\tilde{F}$ matrix such that the poles have a negative real part, and the input p is not saturated. Considering all these conditions, we obtain the following simulations for the linear and the non-linear model (see Figure (3.19)). In both

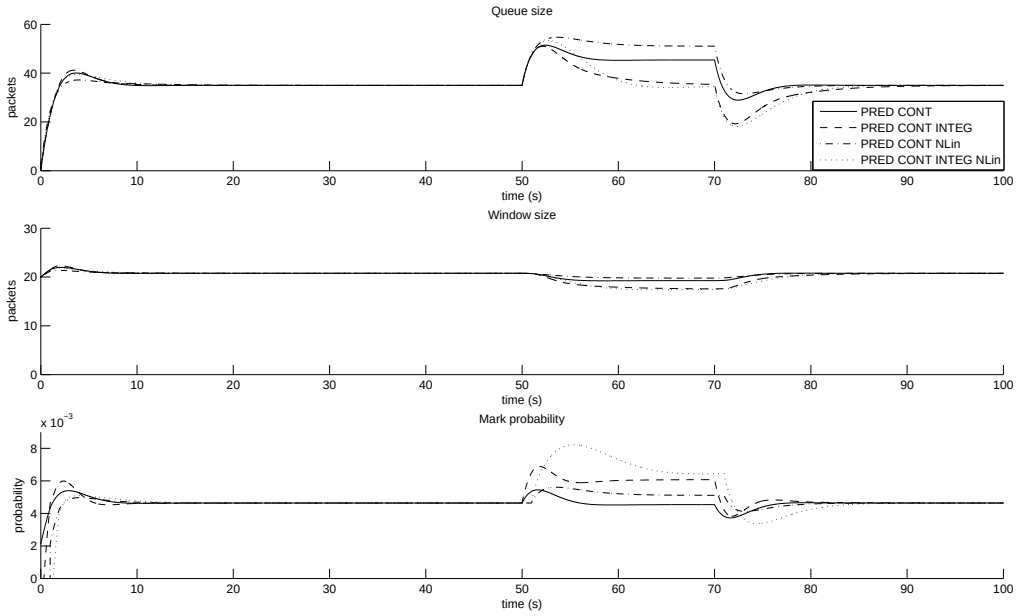


Figure 3.19: Matlab simulations of the linearized system and the non-linear model with the predictive controller without and with the integral part

cases (linearized and non-linear model), we observe that when the perturbation occurs, the integral part removes the static error.

Conclusion

During the Master's thesis, we have designed two different AQMs based on automatic control that we compare to the DropTail AQM. This method is very interesting in terms of delay regulation. Indeed, it leads to constant and lower delay than DropTail AQM, and a response which oscillates less. Concerning the PI controller, we obtain a zero steady-state regulation error, and the method is very easy to implement on NS-simulator. Regarding the predictive controller, the implementation is more complex. We observe that the steady state error is not cancel when the perturbation occurs. However none of the UDP packets are lost which is one of most important goal of this work, despite that we have a steady state error. Conversely, the DropTail AQM poses a huge problem if the UDP packets arrive when the buffer is full.

In order to improve the predictive controller, it could be useful to implement an integral part on NS-simulator, which will cancel the steady-state error, because we observe a good behavior on Matlab. This is the next objective of this work. Secondly one other interesting work that could be done will be to implement the two controllers on a platform available at LAAS (Platine [4]) to make more experimentations.

Another method that could be used for this problem is to have two queues (one for UDP and one for TCP) with different priorities. However in this scheme, we can ask the question of the necessity of AF queue. It will lead to a scheme with an EF queue and an BE queue in the AF queue. Moreover, some problems of TCP instability could occur, because TCP packets will be blocked when UDP packets arrive, so the congestion phenomena will be accentuated (c.f. 4.3).

Finally, from a personal point of view, this work was very rewarding in terms of automatic control background. This work has enabled me to better understand the effect of the delay on a system, as well as control this delay. Moreover, I had the possibility to apply and understand the robust control. Lastly, during this Master's Thesis, I have learned a lot about network theory, domain which was completely unknown for me. The work I realized was the subject of an article [5], which will be presented at the conference Ka and Broadband Communications, Navigation and Earth Observation between the 23rd and 25th of september 2009 in Cagliari. To conclude, this work has allowed me to discover the research field, which corresponds well to my vision of the work. So I have decided to continue my steady by a PhD in France in a laboratory in the Parisian suburbs.

Appendix A

Linearization

A.1 Linearization

The system (1.3) being non-linear, we have to linearize it around the operating point $((W_0, q_0, p_0))$ defined by (1.4).First we define the functions f and g by

$$\begin{array}{rcl} f(W, W_R, q, q_R, p_R) & \doteq & \frac{1}{\frac{q}{C}+T_p} - \frac{WW_R}{2(\frac{q_R}{C}+T_p)}p_R \\ g(W, q) & \doteq & \frac{N}{\frac{q}{C}+T_p}W - C \end{array} \tag{A.1}$$

where $W_R(t) \doteq W(t - h(t))$, $q_R(t) \doteq q(t - h(t))$ and $p_R(t) \doteq p(t - h(t))$ are the delayed states variables and the delayed input and $h(t) = \frac{q}{C} + T_p$.

The partial derivatives of f and g at the operating point are:

$$\begin{array}{rcl} \frac{\partial f}{\partial W} & = & -\frac{W_R}{2(\frac{q_R}{C}+T_p)}p_R = -\frac{W_0}{2(\frac{q_0}{C}+T_p)}p_0 = -\frac{W_0 p_0}{2R_0} = -\frac{N}{R_0^2 C} \\ \frac{\partial f}{\partial W_R} & = & \frac{\partial f}{\partial W} \\ \frac{\partial f}{\partial p_R} & = & -\frac{WW_R}{2(\frac{q_R}{C}+T_p)} = -\frac{W_0^2}{2R_0} = -\frac{C^2 R_0}{2N^2} \\ \frac{\partial f}{\partial q} & = & -\frac{1/C}{(\frac{q}{C}+T_p)^2} = -\frac{1}{C(\frac{q_0}{C}+T_p)^2} = -\frac{1}{CR_0^2} \\ \frac{\partial f}{\partial q_R} & = & \frac{WW_R(1/C)}{2(\frac{q_R}{C}+T_p)^2}p_R = \frac{W_0^2 p_0}{2C(\frac{q_0}{C}+T_p)^2} = \frac{1}{CR_0^2} \\ \frac{\partial g}{\partial W} & = & \frac{N}{\frac{q}{C}+T_p} = \frac{N}{\frac{q_0}{C}+T_p} = \frac{N}{R_0} \\ \frac{\partial g}{\partial q} & = & -\frac{NW(1/C)}{(\frac{q}{C}+T_p)^2} = -\frac{NW_0}{C(\frac{q_0}{C}+T_p)^2} = -\frac{1}{R_0} \\ \frac{\partial g}{\partial p_R} & = & 0 \end{array} \tag{A.2}$$

Appendix B

Simulation

B.1 Matlab/simulink

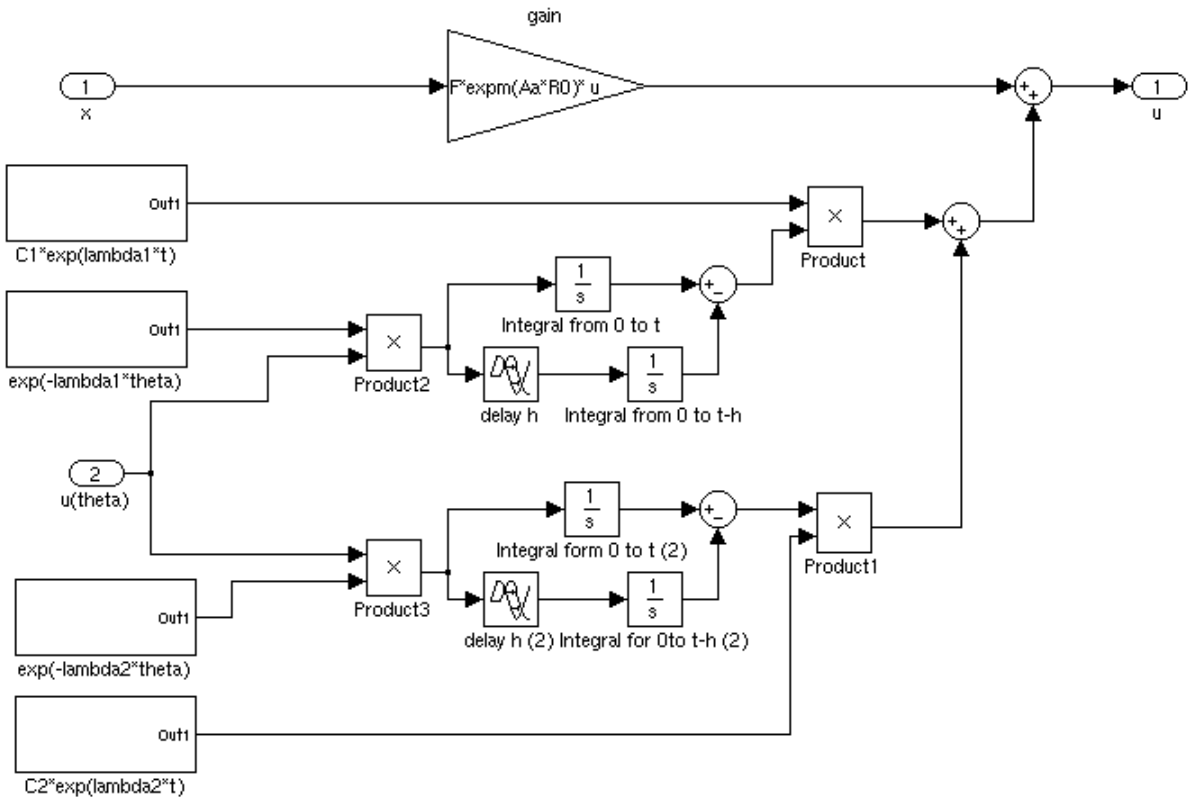


Figure B.1: Simulink model of the predictive controller

B.2 NS2-simulator

B.2.1 PI controller

see figure (B.2)

```
double PIQueue::calculate_p()
{
    double now = Scheduler::instance().clock();
    double p;
    int qlen = qib_ ? q_->byteLength() : q_->length();

    p=edp_.a*(qlen-edp_.qref)-edp_.b*(edv_.qold-edp_.qref)+edv_.v_prob;

    if (p < 0) p = 0;
    if (p > 1) p = 1;

    edv_.v_prob = p;
    edv_.qold = qlen;

    CalcTimer.resched(1.0/edp_.w);
    return p;
}
```

Figure B.2: PI function implement on NS

B.2.2 Predictive controller

see figure (B.3)

```
double PRED_CONTQueue::calculate_p(double CongFen)
{
    double now = Scheduler::instance().clock();
    double p;

    int qlen = qib_ ? q_->byteLength() : q_->length();
    double integ=0.0;

    //State Feedback
    p=edp_.k1*(CongFen-edp_.wref)+edp_.k2*(qlen-edp_.qref);

    //Rectangle
    //integrales from 0 to t
    edp_.integ1=edp_.integ1+(now-edp_.oldnow)*exp(-edp_.lambda1*now)*p;
    edp_.integ3=edp_.integ3+(now-edp_.oldnow)*exp(-edp_.lambda2*now)*p;

    //integrales from 0 to t-R0
    if(now>=edp_.R0)
    {
        edp_.integ2=edp_.integ2+(now-edp_.oldnow)*exp(-edp_.lambda1*(now-edp_.R0))*edp_.pold[edp_.loop2];
        edp_.integ4=edp_.integ4+(now-edp_.oldnow)*exp(-edp_.lambda2*(now-edp_.R0))*edp_.pold[edp_.loop2];
    }

    edp_.oldnow=now;


    //Sum of integrales
    integ=edp_.C1*exp(edp_.lambda1*now)*(edp_.integ1-edp_.integ2)
         +edp_.C2*exp(edp_.lambda2*now)*(edp_.integ3-edp_.integ4);
    edp_.pold[edp_.loop]=p;

    p=p+integ;
    edp_.loop++;

    if (p < 0) p = 0;
    if (p > 1) p = 1;

    return p;
}
```

Figure B.3: Predictive controller function implement on NS

Appendix C

Network Simulator 2

C.1 Writing of a script

The writing of a script start by the creation of a simulator object, the opening of the trace file.

```
#Create a simulator object
set ns [new Simulator]
#Open the nam trace file
set nf [open out.nam w]
$ns namtrace-all $nf
```

The class used to create the nodes is the class Node. Function of the desired architecture, the user connect this nodes with the adapted link. The links have as parameters the bandwidth, the propagation delay and the AQM type. In this work we use DropTail, PI and PRED. The sources nodes are creating using a for-loop, and two nodes are creating for the bottleneck and the destination.

```
#Create bottleneck and destination nodes
set N [$ns node]
set D [$ns node]
#Source nodes
for {set j 1} {$j<=$NumbSrc} { incr j } {
set S($j) [$ns node]
}

#Create links between these nodes
$ns duplex-link $N $D 512kb 300ms AQM_type
#Create links between source and bottleneck
for {set j 1} {$j<=$NumbSrc} { incr j } {
$ns duplex-link $S($j) $N 15Mb 50ms DropTail
}
```

Once the nodes created, the user has to define the transport protocols used, adding agents on the nodes. For each protocol, we define a source agent which will send the traffic and a receiver agent.
The user can then link an application to the chosen agent, it will generate the traffic which

will be passed to the agent. The generated traffics depend of the chosen application.

```
#TCP Sources
for {set j 1} {$j<=$NumbSrc} { incr j } {
set tcp_src($j) [new Agent/TCP]
}
#TCP Destinations
for {set j 1} {$j<=$NumbSrc} { incr j } {
set tcp_snk($j) [new Agent/TCPSink]
}
#Connections TCP
for {set j 1} {$j<=$NumbSrc} { incr j } {
$ns attach-agent $S($j) $tcp_src($j)
$ns attach-agent $D $tcp_snk($j)
$ns connect $tcp_src($j) $tcp_snk($j)
}
#Parametrisation of TCP sources
for {set j 1} {$j<=$NumbSrc} { incr j } {
$tcp_src($j) set packetSize_ 500
$tcp_src($j) set window_ 100
}
```

This is for TCP, we use the same code for UDP, by changing TCP by UDP.
The script finishes by the launching of the simulator and the choice of the starting and finishing time of each applications.

```
#Schedule events for the FTP agents:
for {set i 1} {$i<=$NumbSrc} { incr i } {
$ns at 0.1 "$ftp($i) start"
$ns at Duration "$ftp($i) stop"
}
# Start Simulation
$ns run
```

In order to use the PI controller the predictive controller, we need to specify the parameters as follow:

```
#PI parameters
Queue/PI set a_ ...
Queue/PI set b_ ...
Queue/PI set w_ ...
Queue/PI set qref_ ...
Queue/PI set pref_ ...

#parametre PRED_CONT
Queue/PRED_CONT set qref_ ...
Queue/PRED_CONT set R0_ ...
Queue/PRED_CONT set wref_ ...
Queue/PRED_CONT set pref_ ...
```

Queue/PRED_CONT set lambda1_ ...
Queue/PRED_CONT set lambda2_ ...
Queue/PRED_CONT set lambda3_ ...
Queue/PRED_CONT set C1_ ...
Queue/PRED_CONT set C2_ ...
Queue/PRED_CONT set C3_ ...
Queue/PRED_CONT set k1_ ...
Queue/PRED_CONT set k2_ ...
Queue/PRED_CONT set k3_ ...
Queue/PRED_CONT set w_ ...
Queue/PRED_CONT set NbConnect_ ...

Bibliography

[1] Satlabs group. URL: http://satlabs.org/.

[2] E. Altman and T. Jiménez. Ns simulator for beginners. Lecture notes, December 2003. URL: http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS.

[3] Y. Ariba, Y. Labit, and F. Gouaisbaut. Congestion control of a single router with an active queue management. *International Journal on Advances in Internet Technology*, 2(1), 2009.

[4] C. Baudoin, Mathieu Dervin, Pascal Berthou, Thierry Gayraud, Frédéric Nivor, Baptiste Jacquemin, Didier Barvaux, and J. Nicol. PLATINE: DVB-S2/RCS enhanced testbed for next generation satellite networks. In *International Workshop on IP Networking over Next-generation Satellite Systems (INNSS07)*, volume ISBN-13: 978-0387754277, page 11p., 07 2007.

[5] Romain Delpoux, Pascal Berthou, Yann Labit, and Frederic Gouaisbaut. Satellite terminal quality of service management with AQM control. In *Fifteenth Ka and Broadband Communications, Navigation and Earth Observation Conference*, Cagliari, Italy, 9 2009.

[6] K. Fall and K. Varadhan. The ns manual. notes and documentation about the software ns2-simulator. URL: www.isi.edu/nsnam/ns/.

[7] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1:397–413, August 1993.

[8] T. Gayraud and P. Berthou. A qos architecture for dvb-rcs next generation satellite networks. *EURASIP Journal on Wireless Communications and Networking*, ID 58484 doi:10.1155/2007/58484:200–300, 2007.

[9] T. Gland and L. Ljung. *Control Theory Multivariable and nonlinear methods*. Taylor and Francis, 2000.

[10] S. H. Low, F. Paganini, and J. C. Doyle. Internet congestion control. *IEEE Control Systems Magazine*, feb 2002.

[11] C. V. Hollot, V. Misra, D Towsley, and W. Gong. On designing improved controllers for aqm routers supporting tcp flows. In *IEEE INFOCOM*, volume 3, pages 1726–1734, April 2001.

[12] C. V. Hollot, V. Misra, D. Towsley, and W. Gong. Analysis and design of controllers for aqm routers supporting tcp flows. *IEEE Trans. on Automat. Control*, 47:945–959, jun 2002.

[13] A. Z. Manitius and A. W. Olbrot. Finite spectrum assignment problem for systems with delays. *IEEE Trans. on Automat. Control*, AC-24:541–553, aug 1979.

[14] V. Misra, W. Gong, and D. Towsley. Stochastic differential equation modeling and analysis of tcp windowsize behavior. Technical report, University of Massachusetts, October 1999.

[15] V. Misra, W. Gong, and D Towsley. Fluid-based analysis of a network of aqm routers supporting tcp flows with an application to red. In *SIGCOMM*, pages 151–160, August 2000.

[16] K. K. Ramakrishnan and S. Floyd. A proposal to add explicit congestion notification (ecn) to ip. RFC 2481, January 1999.

[17] I. Tou, M. Gineste, T. Gayraud, and P. Berthou. Quality of service evaluation in satellite systems. *International Workshop on Satellite and Space Communications 2008 (IWSSC 2008)*, pages 133–137, oct 2008.

[18] H. Zhang, C. V. Hollot, D. Towsley, and V. Misra. A self-tuning structure for adaptation in tcp/aqm networks. In *IEEE Globecom*, volume 7, pages 3641–3646, December 2003.