

# Handling Concurrent Admission Control in Multiservice IP Networks

Solange Rito Lima, Paulo Carvalho, and Vasco Freitas

University of Minho

Department of Informatics

4710-057 Braga, Portugal

Email: {solange,pmc,vf}@di.uminho.pt

**Abstract**— This paper debates the problem of handling concurrent admission control decisions in multiservice networks, putting forward solutions to mitigate the negative impact that distributed admission of flows might have on the service level guarantees provided to network customers. Keeping in mind that simplicity is a key factor for deployable solutions, we suggest and discuss the use of (i) a service-dependent concurrency index; (ii) a token-based system and (iii) a rate-based credit system, as alternative or complementary proposals to minimize or solve QoS degradation resulting from AC false acceptance.

## I. INTRODUCTION

Providing QoS in the Internet is a multilevel problem involving enhanced QoS-aware applications, communication protocols and technologies. Given that new policies, rules and traffic control mechanisms have to be in place, a major objective to keep in mind, and likely a key aspect for their deployment in real networks, is to maintain the network control plane complexity as low as possible. In this way, recognized the relevant role of admission control (AC) in multiservice IP networks [1], [2], [6], a lightweight and distributed AC model based on on-line QoS monitoring feedback for managing multiple services quality has been proposed in [8] and recently formalized in [9]. This model also allows to control the utilization of Service Specification Levels (SLSs) both intra and interdomain.

Although, distributed admission control has been widely covered in the literature [4], [5], [7], [8], the problem of concurrent AC has been rarely tackled. The need for handling concurrency, which stems from the distributed nature of AC decisions, is justified as a way to avoid the over/false acceptance of flows entering the network and, consequently, service degradation.

In [3], a given amount of bandwidth, called AC Limit, is defined as a reference value for the acceptable traffic within a class. AC Limits are defined off-line at an initial provisioning phase taking as input: (i) the network topology (ii) the long-term expected traffic matrices; and (iii) the bandwidth sharing policies among classes. The initial static limits can be extended dynamically by sharing unused AC Limits between egress routers.

In this paper, reporting on-going work, we extend these concepts and point out several new proposals that may be adopted to control the admission of concurrent flows, so that the service

level guarantees negotiated with customers are protected from overacceptance. These alternative or complementary proposals include the definition of:

- a per-class concurrency index;
- a token-based system;
- a rate-based credit system controlled by egress nodes.

In the latter approach, an amount of rate credits is assigned to each ingress node, considering: (i) the negotiated (upstream and downstream) SLSs; (ii) the dynamic negotiation of new SLSs; (iii) the AC of flows when sustained or not by an individual SLS. In this system, the concurrency problem is implicitly considered and minimized as long as each ingress node maintains its flows' acceptance level within the available credits of each service class.

The remaining of this paper is structured as follows: the multiservice AC model characteristics, the main network domain entities and the AC criteria are summarized in Section II, the study of concurrent AC and the proposals for tackling concurrency shortcomings are debated in Section III; finally, the conclusions are drawn in Section IV.

## II. AC MODEL SPECIFICATION

### A. Model Overview

This section provides a brief overview of the multiservice AC model proposed in [8], [9] before debating AC concurrency. This model resorts to edge-to-edge on-line monitoring to obtain feedback of each service class's performance so that proper AC decisions are made. To control dynamically customer traffic entering a network domain, the model's underlying AC rules control *both* QoS levels in the domain and the sharing of active SLS between domains. While ingress routers perform explicit or implicit AC depending on the application type and corresponding service class, egress routers perform on-line QoS monitoring and SLS control. *On-line QoS Monitoring*, carried out on an ingress-egress basis, measures specific metrics for each service type, providing a quantitative view of the service level available from each ingress node. *SLS Control* monitors the usage of downstream SLSs at each egress, to ensure that traffic to other domains does not exceed the negotiated profiles. The obtained measures are sent to the corresponding ingress routers periodically to update an Ingress-Egress service matrix used for distributed AC and active service management.

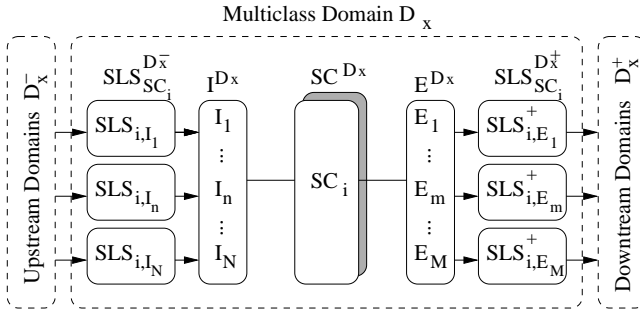


Fig. 1. Domain Elements and Notation

The *end-to-end case*, detailed in [9], is viewed as a repetitive and cumulative process of AC and available service computation only performed at ingress nodes.

### B. Multiservice Domain Specification

Following [9], we specify the following domain entities: (i) service classes; (ii) upstream SLSs; (iii) downstream SLSs and (iv) traffic flows. Network resources are implicitly considered and controlled by the edge-to-edge monitoring process.

1) *Service Classes Specification*: Considering a multiclass domain  $D_x$  comprising  $N$  ingress nodes and  $M$  egress nodes, we define  $I^{D_x} = \{I_1, I_2, \dots, I_N\}$  and  $E^{D_x} = \{E_1, E_2, \dots, E_M\}$  as the set of ingress and egress nodes, respectively<sup>1</sup>. For this domain, we represent the set of supported service classes as  $SC^{D_x} = \{SC_1, SC_2, \dots, SC_Y\}$ . For each service class  $SC_i \in SC^{D_x}$ , the set of QoS parameters under control is defined as  $P_{SC_i} = \{(P_{i,1}, \beta_{i,1}), \dots, (P_{i,P}, \beta_{i,P})\}$  where each  $P_{i,p} \in P_{SC_i}$  is the class parameter target value and  $0 \leq \beta_{i,p} \leq 1$  is the parameter's safety margin. Each parameter's upper bound or threshold, given by  $T_{i,p} = \beta_{i,p} P_{i,p}$ , is used to trigger AC.

In practice, the service classes to be supported in  $D_x$  are closely related to the service levels negotiated with both upstream and downstream customers. Thus,  $D_x$  is a service provider for an upstream domain  $D_x^-$  and a customer of a downstream domain  $D_x^+$ . Lets now consider that  $SLS_{i,I_n}$  identifies a specific SLS accepted for  $SC_i$  with upstream domain  $D_x^-$ , connecting  $D_x$  through  $I_n$ , and  $SLS_{i,E_m}^+$  identifies a specific SLS negotiated for  $SC_i$  with downstream domain  $D_x^+$ , accessible from  $D_x$  through  $E_m$  (see Fig. 1).

2) *Upstream SLSs Specification*: The definition of SLSs [10], apart from being a key aspect for QoS provisioning, provides a valuable input for AC, in special, when admission spans multiple domains. From an AC perspective, an upstream SLS for service class  $SC_i$ , i.e.,  $SLS_{i,I_n}$ , includes elements such as:

1.  $SLS_{i,I_n} \rightarrow Scope$  is specified as a pair  $(I_n, E')$  where  $I_n$  is the access point of the upstream domain  $D_x^-$  to  $D_x$  and  $E' \subseteq E^{D_x}$  represents all possible egress nodes  $E_m$  providing access from  $D_x$  to  $D_x^+$  for this SLS.

<sup>1</sup>To simplify the notation, and without losing generality, each ingress or egress distinct interface is treated as a virtually distinct ingress  $I_n$  or egress node  $E_m$ .

2.  $SLS_{i,I_n} \rightarrow SC_{id}$  identifies the service type to be provided by  $D_x$  to packets belonging to  $SLS_{i,I_n}$ .

3.  $SLS_{i,I_n} \rightarrow TrafficProfile$  specifies the traffic characteristics of  $SLS_{i,I_n}$ , allowing to identify traffic as in or out-of-profile. The rate  $R_{i,I_n}$  represents the global aggregate rate established for  $SLS_{i,I_n}$  within the scope region.

4.  $SLS_{i,I_n} \rightarrow ExpectedQoS$  specifies the expected QoS parameters for  $SLS_{i,I_n}$ , i.e.,  $P_{SLS_{i,I_n}} = \{P_{i,I_n,1}, \dots, P_{i,I_n,P'}\}$ . Each QoS parameter  $P_{i,I_n,p}$  value is bounded by the corresponding service class  $P_{i,p}$ , regardless the incoming  $I_n$  and accepted  $SLS_{i,I_n}$ . Embedding the expected SLS parameters values in the respective class parameter target values is of paramount importance as QoS and SLS control in the domain is clearly simplified. Examples of  $P_{i,I_n,p}$  are  $IPTD_{i,I_n}$ ,  $ipdv_{i,I_n}$ ,  $IPLR_{i,I_n}$ .

5.  $SLS_{i,I_n} \rightarrow ServSched$  determines the time interval  $[t_{i,I_n,0}, t_{i,I_n,f}]$  in which the service is due to be scheduled.

3) *Downstream SLSs Specification*: In a domain  $D_x$ , when defining and negotiating an SLS with a downstream domain  $D_x^+$ , i.e., an  $SLS_{i,E_m}^+$ , the contracted service from an egress node  $E_m$  should foresee the provision of adequate service levels taking into account all active  $SLS_{i,I_n}$  going through  $E_m$ . From an  $E_m$  perspective, specifying a downstream  $SLS_{i,E_m}^+$  follows the SLS template and notation introduced above for upstream SLSs, adding the sign +.

4) *Flow Specification*: Depending on each application ability for signaling its service requirements, a traffic flow  $F_j$  may undergo either implicit or explicit AC. For implicit AC, the relevant fields to consider include the source, destination and service class identifiers, i.e.,  $Src_{id}$ ,  $Dst_{id}$ ,  $SC_{id}$ . For explicit AC, in addition to these fields, specifying a flow includes defining the *TrafficProfile*, the required QoS parameters  $ReqQoS$  and an optional  $QoSTolerance$ . Their notation is similar to the one introduced for  $SLS_{i,I_n}$ .

### C. AC Criteria Specification

The service-dependent AC criteria resort to (i) rate-based SLS control rules and (ii) QoS parameters control rules. These rules follow the notation introduced in Sec. II-B.

**Rate-based SLS Control Rules** - For each ingress node  $I_n \in I^{D_x}$  and each egress node  $E_m \in E^{D_x}$  one or more SLSs can be in place. As each  $SLS_{i,I_n}$  and  $SLS_{i,E_m}^+$  have specified a negotiated rate,  $R_{i,I_n}$  and  $R_{i,E_m}^+$  respectively, a rate-based Measure-Sum algorithm can be applied to control SLSs utilization at each network edge node.

*Explicit AC* - At each ingress node  $I_n$ , verifying if a new flow  $F_j \in SLS_{i,I_n}$  can be admitted involves testing if the  $SLS_{i,I_n}$  can accommodate the new flow traffic profile, i.e.,

$$\tilde{R}_{i,(I_n,*)} + r_j \leq \beta_{i,I_n} R_{i,I_n} \quad (1)$$

In Eq. (1),  $\tilde{R}_{i,(I_n,*)}$  is the current measured rate of flows using  $SLS_{i,I_n}$ ;  $r_j$  is the rate of the new flow  $F_j$ ;  $0 < \beta_{i,I_n} \leq 1$  is a safety margin defined for the negotiated rate  $R_{i,I_n}$ .

When the destination of flow  $F_j$  is outside  $D_x$ , verifying if the new flow can be admitted also involves testing if the

downstream  $SLS_{i,E_m}^+$  can accommodate the new flow traffic profile, i.e.,

$$\tilde{R}_{i,(*,E_m)}^+ + r_j \leq \beta_{i,E_m}^+ R_{i,E_m}^+ \quad (2)$$

In Eq. (2),  $\tilde{R}_{i,(*,E_m)}^+$  is the current measured rate of flows using  $SLS_{i,E_m}^+$ , considering all ingress-to- $E_m$  estimated rates of flows going through  $E_m$ , i.e.,  $\tilde{R}_{i,(*,E_m)}^+ = \sum_{k=1}^N \tilde{r}_{i,(I_k,E_m)}$ ;  $r_j$  is the rate of the new flow  $F_j$ ;  $0 < \beta_{i,E_m}^+ \leq 1$  is the safety margin for the rate  $R_{i,E_m}^+$  defined in  $SLS_{i,E_m}^+$ . This safety margin determines the degree of overprovisioning for  $SC_i$ . The value of  $\beta_{i,E_m}^+$  may result from high-level domain policies defined at service class level, instead of being defined at SLS level.

The rate control rules for the admission of flows not sustained by an SLS, i.e.,  $F_j \notin SLS_{i,I_n}$ , resort to Eq. (2) using the measured rate  $R_{i,I_n}^{\notin SLS}$ , i.e.,  $\tilde{R}_{i,(I_n,*)}^{\notin SLS} + r_j \leq \beta_{i,I_n}^{\notin SLS} R_{i,I_n}^{\notin SLS}$ .

**Implicit AC** - For a service class  $SC_i$  under implicit AC, as flows are unable to describe  $r_j$ , the SLS control equations defined above become similar to the QoS control equation (Eq. (3)), considering  $P_{i,p}$  as a rate-based parameter. Therefore, traffic flows are accepted or rejected implicitly according to the value of a variable  $AC\_Status_{\Delta t_i}$  computed once for  $\Delta t_i$ .

**QoS Parameters Control Rules** - At each ingress node  $I_n$ , the  $AC\_Status_{\Delta t_i}$  variable, used to control the admission of new flows in the monitoring interval  $\Delta t_i$ , is updated after checking the controlled parameters  $P_{i,p}$  of  $SC_i$ , provided by egress nodes, against the corresponding pre-defined thresholds  $T_{i,p}$ , i.e.,

$$\forall (P_{i,p}, \beta_{i,p}) \in P_{SC_i} : \tilde{P}_{i,p} \leq T_{i,p} \quad (3)$$

where  $\tilde{P}_{i,p}$  is the measured value of  $P_{i,p}$  for  $\Delta t_i$ , and  $T_{i,p}$  is the parameter's threshold, as explained in Sec. II-B.1. Eq. (3) is not flow dependent, i.e., it is checked once during  $\Delta t_i$  to determine  $AC\_Status_{\Delta t_i}$ . The  $AC\_Status_{\Delta t_i}$  - *accept* - indicates that the measured QoS levels for  $SC_i$  are in conformance with the QoS objectives and, therefore, new flows can be accepted. The  $AC\_Status_{\Delta t_i}$  - *reject* - indicates that no more flows should be accepted until the class recovers and restores the QoS target values. This will only be checked at  $\Delta t_{i+1}$ .

### III. HANDLING CONCURRENCY

A distributed AC model may involve multiple ingress routers making concurrent AC decisions. Therefore, dealing with concurrency is a key aspect to avoid over or false acceptance. In fact, within a measurement time interval  $\Delta t_i$ , each ingress node  $I_n$  makes AC decisions based on measures estimated for the interval, without knowing the contribution of other ingress nodes to the metrics variation until  $\Delta t_{i+1}$ , i.e., when the next measuring update takes place<sup>2</sup>.

<sup>2</sup>In order to maintain simplicity, reduce overhead and latency associated with the exchange of control information, during  $\Delta t_i$  each  $I_n$  only knows (i) the initial measures provided by each  $E_m$  for that time interval and (ii) its own contribution for the rate metric variation.

The presence of concurrency affects both the measured utilization of the rate related variables (e.g.,  $\tilde{R}_{i,E_m}^+$ ) shared among ingress nodes and the QoS measures. Note that, although these QoS measures reflect the available service between each  $(I_n, E_m)$  pair, the links in the corresponding path may carry traffic resulting from a different pair of nodes. Therefore, the acceptance decisions at any other ingress node  $I_{n'} \neq I_n$  may affect the measured QoS for a specific  $(I_n, E_m)$  pair.

The problem of mis-acceptance within each service class can be reduced resorting to larger safety margins ( $\beta_{i,E_m}^+$ ,  $\beta_{i,p}$ ) to absorb the effect of traffic load fluctuations resulting both from the inherent statistical properties of traffic and from concurrent AC. Here, to reduce or solve the negative effects of concurrent AC might have on service offering, we explore and debate solutions such as: (A) the definition of a concurrency index based on the number of concurrent ingress nodes, affecting explicitly the rate control rules; (B) a token-based system to rule and limit the number of simultaneous AC decisions; (C) a rate-based credit system to control each  $I_n$  admission capacity. These solutions are not mutually exclusive as, for instance, a concurrency index may complement a token-based system. The following topics explore these scenarios, regarding the control of  $SLS_{i,E_m}^+$  utilization.

#### Initial AC scenario

In the case of explicit AC, we consider that satisfying the inequality  $\tilde{R}_{i,(*,E_m)}^+ + r_j \leq \beta_{i,E_m}^+ R_{i,E_m}^+$  determines a positive AC decision. When a new flow acceptance occurs,  $\tilde{R}_{i,(*,E_m)}^+$  can be updated by considering  $r_j$  at the corresponding  $I_n$ , assuring that  $I_n$  does not accept more traffic than the estimated available rate for  $SLS_{i,E_m}^+$  during  $\Delta t_i$ <sup>3</sup>. However, assuming that other concurrent ingress nodes are in place, the total new load is temporarily unknown and the available rate at  $SLS_{i,E_m}^+$  may be exceeded.

#### A. Concurrency index

Considering  $\tilde{I}$  the set of concurrent ingress nodes sharing a common  $SLS_{i,E_m}^+$ , the estimation of  $SLS_{i,E_m}^+$  available rate for  $\Delta t_i$  can be protected by a concurrency index  $\chi_{i,E_m}$ , which depends on the cardinality of  $\tilde{I}$ . In this way, explicit flow AC is ruled by

$$r_j \leq \frac{\beta_{i,E_m}^+ R_{i,E_m}^+ - \tilde{R}_{i,(*,E_m)}^+}{\chi_{i,E_m}} \quad (4)$$

where  $\beta_{i,E_m}^+ R_{i,E_m}^+ - \tilde{R}_{i,(*,E_m)}^+$  represents the estimated available rate of  $SLS_{i,E_m}^+$  to be shared among concurrent ingress nodes. In the case of implicit AC, a similar use of  $\chi_{i,E_m}$  can be applied.

<sup>3</sup>Updating rate estimations leads to a more conservative AC as the rates of new flows are considered but the compensation effect of flows' departure is not taken into account. Keeping rate estimation at  $I_n$  unchanged during  $\Delta t_i$  explores this compensation effect but may increase over acceptance.

## B. Token-based system

Other possible solution to control the number of concurrent ingress nodes performing AC decisions may follow a token-based system, where the level of concurrency allowed is determined by the number of tokens available. In this system, only ingress nodes holding a token can accept new flows in  $\Delta t_i$ . In the limit, when a single token is available in the system, no concurrency is allowed. Nevertheless, if during  $\Delta t_i$  the tokens pass through several ingress nodes, the  $SLS_{i,E_m}^+$  utilization can change without common knowledge of all concurrent nodes,  $\tilde{I}$ . Consequently, overacceptance may still occur. To cope with this, tokens can be used to carry  $SLS_{i,E_m}^+$  updates. If the token assignment remains unchanged during  $\Delta t_i$ , this time interval needs to be carefully defined as it influences directly the domain QoS stability and load balancing, and the AC latency at ingress nodes without tokens.

Apart from the conceptual simplicity of a token-based model to control concurrency, this method reduces the problem but does not solve it completely. Additionally, the signaling required for token exchange among ingress nodes and the time required for  $I_n$  to get a token, which depends on the number of available tokens and the number of concurrent nodes  $|\tilde{I}|$ , may be prohibitive.

## C. Rate-based credit system

To reduce the underlying drawbacks of a token-based system, the strategic view an egress node has of each measured rate  $\tilde{r}_{i,(I_n,E_m)}$  can be used to implement a rate-based credit system to control the bandwidth usage of ingress nodes and, implicitly, concurrency.

Following the defined AC model strategy, in the proposed rate-based credit system, the monitoring information obtained at egress node  $E_m$  is used to control the amount of credits assigned to  $I_n$ , from an  $(I_n, E_m)$  and service class  $SC_i$  perspective. Each egress  $E_m$  manages a pool of unused credits in order to distribute spare resources (bandwidth) dynamically as a complement to the static credit assignment initially defined, considering the ingress nodes grouped into distinct topological areas. The amount of available credits to be shared by ingress nodes that want to reach a specific egress  $E_m$ , therefore, controlled by this one, should consider (i) the network topology, the underlying bottleneck capacity<sup>4</sup> and network core multiplexing effects; (ii) the bandwidth sharing policies among classes [3]; (iii) the already accepted  $SLS_{i,I_n}$  and the corresponding expected traffic matrix; (iv) the  $SLS_{i,E_m}^+$  negotiated rate or the capacity allocated at  $E_m$  for  $SC_i$ ; (v) a safety margin of unused credits at each  $I_n$  to assure that  $I_n$  has a controlled autonomy to make acceptance decisions during  $\Delta t_i$ .

<sup>4</sup>Network bottleneck can be hard to define as it changes dynamically. Different pairs of  $(I_n, E_m)$  may share and be limited in rate by a known bottleneck link; however, a new bottleneck may occur in a different place depending on traffic load and  $(I_n, E_m)$  pairs involved. This concept is not new and is usually expressed by metrics such as: (i) available capacity and (ii) available bandwidth. For an initial credit assignment, (i) determines the bottleneck link between  $(I_n, E_m)$ .

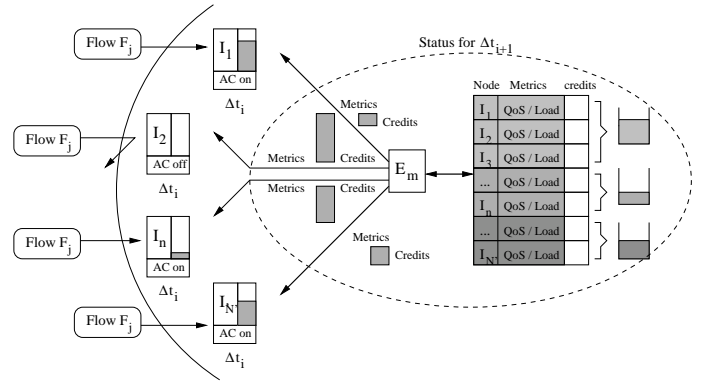


Fig. 2. Rate-based credit system

At each  $\Delta t_i$ , ingress nodes may receive new credits using the QoS metric dissemination process. When an egress node  $E_m$  provides new measures to an ingress node  $I_n$ , it can distribute new credits too, i.e., no specific or additional control messages are needed (see Fig. 2). This strategy avoids several drawbacks of the solution proposed in [3], such as horizontal sharing of credits, use of specific signaling between ingress nodes and holding to many unused resources at each  $I_n$ .

The management of credits can be either measurement-based or explicit, with credits being captured and released according to  $SLS_{i,I_n}$  acceptance and termination. A possible measurement-based approach for managing the distribution of rate credits is detailed in Algorithm 1.

In an explicit approach, two scenarios can be devised: (i) each ingress node  $I_n$  informs explicitly the egress node  $E_m$  of the amount of credits captured or released, keeping the credits captive during the service scheduling period defined in the  $SLS_{i,I_n}$ ; (ii) each egress  $E_m$  uses the measured rate  $\tilde{r}_{i,(I_n,E_m)}$  to determine when  $I_n$  needs additional credits, waiting for an explicit teardown before releasing credits previously assigned. This avoids removing temporarily unused credits of SLSs still active, assuring that new incoming flows  $F_j \in SLS_{i,I_n}$  have credits available.

At domain egress nodes, the amount of available credits in the pool may change for different reasons. Credits are increased when: (i) the negotiated rate  $R_{i,E_m}^+$  and/or the links' capacity are upgraded; (ii) an  $SLS_{i,I_n}$  having  $E_m$  within its scope expires (explicit case, with  $I_n$  returning credits back); (iii) the egress  $E_m$  senses a rate utilization decrease at  $I_n$ , recovering excess credits (measurement-based case). The amount of available credits is decreased in favor of one  $I_n$  when: (i)  $I_n$  is running out of credits, i.e., its previous credit assignment is reaching an usage limit; this can be sensed by egress  $E_m$  when measuring the rate  $\tilde{r}_{i,(I_n,E_m)}$  or (ii) an explicit request occurs from  $I_n$ <sup>5</sup>.

<sup>5</sup>Specific requests of credits from  $I_n$  to  $E_m$  during  $\Delta t_i$  can also be considered, however, it changes the initial concept and assumption of viewing  $\Delta t_i$  as a black-box, reflecting a measurement steady state.

---

**Algorithm 1: Measurement-based Credit Management**

---

/\* Available Rate Credits at  $I_n$  for  $SC_i$  at the end of  $\Delta t_i$  \*/

$$RC_{i,(I_n,E_m)}^{avail} = RC_{i,(I_n,E_m)} - \tilde{r}_{i,(I_n,E_m)}$$

/\* Updating credits for  $\Delta t_{i+1}$  \*//\* if  $RC_{i,(I_n,E_m)}^{avail} < \beta_{i,RC}$ : credits under limit, new credits are distributed \*//\* if  $RC_{i,(I_n,E_m)}^{avail} > \beta_{i,RC}$ : credits over limit, excess credits return to the pool \*//\* if  $RC_{i,(I_n,E_m)}^{avail} < 0$ :  $RC_{i,(I_n,E_m)}$  underestimated, new credits are distributed \*/

$$RC_{i,(I_n,E_m)}^{new} = \beta_{i,RC} - RC_{i,(I_n,E_m)}^{avail}$$

$$CredPool_{i,E_m} = CredPool_{i,E_m} - RC_{i,(I_n,E_m)}^{new}$$

$$RC_{i,(I_n,E_m)} = RC_{i,(I_n,E_m)} + RC_{i,(I_n,E_m)}^{new}$$

---

**Notation:** $RC_{i,(I_n,E_m)}$ : rate credits available at  $I_n$  for  $SC_i$ ;  $E_m$  maintains this information to determine  $RC_{i,(I_n,E_m)}^{new}$  $RC_{i,(I_n,E_m)}^{avail}$ : remaining credits at  $I_n$  according to the estimated rate usage  $\tilde{r}_{i,(I_n,E_m)}$  $RC_{i,(I_n,E_m)}^{new}$ : credits update for  $\Delta t_{i+1}$  $\beta_{i,RC}$ : safety margin of unused credits at ingress nodes for  $SC_i$ . It is service-dependent and defined in kbps $CredPool_{i,E_m}$ : pool of credits at  $E_m$  for  $SC_i$ , shared among  $\tilde{I} \subseteq I^{D_x}$  concurrent nodes

---

### Decoupling AC Decisions

Controlling the utilization of  $R_{i,E_m}^+$  during  $\Delta t_i$  and, consequently, the concurrency control of  $SLS_{i,E_m}^+$  can be simplified if the AC module and corresponding tasks are divided between ingress and egress nodes. Instead of controlling the rates  $R_{i,I_n}$ ,  $R_{i,I_n}^{SLS}$  and  $R_{i,E_m}^+$  at  $I_n$ , the control of  $R_{i,E_m}^+$  can be passed to  $E_m$ . For example, a flow request  $F_j \in SLS_{i,I_n}$  crossing the domain  $D_x$  is accepted at  $I_n$  if Eqs. (1) and (3) are satisfied. When arriving at egress  $E_m$ ,  $F_j$  is accepted and may be forward to the next domain if Eq. (2) is satisfied. When it is rejected at  $E_m$ , a reject notification due to  $R_{i,E_m}^+$  underestimation or by an incorrectly defined statistical multiplexing factor may be reported.

Decoupling AC between  $I_n$  and  $E_m$  nodes, apart from being conceptually correct as  $SLS_{i,I_n}$  is related to  $I_n$  and  $SLS_{i,E_m}^+$  to  $E_m$ , brings other clear advantage. In fact, the overacceptance or concurrency control of  $R_{i,E_m}^+$  during  $\Delta t_i$  becomes straightforward. Since each egress node  $E_m$  can have a global view of all new flow requests trying to use  $SLS_{i,E_m}^+$ , for all  $I_n$ , it can update the previous  $\tilde{R}_{i,E_m}^+$  estimation in  $\Delta t_i$  accordingly. This means that, when  $E_m$  accepts a new flow it can update  $\tilde{R}_{i,E_m}^+$  to  $\tilde{R}_{i,E_m}^+ - r_j$ , maintaining an updated view of global  $SLS_{i,E_m}^+$  occupancy. In this way, overacceptance as regards  $R_{i,E_m}^+$  cannot occur<sup>6</sup>. An obvious disadvantage is the additional computational burden of identifying and processing flow requests at each  $E_m$ .

Thus, the decision of decoupling AC decisions between  $I_n$  and  $E_m$ , apart from the concurrency debate, should consider both (i) the computational overhead balance between ( $I_n$ ,  $E_m$ ) QoS monitoring and AC tasks and (ii) the required state information at edge nodes.

When egress nodes perform  $SLS_{i,E_m}^+$  AC, the credit strategy may still be useful to control each  $I_n$  rate share, SLS AC, traffic entering  $I_n$  not involving an  $SLS_{i,E_m}^+$  and, indirectly, the QoS levels in the involved paths.

<sup>6</sup>Note that when a flow is rejected at  $E_m$ ,  $\tilde{R}_{i,I_n}$  remains overestimated till  $\Delta t_{i+1}$ , as its rate  $r_j$  is incorrectly accounted for.

### IV. CONCLUSIONS

Considering the need for distributed AC in multiservice networks, in this paper, we have debated the issue of how to handle concurrency in these networks so that service level guarantees are protected from overaccepting flows. We have pointed out simple solutions such as the use of a service-dependent concurrency index, a token-based or a rate-based system to reduce the chance of QoS violations that concurrent AC may raise. For the latter approach, a measurement-based credit management algorithm has also been proposed. Current work is focused on tuning and assessing the performance of these solutions.

### REFERENCES

- [1] R. Atkinson, S. Floyd, and I. A. Board. IAB Concerns and Recommendations Regarding Internet Research and Evolution. RFC 3869 (Informational), Aug. 2004.
- [2] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for Diff-Serv Service Classes. draft-baker-diffserv-basic-classes-04.txt, (work in progress), Oct. 2004.
- [3] A. Bak, W. Burakowski, F. Ricciato, S. Salsano, and H. Tarasiuk. Traffic Handling in AQUILA QoS IP Networks. In M. Smirnov, J. Crowcroft, J. Roberts, and F. Boavida, editors, *QoITS'01*, volume 2156, pages 243–260, Sept. 2001.
- [4] S. Bhatnagar and B. Nath. Distributed Admission Control to Support Guaranteed Services in Core-Stateless Networks. In *IEEE INFOCOM'03*, Mar. 2003.
- [5] C. Cetinkaya, V. Kanodia, and E. Knightly. Scalable Services via Egress Admission Control. *IEEE Transactions on Multimedia*, 3(1):69–81, Mar. 2001.
- [6] G. Huston. Next Steps for the IP QoS Architecture. RFC 2990 (Informational), Nov. 2000.
- [7] F. Kelly, P. Key, and S. Zachary. Distributed Admission Control. *IEEE Journal on Selected Areas in Communications (JSAC)*, 18(12), Dec. 2000.
- [8] S. R. Lima, P. Carvalho, and V. Freitas. Distributed Admission Control for QoS and SLS Management. *Journal of Network and Systems Management - Special Issue on Distributed Management*, 12(3):397–426, Sept. 2004.
- [9] S. R. Lima, P. Carvalho, and V. Freitas. Self-adaptive Distributed Management of QoS and SLSs in Multiservice Networks. In *IEEE/IFIP International Conference on Integrated Management (IM 2005)*, Nice, France, May 2005. IEEE Press.
- [10] P. Morand et al. Mescal D1.2 - Initial Specification of Protocols and Algorithms for Inter-domain SLS Management and Traffic Engineering for QoS-based IP Service Delivery and their Test Requirements. Mescal Project IST-2001-37961, Jan. 2004.