

# A Control Theory based Medium Access Control for the downstream satellite uplink of SATIP6 project.

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

Computer Science Department, University of Rome “La Sapienza”

Via Eudossiana, 18 00184 Rome, ITALY

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

## ABSTRACT

This paper describes a controller for allocating shared bandwidth and a scheduler for transmitting IP traffic from the Gateway to the Users' Terminals of the satellite network studied in SATIP6 IST project. The paper describes the algorithms employed in the MAC controller in terms of various components of a classical control system and a fuzzy tuning component. The focus of this work is on the solution of the MAC problem relative to the downstream uplink channel of the SATIP6 satellite network, through the modeling approach proper to the Theory of Control. We conceive a situation where various connections flow from Internet into a Hub Station (Gateway) which broadcasts data towards the satellite, from where flows are redirected towards the various users' Satellite Terminals. The goal of the controller is to calculate an equilibrium to share the Hub Station pre-assigned bandwidth resource 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. In such a way, on the downstream uplink, bandwidth- and resource-consuming algorithms using bandwidth request signalling to the NCC (Network Control Centre) can be avoided.

## I. INTRODUCTION

In this article we propose a solution to the bandwidth resource sharing problem of a GEO satellite network between the various data-streams that convey data from the Internet to the end users of the satellite network. This work has been developed in SATIP6 IST project [1].

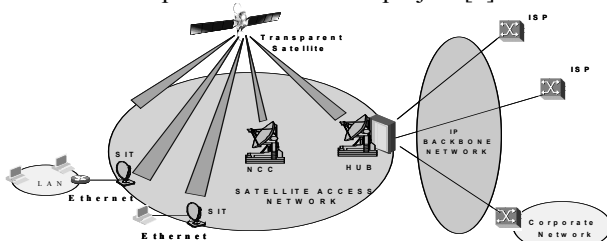


Fig. 1 – SATIP6 general scenario

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 (HS)-Gateway which broadcasts the data towards the satellite, that subsequently redirects the flow towards the various users' Satellite Terminals (SAT) through a transparent satellite.

In this scenario (many others SATIP6 scenarios are described in [2]) the goal is to share the pre-assigned downstream uplink bandwidth of the HS among the various incoming data flows. The most relevant problems are: (a) dealing with the limitedness of the bandwidth given to the HS by the Network Control Centre, which must be used to transmit as many connections as possible, (b) dealing with the bursty nature of the typical internet connections, which forward uneven flows of data packets toward the users, (c) dealing with the QoS that must be guaranteed to each packet, which risks expiration if delayed too much. The solution described in this paper is based on the classic Control Theory.

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 (as will be defined in Section III). For what concerns the latter, the goal is to minimize the waste of the limited (and costly) satellite bandwidth. Most satellite MAC algorithms use a specific bandwidth-request signalling to the NCC in order to dynamically handle the Internet incoming traffic; the backwards of this approach are bandwidth waste for signalling and the difference between the needed bandwidth and the assigned one due to the inaccuracy of the IP traffic prediction that must be done in advance.

The solution proposed in this paper is to be enacted inside the Gateway/Hub Station, which has the duty of managing the bandwidth resource that the Network Control Centre assigned to it. As can be seen in Fig. 2 the architecture of the control system is analogous to a generic feedback

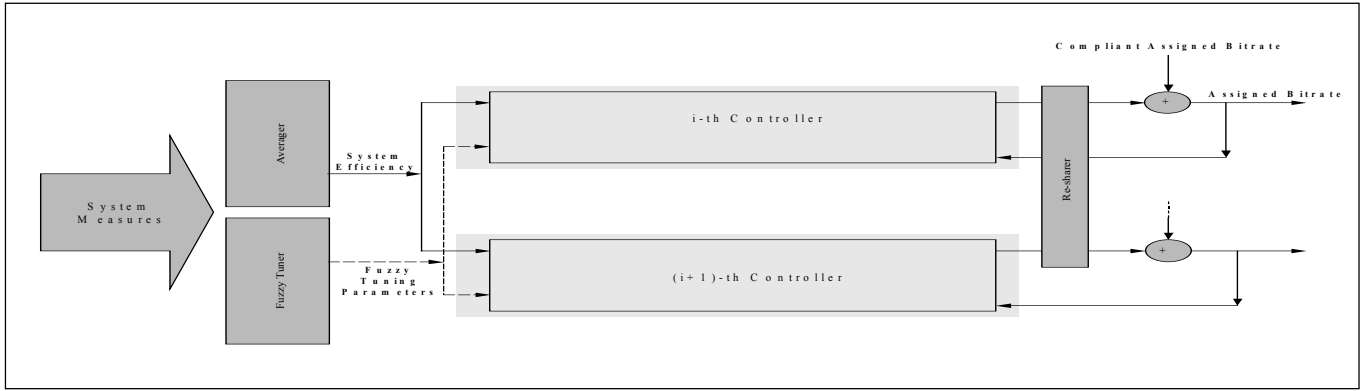


Fig. 2 – Generic scheme of the MAC control system

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. 2 and Fig. 3, and how they affect the behaviour of the control action.

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

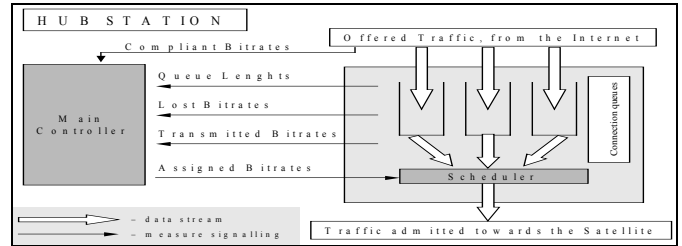


Fig. 4 – Incoming traffic scheme.

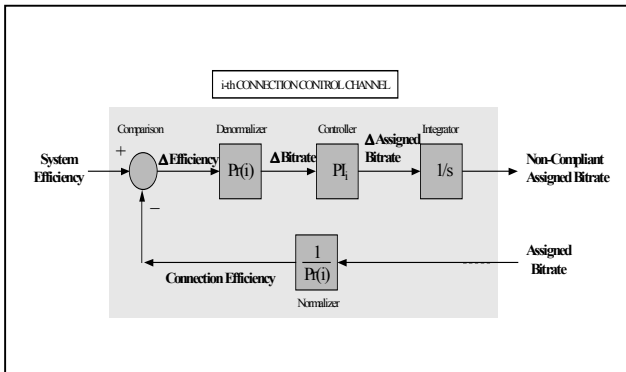


Fig. 3 – Controller scheme.

## II. GENERAL OVERVIEW

The schematisation of the HS of Fig. 1 is shown in Fig. 4. An **Offered Traffic** comes from the Internet in the form of IP packets which are enqueued into one of the **Queues**, based on the flow it belongs to. The MAC controller 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  $T_{CONTROL}$  seconds through various measures: the filling up of the queues (*Queue Lengths* in Fig. 4) 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 Connection (if the connection requires it), 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  $T_{CONTROL}$  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  $T_{CONTROL}$  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 higher layer Scheduler to deal with the discrete size of the queued packets to realize the equilibrium calculated by the Main Controller. The SATIP6 Protocol Stack is depicted in Fig. 5.

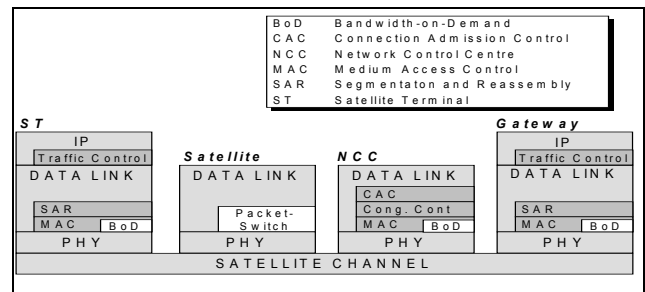


Fig. 5 -.SATIP6 Protocol Stack

### 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 Non-Compliant Bit-rate. In the first stage the controller assigns a part of the *Total System Bit-rate* according to the QoS contract of each connection, while in the second it assigns the remaining available bit-rate according to an original algorithm described in detail later on. At the end of the two stages the sum of these two assigned values will make up the new *Assigned\_Bit-rate(i)* for each connection – this value will be used throughout the whole  $T_{\text{CONTROL}}$  period that follows its calculation.

#### A. Compliant and Non-Compliant Bit-rate Assignment

In the first stage the system checks the *Offered\_Bit-rate(i)* of each connection – that is to say the average bit-rate offered by the connection in the  $T_{\text{CONTROL}}$  period – and compares it with its *Compliant\_Bit-rate(i)*, bestowing onto the connection a *Compliant\_Assigned\_Bit-rate(i)* equal to the lesser of the two values (so as to comply with the QoS contract while avoiding waste of bandwidth). The rationale behind this choice is to assign output bit-rate on the base of the input (*offered*) bit-rate.

In the second stage the system calculates the *Spare System Bit-rate*, which represents what is left of the *Total System Bit-rate* after the assignment of the *Compliant\_Assigned\_Bit-rates* (this value is always positive since the sum of the *Compliant\_Bit-rates* is always less than the *Total System Bit-rate*, as guaranteed by the Connection Admission Control (CAC)). The *Spare System Bit-rate* is shared by the various *Non-Compliant\_Assigned\_Bit-rates*.

At the end each connection will have a *Compliant\_Assigned\_Bit-rate(i)* and a *Non-Compliant\_Assigned\_Bit-rate(i)* which will be summed to obtain the final *Assigned\_Bit-rate(i)*.

#### B. The control algorithm

This second stage is the core of the control system that we are exposing in this article.

The approach used in this stage is different from that of the previous one: the *Spare System Bit-rate* is assigned on the base of a measure of the length of the connection queues. The algorithm evaluates how well the connections are served, in comparison with the congestion of the whole system, and assigns them greater or lesser shares of the *Spare System Bit-rate*. This approach can obtain fairness and efficiency, as will be shown later.

##### B.1. Definitions of Priority and Connection Efficiency

The first step of the algorithm is the definition of a connection's "capacity to offer traffic". We called it

*Priority*, given its meaning of "need for bandwidth" (to avoid the expiration of packets)

$$\text{Priority}(i) = \frac{\text{Queue\_Length}(i)}{T_{\text{CONTROL}} + \text{Transmitted\_Bit-rate}(i) + \text{Lost\_Bit-rate}(i)} \quad (1)$$

Consequently we define the efficiency with which this traffic capacity is served by dividing the *Assigned\_Bit-rate(i)* by *Priority(i)*. We call this *Connection Efficiency*. Great values of *Connection Efficiency* mean that the bit-rate assigned to a connection satisfies to a great extent the need for bandwidth of the connection, which is expressed by *Priority(i)*. This ratio is enacted by the component named "Normalizer" in Fig. 3. In fact it normalizes the bit-rate assigned to a connection by its priority.

##### B.2. Definition of System Efficiency

After introducing the two previous values we extend the idea to the whole system. What we need is a reference value that expresses a theoretical and ideal condition to which all the connections should tend – we will compare *Connection Efficiency* to it, as can be seen in Fig. 3 under the heading "Comparison".

The simplest and most efficient approach is to consider the system's *Priority* as the sum of all the connections' *Priorities*. The *Total\_System\_Bit-rate* is divided by this global *Priority* to obtain the *System\_Efficiency*.

These calculations are made by the component named "Averager" in Fig. 2.

##### B.3. Efficiency Comparison and Control Action

The whole idea behind the controller is to calculate a level of efficiency for each connection, compare it with the system's overall efficiency and use the difference to evaluate how much bandwidth to assign or subtract to each connection.

The algorithm proceeds like this for each connection:

1. *System\_Efficiency* and *Connection\_Efficiency(i)* are compared to see if the connection is under-served or over-served. The result is  $\Delta\_Efficiency(i)$ , which represents the delta that must be imposed to the connection efficiency to align it with the system's level. Positive deltas mean that the connection is under-served and must increase its efficiency, and the other way around.
2.  $\Delta\_Efficiency(i)$  is multiplied by *Priority(i)* to obtain  $\Delta\_Bit-rate(i)$ , which represents the change in bit-rate assignment to be given to the i-th connection to bring its efficiency on par with the system's.
3.  $\Delta\_Bit-rate(i)$  is fed to a PI controller to enhance the control action. The result is  $\Delta\_Assigned\_Bit-rate(i)$ . The PID controller will be discussed later in paragraph B.4.
4.  $\Delta\_Assigned\_Bit-rate(i)$  is summed to the *Non-Compliant\_Assigned Bit-rate* of the connection.

5. After a precautional non-linear resharing to avoid negative *Non-Compliant Assigned Bit-rates*, that could subtract to the *Compliant Assigned Bit-rates*, these two values are summed to obtain the *Assigned Bit-rate(i)*.

The conclusion of the calculation is the *Assigned Bit-rate(i)* value, which defines the total bit-rate that the *i*-th connection will receive in the following  $T_{\text{CONTROL}}$  period.

#### B.4. PID control and Fuzzy tuning

We used a PID controller to enhance the control action: this component has the property of enacting a faster action on the process it controls. The tuner that controls the PID's parameters realizes three different rule-sets: Proportional, Integral and Derivative

The first set concerns the tuning of the overall policy of the controller, which is obtained by changing the  $K_P$  constant. The various connections strive to obtain a common and limited bandwidth, and in situations of great congestion there is little bandwidth to be shared: the aggressiveness of the control must be tuned down 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. 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 the connections that are over-served:

$$Need = \sum_i \max[\Delta\_Assigned\_Bit-rate(i), 0] \quad (2)$$

$$Surplus = \sum_i \max[-\Delta\_Assigned\_Bitrate(i), 0] \quad (3)$$

$$Congestion = Need / (Need + Surplus) \quad (4)$$

Utilizing simple triangular fuzzy sets and the Sugeno deduction, we implement a rule that decreases  $K_P$  as *Congestion* increases. This rule prevents newly established connections from pulling pre-existing connections away from the equilibrium.

The second and third fuzzy rules come from the usual principles of PID tuning (see [3], [4] and [5]). 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.

#### C. Scheduler & Bit Credit

The scheduler of Fig. 4 has the duty of imposing an equilibrium on the system that resembles as much as possible the theoretic one calculated by the Main Controller. The discrete nature of the IP packets makes the realized equilibrium always somewhat different from the theoretic one. This implies that oftentimes some connections receive more bandwidth than they were assigned, while some others are underserved. This phenomenon cannot be completely avoided, but it does not pose such a great problem if the theoretic equilibrium is

realized as an average on the long period. The goal is to keep the error near to zero.

The quantitative solution to realize this mechanism is to implement a bit credit system. In the generic  $T_{\text{CONTROL}}$  sampling interval, a connection is bestowed a certain *Assigned Bit-rate(i)*. This means that in that period it should transmit packets for a total of *Assigned Bit-rate(i)*\*  $T_{\text{CONTROL}}$  bits. But the connection is likely to either use more or less bits. Consequently it may be useful to allow it to respectively make a bit "debt" or to "save" its bit credit. Therefore we create a *Bit\_Credit(i)* variable for each connection: every  $T_{\text{CONTROL}}$  seconds we increment the bit credit by *Assigned Bit-rate(i)*\*  $T_{\text{CONTROL}}$ , and every time it transmits a packet we decrement it by the packet's size. The goal is to keep the credit near 0 as much as possible, since if *Bit\_Credit(i)* is exactly equal to 0 it means that we have perfectly realized the theoretic equilibrium. The bit credit solution allows to use the assigned bit-rate independently of the period in which it was assigned.

An algorithm is needed to choose which connection must be selected to transmit the next packet. The solution we conceived uses a data structure made up of three queues in which the connection handles are continuously reordered so as to serve each of them efficiently. The goal of the algorithm is to offer an enhanced version of circular scheduling tailored to the needs of our system. The reason to seek an enhancement is that when simple circular scheduling is applied to the connections, a connection that misses its turn must wait for another full round before being given another chance to transmit. The triple-queue scheduler solves this problem: Queue 1 is the basis for the circular FIFO algorithm. Queue 2 keeps track of those connections that have missed their turn. Queue 0 handles inactive connections. Further Queues may be implemented to enhance the algorithm. A connection handler cannot be simultaneously in more than one queue.

The algorithm follows this approach:

1. It scans the queue structure and picks the first connection in the highest non-empty queue.
2. If this connection can transmit a packet it does so and then it is put back at the end of queue 1.
3. If it cannot transmit a packet – because it has negative bit credit – it is put in an **higher** queue, so as to allow it to regain bit credit while the next connection in the queue structure is selected to transmit.

The emphasis is on this third step: since a) those connections that must recharge their Bit Credit are put in higher queues and b) higher queues are checked more often, the algorithm manages to keep bit credit around zero for all the connections.

The algorithm uses Queue 0 to manage inactive connections, that is to say connections without packets to transmit. It scans all the connections each turn and does the following:

1. moves the ones which have no packets to transmit in Queue 0.

2. moves the ones that are in Queue 0 and that have new packets to transmit to the end of Queue 1.

In cases of excessive congestion it may happen that all the connections are constantly moved to higher queues and no one is able to transmit. The result is that all the connections end up in the top queue: then the transmission order is ignored and the queue with the **higher Bit Credit** is selected to transmit. This last step somehow breaks the theoretic equilibrium, but the sacrifice is necessary for the sake of obtaining a greater link efficiency.

As a conclusion, this algorithm offers a special FIFO circular scheduling that allows connections to rearrange themselves to minimize the waiting delay.

#### IV. SIMULATIONS

The algorithm has been simulated with the OPNET SW tool by MIL3. 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. 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.

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

Figure 6 shows 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 [6] 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.

Infringement of the QoS contracts is null up to almost 100% *System Congestion*, which is another relevant result.

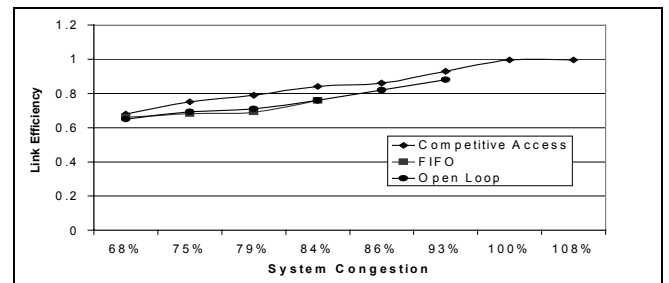


Fig. 6 – Compared Link Efficiencies

#### 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.

#### REFERENCES

- [1] SATIP6 project, IST-2001-34344, <http://satip6.tilab.com/>
- [2] Alain Debray et al., "SATIP6 project Scenarios", submitted to IST Mobile Summit 2003.
- [3] K.J. Astrom and T. Hagglund, Automatic Tuning of PID controllers, Instrument Society of America, 1988.
- [4] K.J. Astrom and T. Hagglund, PID Controllers: Theory, Design and Tuning, Instrument Society of America, 1995.
- [5] V.B.Bajić, Simple rule-based and fuzzy controllers, Technical Report, TR95-07, Control Laboratory, Technikon Natal, RSA, 1995.
- [6] Design and Evaluation of a Feedback Control EDF Scheduling Algorithm, Chenyang Lu, John A. Stankovic, Gang Tao, Sang H. Son.
- [7] Differentiated Service Router Architecture-Classification, Metering and Policing. Daniel Lin and Frank Akujobi, pag.21-23 on WFQ