

# A control based Traffic Controller in wireless and satellite downlink channels.

Francesco Delli Priscoli<sup>1</sup>, Dario Pompili<sup>2</sup>, Giuseppe Sette<sup>3</sup>

<sup>1</sup>dellipriscoli@dis.uniroma1.it, <sup>2</sup>pompili@dis.uniroma1.it, <sup>3</sup>giuseppe.sette@tiscali.it

Computer Science Department, University of Rome "La Sapienza", Via Eudossiana, 18 00184 Rome, ITALY

The authors are listed in alphabetical order

## Abstract

This paper presents an innovative control based Traffic Controller which provides at sharing a single resource, namely the available bit rate, among a set of IP flows characterized by different Quality of Service (QoS) requirements. The main novelty behind the proposed Traffic Controller is that it relies on an original control based approach which exploits a closed-loop architecture and a fuzzy tuner. Simulation results demonstrate that such Traffic Controller outperforms a well known Traffic Controller (namely the one based on the presence of Dual Leaky Buckets and Earliest Deadline First scheduling algorithm).

Key words: Control Theory, QoS, Fuzzy tuning

## I. INTRODUCTION

In this article we propose a solution to the problem of sharing the bandwidth resource of a satellite (or wireless) network between the various data-streams that convey data from the Internet to the end users of the satellite system.

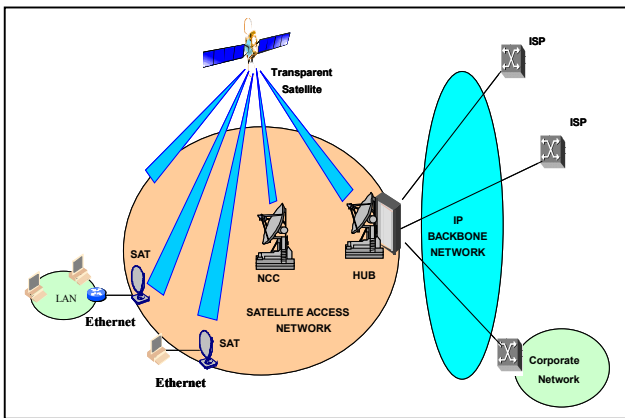


Fig. 1 - Satellite scenario

In greater detail we conceive a situation like the one exposed in Fig. 1 where various connections flow from the Internet into a Hub Station (HUB) which broadcasts the data towards the satellite, that subsequently redirects the flow towards the various users' Satellite Terminals (SAT).

Our goal is to calculate an equilibrium to share the bandwidth resource assigned to the Hub Station between the

various flows so as to respect the assigned Quality of Service (QoS) contract of each connection and lose as few non-compliant packets as possible. This target must be achieved while respecting both principles of **fairness** and **efficiency**. The former principle expresses the need to serve each connection equally with respect of a fairness measure. For what concerns the latter, the goal is to minimize the waste of the limited (and costly) satellite bandwidth.

The solution proposed in this paper is to be enacted inside this generic Hub Station, which has the duty of managing the bandwidth resource that the overall Network Management assigned to it. The schematisation of scenario where we want to operate is shown in Fig. 2. As can be seen in Fig. 3 the architecture of the control system is analogous to a generic feedback control – the main difference being, as we will show, in the definition of the reference and controlled variables.

In Section II we will give an overview of the scenario which has to be controlled through the proposed algorithm.

In Section III we will explain step-by-step the various components of the system shown in Fig. 3, and how they affect the behaviour of the control action.

Finally in Section IV we will propose the results of the simulations and in Section V the relative conclusions.

## II. GENERAL OVERVIEW

Referencing to the scheme of Fig. 2., our system receives an **Offered Traffic** from the Internet in the form of IP packets and inserts each of them into one of its **Queues**, based on the flow it belongs to. Our system can use a generic

*Total System Bit-rate* to broadcast as many inbound connections as possible towards the satellite. The goal of the control is to share this *Total System Bit-rate* so as to serve all the accepted connections respecting their QoS contract – consisting in a minimum bit-rate to be transmitted and in a constraint on the max and min delay of each packet – and complying with the two above-mentioned principles of fairness and efficiency.

To do so the Main Controller monitors the state of the system every  $T_{\text{CONTROL}}$  seconds through various measures: the filling up of the queues (*Queue Lengths* in Fig. 2) and the *Compliant, Lost and Transmitted Bit-rates*.

The *Compliant Bit-rate* is an exogenous variable that expresses the minimum bit-rate that must be assigned to each

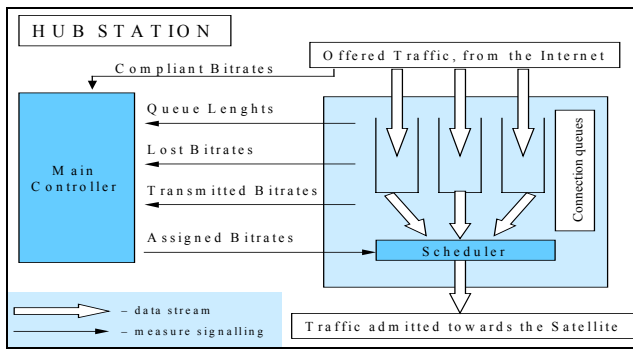


Fig. 2 – Incoming traffic scheme.

TABLE I  
SHORTENED FIG. 3 VARIABLE NAMES

<i>Queue Length(i)</i>	$Q\_Len(i)$
<i>Transmitted Bit-rate(i)</i>	$Tr\_Br(i)$
<i>Lost Bit-rate(i)</i>	$Ls\_Br(i)$
<i>Compliant Bit-rate(i)</i>	$Compl\_Br(i)$
<i>Total System Bit-rate</i>	$Tot\_Sys\_Br$
<i>Spare System Bit-rate</i>	$Sp\_Sys\_Br$
<i>Compliant Assigned Bit-rate(i)</i>	$Comp\_As\_Br(i)$
<i>Non-Compliant Assigned Bit-rate(i)</i>	$NComp\_As\_Br(i)$
<i>Connection Efficiency(i)</i>	$Conn\_Eff(i)$
<i>System Efficiency</i>	$Sys\_Eff$
$\Delta$ <i>Efficiency(i)</i>	$\Delta\_Eff(i)$
$\Delta$ <i>Bitrate(i)</i>	$\Delta\_Br(i)$
$\Delta$ <i>Assigned Bit-rate(i)</i>	$\Delta\_As\_Br(i)$
<i>Connection Assigned Bit-rate(i)</i>	$Conn\_As\_Br(i)$

connection, on the base of its QoS contract. The *Lost* and *Transmitted Bit-rates* are the average bit-rates that the system respectively lost and transmitted in the previous  $T_{\text{CONTROL}}$  period.

Taking into account these four factors the Main Controller must calculate the *Assigned Bit-rate* of each connection. This value is a fraction of the *Total System Bit-rate* that is going to be assigned to the  $i$ -th connection in the next  $T_{\text{CONTROL}}$  period. Once the Main Controller has calculated this share, it forwards it to the Scheduler, which has the duty of broadcasting the IP packets towards the satellite, trying to respect the assignation of resources previously calculated.

In fact the most relevant characteristic of the Main Controller is that it works on a continuous-flow model of the inbound traffic – this means that it only sees continuous measures of bit-rates and bit lengths, and overlooks the discrete nature of the flow of IP packets. Consequently it is up to the Scheduler to deal with the discrete size of the queued packets to realize the equilibrium calculated by the Main Controller.

### III. CONTROL SYSTEM

As we stated previously the control action takes place at the beginning of each control period, namely every  $T_{\text{CONTROL}}$  seconds. The action is divided in two conceptual stages: the assignation of *Compliant Bit-rate* and the assignation of *Spare Bit-rate*. In the first stage we assign a part of the *Total System Bit-rate* according to the QoS contract of each connection, while in the second we assign the remaining available bit-rate according to our own original algorithm. At the end of the two stages the sum of these two assigned values will make up the new *Connection Assigned Bit-rate(i)* for each connection – this value will be used throughout the whole  $T_{\text{CONTROL}}$  period that follows its calculation.

From now on we will shorten all the names of the variables as they are shown in Fig. 3 for the sake of formula compactness. Table I contains these variables and their shortened form. Their meaning will be explained as they are introduced.



*Efficiency* to it, as can be seen in Fig. 3 under the heading “Comparison”.

To do so we introduce the “Super Connection”: this is an extension of the concept of connection to the whole system. The best expression of the *Priority* of such Super Connection is simply the sum of the *Priority(i)* terms, because it accounts for the cumulative capacity to offer traffic of all the connections. As for the equivalent of the *Conn\_As\_Br(i)* in (4), the best choice is *Sp\_Sys\_Br*. Consequently we have :

$$Sys\_Eff = Sp\_Sys\_Br / \sum_i [Priority(i)] \quad (5)$$

By choosing other more intuitive versions of *System Efficiency* the performance of the system would be greatly hampered – i.e. substituting *Sys\_Eff* with an average of the *Conn\_Eff(i)* caused the system to become unbalanced, while changing the numerator of (5) to  $\sum_i [Conn\_As\_Br(i)]$  forced each connection to fight for bandwidth with the other connections, instead of asking it to the overall system. Equation (5) is calculated by the component named “Averager” in Fig. 3.

### B.3. Comparison of the Efficiencies and Control Action

Now that we have a reference level – *Sys\_Eff* – we can compare it to the value that represents the state of the generic connection – *Conn\_Eff(i)* – to determine the entity of the control action. This is what happens in Fig. 3 in the sum component called “Comparison”. We have :

$$\Delta\_Eff(i) = Sys\_Eff - Conn\_Eff(i) \quad (6)$$

As we explained above “efficiency” in this context means how well the bit-rate assigned to a connection serves its need to transmit traffic. Consequently a connection with a lower efficiency than that of the system will have a positive  $\Delta\_Eff(i)$  after the comparison, which means that it deserves more bit-rate – and the other way around.

After the calculation of the value of this incremental efficiency we must de-normalize it to obtain an incremental bit-rate. This happens in the “Denormalizer” component in Fig. 3.

$$\Delta\_Br(i) = \Delta\_Eff(i) * Priority(i) \quad (7)$$

Let’s take a moment to comment the nature of this variable – we defined it as **incremental** because it represents the **variation** of the assigned bit-rate of the i-th connection. But instead of using it directly to modify the assigned bit-rate, we feed it to the “Controller” component, which in our implementation is a digital PID with fuzzy tuning. This way we correct the control action over the *Ncompl\_As\_Br(i)*. From Fig. 3 we simply obtain:

$$\Delta\_As\_Br(i) = Func_{PID}[\Delta\_Br(i)] \quad (8)$$

We used a generic PID controller (Proportional – Integral – Derivative. This definition comes from the three actions that it is able to apply, see [1] and [2]), which is a classic Control component that has the property of enacting a faster action on the process it controls. We applied a modification to its scheme due to the nature of the control we are implementing. We moved the  $K_p$  parameter downstream of the controller – this allows us to tune the  $K_p$  parameter and use it to set the controller’s aggressiveness.

### B.4. Fuzzy Tuning

The tuner that controls the PID’s parameters realizes three different rule-sets.

The first set concerns the tuning of the overall policy of the controller, which is obtained by changing  $K_p$ . The problem is that the various connections strive to obtain a common and limited bandwidth, and in situations of great congestion there is little bandwidth to be shared. Therefore we must tune down the aggressiveness of the control if the system is in such condition, to allow the connections to reach the equilibrium without oscillating while they dispute the little spare bit-rate. This is most important in the transients when new connections are established.

To define “Congestion” we introduce two values: *Need* and *Surplus*. *Need* expresses the cumulative request for bandwidth of all the connections that are under-served. *Surplus* expresses the cumulative yielded bandwidth of those connections that are over-served:

$$Need = \sum_i \max[\Delta\_As\_Br(i), 0] \quad (9a)$$

$$Surplus = \sum_i \max[-\Delta\_As\_Br(i), 0] \quad (9b)$$

Congestion is defined as:

$$Congestion = Need / (Need + Surplus) \quad (10)$$

Utilizing simple triangular fuzzy sets and the Sugeno deduction, we introduce the following rules. Please note that LOW, MEDIUM and HIGH are not fixed constants but rather qualitative indicators of the kind of tuning we employ.

IF <i>Congestion</i> is LOW	THEN $K_p$ is HIGH
IF <i>Congestion</i> is MEDIUM	THEN $K_p$ is MEDIUM
IF <i>Congestion</i> is HIGH	THEN $K_p$ is LOW

These rules prevent newly established connections to pull pre-existing connections away from the equilibrium.

The second and third fuzzy rules come from the usual principles of PID tuning (see [1], [2] and [3]). They control the entity of the  $K_i(i)$  and  $K_D(i)$  parameters, that is to say that they are different for each connection.

For what concerns  $K_i$  we have:

IF $\Delta\_Eff(i)$ is LOW	THEN $K_i(i)$ is LOW
IF $\Delta\_Eff(i)$ is MEDIUM	THEN $K_i(i)$ is MEDIUM

IF  $\Delta_{Eff}(i)$  is HIGH THEN  $K_i(i)$  is HIGH

This is a typical use of the I action, that is raised to speed up the control when the system is distant from equilibrium, and that is turned off when the equilibrium is approached. In particular, this action is useful in the transients when the connection queues fill up: the linear increase of the queues' length causes an almost linear decrease in the *Priority* of the connection, which is properly served by an integrator.

As for the use of a D action, eventually coupled by some kind of tuning, our simulations prove it introduces little to no benefits to the system's efficiency.

The output of the PID is another incremental bit-rate, and will be entered in the "Integrator" component in Fig. 3 to modify the  $Conn\_As\_Br(i)$ .

### B.5. Non-Linear Resharing

The final action of our control consist in the application of a non-linearity that avoids the assignation of a negative  $Conn\_As\_Br(i)$ . This may take place sometimes when load-heavy connections are established in a system that was previously under-loaded (although the fuzzy action reduces this to a point).

Following the application of the non-linearity we must enact a proportional resharing so as to prevent the system from assigning more bandwidth than what is available. This is done in the "Non-Linear Resharer" component in Fig. 3.

## IV. SIMULATIONS

The algorithm has been simulated with the OPNET tool. We have compared its performances with those of some basic scheduling schemes.

In our system we consider four kinds of traffic sources: Voice, FTP, Video, Web.

The sources are simulated through the basic OPNET functions, with the parameters shown in Table II. Makes exception the Video source, which cycles through three stages of a Markov process (**B**, **P** and **I** frames), each with increasing packet interarrival frequency.

Each source is subject to different QoS limitations as shown in Table III.  $R_{min}$  is the minimum bit-rate to be assigned to a connection of a certain class.  $D_{max}$  is the maximum delay that a packet can stand without expiring.  $J_{max}$  is the jitter of the packet, that is to say the difference between the maximum and minimum delay of the packet.

The following figures show static results, connection-wise: that is to say that the number of established connections does not change during the simulation period. We are actually in the process of producing the dynamic results, where there is a greater and variable number of connections running, and where the influence of the PID controller is more evident.

TABLE II  
IP TRAFFIC SOURCES PARAMETERS

	Datagram lenght	Interarrival Times
<b>Voice</b>	Constant 580 bits	Constant 20 ms
<b>FTP</b>	Uniform [320;39680] bits	Uniform [0.05;0.15] secs
<b>Web</b>	Uniform [320;23680] bits	Uniform [0.05;0.55] secs
<b>Video</b>	Uniform [320;23680] bits	Gaussian (avg. value, std.dev) B Frame: 10.6 msec, 5.9 msec P Frame: 8.1 msec, 4.5 msec I Frame: 5.6 msec, 2.9 msec

TABLE III  
IP PACKETS QoS PARAMETERS

	$R_{min}$	$D_{max}$	$J_{max}$
<b>Voice</b>	29 kbps	0.1 sec	0.09 sec
<b>FTP</b>	200 kbps	4 sec	3.8 sec
<b>Web</b>	40 kbps	1.5 sec	1.45 sec
<b>Video</b>	1350 kbps	0.5 sec	0.48 sec

In Fig. 4 we show an example of the behaviour of the control system: the first graph is the reference value, the **System Efficiency**. The other three graphs are the **Efficiencies** of three connections.

It is easy to see the difference between the smooth behaviour of the Voice and Video connection, and the burst-like evolution of the FTP one. It is also evident how the system brings the connection back to the reference value.

For further detail, this behaviour of the **System Efficiency** is typical of a system with high congestion, where there is less bandwidth that what would be needed to transmit the data flow: the queues fill up and **System Efficiency** evolves with inverse proportionality to the time variable. Yet the data loss is kept at minimum.

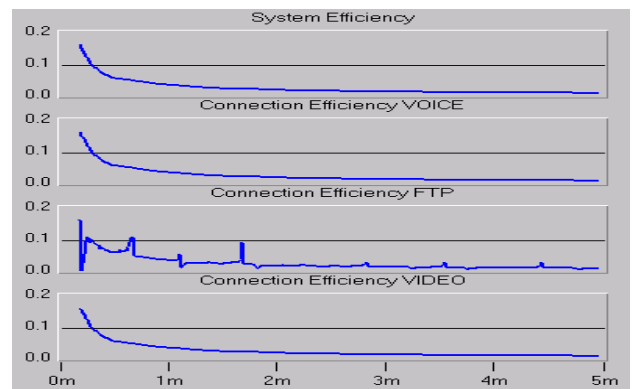


Fig. 4 – Efficiency values , system with high congestion

In Fig. 5 we show the results of the performances of the algorithm, on a simulation period of 10 minutes, compared to those of a FIFO scheduler and of an Open Loop controller (as opposed to the closed-loop philosophy of the controller discussed in this paper and named “Competitive Access”).

In the FIFO scheduler algorithm only one FIFO queue is considered: all the IP packets are enqueued in this queue without taking into account the QoS they are requiring; then packets are served on a First In First Out basis.

In the Open Loop controller DLBs (Dual Leaky Buckets) are used to serve in a differentiated way Compliant packets from Not-Compliant ones; the former are scheduled through an EDF algorithm while the latter are only given extra bandwidth. The name of the scheduling controller is due to the fact that DLBs’ parameters are statically set without any closed loop features.

*System Congestion* is the ratio between the Total System Bandwidth and the total bandwidth transmitted by the incoming connections. It expresses the level of congestion of the system. The Y axes shows the link efficiencies of the various methods used. As can be seen, the algorithm outperforms the simpler versions of congestion control, and it manages to obtain near ideal performances: in fact its link efficiency follows closely the System Congestion, and saturates around 100% congestion without wasting packets. This is almost the best performance that can be obtained out of a system with limited bandwidth.

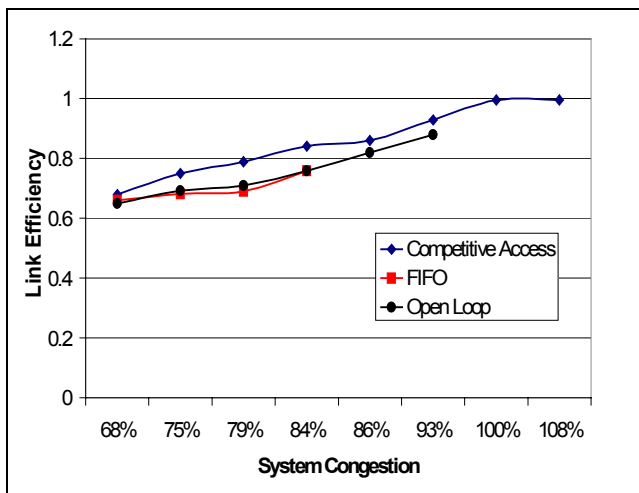


Fig. 5 – Compared Link Efficiencies

## V. CONCLUSIONS

The control system explained in this paper obtains good improvements in downstream satellite uplinks congestion management, as compared to other algorithms, while serving

as many connections as the system bandwidth allows, optimising the use of the valuable bandwidth resource, with no need of special bandwidth request signalling.

The next steps of the research are the study of the dynamic performances of the algorithm – in the most typical cases of a) connection establishment and b) heavier congestion (*System Congestion* growing way beyond 100%) – and the study of the complications of implementing the algorithm in a remote NCC for global network management, in which case long delays are introduced in the control action.

## REFERENCES

- [1] K.J. Astrom and T. Haggund , Automatic Tuning of PID controllers, Instrument Society of America, 1988.
- [2] K.J. Astrom and T. Haggund , PID Controllers: Theory, Design and Tuning, Instrument Society of America, 1995.
- [3] V.B.Baji’c, Simple rule-based and fuzzy controllers, Technical Report, TR95-07, Control Laboratory, Technikon Natal, RSA, 1995.
- [4] A. Silbershatz and P.B.Galvin, Operating System Concepts, Addison-Wesley, 1998.
- [5] Design and Evaluation of a Feedback Control EDF Scheduling Algorithm, Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son.
- [6] Differentiated Service Router Architecture-Classification, Metering and Policing. Daniel Lin and Frank Akujobi, pag.21-23 on WFQ
- [7] R. Braden, D. Clark, and S. Shenker, *Integrated Services in the Internet Architecture: an Overview*, RFC 1633, June 1994.
- [8] K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, RFC 2474, December 1998.
- [9] K.Nichols, V.Jacobson, L.Zhang, *A Two bit Differentiated Services Architecture for the Internet*, RFC 2638, July 1999