대역폭 보장 서비스를 위한 개선된 RIO 알고리듬

김 효 곤†

요 약

이 논문에서 우리는 인터넷 차별화 서비스(Differentiated Services Diff-Serv) 모델 중 대역폭 보장 서비스(Assured Service AS)의 성능을 개선하기 위한 두 가지 방법을 탐구한다. AS의 가장 큰 문제는 프로파일이 큰 플로우가 다수의 작은 프로파일을 갖는 플로우들과 경쟁할 경우 큰 프로파일 플로우의 대역폭을 보장해주지 못한다는 것이다. 그런데 이 문제는 패킷 손실로 야기된 TCP 윈도우 반감을 복구하는 시간이 프로파일의 크기에 따라 다른 데서 시작한다. 현재까지 제안된 해결책들은 따라서 TCP의 동작을 고치는 데 모아지고 있는데, 불행히도 이 방법들은 이미 광범위하게 배치된 인터넷의 기반 구조의 대대적인 변화를 필요로 한다는 점에서 비현실적이다. 이러한 난점을 피하기 위해 우리는 아직까지 배치되지 않은 Diff-Serv 메커니즘들 속에서 현실적인 해결책, 특히AS를 위한 큐 관리 방법인 RIO(RED with IN and OUT)의 역할을 조사한다. 구체적으로, 우리는 대역폭이 만족되지 않는 AS의 문제를 RIO가 악화시키고 있다는 점을 발견한다. 이 발견에 근거하여, RIO의 문제점을 극복하여 AS의 성능을 개선하는 새로운 큐 관리방법인 RI+O를 제안한다. 시뮬레이션을 통하여 우리는 RI+O가 AS의 성능을 상당히 개선시킴을 확인한다.

A modified RIO for Improving Assured Service Performance

Hyo-Gon Kim†

ABSTRACT

In this paper, we explore two ways to improve the bandwidth assurance performance of Assured Service(AS). It is well known that AS fails to meet the bandwidth assurance target for large-profile TCP flows competing with many small-profile flows. This problem originates from the difference between the TCP congestion window recovery times of different profile size flows after the back-offs induced by packet drops. Thus currently proposed solutions to this problem naturally focus on modifying TCP's behavior to counter the unfairness in the TCP dynamics. Unfortunately, these proposals lack practicability in terms of the required changes in the incumbent Internet infrastructure. Admitting this difficulty, we instead look to not yet deployed Diff-Serv mechanisms for practical solutions. In particular, we investigate the role of RIO, RED with IN(in-profile) and OUT(out-profile), queue management scheme in the assurance failure for AS. Specifically, we identify the inadequacy of RIO that aggravates the bandwidth assurance failure. Then we propose a modified scheme, called RI+O, to improve the RIO's performance for AS. Our proposed scheme significantly alleviates the the bandwidth assurance failure problem by separately controlling the out-of-profile packet queue length. Through extensive simulations we demonstrate that RI+O extends the regime where AS consistently provides the bandwidth assurance.

† 정 회 원 : 아주대학교 정보통신대학 교수
논문접수 : 1999년 10월 13일, 심사완료 : 1999년 11월 6일

1. Introduction

The current Internet delivers only the best-effort(BE) service. As Internet grows rapidly, there is a demand to provide a variety of performance guarantees. Recently, the IETF Differentiated Services(Diff-Serv) working group proposed the Diff-Serv architecture, relatively simple and coarse methods of providing differentiated classes of service for Internet traffic, to support various types of applications and specific business requirements. The Diff-Serv approach to providing Quality of Service(QoS) in networks employs a small, well-defined set of building blocks from which a variety of services may be built. A small bit-pattern in each packet, in the IPv4 TOS octet or the IPv6 Traffic Class octet, is used to mark a packet to receive a particular forwarding treatment, or per-hop behavior(PHB), at each network node. The working group has standardized a common layout to be used for both octets, called the 'DS field'. RFC 2474[1] and RFC 2475[2] define the architecture, and the general use of bits within the DS field. The working group also standardized a small number of specific PHBs, the Expedited Forwarding(EF) PHB(RFC 2598)[3] and the Assured Forwarding(AF) PHB group (RFC 2597)[4]. The EF PHB can be used to build low loss, low latency, low jitter, assured bandwidth and end-to-end service through DS domains. Such a service appears to the endpoints like a point-to-point connection or a 'virtual leased line'. This service originally proposed by Nichols and Jacobson[5], described as Premium Service. The EF PHB can be implemented by a mechanism that allows unlimited preemption of other traffic(e.g., a priority queue), and some means to limit the damage that EF traffic could inflict on other traffic(e.g., a token bucket rate limiter). The concept, expected behavior, and the implementation scheme of EF PHB are described in more detail in [3], but it is beyond the scope of this paper.

The AF PHB group is a means for a provider DS domain to offer different levels of forwarding assurances for IP packets received from a customer DS domain. The customer can expect an assurance that the IP packets sent are forwarded with high probability as long as the traffic does not exceed the subscribed information rate(service profile). The packets exceeding the subscribed profile rate are also forwarded but not as high probability as the traffic that is within the profile rate. This service was originally proposed by Clark and Fang[6], and it is referred to as Assured Service(AS) in this paper.

The AF PHB group provides forwarding of IP packets in four independent AF classes. Within each AF class, an IP packet is assigned one of three different levels of drop precedence. And it is still controversial if three drop precedences will indeed yield much better differentiation[7]. But mainly for the reason of simplicity, we will limit ourselves to two levels of drop precedences.

AS is a long-envisioned end-to-end bandwidth assurance service that can be deployed in the Diff-Serv Internet[6]. Constructed from the AF PHB(PHB) group[4], it is designed to provide a statistical bandwidth assurance to a Diff-Serv subscriber. Unfortunately, recent findings show that it fails to provide a consistent assurance. Large profile TCP flows competing with small profile flows can fail to fully utilize their assured bandwidth[12]. This inconsistency originates from the exponential congestion window back-off behavior in TCP congestion control. Namely, the time that takes for a large profile flow to recover from its back-off is larger than that of a smaller profile flow. While the large profile flow grows its window inside its profile envelope towards the profile rate, the smaller flow window can quickly grow out of its envelope and exploit the bandwidth that the large profile flow cannot use.

Thus, currently proposed approaches naturally focus on modifying the TCP behavior accordingly. However, these approaches lack practicability in terms of the required changes in the incumbent Internet infrastructure that already has a large installation base. Admitting this difficulty, in this paper we instead

look to not yet deployed Diff-Serv mechanisms for practical solutions. In particular, we investigate the role of RIO(RED with In and Out) queue management scheme. A major contribution of this paper is that we identify and rectify RIO's conceptual inadequacy that contributes to the assurance failure. Our investigation leads us to a modified version of RIO, which we call RI+O in this paper. Through extensive simulations, we show that the modification significantly alleviates RIO's weakness to the problem. Thereby RI+O effectively extends the regime where AS consistently provides the bandwidth assurance, in an easily deployable manner.

The rest of this paper is organized as follows. Section 2 analyzes the bandwidth assurance failure. In addition to the TCP window dynamics briefly discussed above, we analyze the RIO's contribution to the problem. Section 3 briefly surveys the currently proposed solutions for the problem, and evaluate their practicability in terms of the existing infrastructure changes required by each solution. In section 4, we design RI+O(RED with IN, IN+OUT, and OUT), and compare its bandwidth assurance performance with that of RIO using a wide range of simulation parameter values. We also analyze the performance improvement that RI+O provides. Section 5 concludes the paper.

2. Analysis of the bandwidth assurance failure

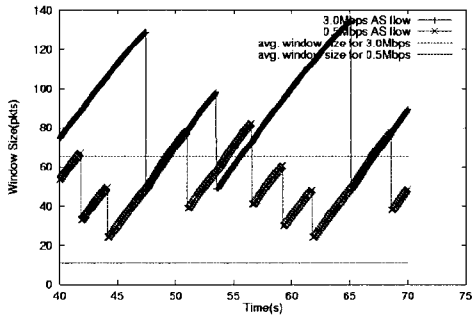
In this section, we consider the causes of the assurance failure in AS. Each AS flow must be able to send packets at least at its contracted profile rate and if there is some unused bandwidth, it must be able to send some more packets although these packets can be dropped with higher probability. But as briefly mentioned above, it was found that the flows with large profile rates cannot reach their contracted profile rates, while the flows with small profile rates and the BE flows exceed their profile rates and even the fair share of the bandwidth.

In this section, we briefly reiterate the existing explanation of the cause appearing in the literatures [12-14]. They all point to the TCP congestion control dynamics for the cause, and indeed there is inherent unfairness towards large profile flows in the dynamics. In addition, we shed a new light on the role of RIO in the failure. We claim that RIO also discriminates against the large profile flows due to its inherent inadequacy. We corroborate this claim by simulations.

2.1 TCP congestion window dynamics

TCP congestion control algorithm exponentially backs off the sender congestion window size whenever a packet is dropped in the network. After the back-off, the TCP sender begins to linearly grow its window size. This linear-increase exponential-decrease control is essential for network stability. For a TCP flow that suffers a packet drop at window size W takes at least W/2 round-trip times(RTTs) to recover W. Since the loss-inducing window size W is proportional to the profile bandwidth, a smaller profile size means a smaller window recovery time. It means that the smaller profile flow more frequently reaches the window size imposed by the profile rate, gaining more chances to exploit excess bandwidth being unused by the larger profile flow growing its window size inside the profile envelope.

(Fig. 1) shows the congestion window size fluctuation of two TCP flows, which contrasted with their



(Fig. 1) Window fluctuation of two different profile flows.

profile rates(translated into the window sizes). The profile sizes for these flows are 3Mbps and 0.5Mbps, respectively. The link capacity shared by these flows is 5Mbps. As easily observed, the smaller profile flow window is always outside the profile envelope, while the larer profile flow alternates in and out of its envelope.

2.2 RIO

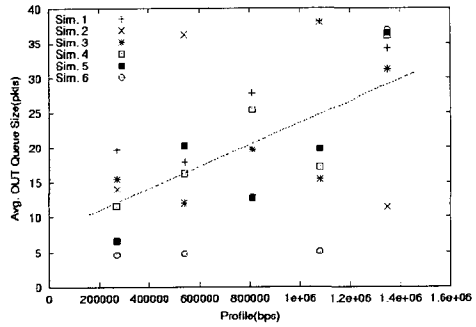
An AF implementation must minimize long-term congestion within each class, while allowing short-term congestion resulting from bursts. So, the AF PHB group RFC[4] suggests that the Diff-Serv router use an active queue management algorithm like Random Early Drop(RED)[11, 16]. But typically, an algorithm called RIO is used. RIO uses the twin RED algorithms for dropping packets, one for INs and the other for OUTs. Upon each packet arrival at the router, the router checks whether the packet is marked IN or OUT. If it is an IN packet, the router calculates avg_{in}, the average queue size for the IN packets; if it is an OUT packet, the router calculates avg_{total}, the average total queue size for both IN and OUT arriving packets. The probability of dropping an IN packet depends on avg_{in}, and the probability of dropping an OUT packet depends on avg_{total}. There are three configurable parameters for each of the twin algorithms. We will call these sets of parameters as IN parameters(min_{in}/max_{in}/P_{maxin}) and IN+OUT parameters(min_{total}/max_{total}/$P_{maxtotal}$), respectively.

In its current form, RIO's avg_{in} tracks only the number of IN packets in the queue where avg_{total} tracks the total(i.e., IN plus OUT) number of packets. avg_{in} is compared with the thresholds for IN packets and avg_{total} is compared with the thresholds for OUT packets(see [11] for the detailed explanation of the RED queue management algorithm). Note that avg_{total}, not avg_{out}, is used to determine the OUT drop probability since the use of avg_{out} would not cover the case where the total queue is growing due to arriving IN packets[6]. However, we observe that determining the OUT drop probability entirely depending on avg_{total} also has a problematic aspect. Below, we explain this problematic aspect using simulation data.

In our simulation, the link capacity is set to 10Mbps, and RIO parameters are set to 30/90/0.05 and 60/180/0.02 for OUT and IN, respectively. Other parameters are set according to the experimental setting in[12]. (Fig. 2) shows $avg_{out} = avg_{total} - avg_{in}$ for 5 different sized AS flows observed as their IN packets are enqueued. Their profile sizes are 270Kbps, 540Kbps, 810Kbps, 1080Kbps and 1350Kbps. Note that since the observer flow is enqueing its IN packets when avg_{in} value is recorded, the avg_{out} queue length reflects the OUT queue length created by *other* flows. If low-pass filtering is heavily used, the "memory" left in the average queue lengths lasts longer, and may blur this distinction. However, to avoid this problem, we use a large w_q for RIO in this experiment. [1]The straight line is obtained through the mean square error(MSE) fitting. First note in the figure that avg_{out} available to the flows other than the flow in its IN burst, i.e., the observer flow, shows a positive correlation to the profile size. Namely, larger profile flows see a significantly reduced avg_{in}[increased avg_{out} (of other flows)] when their IN bursts are on. On the other hand, smaller profile flows see a significantly inflated avg_{in} [reduced avg_{out} for other flows]. It indicates that when a larger profile flow is in its IN burst, i.e., inside its profile envelope, $avg_{out} = avg_{total} - avg_{in}$ is large due to other(smaller profile) flows utilizing excess bandwidth. When it finishes the IN burst and starts to generate OUT packets, it sees a smaller room for the packets. But when smaller profile flows are in their IN bursts, avg_{in} is larger [avg_{out} is smaller], indicating that the larger profile flows are seeing a smaller room for the OUT packets. Note that the avg_{total} remains relatively fixed in RIO. On the surface, this symptom merely confirms the unfairness in TCP window recovery

2) If low-pass filtering is heavily used, the "memory" left in the average queue lengths lasts longer, and may blur this distinction. However, to avoid this problem, we use a large w_q for RIO in this experiment.

discussed previously.



(Fig. 2) avgout created by other flows sen by a flow in IN bursts.

It is surprising, on a second thought, that RIO does not attempt to compensate for the unfairness even though it has sufficient information. In other words, the current RIO implementation does not use the negative correlation information to discriminate against smaller profile flows. This is simply because RIO does not have a way to control the IN/OUT *composition* of the queue. While the total queue avg_{total} is maintained within a certain range between min_{total} and max_{total}, the fraction of avg_{out} can freely fluctuate without being subject to any regulation.

One can argue that since RIO controls avg_{in} and avg_{total}, as a fallout it automatically controls $avg_{out} = avg_{total} - avg_{in}$. However, this argument ignores the fact that IN parameters do not take effect until the link in question reaches its 'saturation point', i.e., the 100% subscription level[18]. One assumption here is $max_{out} <= min_{in}$. Otherwise, IN packets can be dropped when some OUT packets are accepted, which is clearly unacceptable. Overlaping the probabilitic drop region for OUT packets with that for IN packets(i.e. $min_{in} < max_{out}$) allows $1-P_{out} > 0$ and $Pin > 0$ when $min_{in} < avg_{in} <= avg_{total} < max_{out}$. So, if $max_{out} <= min_{in}$, before any IN packet is dropped all OUT packets are dropped. But the IN packet drop only occurs in under-engineered regime, i.e. when the link capacity is smaller than the subscription bandwidth. Therefore, in the well-engineered regime only OUT packet drops occur and they dictate the queue dynamics. Note that AS hinges on the assumption that the network is "adequately provisioned", meaning that the normal operation region for AS is the regime strictly under 100% subscription. Therefore, in this well-engineered regime, avg_{in} is not separately controlled at all. The entire queue dynamics is dictated by the OUT drop probability only. Therefore, avg_{out} and avg_{in}, the constituents of avg_{total} are not controlled by RIO. Unless RIO is modified to control avg_{out} even in the well-engineered regime, RIO remains an accomplice of the assurance failure problem. In Section 5, we propose a modification to remedy the inadequacy of RIO.

3. Existing proposals

The assurance failure problem is significant because if the rate guarantee of AS is severely violated, the Diff-Serv cannot be deployed as the IETF plans. Since the well understood cause of the assurance problem is the difference between the TCP congestion window recovery times of different profile size flows drops, currently proposed solutions naturally focus on modifying TCP's behavior to counter the unfairness in the TCP dynamics. Unfortunately, these proposals lack practicability in terms of the required changes in the incumbent Internet infrastructure that has been already widely installed.

Feng et al. [14] and Yeom/Reddy [13] separately proposed two-window TCP schemes which modify the TCP congestion avoidance algorithm. The TCP congestion window is differently adjusted depending on the sender-detected marking of the dropped packet. The congestion window backs off into the profile envelope only in case of "genuine" congestion, i.e., IN packet drop. Otherwise, the window back-off is done down to the profile rate but not below. At least this solution will shield large profile flows for their profile rates. And eventually all TCP senders need to be replaced by the mark-sensitive version

for AS to provide stable bandwidth assurance in all regimes. It requires the TCP sender code modification, however, which implies that only the modified TCP senders are protected. With the large installation base of TCP, this gradual replacement constraint implies that we need a fast solution for the AS customers that run old version of TCPs. But the bigger problem of this solution is that it may require an additional signaling between the marker and the TCP sender. For the TCP sender to know the marking of the dropped packet, the marker should provide this information. Note that the TCP sender and the marker can be physically apart. For instance, the marker can run from the ingress edge router in the Internet service provider(ISP) premise, where the TCP can be in the subscriber premise. Moreover, if there are a multitude of TCP connections, the signaling between the TCP senders and the marker can become fairly complex. Finally, the marker also needs to examine the transport layer information.

A more feasible solution is to exploit ECN(Explicit Congestion Notification). Since an actual packet drop makes the detection of the marking of the dropped packet prohibitively costly for the TCP sender, we can force the Diff-Serv routers to ECN mark the packet instead of dropping it. Then the TCP receiver can return the ECN and the original packet marking to the TCP sender. But as pointed out earlier, this requires all the Diff-Serv routers on the path to be ECN-compliant, as well as the both TCP parties.

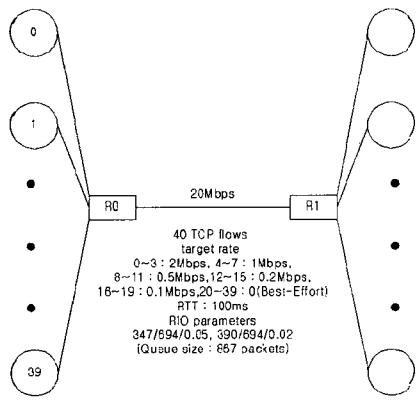
Yeom and Reddy explore the idea of limiting the generation of OUT packets by the TCP sender. The idea here again is that the window back-off should be caused only by IN packet drop. When the sender gets the information that one of its packets is marked OUT from the marker, it reduces the window by a packet. This solution has a couple of problematic aspects. First, as discussed above, it requires an additional signaling between the marker and the TCP sender, not to mention the TCP code modification. If the marker is detached from the TCP sender, the signaling can cause many complications described above. Second, it does not completely eliminate the generation of OUT packets, and the infrequently generated OUT packets can still cause the window back-off into the profile envelope. Last but not least, it contradicts the very spirit of AS model. In AS, a subscriber must obtain not only its profile rate, but also a share of the excess bandwidth. Also note that if a TCP sender backs off as soon as it generates an OUT packet, the effective throughput is limited to 66% of its profile rate[6]. It implies that a different type of marker such as TSW[6] should be used for TCP flows to compensate for the underutilization. However, when the marker should mark multiple TCP flows, the magnitude of underutilization can change due to statistical multiplexing. It will be very difficult, if not impossible, to calculate the changed underutilization level.

Yeom and Reddy also explore an idea called Inverse-rate drop policy. It requires that every packet be stamped with the profile rate. Using the RED drop probability as a function of the rate, the router calculates the appropriate drop probability for each packet carrying the rate information. The drop probability is set lower for higher profile rates. There are visible drawbacks in this solution. First, it requires complex computation for *each* packet on the router's part. This conflicts with the Diff-Serv approach which attempts to boost router throughput by decimating the per-packet processing overhead. Second, it obviously requires an additional IP header field to carry the rate information. However, the IP header does not have more room for such potentially complex information.

Another approach is to use the three drop precedences defined for AF PHB group[4]. In this approach, the marker keeps track of the long-term sending rate of the sender. When a packet has to be marked as OUT, if the long-term sending rate is greater than the profile rate, the packet is marked as OUT-OUT, otherwise OUT-IN. This solution can be used to limit the number of OUT packets of

smaller profile flows, by discriminating against OUT-OUT packets from the smaller profile flows for OUT-IN packets from the larger profile flows. However, this solution still allows the competition between OUT-IN packets and between OUT-OUT packets on an equal footing, regardless of profile sizes. Therefore, it is found to be only partially effective[7, 13].

In summary, existing proposals to solve the assurance failure problem either requires significant changes in the existing infrastructure or yields a marginal improvement. In Section 4 and 5, therefore, we seek to find solutions that do not require significant changes in the already deployed infrastructure. In particular, we try modifying only the newly implemented components of the Diff-Serv architecture, such as the token bucket marker and the RIO mechanism even if the aforementioned problems are not resolved for all conceivable conditions. And we aim to identify the conditions where AS bandwidth assurances are met in a stable manner.



(Fig. 3) Simulation topology.

4. Modifying RIO

In Section 2, we demonstrated that RIO cannot separately control avg_{in} and avg_{out} components of avg_{total}. It is because the IN parameters(min_{in}/max_{in}/$Pmax_{in}$) do not take effect in the well-engineered regime, and the IN+OUT parameters(min_{total}/max_{total}/$Pmax_{total}$) thus entirely dictate the queue dynamics. We also demonstrated that the assurance failure problem manifests itself as a negative correlation between avg_{in} and the profile size. In other words, larger profile flows claim smaller room for OUT packets, effectively shortening OUT bursts. Thus not only the larger profile flows suffer from their longer window recovery period after back-offs, but the duration from the start of an OUT burst till the first OUT drop is also shorter. In essence, the larger profile flows face two kinds of unfairness. The former type of unfairness can only be cured by modifying TCP behavior, but the latter type of unfairness is curable by modifying RIO to actively control avg_{in}(or avg_{out}) in the well-engineered regime (i.e., under 100% subscription level).

```
for each arriving packet P
        calculate avg_in
        Calculate avg_total
        avg_out = avg_total - avg_in
if P is an IN packet
        if min_in < avg_in < max_in
                Calculate p_in
                Drop P with probability p_in
        else if max_in < avg_in
                drop P
else if P is an OUT packet
        if min_total < avg_total < max_total
                Calculate p_total
                drop P with probability p_total
else if max_total < avg_total
        Drop P
        If P has not been dropped
                if min_out < avg_out < max_out
                        calculate p_out
                        drop P with probability p_out
                else if max_out < avg_out
                        drop P
```

(Fig. 4) RI+O algorithm

In this paper, we force RIO to look at avg_{out} in isolation, and apply a separate set of parameters to it. We call this modification RI+O. RI+O is a straight-

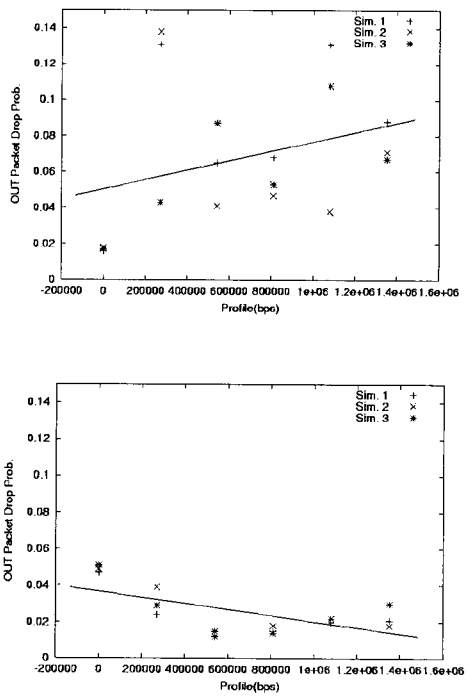
<Table I> Performance comparison of RIO and RI+O

RI+O	prof(bps)	thruput(bps)	markin(pkts)	markout(pkts)	dropout(pkts)	OUT drop rate
	270K	570955	27563	35193	852	0.024
	540K	888546	55999	41001	616	0.015
	810K	1067381	84353	31779	468	0.015
	1080K	1220620	109991	22707	461	0.020
	1350K	1425098	136643	18135	386	0.021
	BE(avg)	236753	0	26936	1270	0.047
RIO	profile(bps)	thruput(bps)	markin(pkts)	markout(pkts)	dropout(pkts)	OUT drop rate
	270K	292272	27567	4771	626	0.131
	540K	563199	55307	6196	401	0.064
	810K	782717	80548	4644	315	0.067
	1080K	961649	102115	2433	318	0.130
	1350K	1221734	130057	2593	227	0.087
	BE(avg)	308349	0	33971	541.8	0.015

forward extension of RIO. In RI+O, whenever a packet arrives, the router calculates avg_{in} and avg_{total}, and with a single subtraction, obtains avg_{out} assuming that the same weighted averaging constant w_q is used for both avg_{in} and avg_{total}. In RI+O, three sets of parameters are used : IN, IN+OUT, and OUT parameters($min_{out}/max_{out}/P_{maxout}$). When an OUT packet arrives, avg_{total} is compared with the IN+OUT threshold and then if the packet is not dropped, avg_{out} is compared with the OUT threshold. (Fig. 4) shows the RI+O algorithm. The emboldened part is the new drop logic added to the original RIO algorithm.

We evaluate the performance of RI+O using the same simulation topology as in (Fig. 3). Here we run 25 TCP flows, 5 of which are AS flows and the remaining 20 are BE flows. An RTT of 100ms is used for all flows. The bottleneck link bandwidth is 10Mbps and RIO parameters are 60/180/0.02 for IN, 30/90/0.05 for IN+OUT. The profile rates of AS flows are set to (flowid+1) 270Kbps. The total subscription level is 40.5%. For each flow, a token bucket marker is attached, and the token bucket size is set to (flowid+1) 320KB. For RI+O, we use the same configuration as RIO, but add one more parameter values for OUT packets. We use 10/30/0.1 as $min_{out}/max_{out}/P_{maxout}$.

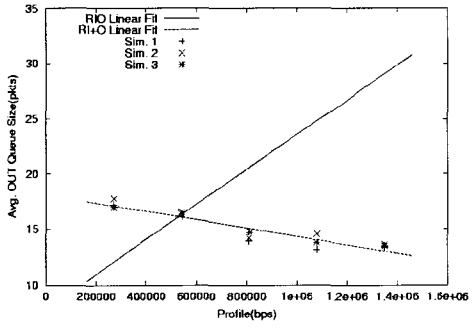
<Table I> compares the performance of RIO and RI+O, respectively, in one simulation instance. The first 5 rows show the performance of AS flows, while the last row in each scheme represents the values averaged over 20 BE flows. We can easily observe that the throughput in RI+O is significantly improved especially for the larger profile flows. And the throughput improvement is obtained by the elongated OUT burst size. The 'markout' is the number of



(Fig. 5) OUT drop probabilities of RIO (left) and RI+O (right)

packets marked OUT at the traffic conditioner. We observe that RI+O significantly increases this number for AS flows at the cost of the BE number. And the OUT packet drop probabilities show that RI+O significantly reduces them. On the other hand, the average OUT packet drop probability of BE flows is much larger in RI+O. This result also bears out our observation made in Section 2. Namely, by controlling avgout explicitly we can achieve an evenly distributed OUT burst size. (Fig. 5) compares the OUT drop probabilities with RIO and RI+O, respectively. As can be seen, the BE drop probability with RIO is much smaller. On the other hand, it becomes larger with RI+O, and consequently the AS flows show smaller drop probabilities.

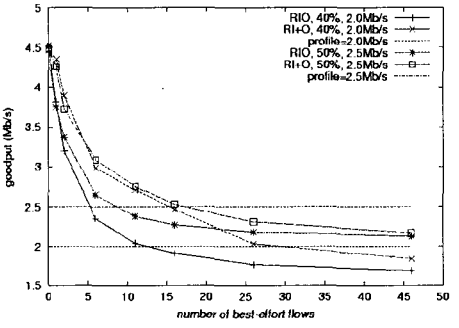
(Fig. 6) plots the avg_{out} created by other flows as a function of profile size. The increasing function is that of RIO taken from (Fig. 2). Compared with RIO's, the correlation between avg_{out} and the profile size is now(slightly) negative in RI+O.



(Fig. 6) avg_{out} created by other flows, as a function of profile size.

One alternative way to control avg_{out} is to reverse-discriminate small profile flows by also using avg_{in} in determining the OUT drop probability. In current RI+O, the OUT drop probability depends only on avg_{out} (until avg_{total} is regulated by IN+OUT parameters). In the reverse discrimination scheme, we could let the OUT parameters be more strict when avg_{in} is smaller. In this way, we could further compensate for the loss of larger profile flows. But we leave it as a future work.

Now, we explore the effect of the number of BE flows on AS flows performance with RIO and RI+O, respectively. At 40% and 50% of the total subscription level(20Mbps bottleneck link), four AS flows contract the same profile rates(2Mbps and 2.5Mbps respectively). Various number of BE flows compete with them for the bottleneck bandwidth. We vary the number of BE flows from 0 to 46(0, 1, 2, 6, 11, 16, 26, 46). (Fig. 7) shows the simulation result.

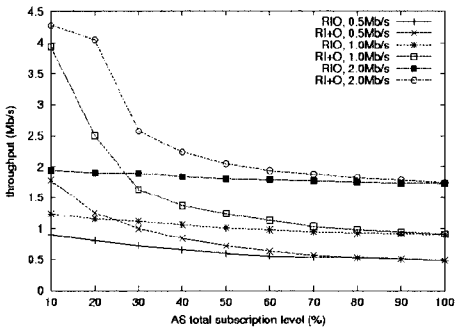


(Fig. 7) Throughput of AS flows vs. number of competing BE flows.

In 40% subscription level, AS flows can maintain their profile rates until the number of BE flows is 13 with RIO and 30 with RI+O, respectively. In 50% subscription level, 9 with RIO and 17 with RI+O. Apparently, the use of RI+O mitigates the impacts of BE flows on AS bandwidth assurance performance.

(Fig. 8) shows another simulation result. The number of BE flows is fixed at 20. AS flows with the same profile rate subscribe 20Mbps bottleneck link. The total subscription level varies from 10% to 100%, and accordingly the number of AS flows varies as well. For each simulation run, the AS flows profile rate is set to either 0.5Mbps, 1Mbps, or 2Mbps. With 0.5Mbps profile rate, the throughputs of RIO and RI+O differ only when the subscription level is low. They both can assure at least the profile rate for all subscrip-

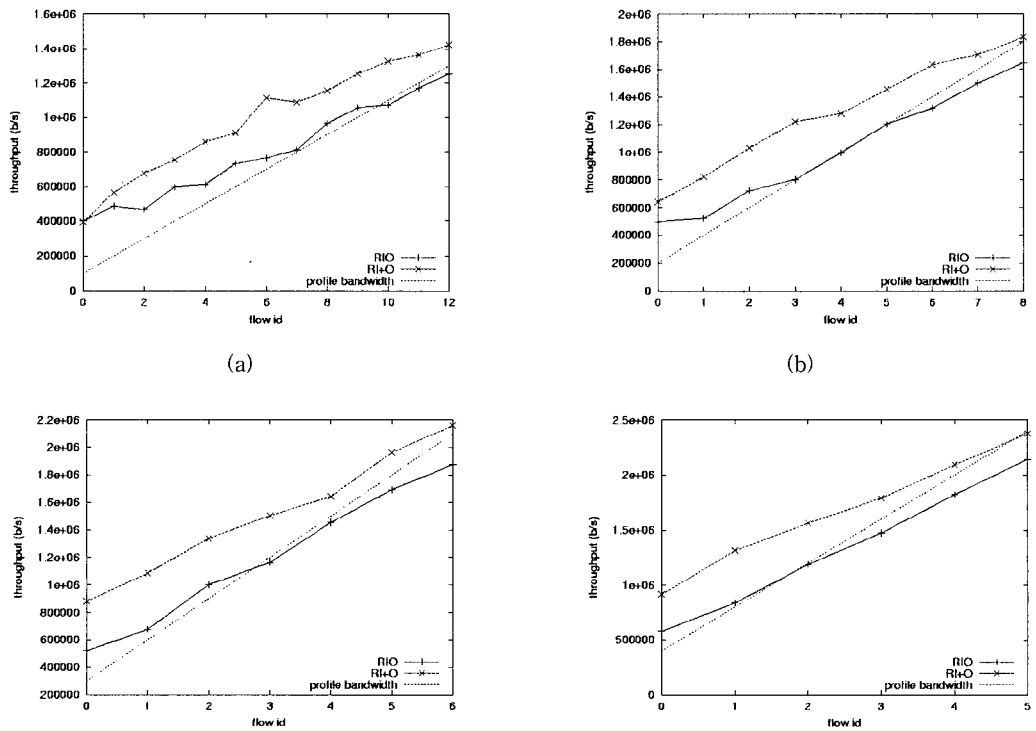
tion levels. But, as the profile rate of each AS flow becomes larger, the performance gap between RIO and RI+O becomes larger. With 1Mbps profile rate,



(Fig. 8) Throughput vs. total subscription level.

RIO can assure the profile rate up to 60% subscription level, but RI+O can assure it up to 80% subscription level. Even with 2Mbps profile rate, RI+O can assure the profile rate until the subscription level reaches 60%, but RIO always fails to assure it. If real networks limit the total subscription level to 50% or below, for instance, RI+O can help meet the profile rate assurance of AS flows while RIO cannot. A general lesson from above experiment is that, (provided the number of BE subscribers is limited) RI+O extends the engineering regime where AS can be offered.

The granularity of profile rate setting is also an important factor in the Diff-Serv network. It is likely that some discrete integrals of a unit bandwidth will be offered to the subscriber, rather than an unlimited spectrum of bandwidth values. Below, we test what unit profile size each queue management scheme can



(Fig. 9) Throughput vs. profile granularity in RIO and RI+O.

efficiently support. It is expected that with a large unit granularity the bandwidth assurance failure will be more pronounced. We performe a simulation where AS flows have linearly increasing profile rates, all integrals of a unit bandwidth. The AS flows share 20Mbps bottleneck link with 20 BE flows. We run 13 AS flows at 0.1Mbps unit granularity, 9 for 0.2Mbps unit 7 for 0.3Mbps unit, and 6 for 0.4Mbps unit, respectively.

(Fig. 9) shows the throughputs of AS flows as we increase the unit profile granularity. (Fig. 9(a)) shows the 0.1Mbps granularity case. RIO fails to meet the assurance for the AS flows with 10 times the unit granularity or larger, but RI+O successfully provides in all profile sizes. As a matter of fact, RI+O does not fail for any four experiments. In the 0.2 Mbps granularity case (Fig. 9(b)), RIO fails for the AS flow with 5 times the unit granularity or larger. (Figs. 9(c)) and (Figs. 9(d)) show that RIO fails from 3 and 2 times the unit granularity, respectively. Also, in allcases, the throughput achieved with RI+O is greater than with RIO. In summary, RI+O allows us to offer coarser granularity profile rates to AS subscribers.

6. Conclusion

In this paper, we explored two ways to remedy the bandwidth assurance failure problem in Assured Service. Our approach contrasts with existing proposals in that only the not yet deployed components of the Diff-Serv architecture are modified. We first investigated the effect of varying the token bucket size in the traffic conditioner. We found that it helps to alleviate the problem, but very large bucket sizes are required. Since it can increase the burst size, stressing the network, it alone is not a desirable solution. Then we looked into the role of RIO queue mechanism in the assurance failure. We demonstrated that RIO is partially responsible for the problem. We showed that RIO has an operational deficiency when used in a normal, well-engineered network. Namely, it cannot control the composition of the total average queue length. Meanwhile, the assurance failure problem manifests as a positive correlation between the profile size and the average out queue length available to the flows other than the flow in its IN burst. Based on this finding, we proceeded to modify RIO in order to make it capable of directly controlling the average OUT queue. As a result, we could successfully adjust the IN burst lengths of different profile flows. This adjustment improved the throughputs of the larger profile flows, at the cost of reduced excess bandwidth share of the smaller profile flows. RI+O also provides improvements over RIO in other aspects of the assurance failure problem. It has better resistance to an increased number of competing BE flows, and accommodates larger profile granularities. Although flow aggregation mitigates the intensity of the assurance failure problem, it is still important to provide a guarantee to individual flows. For instance, high-rate TCP-friendly real-time flows need such guarantee. Or, an AS subscriber testing if its guarantee is met can launch a large FTP session. Moreover, RI+O is a transparent, light-weight mechanism that activates itself only when the queue dynamics indicates the occurrence of unfair treatment towards large profile flows. Still, resolving unfairness when the traffic is highly aggregated, and further refinement of RI+O is under way.

References

[1] K. Nichols and S. Blake, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers," RFC 2474.

[2] D. Black, S. Blake, M. Carlson, E. Davis, Z. Wang and W. Weiss, "An Architecture for Differentiated Services," RFC 2475.

[3] V. Jacobson, K. Nichols and K. Poduri, "An Expedited Forwarding PHB," RFC 2598.

[4] J. Heinanen, F. Baker, W. Weiss and J. Wroclawski, "Assured Forwarding PHB Group," RFC 2597.

[5] K. Nichols, V. Jacobson and L. Zhang, "A Two-

bit Differentiated Services Architecture for the Internet," draft-nichols-diff-svc-arch-02.txt, Internet Draft, April 1999.

[6] D. D. Clark and W. Fang, "Explicit Allocation of Best-effort Packet Delivery Service," IEEE/ACM Transactions on Networking, Vol.6, No.4, August 1998, pp.362-373.

[7] M. Goyal, P. Misra and R. Jain, "Effect of Number of Drop Precedences in Assured Forwarding," draft-goyal-dpstdy-diffserv-00.txt, Internet Draft, March 1999.

[8] K. Nichols and B. Carpenter, "Format for Diffserv Working Group Traffic Conditioner Drafts," Internet Draft, February 1999.

[9] J. Heinanen and R. Guerin, "A Three Color Marker," draft-heinanen-diffserv-tcm-01.txt, Internet Draft, February 1999.

[10] J. Heinanen and R. Guerin, "A Two Rate Three Color Marker," draft-heinanen-diffserv-trtcm-01.txt, March 1999.

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

[12] J. Ibanez and K. Nichols, "Preliminary Simulation Evaluation of an AS," draft-ibanez-diffserv-assured-eval-00.txt, Internet Draft, August 1998.

[13] I. Yeom and A. L. N. Reddy, "Realizing Through put Guarantees in Diff-serv Networks," IEEE ICMCS 99.

[14] W. Feng, D. D. Kandlur, D. Saha and K. G. Shin, "Understanding TCP Dynamics in an Integrated Services Internet," Proceedings of the International Workshop on NOSSDAV, May 1997.

[15] Network simulator v.2 (ns-2), University of California at Berkeley, CA, 1997. Available via http://www-nrg.ee.lbl.gov/ns-2.

[16] References on RED(Random Early Detection) Queue Management. Available via http://www.aciri.org/floyd/red.html.

[17] W. Feng, D. D. Kandlur, D. Saha and K. G. Shin, "A Self-Configuring RED Gateway," Infocom 99.

[18] H. Kim, S. Thomson and W. Leland, "Evaluation of Bandwidth Assurance Service using RED for Internet Service Differentiation," unpublished paper, available via ftp://ftp.bellcore.com/pub/world/hkim/assured.ps.Z.

김 효 곤

e-mail : hkim@madang.ajou.ac.kr

1987년 서울대학교 공과대학 전자계산기 공학과(공학사)

1989년 서울대학교 대학원 전자계산기 공학과(공학석사)

1995년 U. of Pennsylvania 컴퓨터 및 정보과학과(공학박사)

1996년~1999년 Bellcore 인터넷 구조 연구소 연구원

1999년~현재 아주대학교 정보통신대학 조교수

관심분야 : 인터넷 프로토콜, 구조 및 응용전반