Endpoint Admission Control over Assured Forwarding PHBs and its performance over RED implementations

Giuseppe Bianchi[1], Nicola Blefari-Melazzi[2] ,Vincenzo Mancuso[1]

1 University of Palermo, Italy, bianchi@elet.polimi.it
2 University of Perugia, Italy, blefari@diei.unipg.it

ABSTRACT. *The Assured Forwarding Per Hop Behavior (AF PHB) has been devised by the IETF Differentiated Services (DiffServ) working group to provide drop level differentiation. The intent of AF is to support services with different loss requirements, but with no strict delay and jitter guarantees. Another suggested use of AF is to provide differentiated support for traffic conforming to an edge conditioning/policing scheme with respect to non-conforming traffic.*

Scope of this paper is twofold. First, we show that, quite surprisingly, a standard AF PHB class is semantically capable of supporting per flow admission control. This is obtained by adopting the AF PHB as core routers forwarding mechanism in conjunction with an End Point Admission Control mechanism running at the network edge nodes. The performance achieved by our proposed approach depend on the specific AF PHB implementation running in the core routers.

In the second part of the paper, we prove that changes in the customary AF PHB implementations are indeed required to achieve strict QoS performance. To prove this point, we have evaluated the performance of our admission control scheme over a simple AF implementation based on RED queues. Our results show that, regardless of the selected RED thresholds configuration, such an implementation is never capable of guaranteeing tight QoS support, but is limited to provide better than best effort performance.

1 INTRODUCTION

Two QoS architectures are being discussed in the Internet arena: Integrated Services and Differentiated Services. Nevertheless, quoting the recent RFC [R2990], *"both the Integrated Services architecture and the Differentiated Services architecture have some critical elements in terms of their current definition, which appear to be acting as deterrents to widespread deployment... There appears to be no single comprehensive service environment that possesses both service accuracy and scaling properties*". In fact:

1. the IntServ/RSVP paradigm [R2205, R2210] is devised to establish reservations at each router along a new connection path, and provide "hard" QoS guarantees. In this sense, it is far to be a novel reservation paradigm, as it inherits its basic ideas from ATM and the complexity of the traffic control scheme is comparable. In the heart of large-scale networks, the cost of RSVP soft state maintenance and of processing and signaling overhead in the routers is significant and thus there

are scalability problems. In addition to complexity, we feel that the lack of a total and ultimate appreciation in the Internet market of the IntServ approach is also related to the fact that RSVP needs to be deployed in all the involved routers, to provide end-to-end QoS guarantees; hence this approach is not easily and smoothly compatible with existing infrastructures. What we are trying to say is that complexity and scalability are really important issues, but that backward compatibility and smooth Internet upgrade in a multi-vendor Internet market scenario is probably even more important.

2. Following this line of reasoning, we argue that the success of the DiffServ framework [R2474, R2475] does not uniquely stays in the fact that it is an approach devised to overcome the scalability limits of IntServ. As in the legacy Internet, the DiffServ network is oblivious of individual flows. Each router merely implements a suite of scheduling and buffering mechanisms, to provide different aggregate service assurances to different traffic classes whose packets are accordingly marked with a different value of the Differentiated Services Code Point (DSCP) field in the IP packet header. By leaving untouched the basic Internet principles, DiffServ provides supplementary tools to further move the problem of Internet traffic control up to the definition of suitable pricing/service level agreements (SLAs) between peers. However, DiffServ lacks a standardized admission control scheme, and does not intrinsically solve the problem of controlling congestion in the Internet. Upon overload in a given service class, all flows in that class suffer a potentially harsh degradation of service. RFC [R2998] recognizes this problem and points out that *"further refinement of the QoS architecture is required to integrate DiffServ network services into an end-to-end service delivery model with the associated task of resource reservatio*n". It is thus suggested [R2990] to define an *"admission control function which can determine whether to admit a service differentiated flow along the nominated network pat*h".

Scope of this paper is to show that such an admission control function can be defined on top of a standard DiffServ framework, by simply making smart usage of the semantic at the basis of the Assured Forwarding Per Hop Behavior (AF PHB [R2597]). It is obvious that this function must not imply a management of per flow states, which are alien to DiffServ and which would re-introduce scalability problems. The scope of our admission control function is Internet-wise (i.e., not limited to a single domain). It is deployed by pure endpoint operation: edge nodes involved in a communication are in charge of taking an explicit decision whether to admit a new flow or reject it. These edge nodes rely upon the successful delivery of probe packets, i.e., packets tagged with a suitable DSCP label, independently generated by the end-points at flow setup. The internal differentiated management of probes and packets originated by already admitted flows is performed in conformance with the AF PHB definition. Also, following the spirit of DiffServ, the degree of QoS provided is delegated to each individual DiffServ domain, and depends on the AF PHB specific implementation and tuning done at each core router of the domain.

For convenience, we will use the term GRIP (Gauge&Gate Reservation with Independent Probing) to name the overall described operation. GRIP was originally

proposed in [BB01a], although its mapping over a standard DiffServ framework was not recognized until [BB01b]. GRIP combines the packet differentiation capabilities of the AF PHB with both the distributed and scalable logic of Endpoint Admission Control [BRE00], and the performance advantages of measurement based admission control schemes [BJS00]. However, we remark that GRIP is not a new reservation protocol for the Internet (in this, differing from the SRP protocol [ALM98], from which GRIP inherits some strategic ideas). Instead, GRIP is a novel reservation paradigm that allows independent end point software developers and core router producers to inter-operate within the DiffServ framework, without explicit protocol agreements.

The organization of this paper is the following. Section 2 provides an understanding of rationales and limits of Endpoint Admission Control. Section 3 describes the GRIP operation and its support over AF PHB classes. Section 4 first qualitatively discusses the issue of performance achievable by specifically designed AF implementations; then presents numerical results that prove that GRIP's performance over AF PHB routers, implemented with RED queues ([FVJ93]), are just limited to better than best effort support. Finally, conclusions are drawn in Section 5.

2 UNFOLDING ENDPOINT ADMISSION CONTROL

Endpoint Admission Control (EAC) is a recent research trend in QoS provisioning over IP [BRE00]. EAC builds upon the idea that admission control can be managed by pure end-to-end operation, involving only the source and destination host. At connection set-up, each sender-receiver pair starts a Probing phase whose goal is to determine whether the considered connection can be admitted to the network. In some EAC proposals [BOR99, ELE00, BRE00], during the Probing phase, the source node sends packets that reproduce the characteristics (or a subset of them) of the traffic that the source wants to emit through the network. Upon reception of the first probing packet, the destination host starts monitoring probing packets statistics (e.g., loss ratio, probes interarrival times) for a given period of time. At the end of the measurement period and on the basis of suitable criteria, the receiver takes the decision whether to admit or reject the connection and notifies back this decision to the source node.

Although the described scheme looks elegant and promising (it is scalable, it does not involve inner routers), a number of subtle issues come out when we look for QoS performance. A scheme purely based on endpoint measurements suffers of performance drawbacks mostly related to the necessarily limited (few hundreds of ms, for reasonably bounded call setup times) measurement time spent at the destination. Measurements taken over such a short time cannot capture stationary network states, and thus the decision whether to admit or reject a call is taken over a snapshot of the network status, which can be quite an unrealistic picture of the network congestion level.

The simplest solution to the above issue (other solutions are being explored, but their complete discussion and understanding is way out of the aims of the present paper) is to attempt to convey more reliable network state information to the edge of the network. Several solutions have been proposed in the literature. [CKN00]

proposes to drive EAC decisions from measurements performed on a longer time scale among each ingress/egress pair of nodes within a domain. [GKE99, SZH99, KEL00] use packet marking to convey explicit congestion information to the relevant network nodes in charge of taking admission control decisions. [MOR00] performs admission control at layers above IP (i.e., TCP), by imposing each core router to parse and capture TCP SYN and SYN/ACK segments, and forward such packets only if local congestion conditions allow admission of a new TCP flow.

To summarize the above discussion, and to proceed further, we can state that an EAC is, ultimately, the combination of three logically distinct components (although, in some specific solutions – e.g. [BOR99, ELE00] – the following issues are not clearly distinct, this does not mean at all that these three specific issues are not simultaneously present):

1. edge nodes in charge of taking explicit per flow accept/reject decisions;

2. physical principles and measures on which decisions are based (e.g., congestion status of an internal link or an ingress/egress path, and particular measurement technique - if any - adopted to detect such status);

3. the specific mechanisms adopted to convey internal network information to edge nodes (e.g., received probing bandwidth measurement, IP packet marking, exploitation of layers above IP with a well-defined notion of connection or even explicit signaling).

In such a view, EAC can be re-interpreted as a Measurement Based Admission Control (MBAC) that runs internally to the network (i.e., in a whole domain or, much simpler, in each internal router). This MBAC scheme locally determines, according to some specific criteria (which can be as simple as non performing any measure at all, and taking a snapshot of the link state, or as complex as some of the techniques proposed in [BJS00, GRO99]), whether a new call can be *locally* admitted (i.e. as far as the local router is concerned). This set of information (one per each distinct network router) is implicitly (or explicitly) collected and aggregated at the edge nodes of the network; these nodes are ultimately in charge of performing the Y/N decision.

Put in these terms, EAC was sketched as early as in the SRP protocol specification [ALM98]. Unfortunately (see e.g., what stated in [BRE00]), SRP appeared much more like a lightweight signaling protocol, with explicit reservation messages, rather than an EAC technique with increased intelligence within the core routers. Moreover, SRP requires network routers to actively manage packets (via remarking of signaling packets when congestion occurs), and thus it does not fit within a DiffServ framework, where the core routers duty is strictly limited to forwarding packets at the greatest possible speed.

Of the three components outlined above, we argue that, in a DiffServ framework, the least critical issue is how to estimate the congestion status of a router without resorting to per flow operation. In fact, recent literature [GRO99,BJS00] has shown that *aggregate* load measurements are extremely robust and efficient. These schemes do not exploit per-flow state information and related traffic specifications. Instead, they operate on the basis of per-node aggregate traffic measurements carried out at the

packet level. The robustness of these schemes stays in the fact that, in suitable conditions (e.g. flow peak rates small with respect to link capacities), they are barely sensitive to uncertainties on traffic profile parameters. As a consequence, it seems that scalable estimations can be independently carried out by the routers.

The real admission control problem is how to convey the status of core routers (evaluated by means of aggregate measurements) to the end points so that the latter devices can take *learned* admission control decisions, without *violating the DiffServ paradigm.* For obvious reasons, we cannot use explicit per flow signaling. Similarly, we do not want to modify the basic router operation, by introducing packet marking schemes or forcing routers to parse and interpret higher layer information. What we want to do is to *implicitly* convey the status of core routers to the end points, by means of scalable, DiffServ compliant procedures.

3 GRIP: ENDPOINT ADMISSION CONTROL OVER AF PER HOP BEHAVIOR

We name GRIP (Gauge&Gate Reservation with Independent Probing) a reservation framework where Endpoint Admission Control decisions are driven by probing packet losses occurring in the internal network routers. The Gauge&Gate acronym stems from the assumption that network routers are able to drive probing packet discarding (Gate) on the basis of accepted traffic measurements (Gauge), and thus implicitly convey congestion status information to the edge of the network by means of reception/lack of reception of probes independently generated by edge nodes. GRIP concepts were originally introduced in [BB01a], but its mapping over DiffServ was not fully recognized until [BB01b] (as a matter of fact, in [BB01a] we erroneously required the introduction of a new PHB to implement our scheme). The present paper moves further, and shows that the generic GRIP router operation is indeed intrinsically accounted in the Assured Forwarding (AF) PHB. In other words, the AF PHB class, with no further modifications to the specifications described in [R2597], is semantically capable of seamlessly supporting EAC.

AF PHBs have been devised to provide different levels of forwarding assurances within the Differentiated Services Framework [R2474, R2475]. Four AF PHB classes have been standardized, each composed of three drop levels. In what follows, we will use the notation A*Fxj* to indicate packet marks belonging to the AF class x, with drop level j. Conforming to [R2597], within a class x, if i<j, the dropping probability of packets labeled A*Fxi* is lower than that of packets labeled A*Fx*j.

The example services presented in the appendix of [R2597] show that the primary intent of AF is to promote performance differentiation (in terms of packet drop), either among different traffic classes, e.g., marked with different drop levels, as well as within the same traffic class, e.g., marking traffic conforming to a policy specification with a lower drop level than non conforming traffic. However, low loss and low latency traffic support appears to be out of the targets of the AF model. To a larger extent, as discussed in the introduction, QoS guarantees appear not only unfeasible over AF, but also out of reach of the basic DiffServ architectural model, due to the lack of an explicit resource reservation mechanism.

Based on the discussion carried out in Section 2, it is now possible to argument that the AF PHB definition contains all the necessary semantic to support per flow admission control. Quoting [R2597], "*an AF implementation MUST detect and respond to long-term congestion within each class by dropping packets, while handling short term congestion (packet bursts) by queueing packets. This implies the presence of a smoothing or filtering function that monitors the instantaneous congestion level and computes a smoothed congestion level. The dropping algorithm uses this smoothed congestion level to determine when packets should be discarde*d".

This sentence explicitly states that an AF router is capable of supporting an eventually sophisticated measurement criterion that can drive packet discarding. To run EAC over an AF PHB class it is simply necessary to clarify issue (3) presented in Section 2 (i.e., the specific mechanism adopted to convey internal network information to edge nodes). This is done by assigning to a specific AF dropping level the task of notifying internal network congestion to the end nodes *by means of packet dropping* (which, for an AF-compliant router, is the only capability we can rely on).

3.1 AF Router Operation

A particular implementation of the DiffServ router output port operation supporting the AF PHB is depicted in Fig. 1. Packets routed to the relevant output are classified on the basis of their DSCP tag and dispatched to the relevant PHB handler.

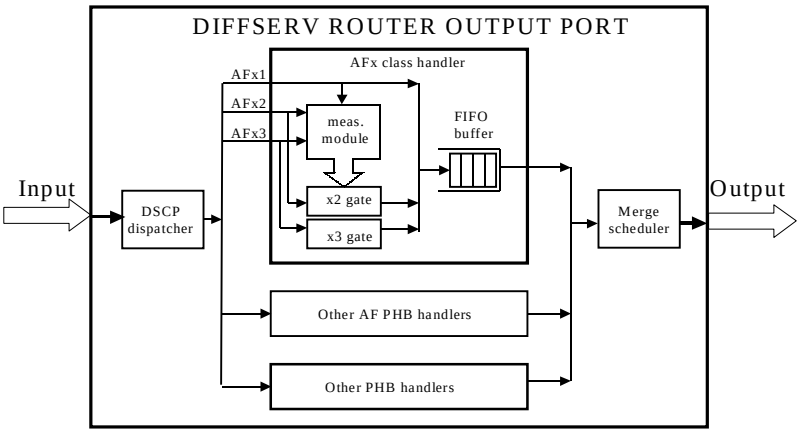

Fig. 1 - Router output port operation

Let us now focus our attention to a specific module in charge of handling AF traffic belonging *to a given class x.*

A measurement module is devised to run-time measure the *aggregate* AF class x traffic (or AFx traffic). The measurement module depicted in the figure does not interact with the AFx1 packets forwarding, i.e., these packets are forwarded to the FIFO buffer placed at the output regardless of the measurements taken. On the basis of such measurements, this module triggers a suitable dropping algorithm on the

AFx2 traffic. With respect to the general AF PHB operation, our AFx2 dropping algorithm depends on AFx traffic measurements. Note also that for simplicity of presentation, the drop level AFx3 is neglected until Section 3.3.

The simplest dropping algorithm is represented by a "gate" (smoother dropping algorithms for AFx2 packets - e.g. RED-like algorithms - may be considered to improve stability). When the measurement module does not detect congestion on the AFx traffic, being the notion of congestion implementation-dependent, it keeps the gate opened (we call this "ACCEPT" state). When the gate is open no AFx2 packet is dropped. Conversely, the measurement module keeps the gate closed ("REJECT" state) when congestion is detected, i.e., it enforces a 100% drop probability over AFx2 packets. Note that this operation does not violate the AF drop level relationship, as AFx1 dropping probability is lower than the AFx2 one.

While the above description is simply a particular implementation of an AF class, we now show its interpretation in terms of implicit signaling, which has important consequences for the definition of our overlay admission control function. In fact, let us assume that: i) the considered AF class x, is devoted to the support of QoS aware flows, requiring an admission control procedure; ii) traffic labeled AFx1 is generated by flows which have already passed an admission control test, iii) AFx2 packets are "signaling" packets injected in the network by flows during the setup phase (in principle, one AFx2 packet per flow).

According to the described operation, an AFx2 packet is delivered to its destination ONLY IF it encounters all the routers along the path in the ACCEPT state. This operation provides an implicit binary signaling pipe, semantically equivalent to a one-bit explicit congestion notification scheme, without requiring explicit packet marking, or, worse, explicit signaling messages, contrary to the DiffServ spirit.

The described router output port operation, combined with an endpoint admission control logic, allows overlaying an implicit signaling pipe over a signaling-unaware DiffServ framework. In fact, when an AFx2 packet reaches the destination, it implicitly conveys the information that all routers encountered across the path have been locally declared themselves in the ACCEPT state, i.e., capable of admitting new connections (see next Section).

Finally, with reference to Fig. 1, the AFx PHB class handler stores packets in a FIFO buffer, to ensure that packets are forwarded in the order of their receipt, as required by the AF PHB specification [R2597]. Packets transmission over the output link is finally managed by a scheduler, which has the task of merging the traffic coming from the different PHB handlers implemented within the router output port.

3.2 End Point Operation

For clarity of presentation, in what follows, we identify the source and destination user terminals as the network end nodes[1] . Consider a scenario where an application

[1] Although, logically, user terminals are the natural nodes where the endpoint admission control should operate, this is clearly not realistic, for the obvious reason that the user may bypass the admission control test and directly send AFx1 packets. Identity authentication and integrity protection are therefore needed in order to mitigate this potential for theft of resources [R2990]. Administrators are then expected to protect network resources by configuring secure policers at interfaces (e.g. access routers) with untrusted customers.

running on a source node within a DS domain wants to setup a one way (e.g., UDP) flow with a destination node, generally in a different DS domain. As shown in Fig. 2, the source node, triggered by the application via proprietary signaling, starts a connection setup attempt by sending in principle *just one single packet* (more discussion about this at the end of this Section), labeled AFx2 through the network. In the same time, a probing phase timeout is started.

The role of the Destination Node simply consists in monitoring the incoming IP packets and detecting those labeled AFx2. Upon reception of an AFx2 packet, the destination node performs a receiver capability negotiation function, eventually based on proprietary signaling, and aimed at verifying whether the destination application is able and willing to accept the incoming flow. We stress that such receiver capability negotiation is recognized as an important functionality for QoS enabled applications [R2990], and it is in an important by-product of our solution. If the destination node is willing to accept the call request, it simply relays, for each incoming probe packet, with the transmission of a feedback packet. For highest probability of delivery, the feedback packet is marked AFx1 (e.g., as an information packet).

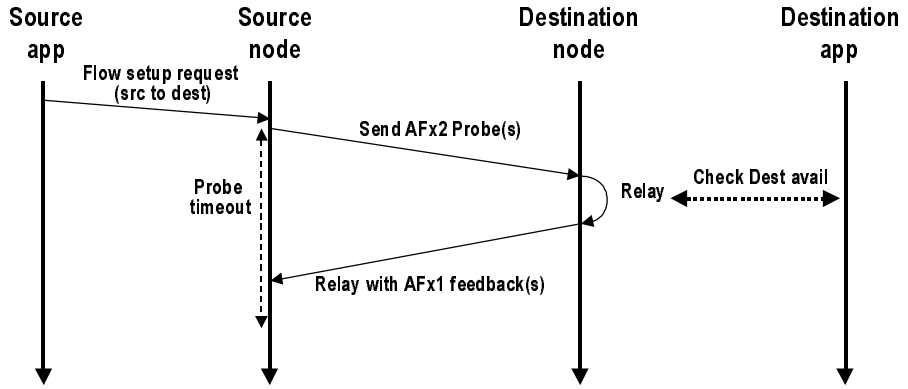

Fig. 2 - End point GRIP operation

The decision whether to admit or reject the call request is driven by the eventual reception of the feedback by the source. When a feedback packet is received in response, the setting up flow is elected at the state of "accepted", and the source node can starts transmitting information packets, labeled as AFx1. Conversely, by not receiving a feedback packet within the probing phase timeout, the source node is made able to implicitly determine that at least one router along the path has declared itself not capable of accommodating additional flows, and thus the source node can abort the flow setup attempt (or reiterate the setup attempt according to some suitable backoff mechanism).

In EAC terms, packets labeled AFx2 have the meaning of probes, while AFx1 packets are meant to support already accepted traffic. The role of the drop level AFx3 is addressed in Section 3.3. We have adopted AFx1 as the label assigned to the feedback packet since the goal of the feedback is to report back to the source the

information that the probe has been correctly received. In the case bidirectional flow setup is aimed at, the feedback packet has the additional task of testing the reverse path, and consequently it will be transmitted with an AFx2 label.

Note that the basic GRIP operation, i.e., a single probe packet and a single feedback packet, is compatible with the H.323 call setup scheme using UDP, which encapsulates a H.225.0v2 call setup PDU into a UDP packet. Our solution seems then perfectly compatible with existing applications. In addition, this scheme leaves the service provider free to provide optional implementation details, including:

- Addition of proprietary signaling information in the probing packet payload or in the feedback packet payload, to be parsed, respectively, at the destination node or at the source node.
- Definition of more complex probing phase operation, e.g., by including reattempt procedures after a setup failure, multiple timers and probes during the probing phase, etc.

As a last consideration, it is quite interesting to remark that this idea is extremely close to what TCP congestion control technique does, but it is used in the novel context of admission control: end points interpret failed receptions of probes as congestion in the network and reject the relevant admission requests.

3.3 Possible roles of the AFx3 level

In the above description, the AFx3 drop level appears in principle unnecessary. However, it may be convenient to use this level. A first possibility is that the AFx3 level be used to mark non-conforming packets, which can be eventually delivered if network resources are available. Second, AFx3 packet marking can be enforced over flows that have not successfully passed the described admission control test. This allows deploying a service model where high QoS is provided to flows that pass the admission control test, while best effort delivery is provided to initially-rejected flows. These latter flows may occasionally retry the setup procedure, by simply marking occasional packets as AFx2 (e.g. by adhering to a suitable backoff procedure), and may eventually receive the upgraded AFx1 marking when network resources become available (as testified by the eventual reception of an AFx1 feedback).

The usage of the AFx3 level as described above is targeted to increase the link utilization. However, [R2597] requires the drop probability for AFx3 to be greater (or at most equal) than AFx2. This implies that the link utilization is bounded by the possibly strict mechanism that triggers AFx2 packets dropping: when AFx2 packets receive a 100% dropping probability, all AFx3 packets must also be dropped to conform to the [R2597] specification. A more effective mechanism would consist in implementing a dropping algorithm for the AFx3 traffic not directly related to the AFx2 drop algorithm. However, this usage of the AFx3 level does not conform to the AF specification, since the AFx3 dropping probability may be eventually lower than AFx2.

A more interesting possible usage of the AFx3 level consists in providing a second control (probing) channel, in addition to AFx2. According to this solution, AFx1 traffic measurements trigger a dropping algorithm on the AFx3 traffic too, with stricter dropping conditions than the AFx2 dropping algorithm (i.e. AFx3 packets are

assumed to detect congestion, and notify it via packet drop, before AFx2 packets). This AFx3 probing class could request admission for flows with e.g., higher peak rate and bandwidth requirements than flows supported via the AFx2 probing class (i.e., we are adding a second implicit signaling pipe). Also, being the AFx3 channel more reactive to congestion conditions, its usage can be envisioned to provide lower access priority to network resources. This would improve fairness and avoid some kinds of sources to “steal” a large part of network resources.

3.4 Arguments for new PHB definitions?

Although this paper leaves untouched the basic AF PHB semantic, we feel that our suggested usage of AF is different (and quite unexpected) from what intended in RFC 2597. The services that are expected to make use of admission control are RTP/UDP streams with delay and loss performance requirements, whose support is currently envisioned by means of the EF PHB. On the contrary, AF appears designed to provide better than best effort support for generic TCP/UDP traffic. Thus, our study raises the case for the transformation of the (single) EF PHB into a PHB class (i.e. by adding an associated, "paired", probing pipe with a different DSCP). An alternative is defining new "paired" PHBs.

On a different prospective, paired PHBs can be envisioned to support more general control functions than admission control. For example, the TCP fast retransmission and recovery algorithm might take advantage of isolated data packets labeled as “control”, and thus expected to encounter loss if (controlled) congestion is encountered in the network.

4 PERFORMANCE ISSUES

The described admission control semantic provides a reference framework compatible with "current" AF implementations. Scope of this section is to provide, in section 4.1, some qualitative insights about the performance achievable by the GRIP operation. Then, in section 4.2 we show that poor performance are provided over RED-like mechanisms customarily used to implement AF PHBs. These results reported in this section allow us to conclude that new explicit traffic measurement module implementations appears necessary, if tight QoS support is aimed at.

4.1 Degree of QoS support in GRIP

Quantitative and tunable performance may be independently provided and specified by each administrative entity. Uniform implementation across a specific domain allows defining a quantitative view (e.g., a PDB [PDB01]) of the performance achievable within a considered DS domain. In this way, the refinements deemed necessary in [R2990] to provide service accuracy in the DiffServ architectural model could be considered as accomplished.

In fact, the performance achievable by the described end point admission control operation depends on the notion of congestion as the triggering mechanism for AFx2 packet discarding, which is left to each specific implementation. Each administrative entity may arbitrarily tune the optimal throughput-delay/loss operational point supported by its routers, by simply determining the aggregate AFx1 traffic target

supported in each router. The mapping of AFx1 throughput onto loss/delay performance in turns depends on the link capacities and on the traffic flow characteristics offered on the AF class x.

With this approach, it is possible to construct PDBs offering quantitative guarantees. A building block of such PDBs is the definition of specific measurement modules and AFx2 dropping algorithms. A generic dropping algorithm is based on suitable rules (or decision criteria). An example of a trivial decision criterion is to accept all AFx2 packets when the measured throughput is lower than a given threshold and reject all AFx2 packets when the AFx1 measurements overflow this threshold. The resulting performance depends upon the link capacity and the traffic model.

It is well recognized that target QoS performance can be obtained by simply controlling throughput (i.e., by aggregate measurements taken on accepted traffic). This principle is at the basis of more sophisticated state of the art MBAC implementations described in [BJS00, GRO99]. As a simple quantitative example, with 32 Kbps peak rate Brady on-off voice calls (see section 4.2) offered to a 2 Mbps (20 Mbps) link, a target link utilization of 75% (92%) leads to a 99th percentile per hop delay lower than 5ms - see figure 3 (reproduced from [BCP00]).

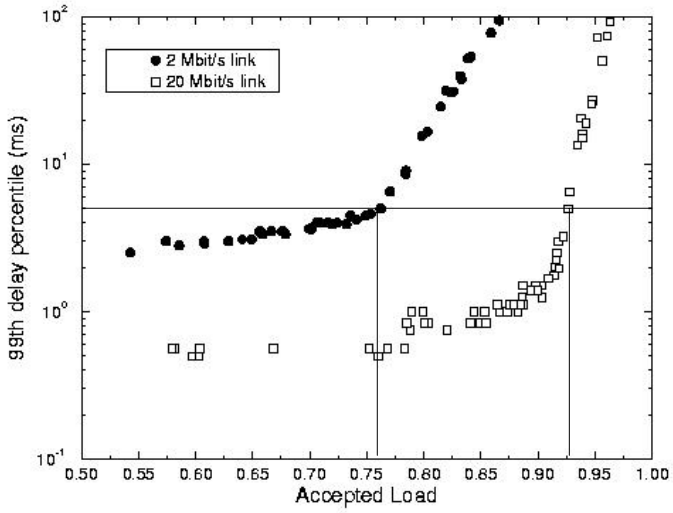

Fig. 3 - 99th delay percentile versus throughput, for different EAC schemes (see [BCP00]) and related parameter settings

Tighter forms of traffic control are possible. As a second example of a decision criterion, we demonstrated that hard (loss and/or delay) QoS guarantees can be provided, under suitable assumptions on the offered traffic (i.e., traffic sources regulated by standard Dual Leaky Buckets, as in the IntServ framework) and with ad hoc defined measurement modules in the routers [BB01a]. When hard QoS guarantees are aimed at, it is furthermore necessary to solve the problem of simultaneous activation of flows. In fact, the Gauge&Gate operation implies that

simultaneous setup attempts (i.e., probes arriving at the router within a very short time frame) may see the router in the ACCEPT state, and thus may lead to concurrent activation of several flows, which can in turns overload the router above the promised QoS level. Actually, this is a common and recognized problem of any MBAC scheme that does not rely on explicit signaling. Although it does not compromise the stability of described operation (overloaded routers close the gate until congestion disappears - see the mathematical formalization of such a problem and the computation of the "remedy" period in [GRO99]), this can be a critical issue if strict QoS guarantees are aimed at. In [BB01a] we have solved this problem by introducing an aggregate stack variable, which takes into account “transient” flows, i.e. flows elected at the state of "accepted" but not yet emitting information packets. This stack protection scheme avoids the concurrent activation of a number of flows, which could overload the router above the promised QoS level: the price to pay is a slight under-utilization of the available link capacity.

Another important issue is what happens when traffic flows with widely different peak rates are offered to the same link. Several approaches may be adopted. The simplest one is to differentiate traffic aggregates into classes of similar traffic characterization (e.g. similar peak rate), and associate to each class a different AF PHB class x, each with its probes AFx2 and data packets AFx1. A second possibility is to multiplex traffic onto a same AF PHB class, but differentiate probing packets by using the AFx3 drop level for traffic flows with higher peak rate (as briefly sketched in section 3.3).

Finally, we note that the AFx2 dropping algorithm must not be necessarily driven by IP-level traffic measurements. In fact, it can be driven by lower layers QoS capabilities (e.g., ATM).

4.2 Performance of GRIP over RED implementations of the AF PHB

In this paper we have demonstrated that, quite surprisingly, admission control can be deployed over the standardized Assured Forwarding PHB. In the previous section 4.1, we have concluded that arbitrary degree of performance guarantees can be obtained by designing specific AF PHB implementations based on runtime traffic measurements. A question that comes out naturally is the following. As long as Random Early Discarding (RED) queue management is customarily considered as the “natural” AF PHB implementation[2] , what are the performance of GRIP when it is operated over RED queues?

In our simulation program, we have assumed, for convenience, only two drop levels, namely AFx1 and AFx2. We have adopted a single buffer for both AFx1 and AFx2 packets. AFx1 packets are dropped only if the buffer is completely full.

[2] We recall that the AF PHB specification [R2597] does not recommend, by any means, a specific implementation. Indeed, RED have emerged as the natural AF PHB implementations, since they provide improved performance when TCP traffic (i.e., the traffic traditionally envisioned for AF) is considered. We now face a very different problem, i.e. what happens to performance when widespread RED AF PHB implementations are used for a completely different purpose, i.e. to support admission controlled (UDP) traffic.

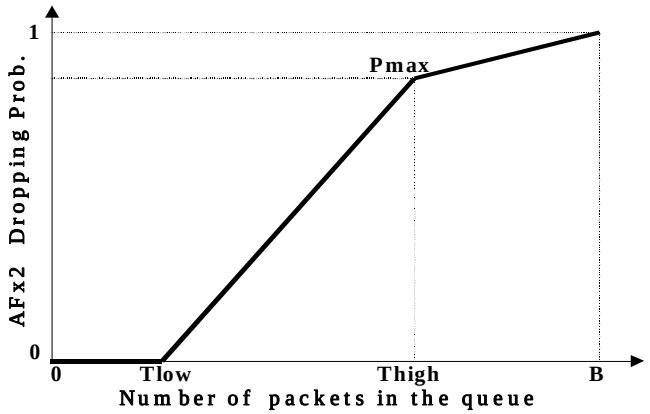

Fig. 4 – Buffer management scheme

Instead, AFx2 packets are dropped according to a Random Early Discarding (RED) management, where the AFx2 dropping probability is a function of the queue occupancy, computed accounting for both AFx1 and AFx2 packets[3]. As shown in figure 4, the AFx2 dropping probability versus the queue occupancy is a piecewise linear curve: no AFx2 packets are dropped when the number of packets stored in the queue is lower than a lower threshold. As the number of packets gets greater than the lower threshold, the AFx2 dropping probability increases linearly, until a value P_{max} is reached in correspondence with an upper threshold. After this value, the dropping probability is either set at 100% (in some RED implementations), or increased linearly until the number of packets fills the buffer capacity (see figure 4). It results that a RED implementation of the AFx2 dropping algorithm is completely specified by means of 4 parameters: the buffer size, the lower and upper thresholds, and the value P_{max}.

Depending on the considered implementation, the AFx buffer occupation is either sampled at the arrival of an AFx2 packet, or suitably smoothed/filtered in order to capture the moving average of the AFx queue occupancy. However, in all implementations proposed in the literature, an AFx2 packet that finds no packets stored in the buffer is always accepted (regardless of the fact that the smoothed AFx queue occupation may give a value different from 0). We will see in what follows that, ultimately, this specific condition prevents GRIP to achieve effective performance, regardless of the RED parameter settings considered.

Performance results have been obtained via simulation of a single network link, loaded with offered calls arriving at the link according to a Poisson process. Each offered call generates a single probe packet. A sufficiently large probing phase timeout has been set to guarantee that calls are accepted when the probing packet is not dropped (in the simulation, we have attempted to simulate conditions as close to ideal as possible, in order to avoid that numerical results were affected by marginal

[3] The described operations can also be seen as a particular case of a standard WRED implementation [CIS], where the T_{low} and T_{high} thresholds for AFx1 packets both coincide with the buffer size.

parameters settings – e.g round trip time, probing phase timer, etc). Accepted calls have been modeled as Brady ON-OFF sources, with peak rate equal to 32 Kbps, and ON/OFF periods exponentially distributed with mean value, respectively, 1 second for the ON period, and 1.35 seconds for the OFF period (yielding an activity factor equal to 0.4255). Each call lasts for an exponentially distributed time, with mean value 120s. The link capacity has been set to 2 Mbps. Therefore, The link results temporarily overloaded when more than 146.9 active connections (2000/(0.4255x32)) are active at a given instant of time.

Figures 5 to 7 report the number of accepted calls versus the simulation time for three different load conditions: underload (normalized offered load equal to 75% of the link capacity), slight overload (110% offered load), and harsh overload (400% offered load).

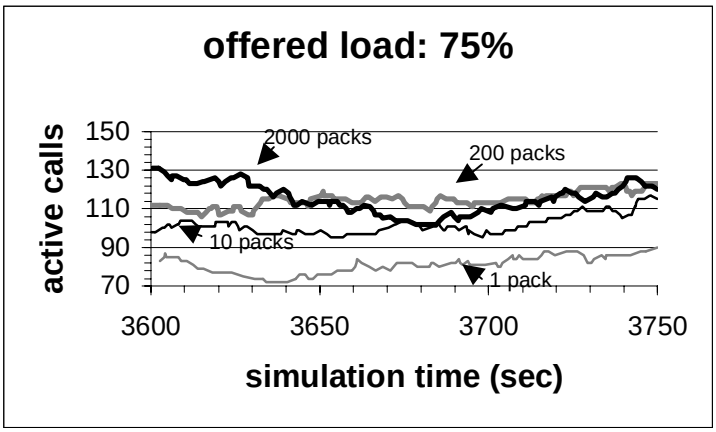

Fig. 5 - Active call vs. simulation time, with 75% offered load; "packs" indicates the AFx2 threshold.

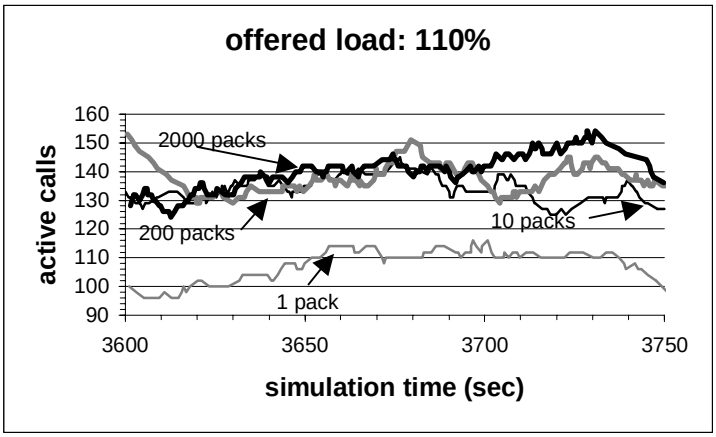

Fig 6 - Active call vs. simulation time, with 110% offered load. "packs" indicates the AFx2 threshold.

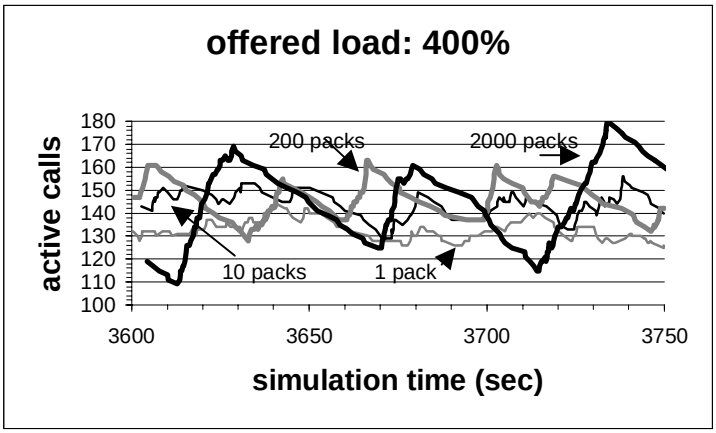

Fig 7 – Active call vs. simulation time, with 400% offered load. "packs" indicates the AFx2 threshold.

In the above figures, we have considered a very large AFx buffer size, such that no AFx1 packet losses occur (thus, QoS of accepted traffic is quantified in terms of AFx1 packet delay). We have tried several different RED parameters configuration, but, for simplicity of presentation, we report results related to a basic parameters settings where a single threshold is considered: all AFx2 packets are accepted whenever the number of AFx packets is lower than this threshold, while all AFx2 packets are dropped when the number of AFx packets is greater than this threshold (very similar results are obtained with more complex RED configurations, as it will be clear from the following discussion). Similarly, for simplicity, no smoothing on the buffer occupancy has been performed.

Figure 5 shows that, in low load conditions, a tight threshold setting (just 1 packet, i.e. an AFx2 packet is dropped as long as just 1 AFx packet is stored in the buffer) is overly restrictive, and exerts a high and unnecessary blocking probability on offered calls. With larger thresholds (200 and 2000 packets), we note from figure 5 that the number of accepted calls fluctuates in the range 100 to 120, meaning that just a small fraction of the offered calls are blocked by the GRIP operation (as it should ideally occur if a stateful admission control algorithm were operated).

Much more interesting and meaningful (for our purposes) are the results presented in figures 6. Here, we see that, in slight overload conditions, the only RED configuration setting that allows to keep the accepted load lower than the target 75% value (i.e. about 110 accepted calls) suggested by figure 3, is the threshold set to just 1 packet. Even with a very small threshold value, such as 10 packets, we see that the average number of accepted calls gets much greater than 110, thus resulting in unacceptable delay performance for accepted traffic.

Throughput and delay performance are quantified in table 8, for 110% and 400% offered load conditions. The table reports the AFx1 throughput, as well as the 95th and 99th delay percentiles experienced by accepted flows. Confidence intervals corresponding to a 95% confidence level are also reported in the table to quantify the accurateness of the numerical results.

AFx2 thresh	Offered load	Throughput		95th delay percentile (ms)		99th delay percentile (ms)	
1 pack	110%	68,64%±	0,28%	2,3±	0,014	3,3±	0,019
1 pack	400%	90,89%±	0,08%	80,6±	1,2	175,2±	1,3
10 packs	110%	88,71%±	0,22%	55,4±	0,7	148,444±	3.0
10 packs	400%	97,72%±	0,03%	586,3±	4,8	987,2±	10,2
200 packs	110%	93,11%±	0,28%	235,9±	4,0	433,8±	3,5
200 packs	400%	99,02%±	0,02%	1282,8±	15,9	2023,3±	75,1
2000 packs	110%	96,39%±	0,24%	1433,9±	20,0	1975,6±	26,9
2000 packs	400%	99,77%±	0,03%	4656,0±	24,9	5921,8±	58,2

Fig. 8 - Throughput and delay perfomance.

From the table, we see that the only case in which we meet target QoS performance for IP telephony (i.e. 99 th delay percentile of the order of few ms) is the case of threshold set to 1 packet, and light overload. It is quite impressive to note that the smallest possible RED threshold (i.e. drop an AFx2 packet whenever the AFx queue is not strictly empty) does not succeed in guaranteeing QoS in large overload conditions. This result allows us to conclude that, regardless of the RED parameters configuration, a RED implementation is never capable of guaranteeing QoS in all load conditions. As long as a RED implementation always accepts an AFx2 packet when the AFx queue is empty[4] , performance will be at best equal to that reported in table 8, for a threshold equal to 1 packet.

As figure 7 shows, in high overload conditions, thresholds greater than 1 packet cannot even avoid that, temporarily, the number of accepted calls is greater than 146.9, i.e. that the link is overloaded. In such a case, load oscillation phenomena occur: as clearly shown in figure 7, the link alternates between periods of significant overload (in which the AFx buffer fills up), and "remedy" [GRO99] periods, where the router locks in the REJECT state, until congestion disappears. The result is that 95th and 99th delay performance are of the order of several seconds (see table 8).

To conclude this section, we observe that, although RED implementations are intrinsically uncapable of providing performance guarantees, indeed a proper parameter setting allows to achieve reasonably better than best effort performance. For example, with a threshold set to 10 packets, table 8 shows that, in very high (unrealistic) load conditions, the 99th delay is still lower than 1 second (although the link has already been congested, as proven by the link load fluctuations, of the order of 15% of the link capacity, and leading to temporary link overload, as shown in figure 7). Notably, with the same 10 packets thresholds, the 99th delay percentile drops down to less than 150ms when light overload conditions are considered. Such

[4] We recall that this specific rule is proper of all RED implementations. I.e., regardless of the smoothing and filtering scheme adopted on the number of AFx packets, an AFx2 packet is never dropped when an empty AFx buffer is found. Changing this rule means changing the intrinsic logic of the RED approach, i.e. moving from queue status measurements to crossing traffic measurements. What we have proven here is that such a leap is required in AF implementations if QoS guaranteed admission controlled services are to be supported.

degree of QoS support might be considered sufficient in a short-term perspective, where current AF implementations might be utilized to support admission control.

5 CONCLUSIONS

In this paper, we have shown that the standard DiffServ AF PHB is semantically capable of supporting stateless and scalable admission control. The driving idea is that accept/reject decisions are taken at the edge of the network on the basis of probing packet losses, being probes tagged with a different AF level than packets generated by accepted traffic. In turns, these losses are driven by the dropping algorithm adopted in the specific AF implementation running at each network router. It is important to understand that, following the spirit of DiffServ, the above described operation does not aim at providing a quantified level of assured QoS, but, in conformance with PHB and PDB specifications, it provides a reference framework over which quantitative performance specification may be deployed. To the purposes of this paper, it was in our opinion sufficient to show that the described operation is compliant with the specification of the AF PHB.

The key to QoS guarantees is left to each specific implementation, i.e. is left to the implementation-dependent quantification of the notion of congestion, which triggers the gate mechanism for AFx2 packet discarding. Each administrative entity is in charge of arbitrarily determine the optimal throughput/delay operational point it wants to support.

The AF PHB implementations so far considered in the literature, based on (RED) thresholds set on the queue occupancy, do not affect the described endpoint operation and thus may be considered to support admission controlled traffic. However, an addition important contribute of this paper was to prove that RED implementations are uncapable of achieving tight QoS support, regardless of their parameter settings. In fact, the "measurement" mechanism adopted is overly simple, and this translates into a poor QoS support, but still much better than best effort, since admission control is still enforced on setting up connections. Indeed, this guarantees that the described GRIP operation can be supported over already deployed AF routers to seamless improve QoS with no internal routers modification: it could be sufficient for short-term perspectives, but deeper enhancements are required in order to satisfy coming needs, in a long-term perspective.

REFERENCES

[ALM98] W.Almesberger, T.Ferrari, J. Y. Le Boudec: "SRP: a Scalable Resource Reservation Protocol for the Internet", IWQoS'98, Napa (California), May 1998.

[BB01a] G. Bianchi, N. Blefari-Melazzi: " A Migration Path for the Internet: from Best-Effort to a QoS Capable Infrastructure by means of Localized Admission Control", Lecture Notes on Computer Science, Springer-Verlag, volume 1989, January 2001 (a more detailed technical report can be found at http://drake.diei.unipg.it/netweb/GRIP_tech_rep.pdf).

[BB01b] G. Bianchi, N. Blefari-Melazzi: "Admission Control over AF PHB groups", internet draft, draft_bianchi_blefari_admcontr_over_af_phb.txt, March 2001, work in progress.

[BCP00] G. Bianchi, A. Capone, C. Petrioli, "Packet Management Techniques for Measurement Based End-to-end Admission Control", KICS/IEEE Journal on Communications and Networking, June 2000.

[BJS00] L. Breslau, S. Jamin, S. Schenker: "Comments on the performance of measurement-based admission control algorithms", Proc. of IEEE Infocom 2000, Tel-Aviv, March 2000.

[BOR99] F. Borgonovo, A. Capone, L. Fratta, M. Marchese, C. Petrioli, "PCP: A Bandwidth Guaranteed Transport Service for IP networks", IEEE ICC'99, June 1999.

[BRE00] L. Breslau, E. W. Knightly, S. Schenker, I. Stoica, H. Zhang: "Endpoint Admission Control: Architectural Issues and Performance", ACM SIGCOMM 2000, Stockholm, Sweden, August 2000.

[CIS] Cisco IOS Quality of Service Solutions Configuration Guide—Online Manual. http://www.cisco.com/univercd/cc/td/doc/product/software/ios120/12cgcr/qos_c/

[CKN00] C. Cetinkaya, E. Knightly, "Egress Admission Control", Proc. of IEEE Infocom 2000, Tel-Aviv, March 2000.

[ELE00] V. Elek, G. Karlsson, "Admission Control Based on End-to-End Measurements", Proc. of IEEE Infocom 2000, Tel Aviv, Israel, March 2000.

[FVJ93] S. Floyd and V. Jacobson. "Random Early Detection Gateways for Congestion Avoidance". IEEE/ACM Transactions on Networking, vol. 1, no. 4, pp. 397-413, August 1993.

[GKE99] R. J. Gibbens, F. P. Kelly, "Distributed Connection Acceptance Control for a Connectionless Network", 16th ITC, Edimburgh, June 1999.

[GRO99] M. Grossglauser, D. N. C. Tse: "A Framework for Robust Measurement Based Admission Control", IEEE/ACM Transactions on Networking, Vol. 7, No. 3, June 1999.

[KEL00] F. P. Kelly, P. B. Key, S. Zachary: " Distributed Admission Control", IEEE JSAC, Vol. 18, No. 12, December 2000.

[MOR00] R. Mortier, I. Pratt, C. Clark, S. Crosby: "Implicit Admission Control", IEEE JSAC, Vol. 18, No. 12, December 2000.

[PDB01] K. Nichols, B. Carpenter, "Definition of Differentiated Services Per Domain Behaviors and Rules for their Specification", draft-ietf-diffserv-pdb-def-03, January 2001, Work in progress.

[R2205] R. Braden, L Zhang, S. Berson, S. Herzog, S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC2205, September 1997.

[R2210] J. Wroclawsky, "The use of RSVP with IETF Integrated Services", RFC2210, September 1997.

[R2474] K. Nichols, S. Blake, F. Baker, D. Black, "Definitions of the Differentiated Service Field (DS Field) in the Ipv4 and Ipv6 Headers", RFC2474, December 1998.

[R2475] S. Blade, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services", RFC2475, December 1998.
[R2597] J. Heinanen, F. Baker, W. Weiss, J. Wroclavski "Assured Forwarding PHB Group", RFC 2597, June 1999.
[R2990] G. Huston, "Next Steps for the IP QoS Architecture", RFC2990, November 2000.
[R2998] Bernet, Y., Yavatkar, R., Ford, P., Baker, F., Zhang, L., Speer, M., Braden, R., Davie, B., Wroclawski, J. And E. Felstaine, "A Framework for Integrated Services Operation Over DiffServ Networks", RFC 2998, November 2000.
[SZH99] I. Stoica, H. Zhang, "Providing guaranteed services without per flow management", Proc. of ACM SIGCOMM 1999, Cambridge, MA, September 2000.