# Fair Assured Services without Any Special Support at the Core*

Sergio Herrería-Alonso, Manuel Fernández-Veiga, Andrés Suárez-González, Miguel Rodríguez-Pérez, and Cándido López-García

Departamento de Enxeñería Telemática, Universidade de Vigo
Campus universitario, 36310 Vigo, Spain
sha@det.uvigo.es

**Abstract.** Many users require IP networks with the capacity to guarantee a minimum throughput even during periods of congestion. Furthermore, it is also desirable to share the excess unsubscribed bandwidth among active users if aggregate demand does not exceed network capacity. This kind of service, named assured service, can be provided through the *Assured Forwarding* (AF) *Per Hop Behavior* (PHB) defined in the DiffServ architecture. DiffServ mechanisms require special networking support at both the edge and the core nodes to guarantee the differentiated service. In this paper we propose the *Ping Trunking* scheme as a suitable mechanism to provide assured services to network users without the need for modifying core nodes. *Ping Trunking* is an edge-to-edge management technique that completely addresses the regulation of aggregate traffic streams at the edge of the network. In addition, it also overcomes some unfairness issues found in AF when sharing the available bandwidth among heterogeneous aggregates. Simulation results have validated the effectiveness of our proposal.
**Keywords**: DiffServ, assured services, aggregated traffic, congestion control.

## 1 Introduction

The Internet best effort model with no service guarantee is no longer acceptable in view of the proliferation of interactive applications such as Internet telephony, video conferencing or networked games. The growing importance of these recent applications with stringent constraints behooves the research community to develop a new range of network services able to accommodate heterogeneous application requirements and user expectations.

Among the new services demanded, assured services are one of the most popular. Assured services must provide different levels of forwarding assurances for IP packets. For instance, many users just require a guarantee that IP packets

* This work was supported by the "Ministerio de Educación y Ciencia" through the project TIC2003-09042-C03-03 of the "Plan Nacional de I+D+I" (partially financed with FEDER funds) and by the "Secretaría Xeral de Investigación de la Xunta de Galicia" through the grant PGIDT04PXIC32203PN.

are forwarded with high probability as long as their aggregate traffic streams do not exceed their committed information rate. In addition, it is also desirable that users may exceed their subscribed profiles with the understanding that the excess traffic is not forwarded with as high probability as the traffic that is within the profile. This kind of services can be provided through the *Assured Forwarding* (AF) [1] *Per Hop Behavior* (PHB) defined in the DiffServ architecture [2]. The differentiated service is obtained through traffic conditioning and packet marking at the edge of the network along with differentiated forwarding mechanisms at the core. Consequently, every node in a DiffServ network must be adapted to provide the required differentiation.

In [8] we proposed a new edge-to-edge management scheme named *Ping Trunking* able to provide some service guarantees to aggregate traffic streams. Our proposal establishes a Vegas-like control connection between the ingress and egress node of each aggregate. This connection regulates the flow of the aggregate traffic stream into the core of the network enforcing congestion control for the managed aggregate at its ingress node. In this paper, we argue that *Ping Trunking* can be used to offer assured services without requiring any special support at the core. Our proposal addresses all control tasks at the edges of the network so that the core nodes do not need to support any particular function for service differentiation. This feature makes our proposal easily deployable and improves its interest considerably.

In addition, *Ping Trunking* also overcomes some unfairness issues found in AF when sharing the available bandwidth. Several studies have shown that the number of flows in aggregates, the round trip time, the mean packet size and the TCP/UDP interaction are key factors in the throughput obtained by competing aggregates [3, 4]. With the help of some simulation experiments, we show how our proposal is able to distribute the available bandwidth among heterogeneous aggregates in a fair manner.

The rest of the paper is organized as follows. Section 2 gives a brief overview of AF PHB. In Sect. 3, we illustrate the *Ping Trunking* mechanism. Section 4 describes the simulation configuration used for the evaluation of both techniques. In Sect. 5, we present the results obtained from the simulation experiments. Section 6 briefly describes other approaches proposed to improve fairness requirements for assured services. We end this paper with some concluding remarks and future lines in Sect. 7.

## 2 Assured Forwarding PHB

AF distinguishes four classes of delivery for IP packets and three levels of drop precedence per class. Each AF class has a certain amount of buffer space and bandwidth reserved in each node. Within each class, IP packets are marked based on conformance to their target throughputs. The *Time Sliding Window Three Color Marker* (TSWTCM) is one of the most interesting packet marking algorithms proposed to work with AF [5]. In this algorithm, two target rates are defined: the *Committed Information Rate* (CIR) and the *Peak Information*

*Rate* (PIR). Under TSWTCM, the aggregated traffic is monitored and when the measured traffic is below its CIR, packets are marked with the lowest drop precedence, *AFx1*. If the measured traffic exceeds its CIR but falls below its PIR, packets are marked with a higher drop precedence, *AFx2*. Finally, when traffic exceeds its PIR, packets are marked with the highest drop precedence, *AFx3*.

At the core of the network, the different drop probabilities can be achieved using the RIO (*RED with In/Out*) scheme [6], an active queue management technique that extends RED gateways [7] to provide service differentiation. RIO is configured with three different sets of RED parameters, one for each of the drop precedence markings. These different RED parameters cause packets marked with a higher drop precedence to be discarded more frequently during periods of congestion than packets marked with a lower drop precedence.

## 3 Ping Trunking

*Ping Trunking* [8] is an edge-to-edge management technique that provides some service guarantees to aggregate traffic streams. Our proposal is based on a technique named *TCP Trunking* [9, 10], but we have extended and improved the original scheme to manage aggregates in a simpler and smoother way.

A ping trunk is an aggregate traffic stream where data packets are transmitted at a rate dynamically determined by a preventive congestion control algorithm. Each trunk carries a varying number of user flows for common treatment between two nodes of the network (the ingress and the egress nodes). The flow of the aggregated traffic is regulated by a single control connection established between the two edges of the trunk. This control connection injects control packets into the network to probe its congestion level. The introduction of control packets is not conditioned by the user data protocols, but it is only determined by the control connection itself. In addition, a trunk will not retransmit user packets if they are lost. If it is required, retransmissions should be handled by user applications on the end hosts.

Figure 1 provides greater detail on the operation of this mechanism. Incoming user packets at the ingress node are classified as belonging to a particular trunk and queued in the corresponding trunk buffer. User packets can only be forwarded when credit for their trunk is available. The credit value represents the amount of user data allowed to be forwarded. When a user packet is sent, the credit is decremented by the size of the packet. When a control packet is sent, the credit is incremented by the size of the control congestion window (*cwnd*). Therefore, the transmission of user data is regulated by both the forwarding of control packets and the *cwnd* value.

It is important to point out that both user and control packets must follow the same path between the edges of the trunk to ensure that control connections are probing the proper available bandwidth. This assumption can be absolutely guaranteed if trunks are run on top of ATM virtual circuits or MPLS label-switched paths [11].

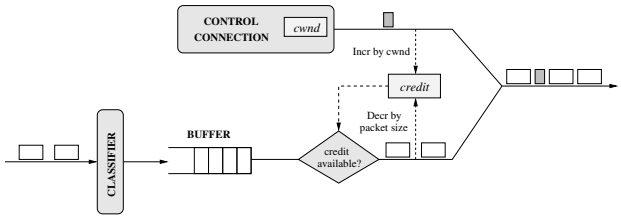



**Fig. 1.** Block diagram. This figure includes several simplifications. In fact, each trunk has its own buffer, credit bucket and control connection.

### 3.1 Operations of the Control Connection

The control connection is in charge of measuring the round-trip time (RTT) between the edges of the trunk accurately. Then, a Vegas-like congestion control mechanism will employ this RTT estimate when adapting the transmission rate of the trunk. Let us start by giving a brief description of the operations accomplished by this connection.

When the first user packet arrives to the ingress node of the trunk, a control packet is generated and sent. For each control packet that reaches the egress node of the trunk, its corresponding acknowledgment (ack) is generated. The arrival of an ack back at the ingress node triggers the transmission of a new control packet. Therefore, the control connection only sends control packets on reception of acks, i.e., once per RTT.

To avoid the starvation of control connections, a waiting time-out timer is needed. This timer is started every time a control packet is sent. If the ack of the last control packet sent does not arrive to the ingress node before the timer expires, the connection will consider that the packet has been lost and it will send a new control packet.[1]

Control connections use a method similar to that used in the TCP estimation of RTT. To carry out this task, they timestamp with its local time every control packet and this timestamp is echoed in the acks. The value of the last RTT sample observed is computed as the difference between the current time and the timestamp field in the ack packet. The RTT is eventually estimated using an exponential moving average taken over RTT samples.

### 3.2 Vegas-Like Congestion Control Mechanism

The transmission rate of each trunk should be able to update dynamically according to current network conditions. We propose the use of a Vegas-like congestion control mechanism to discover the available bandwidth that each trunk should obtain. TCP Vegas [12] is an implementation of TCP that employs proactive techniques to increase throughput and decrease packet losses. The congestion

[1] The control connections employed behave like the *ping* command used to send ICMP ECHO_REQUEST/REPLY packets to network hosts. Hence the name of our proposal.

control mechanism introduced by Vegas gives TCP the ability to detect incipient congestion before losses are likely to occur, so we have devised a similar mechanism adapted to trunks.

Upon receiving each ack, control connections calculate the expected throughput and the current actual throughput as in TCP Vegas. If it is assumed that trunks are not overflowing the path, the expected throughput can be calculated as $cwnd/d$, where $d$ is the round-trip propagation delay and can be estimated as the minimum of all measured RTTs. On the other hand, the actual throughput is given by $cwnd/D$, where $D$ is the RTT estimation. These throughputs are compared and then, control connections adjust their congestion windows accordingly. Let *Diff* be the difference between the expected and the actual throughput:

$$Diff = Expected - Actual = \left(\frac{cwnd}{d} - \frac{cwnd}{D}\right)d \ . \tag{1}$$

The *Diff* value has been scaled with the minimum RTT so that *Diff* can be seen as the amount of user data in transit.

There are two thresholds defined: $\alpha$, $\beta$, with $\alpha \leq \beta$. When $Diff < \alpha$, the trunk is allowed to increment its amount of user data in transit, and therefore, the control connection can increase its congestion window linearly. If $Diff > \beta$, the trunk is forced to decrease its congestion window linearly. In any other case, the congestion window remains unchanged.

This mechanism stabilizes the value of the congestion window and reduces packet drops. If a control packet loss is detected, the available bandwidth is halved, but this should happen sporadically.

### 3.3 Setting of Vegas Parameters

We should determine the suitable values of Vegas parameters that must be assigned to each trunk so that they can obtain their fair share of the available bandwidth. Consider a bottleneck shared by a set of trunks indexed by $i$. Let $C$ denote the bottleneck capacity. Each trunk $i$ has associated a subscribed target rate $r_i$. The overall demand $R$ is the sum of the subscribed target rates for all active trunks. If $R < C$, the excess unsubscribed bandwidth should be distributed among trunks in proportion to the contracted target rates. Therefore, the fair bandwidth $f$ that should be ideally allocated to each trunk $i$ is obtained as

$$f_i = r_i + (C - R)\frac{r_i}{R} = \frac{r_i C}{R} \ , \tag{2}$$

where $R = \sum_i r_i$.

According to one interpretation of Vegas [13], congestion windows of control connections must satisfy the following equation in the equilibrium (we assume for simplicity that $\alpha_i = \beta_i$):

$$\left(\frac{cwnd_i}{d_i} - \frac{cwnd_i}{D_i}\right)d_i = \alpha_i \ . \tag{3}$$

On the other hand, the transmission rate $x$ of a given trunk is determined by the *cwnd* value of its corresponding control connection ($cwnd = xD$). Substituting this in (3), we have

$$\left(\frac{x_i D_i}{d_i} - \frac{x_i D_i}{D_i}\right) d_i = \alpha_i \ , \tag{4}$$

and, from (4), it follows that

$$x_i = \frac{\alpha_i}{D_i - d_i} \ . \tag{5}$$

The RTT can be calculated as the sum of two delays: the round-trip propagation delay ($d$) and the queueing delay ($B/C$), where $B$ denotes the total backlog buffered in the network. Then, from (5), and using $D_i = d_i + B/C$, the transmission rate of each trunk can be expressed as

$$x_i = \frac{\alpha_i C}{B} \ . \tag{6}$$

Finally, equating (2) and (6) yields the suitable value of $\alpha$ threshold that permits to allocate to each trunk its desired share of bandwidth:

$$\alpha_i = \frac{r_i B}{R} \ . \tag{7}$$

Therefore, to compute the $\alpha$ threshold, each trunk must know both $B$ and $R$ parameters. The $B$ parameter should have a fixed low value set by the network manager to encounter small queues at the core. However, the necessity of determining the overall aggregated demand in all edge nodes may complicate our proposal substantially.

Fortunately, we can demonstrate that it is not required to know the value of the overall demand very accurately. Assume $R' \neq R$, $R' > 0$, was used as the aggregated demand. The total backlog $B'$ actually buffered in the network is obtained as the sum of the $\alpha$ thresholds of all competing trunks. Then,

$$B' = \sum_i \alpha_i = \sum_i \frac{r_i B}{R'} = \frac{B}{R'} \sum_i r_i = \frac{BR}{R'} \ . \tag{8}$$

From (6), and using $B'R' = BR$ derived from (8), we can conclude that the fairness condition is still satisfied although the $R'$ value employed is false:

$$x_i = \frac{\alpha_i C}{B'} = \frac{r_i BC}{R'B'} = \frac{r_i C}{R} \ . \tag{9}$$

Taking into account this analysis, we propose to assign to each trunk the following value of $\alpha$:

$$\alpha_i = \frac{r_i B}{C} \ . \tag{10}$$

Computing $\alpha$ thresholds in this manner gives to each trunk its proportional share of the available bandwidth as desired but, in addition, this value is completely independent of the changing number and features of competing trunks.

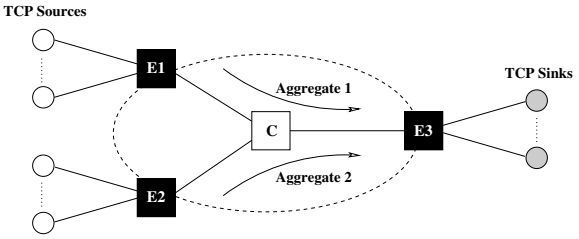



**Fig. 2.** Network topology. Every link has a 100 Mbps capacity and a propagation delay of 1 ms.

## 4 Simulation Configuration

We have implemented *Ping Trunking* in the ns-2 simulator [14]. Figure 2 shows the network topology employed. It consists of three edge nodes and one core node belonging to a particular domain. The edge nodes E1 and E2 are connected to several TCP traffic sources whereas TCP sinks are connected to the edge node E3. We consider two competing aggregates: aggregate 1 comprises all the traffic that flows from E1 to E3, and aggregate 2 comprises all the traffic between E2 and E3. Therefore, both aggregates pass through a single bottleneck (link C-E3). Each aggregate consists of 50 TCP flows. All TCP connections established are modeled as eager FTP flows that always have data to send and last for the entire simulation time. We use the TCP New Reno implementation [15] and the size of data packets is set to 1 000 bytes.

We consider two different scenarios. The first one represents a DiffServ domain with the two competing aggregates belonging to the same AF class. The marking scheme used in the edge routers is TSWTCM. The core node implements the RIO scheme: three sets of RED thresholds are maintained, one for each drop precedence. For the configuration of RIO parameters, the staggered setting [16] has been selected. The drop threshold values are shown in Fig. 3. The physical queue is limited to 150 packets and $w_q$ equals 0.002.

In the second scenario, instead of DiffServ facilities, we employ *Ping Trunking* to regulate user data transmission. In this case, a trunk is used to manage each aggregate traffic stream. Each trunk buffer is a simple FIFO queue with capacity for 25 packets. The core queue is also a FIFO buffer limited to 150 packets. Control connections send 48-byte packets.[2] The maximum total backlog $B$ is fixed to 50 packets.

Simulations run for 50 seconds. Each simulation experiment is repeated 10 times changing slightly the initial transmission time of each TCP flow and then, an average of the measured parameter and a 95% confidence interval for the mean value are taken over all runs. In any case, confidence intervals will not be represented in the graphs because they are lower than $\pm 1\,\%$.

[2] Control connections do not actually transmit any real data, so control packets only consist of the TCP/IP header plus the overhead of the timestamp option required to estimate RTTs.

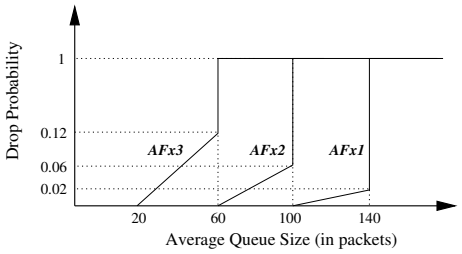
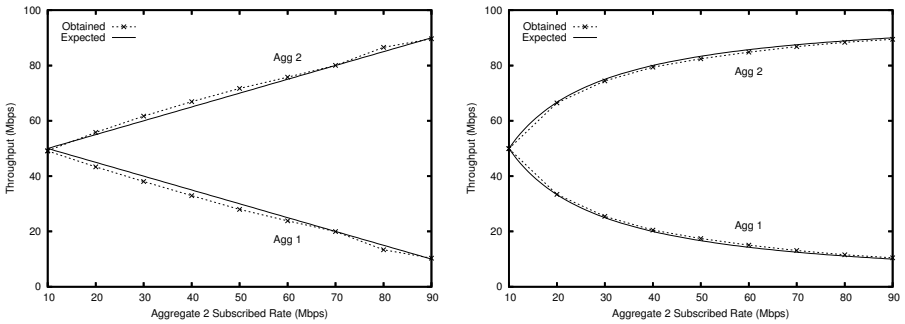



**Fig. 3.** Core RIO parameters in the AF scenario.

## 5 Experimental Results

### 5.1 Bandwidth Distribution

In this experiment, we evaluate the effectiveness of both mechanisms when sharing the network bandwidth. We consider that aggregate 1 contracts a fixed throughput of 10 Mbps while the subscribed target rate of aggregate 2 varies from 10 to 90 Mbps. In the DiffServ scenario, the CIR value of each aggregate is set to its corresponding subscribed rate and the PIR value is set to the CIR value plus 10 Mbps. Figure 4 shows the throughput obtained by each aggregate with both techniques. The proportional share of bandwidth that should be assigned to each aggregate is also shown. With AF, there is an even distribution of excess bandwidth irrespective of the subscribed rates. In contrast, *Ping Trunking* divides the excess bandwidth in proportion to the subscribed rates. Though both solutions are acceptable, we consider it is more desirable that users with higher target rates obtain higher shares of excess bandwidth since target rates depend on the price that users pay.




(a) AF scenario.

(b) Ping Trunking scenario.

**Fig. 4.** Bandwidth distribution experiment results.

### 5.2 Fairness Evaluation

In the previous simulations, both AF and *Ping Trunking* mechanisms have dealt with homogeneous aggregates, but, unfortunately, this is not always the usual scenario. Aggregates can be composed of different numbers of flows, can send packets of different sizes and can have different RTTs. In addition, aggregates can carry UDP flows, which are not congestion aware. In this section, we have conducted several simulations to verify that our proposal can be used to share the bandwidth among heterogeneous aggregates in a fair manner. We consider that the two competing aggregates have contracted the same target rate (10 Mbps).

The first key factor studied has been the number of flows in the competing aggregates. In this experiment, we consider that aggregate 2 contains a different number of TCP flows, varying from 25 to 75. Figure 5(a) shows the obtained results. In the AF case, the aggregate with a larger number of TCP flows obtains a greater share of the available bandwidth. However, with our proposal, each aggregate obtains an equal amount.

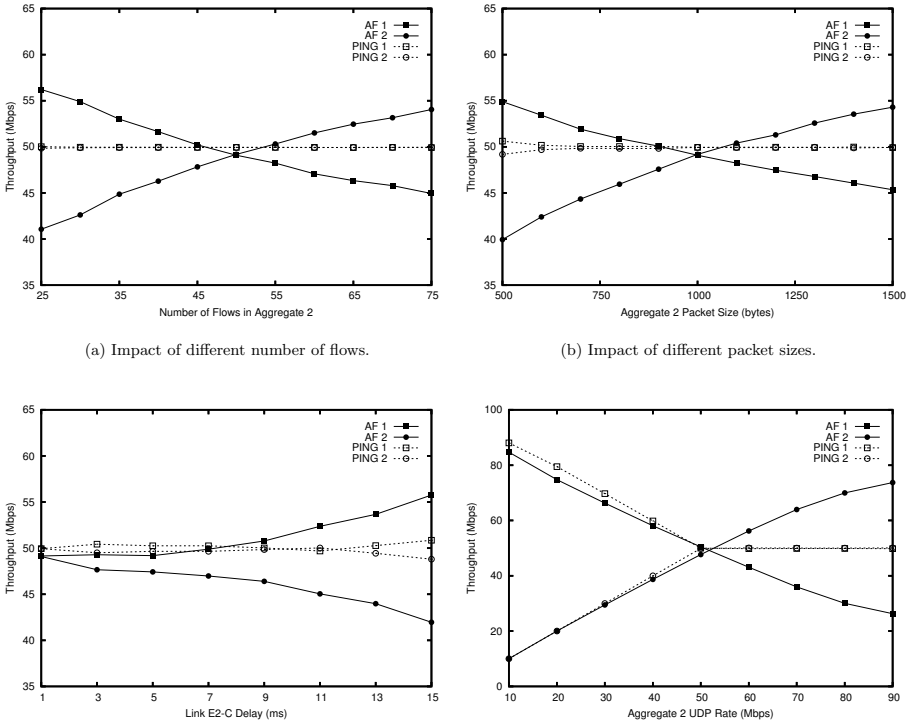
Fairness is also desired between aggregates carrying packets of different sizes. We have simulated a second experiment where aggregate 2 packet size increases from 500 to 1500 bytes. The results are shown in Fig. 5(b). Through AF, the aggregate that is sending larger packets consumes more of the available bandwidth. Under *Ping Trunking*, the sharing of bandwidth can be made insensitive to packet sizes.

Another important factor in the share of bandwidth is the RTT of the competing aggregates. In order to compare the throughput obtained by aggregates with different RTTs, we consider that the delay of the link that joins edge node E2 with the core node has been changed from 1 to 10 ms. As showed in Fig. 5(c), through AF, aggregates with different RTTs cannot achieve a fair share of bandwidth and the shorter the RTT, the higher the obtained throughput. In contrast, *Ping Trunking* is able to amend such unfairness giving to each aggregate its fair share.

Finally, it is important to protect responsive TCP flows from non-responsive UDP flows since this unresponsive traffic may impact the TCP traffic adversely. In this last experiment, aggregate 1 contains 50 TCP flows, while aggregate 2 has a single UDP flow with a sending rate increasing from 10 to 90 Mbps. Figure 5(d) shows the obtained results. Under the AF case, as the UDP rate increases, the amount of bandwidth obtained by the TCP aggregate decreases. With our proposal, this unfairness problem is absent and the bandwidth can be shared in a TCP-friendly manner.

## 6 Related Work

Many smart packet marking mechanisms have been proposed to overcome these unfairness issues found in AF. *Adaptive Packet Marking* [17] is one of these schemes able to provide soft bandwidth guarantees, but it has to be implemented inside the TCP code itself and thus, requires varying all TCP agents. Intelligent




(a) Impact of different number of flows.

(b) Impact of different packet sizes.

(c) Impact of different RTTs.

(d) TCP/UDP interaction.

**Fig. 5.** Fairness evaluation experiment results.

traffic conditioners proposed in [18] handle a subset of these fairness problems using a simple TCP model when marking packets. However, these conditioners require external inputs and cooperation among markers for different aggregates complicating both implementation and deployment. Another marking algorithm based on a more complex TCP model is *Equation-Based Marking* [19]. This scheme solves the fairness problems associated with heterogeneous TCP flows under diverse network conditions. Its behavior depends on the quality of the estimation of the current loss rate seen by TCP flows. Unfortunately, the calculation of this estimate is not an easy problem and complicates the deployment of the scheme extremely. In [20], an RTT-RTO aware conditioner is proposed, but this scheme only mitigates RTT bias. The *Counters-Based Modified* traffic conditioner [21] is able to cope with TCP flows with variable target rates and RTTs, but it cannot oversee UDP traffic. In [22], an adaptive token bucket algorithm able to provide each aggregate with its fair share of the available bandwidth in proportion to the target rate is presented. This approach is based on edge-to-edge feedback information conveyed in TCP acknowledgements, so it cannot be used to manage aggregates just containing UDP flows exclusively.

A different approach addresses these problems by enhanced RIO queue management algorithms. Examples of this technique are DRIO [23] and DAIO [24] schemes. Both techniques require maintaining state information of each individual flow at core routers. Since there can be thousands of active flows, these solutions need to store and manage a great amount of state information at the core of the network and therefore, they are not scalable. Other enhanced RIO algorithms such as EDRIO [25] and URIO [26] use proper buffer usage policing at the aggregate level to avoid scalability issues, but the need to adjust all the core routers still hinders their deployment.

## 7 Conclusions and Future Work

Both AF PHB and *Ping Trunking* mechanisms can be used to provide assured services to network users. However, the edge-to-edge management carried out by our proposal provides this service without the need for modifying core nodes. This feature is very interesting because it facilitates the deployment of our proposal substantially. In addition, our proposal guarantees the required fairness on bandwidth sharing among heterogeneous aggregate traffic streams.

In future work we plan to take advantage of *Ping Trunking* features to prioritize user traffic strictly based on the application type. For example, users could mark packets from highly interactive applications, such as Telnet or Web browsing, with a high priority, and packets from less interactive applications, such as FTP, with a lower one. Thus, if congestion occurs, low priority packets could be dropped more frequently at the trunk buffer using a suitable active queue mechanism. Following this approach, we will be in a position to fairly distribute the network bandwidth among competing aggregates while protecting interactive applications at the same time.

## References


1. Heinanen, J., Baker, F., Weiss, W., Wroclawski, J.: Assured Forwarding PHB group. RFC 2597 (1999)
2. Blake, S., Black, D., Carlson, M., Davis, E., Wang, Z., Weiss, W.: An architecture for differentiated services. RFC 2475 (1998)
3. de Rezende, J.F.: Assured service evaluation. In: Proceedings of IEEE GLOBECOM. (1999) 100–104
4. Seddigh, N., Nandy, B., Pieda, P.: Bandwidth assurance issues for TCP flows in a differentiated services network. In: Proceedings of IEEE GLOBECOM. (1999) 1792–1798
5. Fang, W., Seddigh, N., Nandy, B.: A time sliding window three colour marker (TSWTCM). RFC 2859 (2000)
6. Clark, D.D., Fang, W.: Explicit allocation of best effort packet delivery. IEEE/ACM Transactions on Networking **6** (1998) 362–373
7. Floyd, S., Jacobson, V.: Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on Networking **1** (1998) 397–413

8. Herrería-Alonso, S., Fernández-Veiga, M., Rodríguez-Pérez, M., Suárez-González, A., López-García, C.: Ping Trunking: A Vegas-like congestion control mechanism for aggregated traffic. Lecture Notes in Computer Science (QofIS'04) **3266** (2004) 104–113
9. Chapman, A., Kung, H.T.: Traffic management for aggregate IP streams. In: Proceedings of 3rd Canadian Conference on Broadband Research. (1999) 1–9
10. Kung, H.T., Wang, S.Y.: TCP Trunking: Design, implementation and performance. In: Proceedings of 7th Int. Conference on Network Protocols. (1999) 222–231
11. Rosen, E., Viswanathan, A., Callon, R.: Multiprotocol label switching architecture. RFC 3031 (2001)
12. Brakmo, L., O'Malley, S., Peterson, L.: TCP Vegas: New techniques for congestion detection and avoidance. In: Proceedings of ACM SIGCOMM. (1994) 24–35
13. Low, S., Peterson, L., Wang, L.: Understanding Vegas: A duality model. In: Proc. of ACM SIGMETRICS. (2001)
14. ns-2.27: The network simulator (2004)
15. Fall, K., Floyd, S.: Simulation-based comparison of Tahoe, Reno, and Sack TCP. Computer Communication Review **26** (1996) 5–21
16. Makkar, R., Lambadaris, I., Salim, J.H., Seddigh, N., Nandy, B., Babiarz, J.: Empirical study of buffer management schemes for diffserv assured forwarding PHB. Technical report, Nortel Networks (2000)
17. Feng, W., Kandlur, D., Saha, D., Shin, K.: Adaptive packet marking for maintaining end-to-end throughput in a differentiated-services internet. IEEE/ACM Transactions on Networking **7** (1999) 685–697
18. Nandy, B., Seddigh, N., Pieda, P., Ethridge, J.: Intelligent traffic conditioners for assured forwarding based differentiated services networks. Lecture Notes in Computer Science (Networking'00) **1815** (2000) 540–554
19. El-Gendy, M., Shin, K.: Equation-based packet marking for assured forwarding services. In: Proceedings of IEEE INFOCOM. (2002) 845–854
20. Habib, A., Bhargava, B., Fahmy, S.: A round trip time and time-out aware traffic conditioner for differentiated services networks. In: Proceedings of IEEE International Conference on Communications (ICC). (2002) 981–985
21. Cano, M.D., Cerdan, F., Garcia-Haro, J., Malgosa-Sanahuja, J.: Performance analysis of the counters-based modified traffic conditioner in a diffserv network. In: Proceedings of IEEE International Symposium on Computers and Communications (ISCC). (2003) 305–311
22. Park, E.C., Choi, C.H.: Proportional bandwidth allocation in diffserv networks. In: Proceedings of IEEE INFOCOM. (2004) 2039–2050
23. Lin, W., Zheng, R., Hou, J.: How to make assured services more assured. In: Proceedings of ICNP. (1999)
24. Su, L., Hou, J.: An active queue management scheme for internet congestion control and its application to differentiated services. In: Proceedings of IEEE ICCCN. (2000) 62–68
25. Herrería-Alonso, S., Fernández-Veiga, M., López-García, C., Rodríguez-Pérez, M., Suárez-González, A.: Improving fairness requirements for assured services in a differentiated services network. In: Proceedings of IEEE International Conference on Communications (ICC). (2004) 2076–2080
26. Herrería-Alonso, S., Rodríguez-Pérez, M., Fernández-Veiga, M., Suárez-González, A., López-García, C.: Unbiased RIO: An active queue management scheme for diffserv networks. Lecture Notes in Computer Science (ECUMN'04) **3262** (2004) 60–69