Internet Engineering Task Force
INTERNET DRAFT
Expires February, 1999

J. Ibanez
K. Nichols
Bay Networks
August, 1998

Preliminary Simulation Evaluation of an Assured Service
<draft-ibanez-diffserv-assured-eval-00.txt>

Status of this Memo

Abstract

This draft presents a simulation analysis of Assured Service, an end-to-end service based on the differentiated services enhancements for IP. Assured Service has been the subject of much discussion in the past year in the IETF, but solid information on its applicability to the entire Internet has been lacking. This report is aimed at providing a first step in this direction. Assured Service requires an active queue managment algorithm with preferential packet drop. The RIO algorithm (an extension of the RED algorithm) has been the primary method suggested and is the one evaluated here. Our results show that Assured Service does not provide clearly defined and consistent rate guarantees; the advantage gained by connections using Assured Service is not a quantifiable one. Further work would be required to determine an appropriate use of the Assured Service. A pdf version of this document is available and recommended for the figures it contains.

1. Introduction

The Assured Service (AS) as first defined in [SVCALLOC] is an example of an end-to-end

service that can be built from the proposed differentiated services enhancements to IP [HEADER] using a single PHB. This type of service is appealing in its apparent ease of deployment, but at the same time insufficient measurement and analysis has been done to determine whether its range of applicability and whether it scales to the entire Internet. This draft is a first step at giving some clues towards the answer.

This report analyzes Assured Service through use of simulations performed with ns-2 [ns], a network simulator developed by UC Berkeley, LBL, USC/ISI and Xerox PARC. A drop preference queue management mechanism is the required PHB to deploy AS. Though other queue managment techniques based on drop preference may be used, this document explores only the use of RIO, an extension of RED, and the PHB originally proposed to implement Assured Service [SVCALLOC, 2BIT].

The starting point for this work was Clark and Fang's "Explicit Allocation of Best Effort Delivery Service" [EXPALLOC], which presents interesting results, but in our opinion does not use appropriate traffic models and scenarios. This report goes a step further toward realistic Internet traffic patterns by mixing Assured traffic with best-effort traffic. As a result, we reach very different conclusions. We conclude that Assured Service cannot provide clearly defined and consistent guarantees, at least when applied to the whole Internet.

2. Differentiated Services, Network Model and Assured Service

2.1 Differentiated services

Differentiated services is described in [HEADER] and [ARCH] and an earlier version with discussion of AS in [2BIT]. In this draft, we have only introduced Assured service into a network model.

2.2 The network model

Simulations are discussed in more detail in the Appendix. Unless stated otherwise, simulations use the topology of figure 1. 50 point-to-point connections share a bottleneck link of 50 Mbps (6.25 Mbytes/s). 10 Mbps links connect each host to its respective router. Host hi is connected to host hi+50 and connections are identified by the sending host ID; for instance connection 0 is associated to host h0. The connections are "infinite" FTPs. Transfers are unidirectional, and ACKs are never lost, which implies that the performance of the communications will be better than in a real situation where ACKs can be lost in addition to data. A profiler with a target rate of 1 Mbps (125 Kbytes/s) is attached to each AS capable host; an aggregate policer is installed in the router just before the bottleneck link. RTTs are chosen randomly in the range 50..150ms for each connection, and starting times are randomly distributed within the first second of simulation time. Packet are 576 bytes.

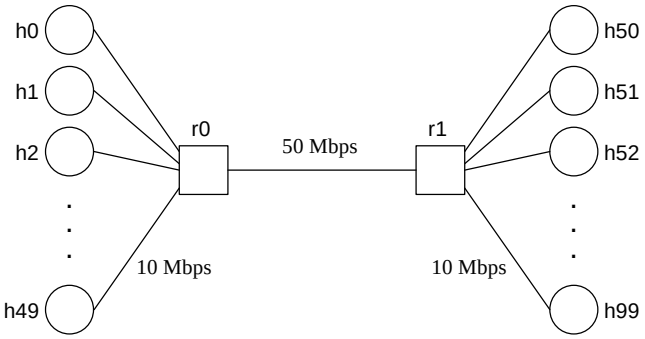


Figure 1: 50 point-to-point connections topology

2.3 Assured Service

Assured Service was first introduced by Clark and Wroclawski in [SVCALLOC], and is also considered in [2BIT]. The idea behind AS is to give the customer the assurance of a minimum throughput, even during periods of congestion, while allowing him to consume more bandwidth when the network load is low. Thus a connection using the assured service should achieve a throughput equal to the subscribed minimum rate, also called target rate, plus some share of the remaining bandwidth gained by competing with all the active best-effort connections.

The mechanism to achieve this goal is as follows. A profile describing the target rate and possibly a burst rate is contracted by the user. The edge device to which the host is connected, or the host itself if it is AS capable, measures the rate of the connection and as long as this rate is below the target rate, packets are marked as IN-profile. When the target rate is exceeded, packets are marked as OUT-of-profile. If a router somewhere in the path experiences congestion, it will preferentially drop packets marked as OUT. One implementation of this is to assume that OUT packets are treated the same as packets which are not profiled and thus are simply not marked at all. In this case, we simply say that a packet is "marked" to mean it is "marked as IN".

Though the goal of AS is to assure a minimum throughput to a connection while enabling it to get even better performance when the network load permits. However there is some concern about the fairness of the assured service against best-effort service. In severe congestion where there is not enough bandwidth to satisfy all the contracts, assured connections would compete against each other for the available bandwidth, depriving best-effort connections of any bandwidth. This situation is expected to be very unusual.

2.1.1 The RIO algorithm

The RIO algorithm allows two traffic classes within the same queue to be treated differently by applying a drop preference to one of the classes. RIO is an extension of RED [RED], "RED with In and Out", and is discussed in [SVCALLOC] and [EXPALLOC]. RIO can be viewed as the combination of two RED algorithms with different drop probability curves, chosen to give one group of packets preference. We assume familiarity with RED at the level of [RED].

For OUT packets, as long as the average queue size is below minth_out no packets are dropped.

If the average queue size exceeds this, arriving packets are dropped with a probability that increases linearly from 0 to maxp_out. If the average queue size exceeds maxth_out, all OUT packets are dropped. Note that the average queue size is based on the total number of packets in the queue, regardless of their marking.

For IN packets, the average queue size is based on the number of IN packets present in the queue and the parameters are set differently in order to start dropping OUTs well before any INs are discarded. However, when there are only OUT (or best-effort) packets, RIO has to perform much like RED. Therefore we have to set OUT parameters following almost the same rules as for RED. We observed in simulation that IN and OUT parameters need not be very different; the inherent discrimination produced by the average queue size calculation is enough.

2.1.2 Choice of the RIO parameters

We applied the general rules for RED [RED] and set a queue weight of 0.002, a minth_out of about 0.4·maxq_size, and maxth_out to about 0.8·maxq_size and maxp_out of 0.05. For IN parameters we took a different approach from that proposed in [EXPALLOC] or [SVCALLOC]. Those papers suggest that the thresholds be chosen much lower for OUT packets than for INs, to achieve enough differentiation. We found that by calculating the drop probability for OUT packets based on the total average queue size, and for IN packets based only on the average number of IN packets in the queue, the discrimination is already significant. Furthermore, if we set IN parameters more leniently, that may cause trouble if most of the packets arriving are marked IN. Choosing the thresholds for INs close to those of OUTs maintains consistent behavior with any proportion of marked traffic. We set minth_in to about 0.45·maxq_size and maxth_in to the same value as maxth_out, 0.8·maxq_size. For maxp_in we used 0.02.

The notation used throughout this report to represent the RIO parameters is:

minth_out /maxth_out/maxp_out minth_in/maxth_in /maxp_in

Following this notation and rounding the values, the simulation parameters we used are:

420/840/0.05 for IN parameters, and 500/840/0.02 for OUT parameters

2.1.3 Policing mechanism: average rate estimator or token bucket?

An interesting point when implementing the architecture is the choice of mechanism to check conformity to the contracted profile. This policing can be performed by the marker in the edge device or by the policer at a boundary between two domains.

We experimented with two different mechanisms: an average rate estimator, as presented in [EXPALLOC], and a token bucket. When the average rate estimator measures a rate that exceeds the contracted target rate for a given flow, the policer marks OUTs with a linearly increasing probability. This was designed for TCP connections, but for other applications the average rate

estimator may allow more IN packets to enter the network than what has been contracted.

We used our simulation model to to explore relative performance of the two mechanisms. We simulated a mix of best-effort and AS connections out of a total of 50 connections. In our simulations, AS connections achieved better performance when using token buckets and the discrimination between AS and best-effort connections is much better as seen from the lack of overlap between the two classes. The superiority of the token bucket is due to its permitting transmission of a deterministic burst of IN packets, whereas an average rate estimator probabilistically marks some packets beyond the target rate as IN and some as OUT. A correctly configured token bucket will therefore allow for the natural burstiness of TCP by marking as IN all the packets within a burst, while with an average rate estimator some packets will be marked OUT giving them drop preference.

In addition, the probabilistic marking is problematic for metering a CBR source. A profile meter using an average rate estimator would allow the source to transmit at a sustained rate higher than the contracted one. Token buckets do not permit this. Thus we used token bucket in our simulations. We found that a token bucket depth of 20 Kbytes at the profilers and 25 Kbytes at the policer to keep the remarking rate low (under 3%).

3. Results for Assured Service

3.1 Influence of the round-trip time

TCP performance is well-known to be sensitive to a connection's round-trip time (RTT). The larger the RTT, the more time needed to recover after a packet loss. It is therefore interesting to see if this effect is lessened by the use of AS.

Simulations were run 5 times with 10, 25 and 40 AS connections out of 50, the others being standard best-effort connections. Every AS capable host has a target rate of 1 Mbps (125 Kbytes/s), therefore the portion of the bottleneck link bandwidth allocated to assured traffic amounts to 20% in the case of 10 AS connections, 50% with 25 of them, and 80% with 40 of them. Results are shown in figures 2, 3, and 4. Each point represents the measured rate for one connection, averaged over 5 runs. The trend lines correspond to an exponential regression and are included to give a rough idea of the way achieved rate varies with the RTT.

The dependency of achieved rate on the RTT is also noticeable for AS connections. However, by comparing the spread of the measures in figure 2 and figure 4, we notice that when the AS connections are in large number, they show less deviation (with respect to RTT) than the best-effort connections do in the reverse situation. There is a more critical observation: notice that some connections do not achieve their target rate, while others exceed the target rate. (With average rate estimators, we had some best-effort connections getting more bandwidth than some AS ones.)

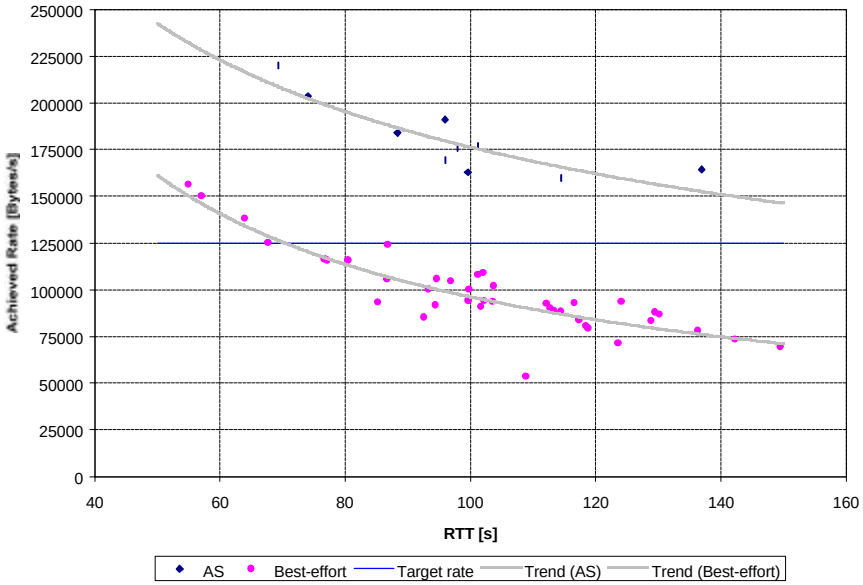


Figure 2: Average rate vs. RTT with 10 AS connections out of 50

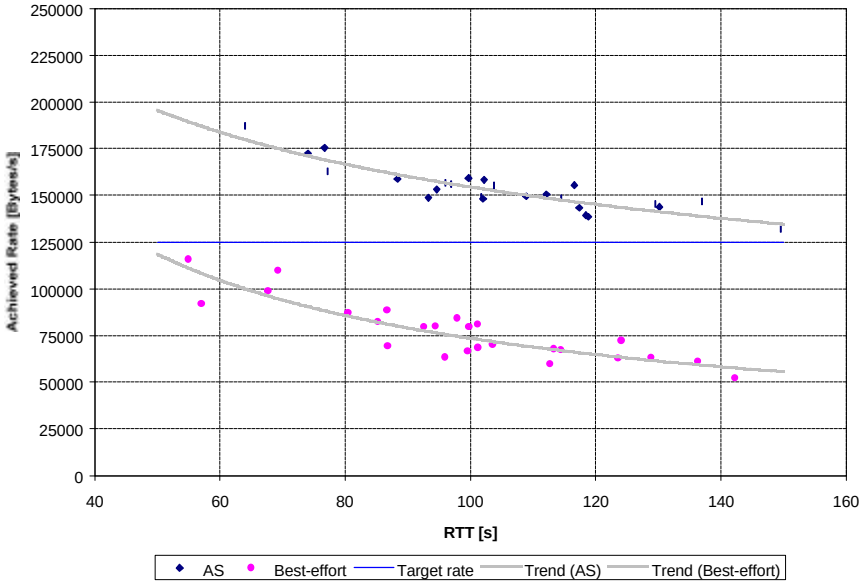


Figure 3: Average rate vs. RTT with 25 AS connections out of 50

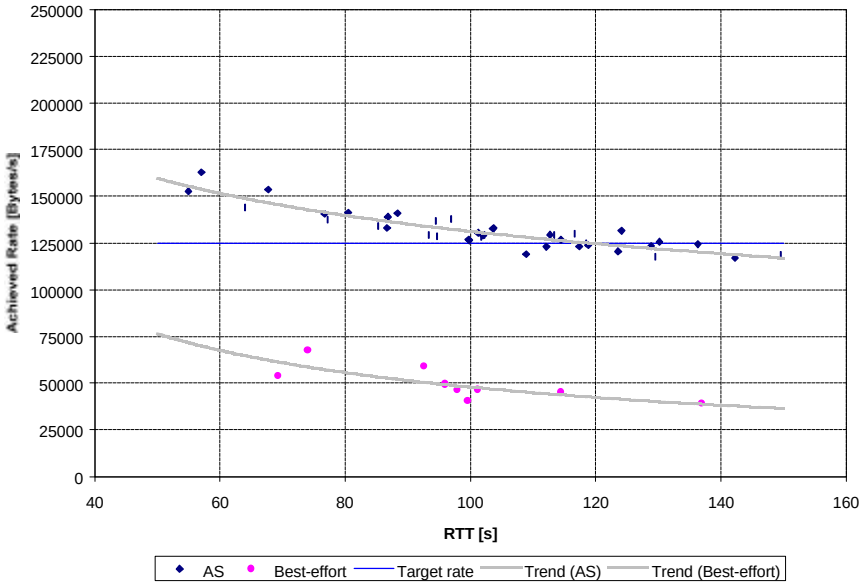


Figure 4: Average rate vs. RTT with 40 AS connections out of 50

Figure 5 shows the difference between the case where there are only best-effort connections and the one where there are only AS connections, the two scenarios being plotted on the same graph.

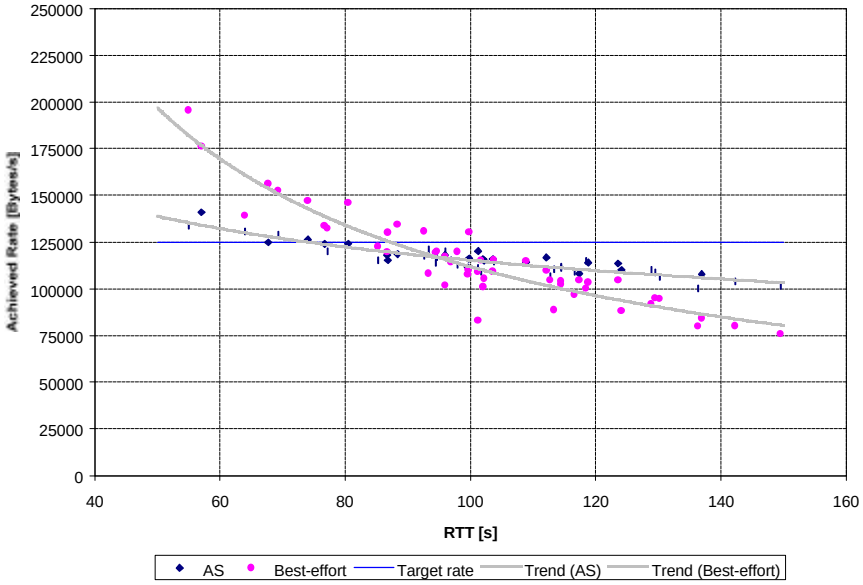


Figure 5: Average rate vs. RTT with best-effort connections only and AS connections only

In the all AS connections case there is less RTT unfairness than in the all best-effort connections case. To understand this phenomenon we need to look at the RIO queue size, which is plotted in figure 6. What happens is that since a connection with a small RTT gets more bandwidth by opportunistically exceeding the target rate and sending OUT packets, many of those packets will be dropped, causing the connection to decrease its sending rate. The more OUT packets a source sends, the higher the probability that one or more of its packets will be dropped within a certain

time interval. That has the effect of mitigating the gain of connections with a smaller RTT. Figure 6 demonstrates how in the best-effort only case the average queue size oscillates substantially, leaving room for small RTT connections to opportunistically send more packets.

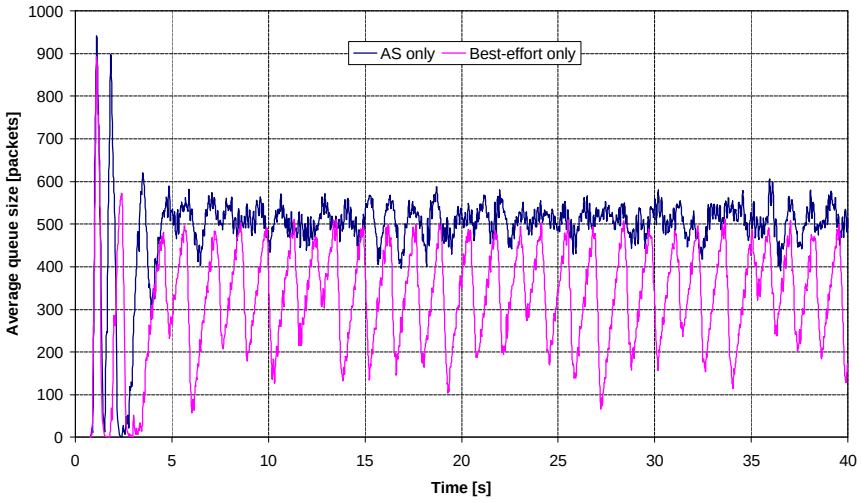


Figure 6: Average queue size for all AS connections and all best-effort connections

Note, however, that this example is artificial since all the available bandwidth is allocated to assured service, and consequently only a few connections reach their target rate. Having said that, the goal of this example was merely to illustrate this point by way of an extreme situation.

If we take a close look at figure 2 and compare it with figure 4, we observe that having a significant amount of assured traffic not only lowers the dependency on the RTT for AS connections, but also for best-effort connections. The cause is the same as above because connections with a small RTT send more packets than those with a large RTT, thus having more chances to undergo a packet drop in a certain time interval.

3.2 Performance against target rates

In this section we explore how well AS TCP connections acheive their target rates. consistent manner. We simulate 40 connections, 20 of which are AS and the remaining 20 are best-effort. An RTT of 100ms was used for all the connections. The bottleneck link bandwidth is 20 Mbps (2.5 Kbytes/s) and we use RIO parameters 347/694/0.05 390/694/0.02. The distribution of target rates among AS capable hosts is as follows:

Host ID	Target rate
0-3	2.0 Mbps
4-7	1.0 Mbps
8-11	0.5 Mbps
12-15	0.2 Mbps
16-19	0.1 Mbps

76% of the total bandwidth is allocated to the AS connections. Other scenarios with different

numbers of connections, with and without best-effort connections, were tried and yielded comparable results. Simulation results were averaged over 10 runs. Table 1 lists the results for each AS connection and two of the best-effort connections (others are comparable thus omitted). If the excess bandwidth is allocated equally among the 40 connections, each would get an additional 2.5% of the excess bandwidth, or 120 Kbps.

Table 1: Performance against target rate and target rate plus equal share

Conn. ID	Measured rate (Mbps)	target rate	target rate plus 2.5% of excess
0	1.32	2.0	2.12
1	1.27	2.0	2.12
2	1.32	2.0	2.12
3	1.31	2.0	2.12
4	0.93	1.0	1.12
5	0.94	1.0	1.12
6	0.95	1.0	1.12
7	0.95	1.0	1.12
8	0.60	0.5	0.62
9	0.58	0.5	0.62
10	0.58	0.5	0.62
11	0.58	0.5	0.62
12	0.38	0.2	0.32
13	0.35	0.2	0.32
14	0.39	0.2	0.32
15	0.36	0.2	0.32
16	0.29	0.1	0.22
17	0.30	0.1	0.22
18	0.30	0.1	0.22
19	0.30	0.1	0.22
20	0.25	0	0.12
21	0.21	0	0.12

In figure 7 the average measured rate is plotted against the target rate (columns 2 and 3 of table 1). The curve labeled "ideal" represents the case where each source achieves its target rate plus an equal share (1/40 or 0.025) of the excess bandwidth, or the last column of table 1.

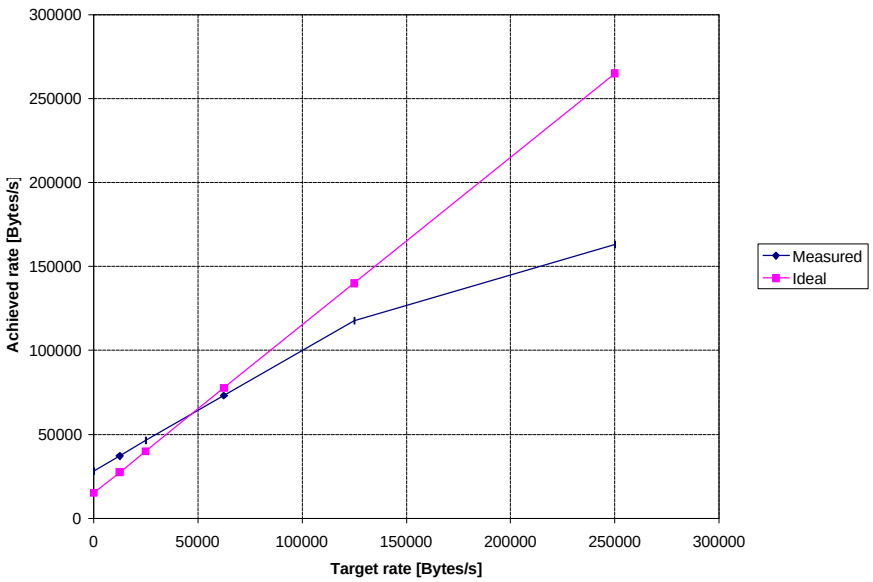


Figure 7: Achieved rate vs. target rate

Only the connections with small target-rates reach or exceed it. Connection 4 was assigned half the target rate of connection 0, but actually gets 70% of it. The only visible discrimination is that as the target rate increases, the achieved rate increases as well, but not proportionally. The explanation is the variation of the congestion window. After the window has closed due to packet losses, the connections with small-target rates return to their former window size quicker than those with bigger target-rates, thus starting sooner to compete for the excess bandwidth. Small target-rate connections make the most of the time during which the large target-rate connections are increasing their window to capture excess bandwidth. This is comparable to the opportunism of small RTT connections at the expense of those with large RTT.

Figures 8 and 9 show the measured rate of connections 0 and 18, which have a target rate of 2.0 Mbps (250 Kbytes/s) and 100 Kbps (12.5 Kbytes/s) respectively. Connection 18 oscillates above its target-rate, whereas connection 0 oscillates around its target-rate, but with an average value below it. Between 12 seconds and 15 seconds both connections are increasing their rate, but connection 0 is well below its target rate, while connection 18 is far above target rate, taking advantage of the bandwidth freed by the former (and some other connections).

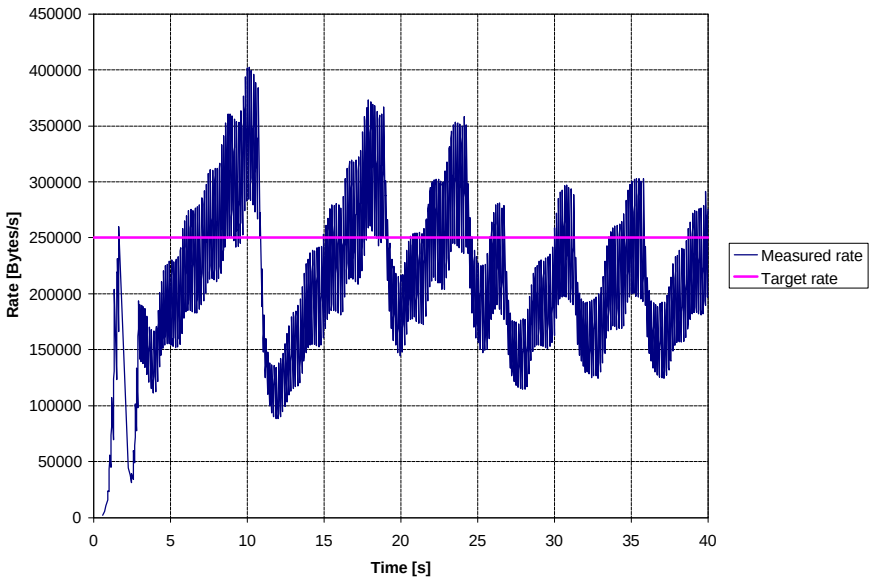


Figure 8: Measured rate for connection #0

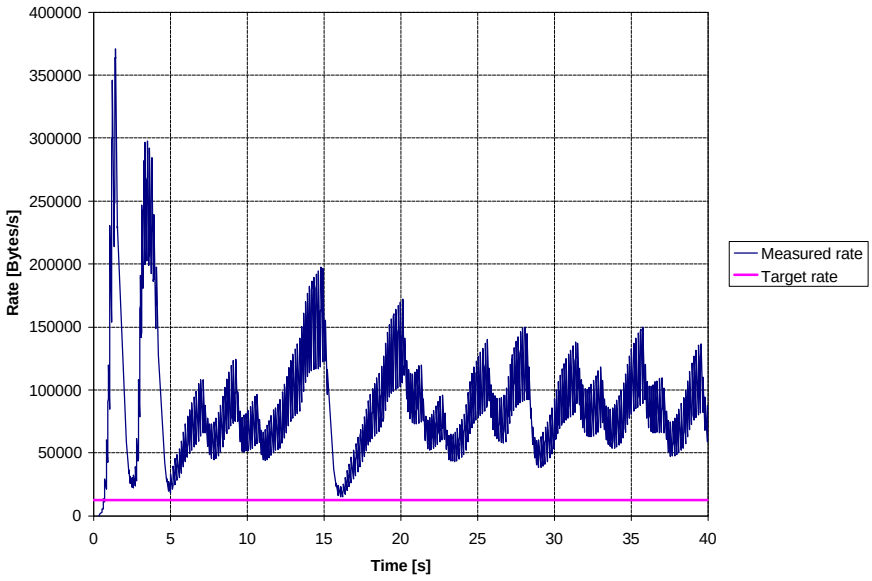


Figure 9: Measured rate for connection #18

3.3 Effect of a non-responsive source

In this section we add a non-responsive source. Many applications that are candidates for quality of service are real-time applications that do not require a reliable transport protocol. They run over UDP and are non-responsive to congestion indication through loss.

Our non-responsive source is a constant bit rate(CBR) source. We simulated 20 point-to-point connections and a bottleneck link of 20 Mbps (2.5 Kbytes/s) and RIO parameters 173/347/0.05

195/347/0.02. The RTT is about 100 ms for every connection. There are 10 AS connections with a target rate of 1 Mbps ($125 \cdot 10^3$ bytes/s) each and 10 best-effort connections including the one associated to the CBR source. Simulations were run 5 times with several values for the CBR output rate. Figure 10 shows the results averaged over five runs. There is one point per connection for each value of the CBR output rate. Figure 11 shows the same results with enhanced resolution around the operating region of the TCP connections.

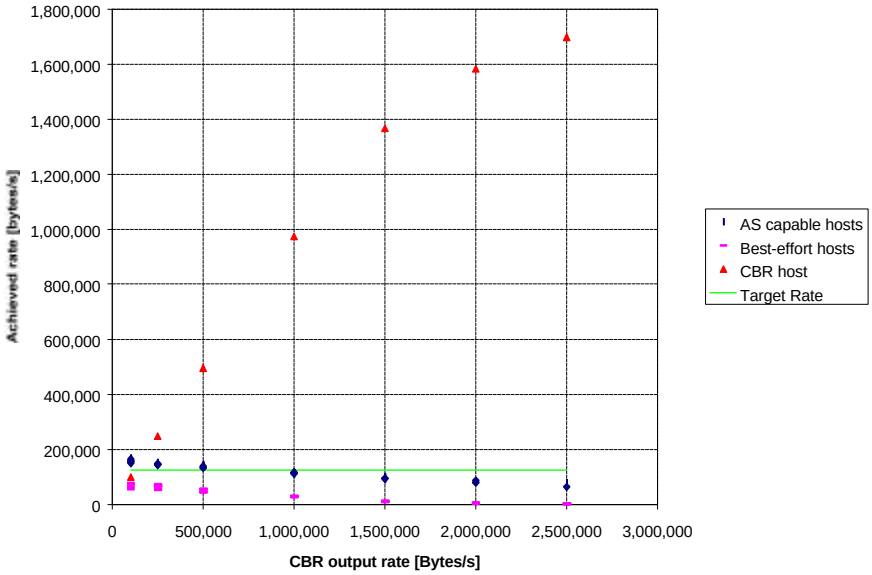


Figure 10: Achieved rate in presence of a non-responsive source (CBR)

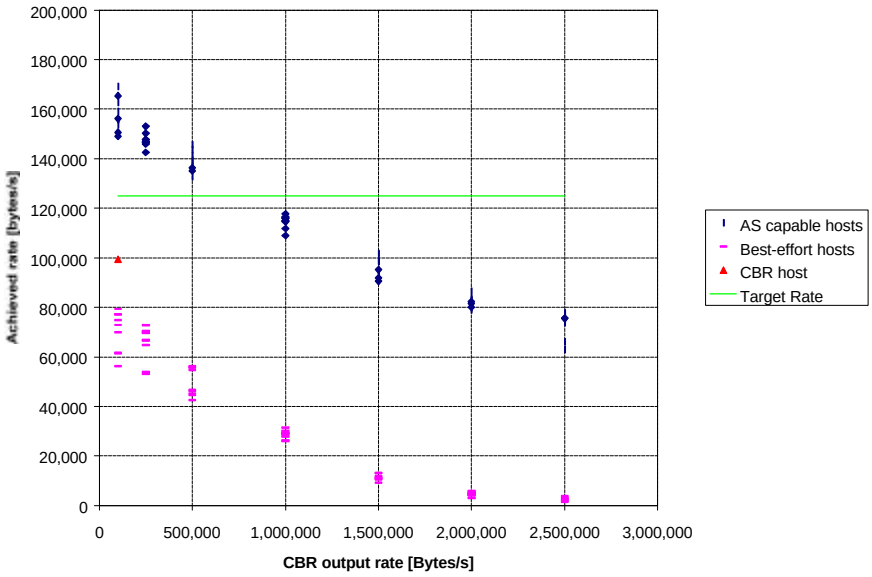


Figure 11: Achieved rate in presence of a non-responsive source (CBR)

As the CBR source increases its sending rate, all the TCP connections get degraded, with the best effort connections being pushed toward starvation. Of course the CBR source experiences

increasing loss, but it still captures a lot of bandwidth. The bandwidth captured by the CBR source has a limit which corresponds to the situation where any OUT packet issued by any AS-enabled TCP connection gets dropped because the maximum threshold for OUT packets of the RIO algorithm is constantly reached. In this situation, AS-enabled TCP connections alternate slow-start phases and congestion avoidance phases, never going through a fast-recovery phase. As a result, almost all the packets leaving the hosts are marked IN and will make their way through the congested link, preventing the CBR source from grabbing more bandwidth.

Next we made the non-responsive source AS-capable. We add a to the output of the CBR source and set its target rate at 4 Mbps (500 Kbytes/sec) for one run and 8 Mbps(1 Mbytes/sec) for another. That leads to a total subscription of 70% or 90% of the bottleneck link bandwidth respectively. The results in terms of achieved rate are almost identical to those shown in figures and . Thus a non-responsive source produces the same impact on other connections, whether it is AS capable or not. The actual difference lies in the number of packets sent by the CBR source which later get dropped. This is represented in figure 12 by the packet drop rate.

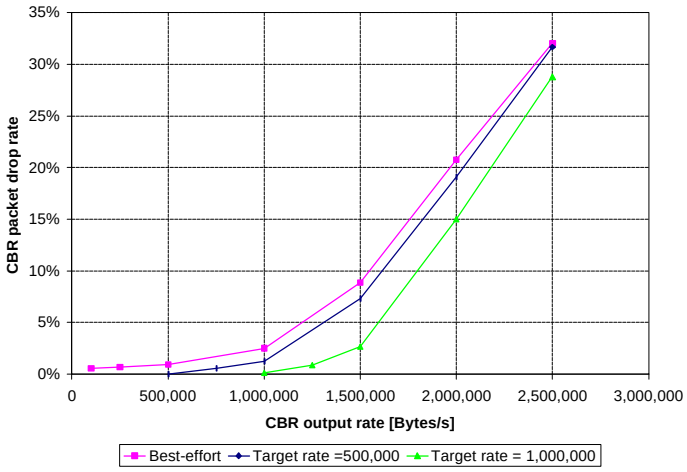


Figure 12: Drop rate vs. output rate for the CBR source

Virtually no packets get dropped when the CBR source transmits at its target rate. When the CBR source has a target rate of 1 Mbyte/sec, 90% of the bottleneck bandwidth is allocated to IN-marked packets. This leads to early dropping a non-negligible number of IN packets, showing that the min_threshold of RIO for IN packets is exceeded. When the CBR target rate is equal to 500 Kbytes/s and 70% of the bottleneck bandwidth is allocated to the assured service, no IN packet gets early dropped.

As long as a CBR source sends packets at a rate close to the contracted one and the allocation of IN-marked packets leaves sufficient headroom on bottleneck links, the packet loss rate will be very low, if not null. On the other hand, a CBR (non-responsive) source, whether AS enabled or not, will grab most of the bandwidth it needs at the expense of TCP connections.

3.4 Effect of AS on best-effort traffic

In this section we explore how IN-marked traffic interacts with best-effort traffic. In the previous

sections we saw how as the number of AS connections increases, best-effort connections' rates are decreased. This is reasonable and acceptable since for AS connections to get their subscribed bandwidth, some other connections must lose bandwidth. Our concern is whether AS and best-effort connections compete for the excess bandwidth on an equal footing.

Each AS connection has a target rate of 1 Mbps (125 Kbytes/s). We did five simulation runs and made histograms of the percentage of connections whose average rate over the run fell into a particular bin. AS connections and best-effort connections are recorded separately. In figure 13, we see that the competition is not fair and that best-effort traffic has the advantage. There are 25 AS connections with a target rate of 125 Kbytes/s, 25 best-effort connections, and a bottleneck link bandwidth of 6.25 Mbytes/s. There is 2.5 Mbytes/s of excess bandwidth. If shared equally among the 50 connections, each should get 50 Kbytes/s. Thus, best-effort connections should get 50 Kbytes/s and AS connections should get this amount added to their target rates, or 175 Kbytes/s. The results show that the AS connections average 155 Kbytes/s and the best-effort connections average 75 Kbytes/s, more than an equal share of the excess.

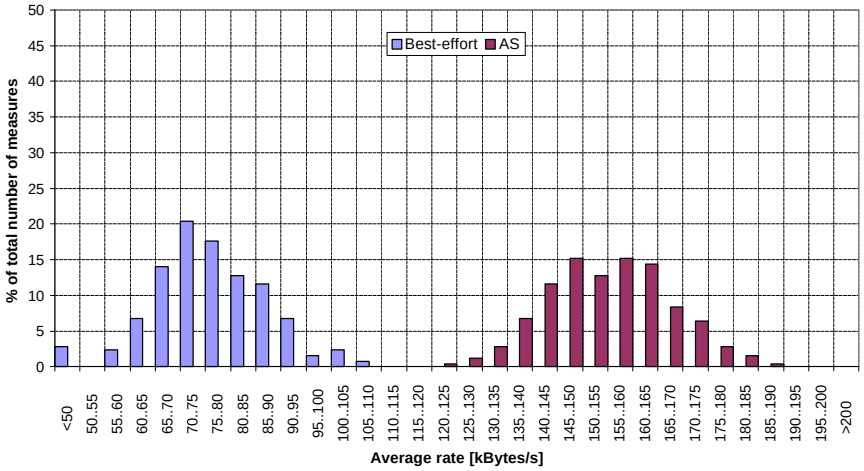


Figure 13: 25 best-effort + 25 AS connections

An AS connection sending OUT packets will experience some drops, just like any best-effort connection, resulting in its sending window closing and the corresponding decrease of the sending rate. Meanwhile, other connections opportunistically increase their rate. Since AS connections send at a higher rate than best-effort connections do, their window size is larger and therefore, once it has closed, requires more time to get to the original size. During this time, best-effort connections can open their windows. In other words, a drop causes the window to close not only with respect to the excess bandwidth, but also with respect to the assured bandwidth, as there is only one single window handling IN and OUT packets as a whole.

3.5 Effects of traffic bursts on AS

The simulations thus far have used a very simple topology. This has been useful, but results from such simulations are, of necessity, optimistic. The Internet is much more complex, composed of many networks merging together. At each merge point traffic gets aggregated, and bursts can accumulate throughout the network. In this section, we discuss results from a more complex

topology, shown in figure 14. It is still relatively simple, but allows us to look at more performance issues.

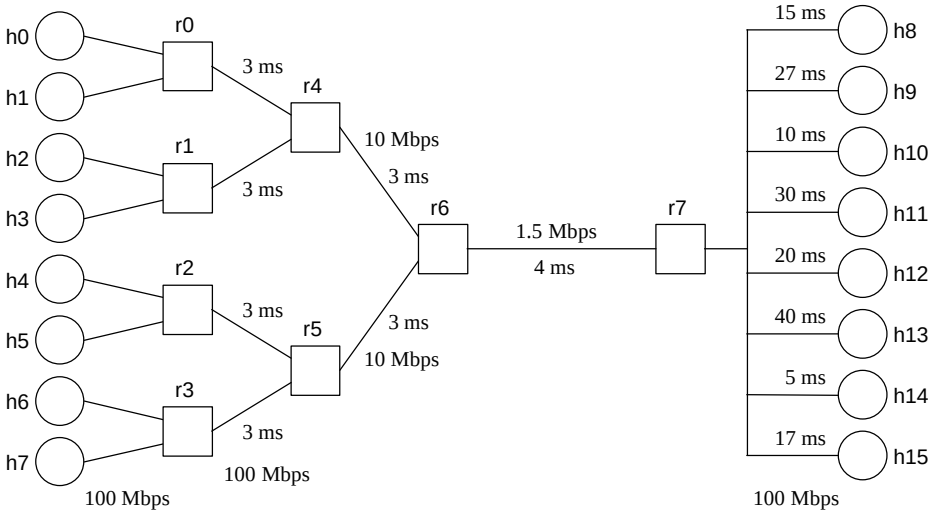


Figure 14: A merging topology

All hosts are AS capable with a target rate of 100 kbps (12.5 Kbytes/s). Aggregate IN-marked traffic is policed at each node. Policers have the following target rates: 200 kbps for nodes r0 to r3, 400 kbps for nodes r4 and r5, and 800 kbps for r6. Token buckets are used for metering, and are 4 packets deep for the profilers, and 6 packets deep for the policers. Packet size is 1500 bytes (including headers). The topology represents a decreasing bandwidth hierarchy ending in a T1 (1.5 Mbps) bottleneck link between r6 and r7. We used a maximum queue size of 24 packets for the T1 link and RIO parameters 4/9/0.05 5/9/0.02. Results are shown in table 2.

Table 2: Results for the merging topology (7.5% of IN packets demoted)

Conn. ID	Measured rate (Kbps)	Overflow IN drops
0	190	3
1	168	1
2	168	2
3	170	0
4	197	2
5	151	0
6	221	0
7	160	1

There are no early drops of IN packets, but some INs are dropped due to queue overflow, and the queue overflows due to bursts of OUT packets. Thus marked traffic is being dropped even when it conforms to its profile. In addition, about 7.5% of IN packets are demoted, becoming candidates for preferential drop should this topology be embedded in a more complex merging hierarchy. It's possible that "push-out" models of drop preference would be better suited to to the implementation of AS than RIO since there would be no overflow drops of INs when there are OUTs in the queue. Still, push-out queues are probably more complex to implement and wouldn't solve the problem of IN packets arriving to a queue full of IN packet bursts (further

study would be required to determine how prevalent this case might be).

The results presented here are optimistic for two main reasons: first, the merging is quite limited and second, we did not employ bursty traffic models like HTTP for either best-effort or IN-marked traffic. Typical Internet paths have on the order of 18 hops and thus would be more complex than the scenario simulated here. The majority of connections and packets in the Internet today are HTTP. However, this scenario reveals some issues related to traffic merging and burstiness accumulation that need further investigation.

3.6 Discussion of results

The big question about AS is the meaning of the "assurance". Is it sufficient to tell a subscriber that a flow using AS will tend to get "better effort" than a best-effort flow? Our results show that it is unlikely that the subscription rate can be viewed as a hard contract. Depending on other traffic and specific topologies, a subscriber may get more or less than the subscribed rate. In addition, it appears that the merging structure of the Internet leads to remarking of IN packets and to their dropping due to queue overflows which may be caused by OUT packets. Further, it seems that compliance may be difficult to check in any case other than a long-lived connection. Although AS may give a "better effort" for web browsing, it will be difficult to quantify this.

If we look at AS in a more general way, it appears it can assure that any packet marked as IN will reach its destination with a higher probability than an unmarked packet as long as all the networks involved in the transmission are adequately provisioned.

4. Summary

In this draft, Assured service was evaluated based on simulations. The results give insight into the workings of the assured service, thus serving as a base for anyone interested in this architecture. Nevertheless the topologies used are quite simple and only long-lived connections were simulated. Since the Internet has a complex topology and a large portion of the traffic consists of short-lived HTTP connections, these simulations are optimistic in that they reflect only the simplest dynamics. In particular, more work needs to be done with traffic merging and burstiness accumulation.

Our main conclusion is AS cannot offer a quantifiable service to TCP traffic. In particular, AS does not allow a strict allocation of bandwidth between several users. Another important conclusion is that, due to the use of a single queue for IN and OUT packets, there is a strong dependency on each other. As a result some IN packets can be dropped in consequence of OUT traffic overflowing the queue. In the real Internet, where the burstiness of best-effort traffic can be significant, this point is very critical and should not be disregarded.

Assured service appears to have potential as a "better effort" service but its characteristics should be further studied before deployment.

5. Security Considerations

There are no security considerations of this draft.

6. References

[ns] UCB, LBNL, VINT Network Simulator - ns
http://www-mash.cs.berkeley.edu/ns/ns.html
[nsdoc] UC Berkeley, LBL, USC/ISI and Xerox PARC, "Ns Notes and Documentation"
http://www-mash.cs.berkeley.edu/ns/nsDoc.ps.gz
[ARCH] D. Black, S. Blake, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", Internet Draft draft-ietf-diffserv-arch-00.txt, May 1998.
[HEADER] K. Nichols, S. Blake, "Definition of the Differentiated Services Field (DS Byte) in the IPv4 and IPv6 Headers", Internet Draft draft-ietf-diffserv-header-02.txt, August 1998.
[SVCALLOC] D. Clark and J. Wroclawski, "An Approach to Service Allocation in the Internet", Internet Draft draft-clark-diff-svc-alloc-00.txt, July 1997.
[EXPALLOC] D. Clark and W. Fang, "Explicit Allocation of Best Effort Delivery Service"
http://diffserv.lcs.mit.edu/Papers/exp-alloc-ddc-wf.ps
[2BIT] K. Nichols, V. Jacobson, and L. Zhang, "A Two-bit Differentiated Services Architecture for the Internet", Internet Draft draft-nichols-diff-svc-arch-00.txt, November 1997,
ftp://ftp.ee.lbl.gov/papers/dsarch.pdf
[RED] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, August 1993.

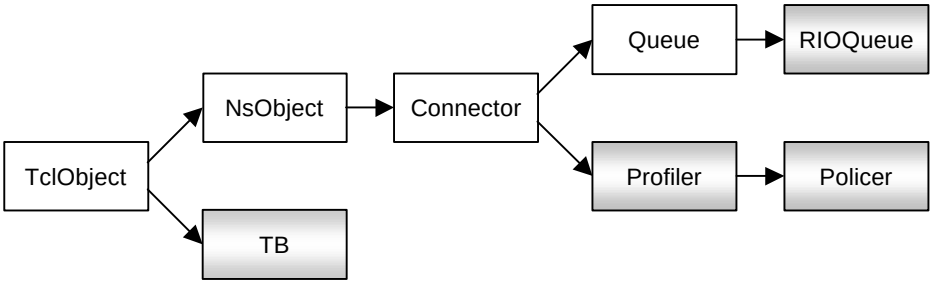
7. Authors' addresses

Juan-Antonio Ibanez
Bay Networks
4401 Great America Parkway, SC01-04
Santa Clara, CA 95052-8185
ibanez@eurecom.fr

Kathleen Nichols
Bay Networks
knichols@baynetworks.com

Appendix: Implementation of differentiated services in ns-2

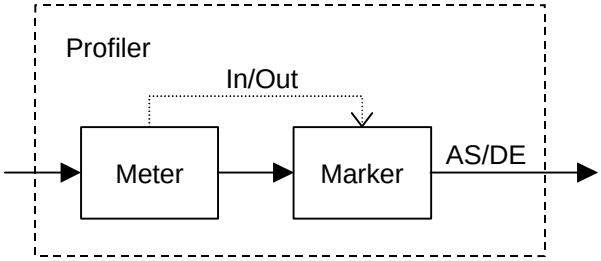
Simulations have been performed with ns-2 (version ns2.1b1). Some modifications or additions were required to implement the differentiated services. New modules are represented hereafter with the inheritance tree and are identified by gray boxes:



The core of these modules has been implemented in C++, and some OTcl code was required to implement their interface with the front-end interpreter. TclObject and NsObject are the two base classes from which any object is derived; they provide all the basic functions allowing objects to interact with each other and with the front-end interpreter, such as mirroring, variable tracing, etc.

Besides these new classes I have also made some minor modifications to standard modules in order to accommodate some specific requirements. The class TB implements a token bucket which is used by the Profiler.

The profiler, also called a profile meter, is attached directly to a packet source and measures the rate at which it is sending data, either by mean of an average rate estimator, or a token bucket. If the measured rate is below the target rate, it marks packets IN, but when the target rate is exceeded they are left unmarked.



A profiler is usually attached to a single source of traffic, but it can also process an aggregate, at the egress of a domain. For this reason the profiler's interface is written in such way that it can be attached either to an agent, or to a link. The second option allows the policing of an aggregate by attaching the profiler to the link exiting the domain. Two different meters were implemented for the profiler: an average rate estimator, and a token bucket. In [EXPALLOC] only the average rate estimator is used.

The average rate estimator is the time sliding window algorithm described in [EXPALLOC].

Initialization:

win_length = a constant
avg_rate = 0
last_arrival = 0

`Each time a packet is received:`

$$\text{avg\_rate} = \frac{\text{avg\_rate} \cdot \text{win\_length} + \text{pkt\_size}}{\text{now - last\_arrival} + \text{win\_length}}$$

last_arrival = now

The window length determines the worth of past history the algorithm remembers, or in other words the weight of the past against the present. After the rate estimation the following tagging algorithm is executed.

```
if  avg_rate <= target_rate
        set DS field to IN
else
        with probability Pout = (avg_rate – target_rate) / target_rate
                set DS field to OUT
        else
                set DS field to IN
```

This probabilistic marking allows for the well-known oscillations of a TCP connection in pursuit of its maximum rate. As a matter of fact, for a TCP connection to achieve a given average rate, the connection should be allowed to oscillate around that value. The drawback is that this mechanism will permit other types of connections, like a CBR-like connection, to get in average more than the target rate.

The policer is attached to an intermediate node representing a router and monitors the aggregate of traffic which enters (or leaves) the node. If the rate of the aggregate is below its target rate, packets are forwarded unchanged, but if the target rate is exceeded, the packets arriving with marked IN get remarked (demoted) to OUT before being forwarded.