Distributed Admission Control in Multiservice IP Networks: Concurrency Issues

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—In distributed admission control (AC) schemes, handling concurrent AC decisions assumes a relevant role in avoiding over or false acceptance and, consequently, service quality degradation. This problematic is even more intricate in multiservice network environments where distinct service levels need to be fulfilled. This paper debates and points out 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 AC 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.

Index Terms—Concurrent Admission Control; Quality of Service; Differentiated Services; Multiservice Networks

I. INTRODUCTION

Supporting QoS in the Internet launches new demands and challenges on the provision and management of multiservice networks and underlying resources. To guarantee that QoS commitments are honored, traffic control mechanisms such as admission control (AC) are recommended for keeping service classes under controlled load and assuring QoS levels [1]–[3]. Despite this need, 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 context, a lightweight and distributed AC model based on on-line QoS monitoring feedback for managing multiple services quality has been proposed in [4] and formalized in [5]. This model allows to control QoS and the utilization of Service Specification Levels (SLSs) both intra and interdomain.

Distributed AC models, by their nature, are likely to involve multiple and simultaneous AC decisions. Therefore, the need for handling concurrency, which stems

from having multiple decision points, is justified as a way to avoid over/false acceptance of flows entering the network causing, consequently, resources overload and service degradation. This degradation, more likely to occur when the number of ingress nodes accepting incoming calls increases, may be unacceptable for QoS demanding services such as IP telephony.

Although distributed AC has been matter of intense study (see Section II), the problem of concurrent AC in multiservice networks has been rarely tackled. In [6], 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.

This paper extends these concepts and points 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: current proposals focusing on distributed AC are debated in Section II; the characteristics of the multiservice AC model used as case study, the main network domain entities and the AC criteria are summarized

in Section III, the study of concurrent AC and the proposals for tackling concurrency shortcomings are debated in Section IV; finally, the conclusions are drawn in Section V.

II. RELATED WORK

Common AC approaches for class-based IP networks are either centralized (e.g. based on bandwidth brokers [7], [8]) or distributed, parameter or measurement-based differing on the type of services being supported.

The main advantage of centralized AC approaches is that centralizing state information and control tasks allows a global vision of the domain's QoS and operation, relieving the control plane inside the network. This centralization process also facilitates creating and changing service policies and control mechanisms such as AC algorithms. The cost of centralized approaches is however high. Central entities need to store and manage large amounts of information, which in large and highly dynamic networks with many signaling messages and information state updates needing to be processed in real-time is even hard or prohibitive. The congestion and functional dependence on a single entity is another well-known problem of centralization.

Therefore, to improve reliability and scalability in large network domains, several approaches consider distributed AC with variable control complexity depending on the QoS guarantees and predictability required. To provide guaranteed services (e.g., for hard real-time traffic), AC proposals tend to require significant network state information and, in many cases, changes in all network nodes [9], [10]. To provide predictive services (e.g., for soft real-time traffic) measurement-based AC (MBAC) [11], [12] and end-to-end MBAC (EMBAC) solutions [13]–[15] have deserved special attention. These solutions leads to reduced control information and overhead, but eventually to QoS degradation. To control elastic traffic, for more efficient network utilization, implicit AC strategies, i.e., without requiring explicit signaling between the application and the network, have also been defined [16], [17].

As far as distributed AC is concerned, this paper extends the former studies by focusing on debating the problem of concurrent AC and putting forward solutions to mitigate the negative impact concurrency might have on the provided service level guarantees. To sustain this debate a distributed AC proposal is presented and summarized next.

III. AC MODEL SPECIFICATION

A. Model Overview

This section provides a brief overview of the multi-service AC model proposed in [4], [5] 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 customers' 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.

As illustrated in Fig. 1, 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 and packet drop will not occur due to a simple and indiscriminate traffic conditioning process. The obtained measures are sent to the corresponding ingress routers to update an Ingress-Egress service matrix used for distributed AC and active service management. This notification is carried out periodically, when a metric value or its variation exceeds a limit or the SLS utilization exceeds a safety threshold.
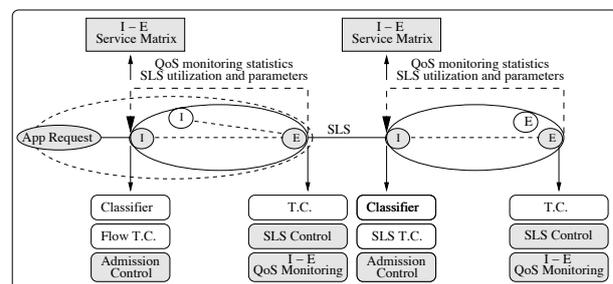
Fig. 1. Location of tasks in a multiclass domain

The *end-to-end case*, detailed in [5], is viewed as a repetitive and cumulative process of AC and available service computation only performed at ingress nodes.

As the proposed model is multiservice, explicit and implicit AC can be in place depending on the application or service characteristics. Explicit flow AC is oriented to applications able to signal the network with their traffic profile and QoS objectives. In this case, the AC decision requires two initial verifications regarding SLS utilization control and QoS control following the rules defined in III-C.

Implicit flow AC, oriented to applications which do not use signaling and in particular to elastic applications, use implicit detection/rejection of flows [16]. In this type of AC, neither specific flow's information nor measures from upstream domains are considered.

B. Multiservice Domain Specification

Considering the overview of the AC model operation described above, we focus now the main components of a generic network domain comprising multiple ingress and egress routers, as regards the provision of multi-services to customers (individuals or other domains). Following [5], 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. When possible, the entities under specification use indexes based on the corresponding service class and involved ingress and egress nodes. As the AC model is class-based and operates edge-to-edge, this approach enriches semantically the notation, while keeping it intuitive.

Service Classes

Considering a multiclass domain D_x comprising N ingress nodes and M egress nodes, we define $I^{D_x} = \{I_1, I_2, ..., I_N\}$ and $E^{D_x} = \{E_1, E_2, ..., E_M\}$ as the set of ingress and egress nodes, respectively[1]. For this domain, we represent the set of supported service classes as $SC^{D_x} = \{SC_1, SC_2, ..., SC_Y\}$. For each service class $SC_i \in SC^{D_x}$, the set of QoS parameters under control is defined as $P_{SC_i} = \{(P_{i,1}, \beta_{i,1}), ..., (P_{i,P}, \beta_{i,P})\}$ 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. 2).

The case of flows entering the domain D_x without pre-negotiated SLSs (usually dial-up users) is also covered, and the notation $\notin SLS$ is introduced for this purpose. The global rate share of these users is controlled by $R_{i,I_n}^{\notin SLS}$. Therefore, $R_{i,I_n}^{\notin SLS}$ is a rate-based parameter defined to limit traffic not sustained by a specific SLS. This allows a better control of the rate share in D_x and of SLS_{i,E_m}^+ utilization, while avoiding possible denial-of-service to flows belonging to SLS_{i,I_n}.

---

[1]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.
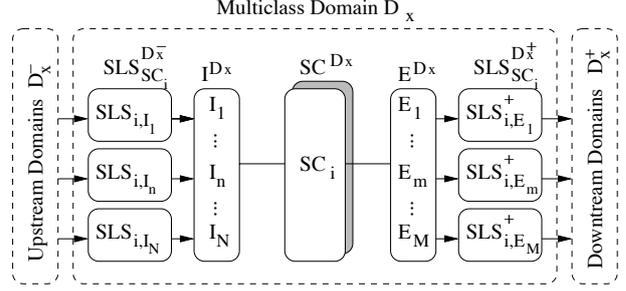
Fig. 2.   Domain Elements and Notation

Upstream SLSs

The definition of SLSs [18], 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 those included in Table I.

TABLE I
COMMON SLS_{i,I_n} ELEMENTS

Item	Notation	Example
1) Scope	$SLS_{i,I_n} \rightarrow Scope$	(I_n, E')
2) Service ID	$SLS_{i,I_n} \rightarrow SC_{id}$	$DSCP$
3) Traffic Profile	$SLS_{i,I_n} \rightarrow TProf$	$TB(R_{i,I_n}, b_{i,I_n})$
4) Expected QoS	$SLS_{i,I_n} \rightarrow EQoS$	$IPTD_{i,I_n}$
5) Validity	$SLS_{i,I_n} \rightarrow Sched$	$[t_{i,I_n,0}, t_{i,I_n,f}]$

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. At this point, the scope of SLS_{i,I_n} is limited to a single domain D_x, which is responsible for identifying E' according to the destination domains D_x^+ defined in SLS_{i,I_n}.

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}. The DS Code Point is a possible SC_{id} in Diffserv domains.

3) $SLS_{i,I_n} \rightarrow TProf$ specifies the qualitative and/or quantitative 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) $SLS_{i,I_n} \rightarrow EQoS$ specifies the expected QoS parameters for SLS_{i,I_n}, i.e., $P_{SLS_{i,I_n}} = \{P_{i,I_n,1}, ... , P_{i,I_n,P'}\}$. 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) $SLS_{i,I_n} \rightarrow Sched$ determines the time interval $[t_{i,I_n,0}, t_{i,I_n,f}]$ in which the service is due to be scheduled, where $t_{i,I_n,0}$ expresses the SLS starting time and $t_{i,I_n,f}$ the SLS expiring time.

Downstream SLSs

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 +.

Traffic Flows

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 traffic profile $TProf$, 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. III-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.,

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

In (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.,

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

In (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.,

$$\tilde{R}_{i,(*,Em)}^+ = \sum_{k=1}^{N} \tilde{r}_{i,(I_k,E_m)}; \tag{3}$$

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 β_{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 (2) using the measured rate $R_{i,I_n}^{\notin SLS}$, i.e.,

$$\tilde{R}_{i,(I_n,*)}^{\notin SLS} + r_j \leq \beta_{i,I_n}^{\notin} R_{i,I_n}^{\notin SLS}. \tag{4}$$

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 (see (5)), 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 Δt_i.

QoS Parameters Control Rules - When controlling the QoS levels in a domain, the QoS parameters and corresponding thresholds may vary depending on each service class SC_i commitments, the statistical properties of the traffic and degree of overprovisioning. 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 Δt_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.,

$$\forall (P_{i,p}, \beta_{i,p}) \in P_{SC_i} : \tilde{P}_{i,p} \leq T_{i,p} \tag{5}$$

where $\tilde{P}_{i,p}$ is the measured value of $P_{i,p}$ for Δt_i, and $T_{i,p}$ is the parameter's threshold, as explained in Sec. III-B. Equation (5) is not flow dependent, i.e., it is checked once during Δt_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 Δt_{i+1}.

IV. HANDLING CONCURRENCY

A distributed AC model may involve multiple nodes making concurrent AC decisions. Therefore, dealing with concurrency is a key aspect to avoid over or false acceptance. This problematic is present whenever distributed and simultaneous AC decisions are taken, being more notorious on measurement-based approaches.

Focusing on the distributed AC model presented above, within a measurement time interval Δt_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 Δt_{i+1}, i.e., when the next measuring update takes place[2].

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. Regarding $\tilde{R}^+_{i,E_m}$, when a downstream SLS^+_{i,E_m} is shared among all possible $(*, E_m)$ pairs (see (2)), the available rate of SLS^+_{i,E_m} in Δt_i viewed by each I_n, i.e., $\beta^+_{i,E_m} R^+_{i,E_m} - \tilde{R}^+_{i,(*,E_m)}$, will change upon each ingress I_n acceptance decision. Even if each I_n, individually, would update the measured rate $\tilde{R}^+_{i,(*,E_m)}$ according to new flow's accepted rate (r_j), the side effect of other ingress admissions would be unknown until Δt_{i+1}. So, false or over acceptance could still occur during Δt_i.

The problem of mis-acceptance within each service class can be reduced resorting to larger safety margins (β^+_{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 Δt_i[3]. 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 $\ddot{I}$ the set of concurrent ingress nodes sharing a common SLS^+_{i,E_m}, the estimation of SLS^+_{i,E_m} available rate for Δt_i can be protected by a concurrency index χ_{i,E_m}, which depends on the cardinality of $\ddot{I}$. In this way, explicit flow AC is ruled by

$$r_j \leq \frac{\beta^+_{i,E_m} R^+_{i,E_m} - \tilde{R}^+_{i,(*,E_m)}}{\chi_{i,E_m}}, \tag{6}$$

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 χ_{i,E_m} can be applied.

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 Δt_i. In the limit, when a single token is available in the system, no concurrency is allowed. Nevertheless, if during Δt_i the tokens pass through several ingress nodes, the SLS^+_{i,E_m} utilization can change without common knowledge of all concurrent nodes, $\ddot{I}$. 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

---

[2]In order to maintain simplicity, reduce overhead and latency associated with the exchange of control information, during Δt_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.

[3]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 Δt_i explores this compensation effect but may increase over acceptance.
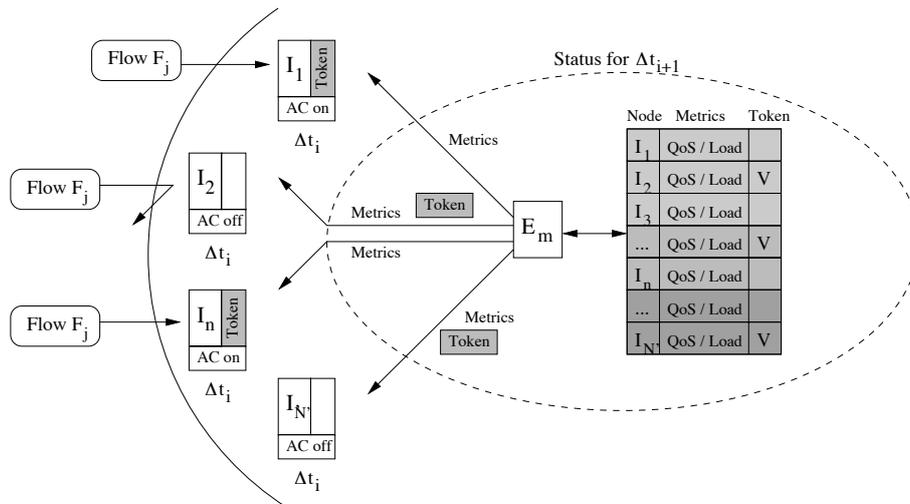
Fig. 3. Token-based system

unchanged during Δt_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 $|\ddot{I}|$, may be prohibitive. To reduce this overhead, the number of available tokens and the choice of ingress nodes which might get one are controlled by the egress nodes based on their global view of each ingress rate usage. Topology information may also be used to group ingress nodes into topological independent areas according to the bottleneck link sharing [6]. Each defined group would share a single token or more tokens according to the degree of concurrency allowed. However, it should be noticed that a token based strategy may impair the admission of flows $F_j \in SLS_{i,I_n}$, i.e., a flow request not exceeding the available SLS_{i,I_n} rate may be rejected or delayed by an I_n not having a token. In this way, this strategy is more suitable to control the admission of concurrent $F_j \notin SLS_{i,I_n}$.

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. This system should respect the I_n commitments previously assumed as regards the accepted SLS_{i,I_n}. Therefore,

above all, this credit system should drive SLS AC and could also be explored to guide dynamic SLS AC decisions.

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:

- the network topology, the underlying bottleneck capacity[4] and network core multiplexing effects;
- the bandwidth sharing policies among classes [6];
- the already accepted SLS_{i,I_n} and the corresponding expected traffic matrix;
- the SLS_{i,E_m}^{+} negotiated rate or the capacity allocated at E_m for SC_i;
- internal policies regarding $R^{\in SLS}$ and $R^{\notin SLS}$ shares;
- a safety margin of unused credits at each I_n to assure that I_n has a controlled autonomy to make acceptance decisions during Δt_i.

At each Δt_i, ingress nodes may receive new credits

---

[4]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).
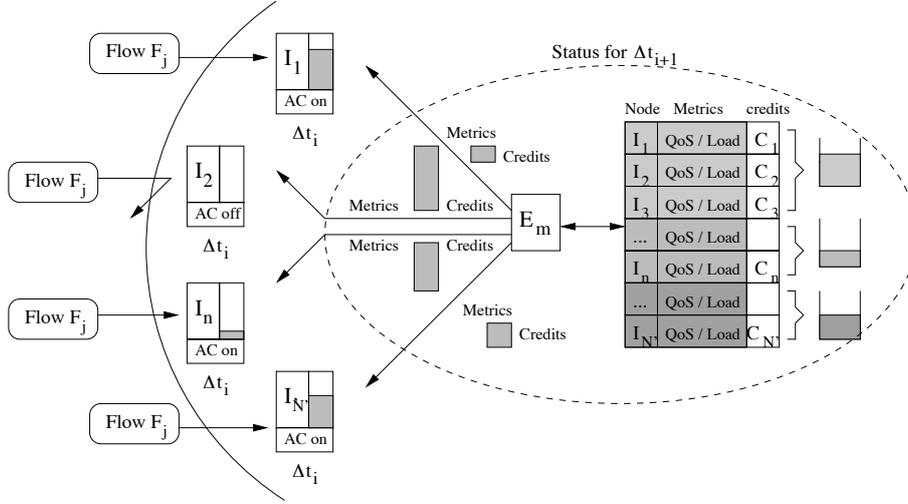
Fig. 4. Rate-based credit system

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. 4). This strategy avoids several drawbacks of the solution proposed in [6], such as horizontal sharing of credits, use of specific signaling between ingress nodes and holding to many unused resources at each I_n.

The pool of unused credits at E_m can either have one type of credits shared between $R^{\in SLS}$ and $R^{\notin SLS}$ or differentiated credits and bounds for the two cases. While both rates use (I_n, E_m) path resources and indirectly interfere with QoS metrics, they can have a different weight in the global rate/credits usage. Distinguishing two types of credits assumes that E_m can distinguish I_n traffic as belonging or not to an SLS_{i,I_n} when measuring $\tilde{r}_{i,(I_n,E_m)}$. To simplify the proposed credit system we can consider just one type of credits and let I_n control the rate of each type of traffic applying pre-defined shares to the globally assigned credits. This would reduce the state information and avoid extra filters in $\tilde{r}_{i,(I_n,E_m)}$ estimation.

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:

- 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};
- 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[5].

Decoupling AC Decisions

Controlling the utilization of R^+_{i,E_m} during Δt_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}, $R^{\notin SLS}_{i,I_n}$ 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 (5) are satisfied. When arriving at egress E_m, F_j is accepted and may be

---

[5]Specific requests of credits from I_n to E_m during Δt_i can also be considered, however, it changes the initial concept and assumption of viewing Δt_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 Δt_i */

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

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

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

Notation:

$RC_{i,(I_n,E_m)}$: rate credits available at I_n for SC_i; E_m maintains this information to determine $RC^{new}_{i,(I_n,E_m)}$

$RC^{avail}_{i,(I_n,E_m)}$: remaining credits at I_n according to the estimated rate usage $\tilde{r}_{i,(I_n,E_m)}$

$RC^{new}_{i,(I_n,E_m)}$: credits update for Δt_{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 $\breve{I} \subseteq I^{D_x}$ concurrent nodes

---

forward to the next domain if (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 Δt_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 Δt_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[6]. 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.

V. CONCLUSIONS

In this paper, we have addressed the issue of controlling concurrent AC in multiservice IP networks so that service level guarantees are protected from flows' over-acceptance. This overacceptance results essentially from having multiple ingress nodes handling simultaneously multiple flow admission requests. 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 considering a multiservice domain.

REFERENCES

[1] G. Huston. Next Steps for the IP QoS Architecture. RFC 2990 (Informational), November 2000.

[2] R. Atkinson, S. Floyd, and Internet Architecture Board. IAB Concerns and Recommendations Regarding Internet Research and Evolution. RFC 3869 (Informational), August 2004.

[3] J. Babiarz, K. Chan, and F. Baker. Configuration Guidelines for DiffServ Service Classes. draft-baker-diffserv-basic-classes-04.txt, (work in progress), October 2004.

[4] Solange Rito Lima, Paulo Carvalho, and Vasco Freitas. Distributed Admission Control for QoS and SLS Management. *Journal of Network and Systems Management - Special Issue on Distributed Management*, 12(3):397–426, September 2004.

[5] Solange Rito Lima, Paulo Carvalho, and Vasco 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.

[6] 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, *QofIS'01*, volume 2156, pages 243–260, September 2001.

[7] Z. Duan, Z. Zhang, Y. Hou, and L. Gao. A Core Stateless Bandwidth Broker Architecture for Scalable Support of Guaranteed Services. *IEEE Transactions on Parallel and Distributed Systems*, 15(2):167–182, February 2004.

---

[6]Note that when a flow is rejected at E_m, $\tilde{R}_{i,I_n}$ remains overestimated till Δt_{i+1}, as its rate r_j is incorrectly accounted for.

[8] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson V. Narayan, and F. Reichmeyer. Internet2 QBone: building a testbed for differentiated services. *IEEE Network*, 13(5):8–16, "Sep./Oct." 1999.

[9] I. Stoica and Hui Zhang. Providing Guaranteed Services Without Per Flow Management. In *ACM SIGCOMM'99*, October 1999.

[10] L. Westberg. Resource Management in Diffserv (RMD) Framework. IETF draft: draft-westberg-rmd-framework-04.txt (work in progress), September 2003.

[11] S. Jamin, P. Danzig, S. Shenker, and L. Zhang. A Measurement-Based Call Admission Control Algorithm for Integrated Services Packet Networks (Extended Version). *IEEE/ACM Transactions on Networking*, pages 56–70, February 1997.

[12] L. Breslau and S. Jamin. Comments on the Performance of Measurement-Based Admission Control Algorithms. In *IEEE INFOCOM'00*, March 2000.

[13] C. Cetinkaya, V. Kanodia, and E. Knightly. Scalable Services via Egress Admission Control. *IEEE Transactions on Multimedia*, 3(1):69–81, March 2001.

[14] V. Elek, G. Karlsson, and R. Rnngren. Admission Control Based on End-to-End Measurements. In *IEEE INFOCOM'00*, 2000.

[15] L. Breslau, E. Knightly, S. Shenker, I. Stoica, and H. Zhang. Endpoint Admission Control: Architectural Issues and Performance. In *ACM SIGCOMM'00*, 2000.

[16] R. Mortier, I. Pratt, C. Clark, and S. Crosby. Implicit Admission Control. *IEEE Journal on Selected Areas in Communication*, 18(12):2629–2639, December 2000.

[17] S. Fredj, S. Oueslati-Boulahia, and J. Roberts. Measurement-based Admission Control for Elastic Traffic. In *17th International Teletraffic Congress*, June 2001.

[18] P. Morand, M. Boucadair, P. Levis, R. Egan, H. Asgari, D. Griffin, J. Griem, J. Spencer, P. Trimintzios, M. Howarth, N. Wang, P. Flegkas, K. Ho, S. Georgoulas, G. Pavlou, P. Georgatsos, and T. Damilatis. 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, January 2004.

Solange Rito Lima graduated in 1991, received her MSc and PhD degrees in Computer Communications from the University of Minho, Braga, Portugal, in 1997 and 2006, respectively. She is currently an Assistant Professor in Computer Communications at the same University. Her research interests include multiservice IP networks and protocols, QoS, traffic control and monitoring issues.

Paulo Carvalho graduated in 1991 and received his PhD degree in Computer Science from the University of Kent at Canterbury, United Kingdom, in 1997. He is currently Assistant Professor of Computer Communications, Department of Informatics, at the University of Minho, Portugal. His main research interests include broadband technologies, multiservice networks and protocols, and mobile networks.

Vasco Freitas graduated in 1972 and obtained his MSc and PhD degrees in Control and Computer Communications at the University of Manchester, UK, in 1977 and 1980. From 1989 until 1994 he was Director of Networking at the Portuguese Foundation for Scientific Computing to establish the National University Data Network. He is currently Professor of Computer Communications at the University of Minho, Portugal.