SACRIO: An Active Buffer Management Scheme for Differentiated Service Networks

Saikrishnan Gopalakrishnan
Cisco Systems Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
saig@cisco.com

A. L. Narasimha Reddy
Dept. of Electrical Engineering
Texas A & M University
College Station, TX 77843-3128
reddy@ee.tamu.edu

ABSTRACT

In this paper, we propose an active resource management approach for Differentiated Services networks. The proposed approach, SACRIO, employs caching and localized packet remarking within a router. It is shown that SACRIO is simple and can be implemented transparently within the diff-serv architecture. It is shown that the packet handling cost remains O(1) with SACRIO. SACRIO is shown to be quite effective and scalable through both ns-based simulations and real-world trace-driven simulations.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]; C.2.3 [**Network Operations**]: Network management; C.2.6 [**Internetworking**]: Routers; C.4 [**Performance of Systems**]: Design studies

Keywords

QOS, Differentiated Services, RIO, active buffer management, packet remarking.

1. INTRODUCTION

Diff-serv architecture is proposed to provide different levels of services [1, 2]. In a diff-serv network, routers are divided into two categories viz., Edge routers and the Core Routers as shown in figure 1. The core routers maintain no state information. This makes the network scalable since the routers that maintain full state viz., the edge routers, have only a limited number of flows going through them. The Edge routers have the components called the *markers* and the core routers have *droppers*. Essentially the markers mark the headers of the packets depending upon certain factors that are discussed below and the core routers recognize the marking that are done by the edge routers and use that information for resource management. There are currently two per-hop behaviors (PHBs) standardized by IETF, Expedited Forwarding (EF) and Assured Forwarding (AF). This paper deals with bandwidth mangement issues in supporting AF PHB.

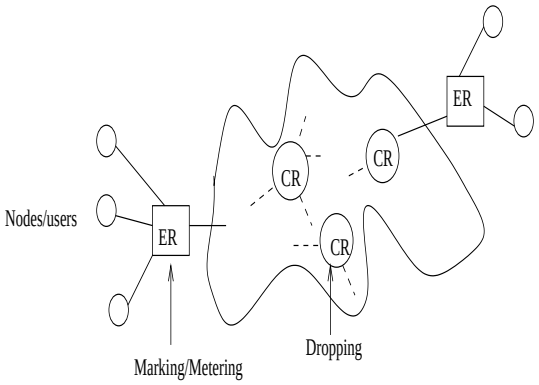

Figure 1: Differentiated Services network

Considerable amount of work has been done in realizing the target rates in a diff-serv network by deploying different marking and dropping strategies. One such marking and dropping strategy is discussed below. Based on the contract (assured throughput guarantees for example), the packets are marked IN if the flow is in-profile and OUT if the sending rate of the flow is out of profile. At the time of congestion, the core router drops the OUT packets first. This two-drop precedence is first proposed in RIO (RED with IN/OUT) [2]. Discussion on RED (Random Early Detection) can be seen in [11]. Like the two-drop precedence discussed above, three-drop precedences also have been studied in [9, 10].

Recent studies [4, 12, 13, 17] have pointed to the difficulty of realizing performance targets when the OUT packets constitute a significant fraction of the link bandwidth: (a) contract rates may not be met, (b) nonresponsive sources may claim a siginificantly more BW than their contract rates, and (c) RTTs impact the realized rates. Analytical work [13, 17] has shown that, without additional mechanisms, BW goals may not be met in certain situations within the current diff-serv framework. This paper proposes an active buffer mangement scheme that improves the realization of performance targets.

The paper makes the following signficant contributions: (a) proposes localized remarking of packets within a router for active management of link bandwidth within a diff-serv

NOSSDAV'01, June25-26, 2001, Port Jefferson, New York, USA

network, (b) proposes the employment of caching (partial state) to monitor high-rate flows, (c) presents extensive simulations to demonstrate the effectiveness of the proposed approach based on these ideas.

The rest of the paper is organized as follows. In section 2, we propose SACRIO (Sampling and Caching using RIO), an approach to active resource management in differentiated service network. In section 3, the implementation details of SACRIO are given. Section 4 presents the simulation results of SACRIO's performance under various conditions. Section 5 discusses the scalability issues of SACRIO through a simulation analysis of real-world traces. Section 6 concludes the paper.

2. SACRIO - AN APPROACH FOR ACTIVE RESOURCE MANAGEMENT

Our approach attempts to manage the excess bandwidth effectively to improve the realization of performance goals. Every router will know the capacity of its output link C. If it knows the amount of IN traffic going through it, then it can find how much of excess bandwidth E it has. In order to manage this bandwidth E effectively, a router converts an amount of OUT traffic at the rate of E into another class of packets, IN2, as shown in figure 2. Since the capacity of this link would not be equal to that of another router down the network path, it cannot assume that this conversion of $OUT \rightarrow IN2$ is valid for the other router. So it has to rename the packets which it converted to IN2 as OUT before it puts them on its output link. As shown in figure 2 just before dequeuing, we convert those packets with IN2 as OUT.

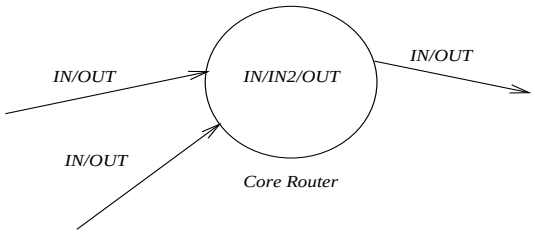

Figure 2: 2-levels in the network and 3-levels in the core router

The changes in packet markings are localized and hence no standardization across the network is needed. This makes this scheme easily deployable. Now the buffer management of SACRIO will need to identify 3 levels of packets IN, IN2, and OUT. It is graphically shown in figure 3. The new IN2 marked packets have a precedence between IN and OUT packets. The thresholds for OUT packets are moved further down. As a result, very few OUT packets, that are not converted into IN2, get through the routers.

By effectively managing the remarking scheme, different performance goals can be achieved. Limiting the OUT bandwidth of a flow to a target rate is used as an example goal in this paper. The proposed approach can be generalized to work with different goals.

In the rest of this section, we will discuss how to identify the excess bandwidth and on what basis we can convert $OUT \rightarrow IN2$. The router can find the rate of the IN traffic without maintaining per-flow state information. Hence it can easily estimate rate at which the OUT packets need to get converted into IN2. The important question is which OUT packets should be converted into IN2? It is not advisable to convert, indiscriminately, all the OUT packets to IN2 at the rate of the excess bandwidth. For, such probabilistic marking still allows a large number of packets of a non-responsive flow to pass through the router.

In order to facilitate a more intelligent remarking of packets, we propose an approach which maintains a cache at each core-router. The cache carries information about some of the flows that require individualized bandwidth management. For example, when the goal is to limit resource consumption, we maintain information about flows that are exceeding the target rate. For these flows, for which the state is maintained, we can accurately manage their resource consumption at the routers. For the other flows, for which the state information is not maintained, we allow aggregate policies to effectively manage their resource consumption.

This approach is similar to how caches are employed in modern computer systems. Caches are an engineering solution employed in an architecture-transparent fashion to improve performance. Similarly, we plan to use a cache to save state for a few flows that are sending at a high rate. By actively managing the resource usage of these flows, it is expected that we may be able to improve the realization of the performance goals. This cache is transparent to the QOS framework. In [5], the authors show that this can improve fairness in sharing link bandwidth. Our work here targets the specific problems of diff-serv networks and the solution we propose here is much simpler and different from [5]. Caching has been used for router-lookup and other functions in routers [6]. Sampling has been employed for network management in [7].

Since there are a limited number of entries in the cache, sampling is used to determine the candidates for state maintenance. Sampled flows are monitored to see if they require individualized resource management and control. If not, the state space in the cache is allowed for other sampled flows. For example, a flow sending at a rate higher than the target rate can be allocated space in the cache when limiting a flow's resource consumption is a performance goal. The state management policy decides how long the flow's state is maintained in the cache.

These two general ideas, (a) localized packet remarking and (b) sampling and caching to facilitate state maintenance for a few select flows form the basis of our proposed scheme SACRIO.

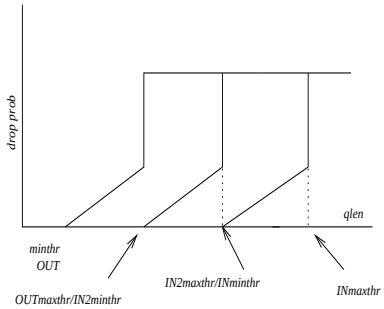

Figure 3: SACRIO with IN/IN2/OUT

Continuing with the example goal of limiting a flow's share of excess bandwidth, if we maintain state information for each flow, we could calculate the OUT rate of every flow OUT_i. Then, we can calculate the ideal dropping rate for this flow as 1 - α/OUT_i, where α is the target rate for a flow's excess bandwidth. Such a strategy is adopted by [8] where edge routers measure the flow rates and this state information is carried as a part of the packet header. This mechanism requires that routers be standardized to mark the packet state uniformly. Diff-serv architecture already uses IP packet TOS byte to mark the packets differently and hence the approach in [8] is not directly compatible.

SACRIO employs caching to solve this problem. The flows are sampled, cached and are monitored to see whether the rate of their OUT packets is more than the target share of the excess bandwidth. If that is the case, they are individualized. If not, aggregate policies manage these flows.

SACRIO does not explicitly drop the OUT packets that are not converted into IN2. We let the buffer management scheme control the rate at which these packets get dropped. As shown in figure 3, the OUT packets are set to have low queue thresholds and hence will be dropped at a higher rate whenever the queues build up. It is emphasized that SACRIO actively allocates excess bandwidth so that IN2 marked packets get through the router at a higher probability than otherwise.

Although SACRIO requires the core router to recognize 3 different kinds of packets IN/IN2/OUT, it is not similar to the three-drop precedence mentioned in [9]. Where as three-drop precedence requires the R/Y/G to be marked at edge routers depending upon a pre-determined R/Y/G levels, the method that is used in this paper converts $OUT \rightarrow IN2$ at every core router based on the excess bandwidth on that particular output link.

In the next section, we discuss the implementation details.

3. IMPLEMENTATION OF SACRIO

The implementation description is based on a goal of limiting OUT bandwidth to a target rate of α. This is expected to serve as a concrete example. Again, it is emphasized SACRIO can implement more general performance goals. SACRIO recognizes 3 levels of packet markings IN/IN2/OUT when the edge router employs IN/OUT markings. SACRIO implementation with three color edge marking [9] is discussed later. SACRIO consists of 2 main parts. The first part identifies the flows whose OUT rate (OUT_i) exceeds their share of excess bandwidth. This is done through sampling and caching. Flows are sampled and monitored to see if their OUT_i exceeds α. If it is observed that a flow's outrate is greater than α, that flow is pinned. State information, OUT_i, is maintained for all cached flows. The OUT packets of pinned flows are converted into IN2 at the rate of α/OUT_i. The OUT packets of non-pinned flows are converted into IN2 at an aggregated rate as described below.

The second part takes care of dropping of packets at the time of congestion. This is done through active buffer management. SACRIO adjusts the queue thresholds appropriately such that IN/IN2 marked packets get through the router and very few packets marked OUT (not converted into IN2) get through the router.

In this section, we describe the details of the first part. The flowchart shown in figure 4 depicts the different modules and the overall flow of SACRIO. The following description of SACRIO's operation closely follows the flowchart in Fig. 4.

If the arriving packet is marked IN, a counter for estimating the total IN rate is incremented and the packet is enqueued for the output link. If the arriving packet is marked OUT, the cache is checked to see if the packet belongs to a cached flow. If the packet doesn't belong to a cached flow, it is checked to see if this flow can be cached. If there is room in the cache, this flow is cached and the out packet counter for this flow is initialized.

If the arriving OUT packet belongs to a cached flow, the packet count is incremented for this flow. If the observation period for this flow has expired, the out rate is calculated by dividing the packet count by the observation period. If the resulting outrate is below α, this cache entry is marked invalid making room for another flow. If the out rate is above α, this entry is *pinned*, and the OUT packet of this flow is converted to IN2 with a probability of α/OUT_i, where OUT_i is the outrate observed in the last observation period.

If the arriving OUT packet belongs to a non-pinned flow, it is converted into IN2 with a probability of p_{agg}. Let OUT_{total} be the total out rate, OUT_{pinned} be the out rate of the *pinned* flows in cache and l be the number of flows that are pinned in the cache. The probability that an OUT packet of a non-pinned flow is converted from OUT to IN2 is

$$p_{agg} = (C - IN - \alpha * l)/(OUT_{total} - OUT_{pinned}) \quad (1)$$

In the above equations, IN, OUT_{total} and OUT_{pinned} can be easily measured through counters.

3.1 Cost analysis of SACRIO

As mentioned earlier, there are 2 main parts in SACRIO viz., (i) identifying a non-cached flow plus calculation of $OUT \rightarrow IN2$ probability (ii) dropping packets if needed according to the OUT/IN2/IN thresholds.

The IN packets will go through only the 2nd part. Please refer to figure 4. The additional cost in SACRIO for an IN packet is the cost of updating the IN packet counter.

The OUT packet will go through identification and probability calculation phases. In the identification phase, cache insertion , searching and deletion are involved. Depending upon the associativity of the cache, the search time will vary. Higher the associativity, higher the time for search without parallel hardware assistance. Cache lookup can be done in O(1) time in hardware. Also at the end of an observation period, the *outrate* of a flow is calculated. As shown in previous subsection, the *outrate* involves division. But by choosing suitable observation periods (for example, a power of 2), this division can be done using simple shift operations. In addition to these above mentioned calculations, we need to keep track of number of OUT packets, IN2 conversion rate for non-cached flows and the cache hit rate. Probability calculations are done only at the end of an observation period. All of these calculations are O(1) per packet arrival. Work done per cached flow is higher than non-cached flows and as more processing (and memory) resources become available, larger caches can be employed.

The memory cost directly depends on the amount of state employed at each router. Since SACRIO is designed to work with partial state, the actual amount of state and hence the cost can be controlled by the designer. If the cache is designed to monitor p flows, the memory cost is $O(p)$.

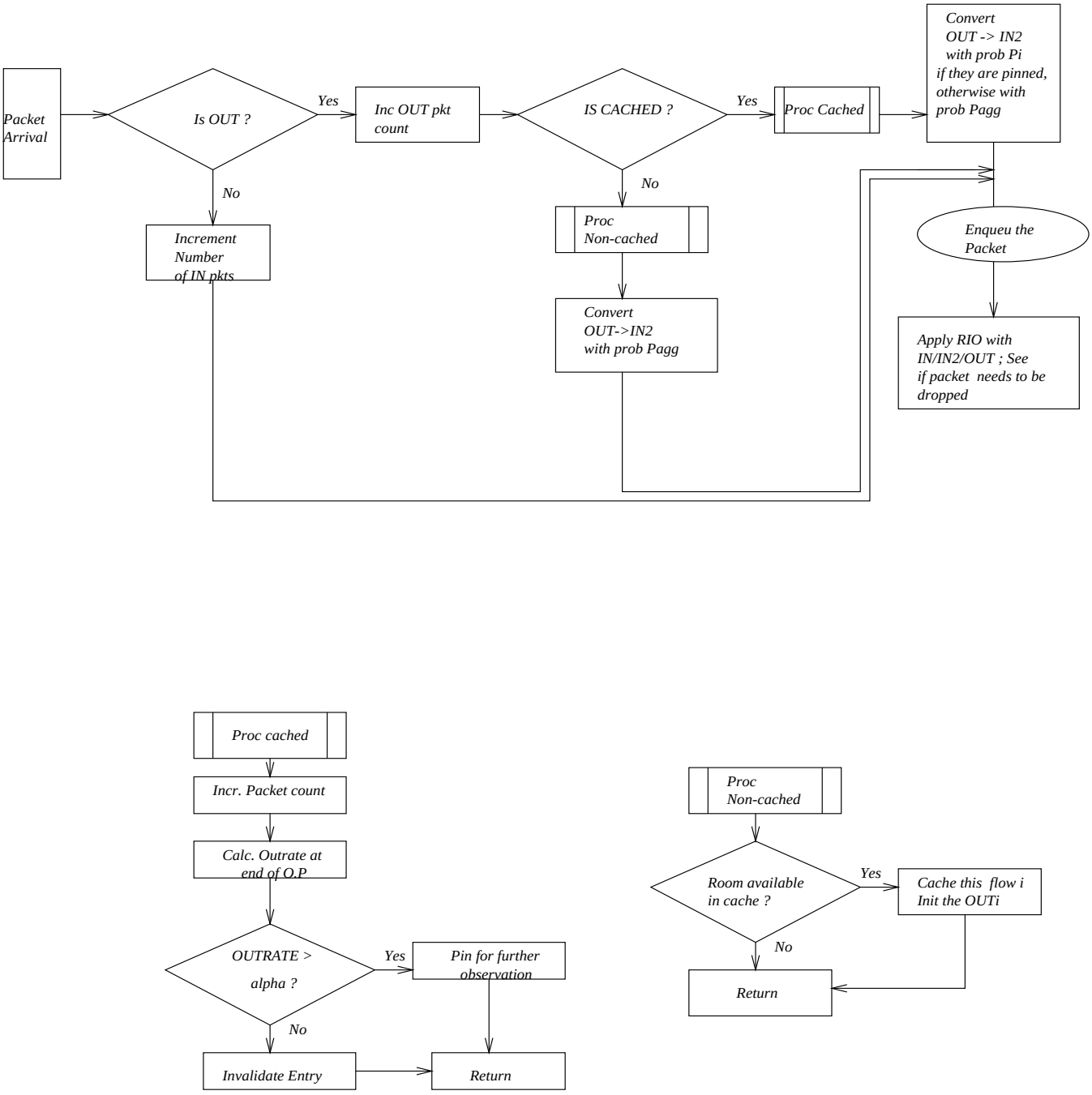

Figure 4: Flow chart describing the operation of SACRIO

3.2 SACRIO with 3-color edge marker

The 3-color edge marker is discussed in [9]. In this method, the edge router differentiates the packets as RED(R), YELLOW(Y) or GREEN(G) based on the pre-determined thresholds for R,Y,G. With the 3-color edge marker, SACRIO cannot convert $R/Y/G \rightarrow OUT/IN2/IN$ directly. We propose the following method to integrate [9] with SACRIO.

At each router, we would like to convert as many yellow packets as possible to IN2 before any red packets are converted into IN2. This brings up two possible cases. Let G, Y, and R denote the total rate of Green, Yellow and Red packets respectively at a router. If $Y > C - G$, then do not convert any $R \rightarrow IN2_R$ and convert $Y \rightarrow IN2_Y$ with a probability,

$$P_{Y \rightarrow IN2_Y} = (C - G)/Y \quad (2)$$

If $Y < C - G$, then we convert all $Y \rightarrow IN2_Y$. The probability for $R \rightarrow IN2_R$ depends upon whether a flow is cached or not. If the packet belongs to a flow i that is pinned in the cache, the probability of conversion to IN2 is given by:

$$P_{R_i \rightarrow IN2_R} = (C - G - Y)/R_i \quad (3)$$

If the packet belongs to a flow which is not pinned and let $R_{agg} = R - R_{pinned}$, then the probability that R is converted into IN2 is given by

$$P_{R_i \rightarrow IN2_R} = (C - G - Y)/R_{agg} \quad (4)$$

Since we have to convert the IN2s into R and Y on the output link, we convert the $Y \rightarrow IN2_Y$ and the $R \rightarrow IN2_R$. This allows us to preserve the original markings of Y and R as the packets exit the router. The above equations show that for the cached flows, we only need to maintain state for counting the Red packets. Only aggregate states for Y and G are necessary. Since a flow cannot send unlimited *yellow* packets, individual flow state is not needed when $Y > C - G$, as observed in equation 2.

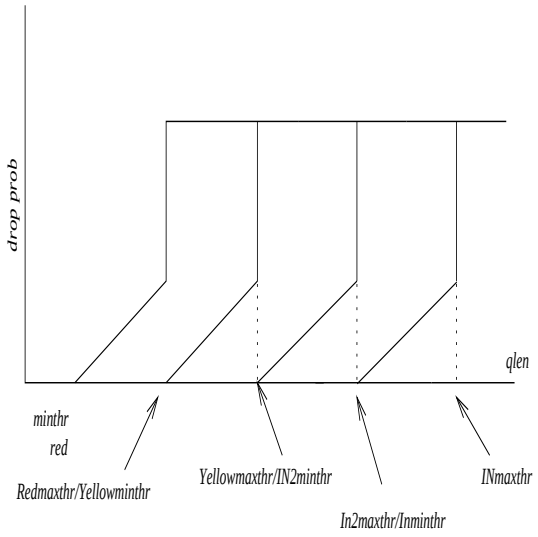

Figure 5: SACRIO with RYG

4. SIMULATION RESULTS FOR SACRIO

In this section, we present the results of the simulations under different conditions for SACRIO. The common setup is as shown in figure 6. There are totally 40 flows out of which 8 are UDP and the rest are TCP-Reno. The UDP flows do not respond to congestion. In the EdgeRouter R1, we have the marker installed and in the router R2, we have SACRIO running. The RTT for all the flows are 8ms and the individual link to the router has 10Mb capacity. SACRIO is configured with 0/0/500/600 for the minthreshhold for OUT, maxthreshhold for OUT, maxthreshhold for IN2, maxthreshhold for IN respectively. The observation period is 20ms. Unless otherwise mentioned, flows 1-8 have 0Mbs contract, flows 9-16 have 0.1Mbs contract, flows 17-24 have0.5 Mbs contract, flows 25-32 have 1Mbs contract and flows 33-40 have 2Mbs contract.

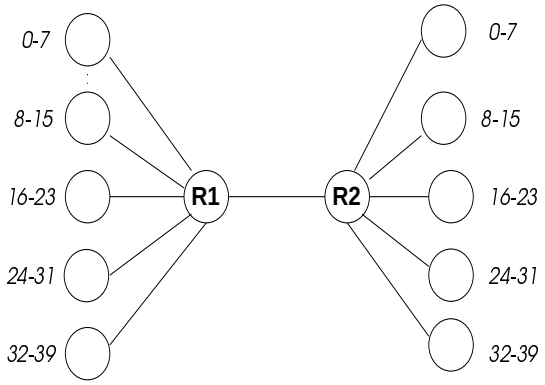

Figure 6: Simulation Setup

4.1 Containment of non-responsive flows

We employ UDP flows as example non-responsive sources. In this experiment, the UDP flows have no contract and hence should receive only best-effort service. The bandwidth in the link R1-R2 is 115.2Mbps, and the contracted rates of other flows amount to a 25 % reservation on the link. Figure 7 is plotted for the plain RIO and SACRIO with 12.5%, 25%, 100% state information. It is evident from the graph that SACRIO has contained non-responsive UDP flows and has distributed the excess bandwidth equally among all the flows. RIO fails to contain these UDP flows and hence they claim a significant amount of the link bandwidth while responsive TCP flows fail to get their fair share of the bandwidth. It is also observed that a 3-color edge marker doesn't offer significantly better performance. When the state information exceeds the number of UDP flows, they are all cached and contained to get only the fair share of excess bandwidth. When state information is not sufficient, some UDP flows escape observation and hence obtain larger than their fair share of bandwidth. With enough state information, all the flows achieve throughput closer to their target rates.

In figure 8, we show how SACRIO performs when the link capacity is 115.2Mb, 57.6Mb, 38.4Mb and 32Mb i.e., when respectively 25%, 50%, 75%, 90% of the total link capacity is reserved. It is observed that in all the cases, SACRIO contains UDP flows to their fair share irrespective of the reserved load on the link.

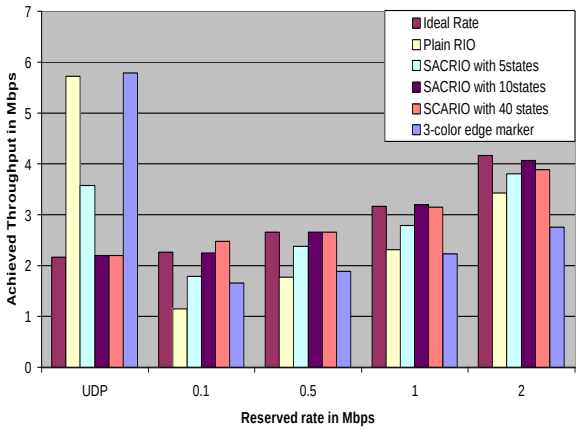

Figure 7: SACRIO performance with UDP contract 0Mbps

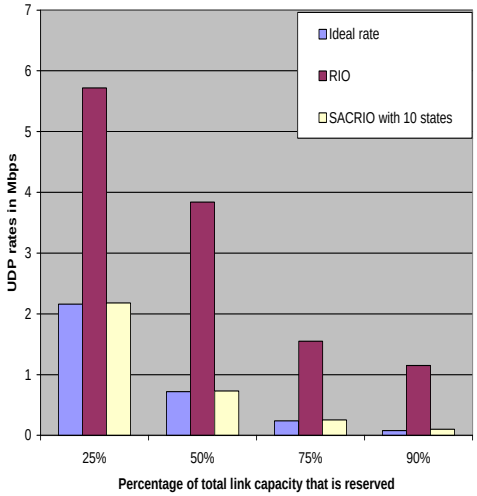

Figure 8: UDP containment with different link capacity reserved

We conducted another experiment to test the efficacy of containing non-responsive flows. In this simulation, the UDP flows have a contract rate of 0.5Mb. The results shown in figure 9 demonstrate that SACRIO does give the UDP flows their share of contracted bandwidth and does not blindly punish the non-responsive flows.

Also figure 9 plots the results for the simulation with queue thresholds of 10/80/500/600. It is observed that the results are nearly the same as with queue thresholds of 0/0/500/600. Our experiments have shown that as long as the IN2 packets have significantly higher queue thresholds, they are protected from the OUT packets of non-responsive flows.

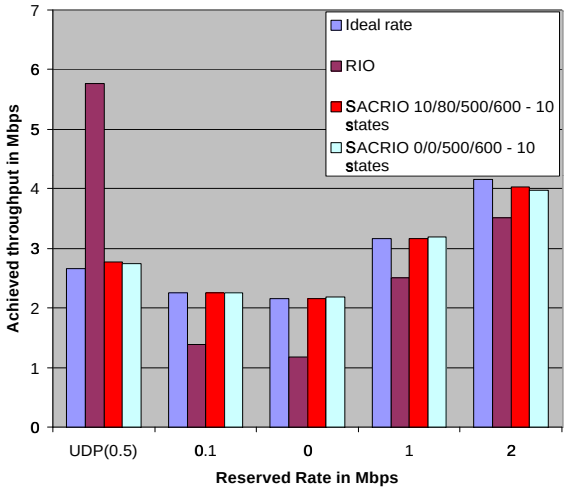

Figure 9: UDP Containment with UDP contract 0.5Mbps

These results indicate that SACRIO is effective in identifying non-responsive sources and at limiting their bandwidth consumption to their fair share. It is also observed that the realized rates are close to target rates irrespective of the contract rates.

4.2 Reducing RTT bias

To observe SACRIO's efficacy at reducing the RTT bias, we conducted simulations with TCP flows with different RTTs. The RTTs of TCP flows within each contract are varied from 8ms-36ms. Let *diffmaxmin* be the difference between max. and min. throughput of the flows in the group with the same contract. Figure 10 plots the ratio of *diffmaxmin* to the target rates of the flows. It can be clearly seen from figure 10 that SACRIO reduces RTT bias compared to plain RIO.

4.3 Target rate = fair share of excess bandwidth

Here we consider the case where α is set equal to the fair share of excess bandwidth. The fair share of the excess bandwidth $= E/N = (C - IN)/N$, where N is the number of flows going through the router. The excess bandwidth

E can be accurately estimated by maintaining a counter for the total number of IN packets at the network element. The only challenge is how to estimate the number of flows, N, going through the router.

Determining the number of flows accurately is a difficult task because we don't want to keep per-flow state information. We estimate the number of flows as accurately as possible. In [18], the authors have approached this problem by randomly sampling the incoming packets to see if they get a hit against a cache full of flows. The number of flows then can be estimated based on this cache hit ratio and the number of entries in the cache. If all the flows send packets at an equal rate, $p_{hit} = K/N$, where K is the number of entries in the cache. By measuring p_{hit}, we can then calculate N. However, in the general case, not all the flows are sending packets at the same rate. Moreover, SACRIO tries to cache the flows that are sending packets at a higher rate than others. Hence, observed p_{hit} will most likely be higher than K/N. If we calculate $N = K/p_{hit}$, we will be underestimating the number of flows. As a result, we will overestimate α. If the OUT packets are converted into IN2 based on α/OUT_i, since α is overestimated, we will be converting a higher fraction of packets of the non-responsive flows into IN2. This is undesirable.

In our implementation, flows observed to exceed the target rate α are *pinned*. These are the flows with high rate of OUT packets. At the time of calculation of p_{hit}, we can consider these flows separately to improve the accuracy of estimating N. We calculate a different probability $p_{hit}^{'}$ as follows: $p_{hit}^{'}$ is the probability of hits against flows that are in the cache, but not pinned. Then the number of flows $N^{'}$ can be estimated as $m/p_{hit}^{'} + l$, where l is the number of pinned entries in the cache and m is the number of unpinned entries in the cache. Through experimentation, we found this estimate to be much more accurate than the earlier estimate. We used another estimate $N^{''} = (OUT/OUT_{pinned} + 1)N^{'}$. We will present results later with both estimates of $N^{'}$ and $N^{''}$. We will show that results are pretty close to the ideal rates with either of these estimates. We will also present results to show the impact of misestimation of α.

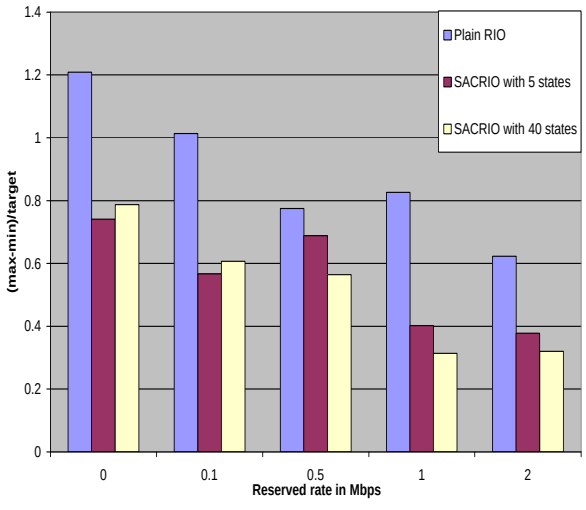

Figure 10: SACRIO with TCP flows of different RTTs

Figure 11 compares the performance of SACRIO when we know the number of flows, N, that are going through the router with that of SACRIO when we estimate N. Estimate 1 and Estimate 2 in figure 11 refer to $N^{'}$ and $N^{''}$ discussed above. From the figure 11, we find that performance with $N^{''}$ estimate is close to the ideal case when the number of flows is exactly known. It is observed that the performance with $N^{'}$ estimate is slightly worse. In all cases, SACRIO's performance is much better than that of RIO.

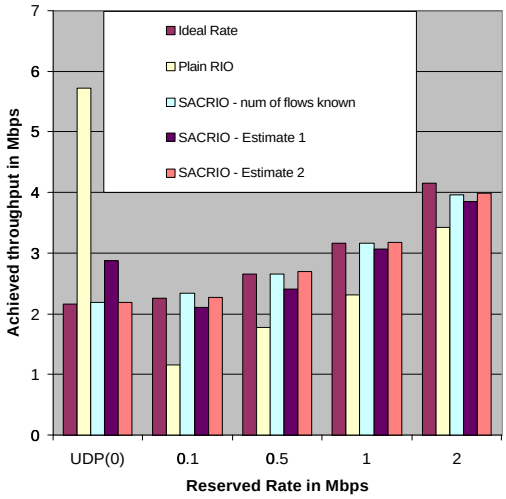

Figure 11: Performance of SACRIO with different estimates of flows

To study the impact of miscalculation of excess bandwidth further, we conducted another experiment. In this simulation, the fair share excess bandwidth parameter α is artificially set either too high or too low. In one case, we set it to 200% of the actual value of α and in another case, we set it to 50% of the actual value of α. This is expected to illustrate the potential range of SACRIO performance. From figure 12, it is observed that even when we over-estimate α by 100%, obtained results are better than RIO's results. Underestimation of α leads to aggressive containment (or punishment) of non-responsive sources.

4.4 Multiple SACRIO routers

Earlier experiments have shown that SACRIO can be effective in improving the provided bandwidth service at a single router. How well does performance improve as more routers in the network deploy SACRIO? We also wanted to see if multiple routers can work together to contain non-responsive sources when the state at a single router is not enough to cache all the non-responsive flows. In this experiment we setup 3 routers, so that routers R2, R3 run SACRIO. The setup is as shown in figure 13. The link R1-R2 as well as R2-R3 have 25% of their link bandwidth reserved.

We have just 5 states in the cache of R2 while there are 8 non-responsive flows. We observe from the simulation re-

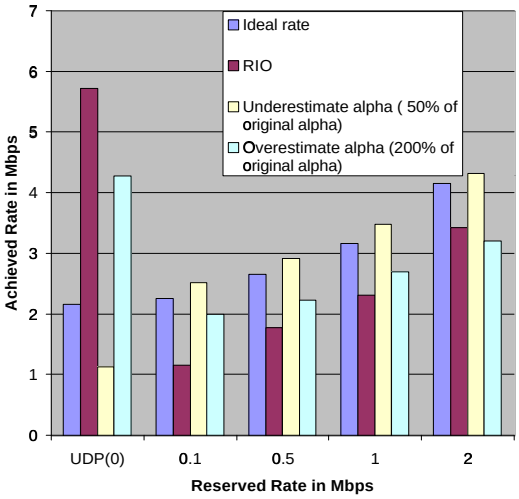

Figure 12: Impact of overestimating or underestimating alpha

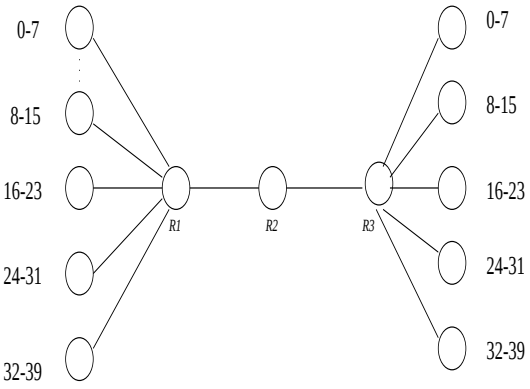

Figure 13: The 3 router Setup

sults that the 3 UDP flows that escape R2 get caught in R3. This is shown in the figure 14. This shows that as more SACRIO routers are employed, the provided service is improved. This shows that the state of all the routers can be combined and employed to contain non-responsive sources.

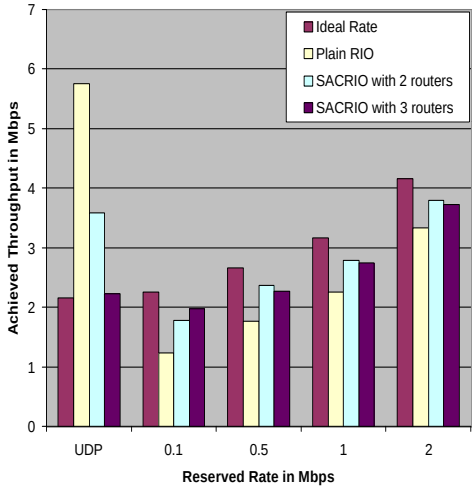

Figure 14: Impact of multiple SACRIO routers.

To observe the performance of SACRIO in more detail, we did simulations in a complex network topology with multiple routers. The setup is as shown in figure 15. The results are shown in 16. The results clearly show that SACRIO performs better than RIO by containing the UDP flows and by realizing throughputs close to the ideal target rates.

We tested SACRIO in many more settings: (a) with different RED parameters for the different classes of packets, (b) in settings where the number of flows were constantly changing, and (c) with aggregate contracts. Results for these experiments can be found in [19].

5. SCALABILITY OF SACRIO

Our results in the earlier section have shown that SACRIO is effective in achieving bandwidth allocation goals. We have shown that SACRIO can effectively contain non-responsive sources when the amount of available state is larger than the number of non-responsive flows. We have also shown that when the amount of state is less than the number of non-responsive flows, SACRIO improves the performance over RIO. We have also shown that the amount of state available across multiple routers can be employed in an additive fashion to contain non-responsive flows even when no single router has enough state to monitor all the non-responsive flows. The simulations have shown that SACRIO provides *flexible* service (the larger the cache, the better the performance) and *additive* service (the more routers employ it, the better the performance).

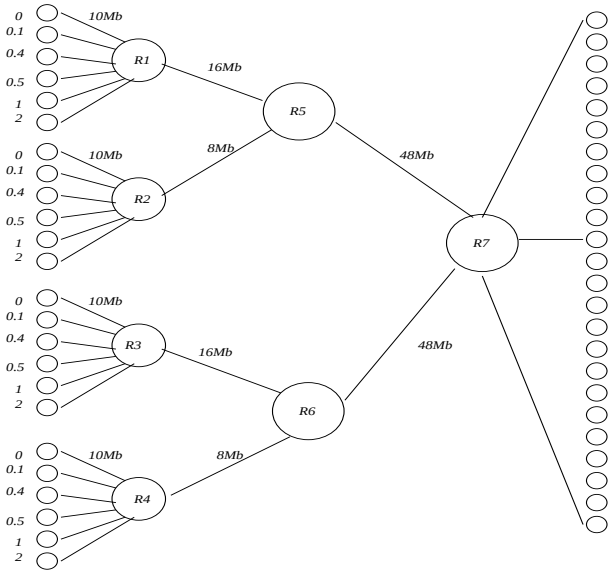

Figure 15: Simulation topology in a heterogeneous network

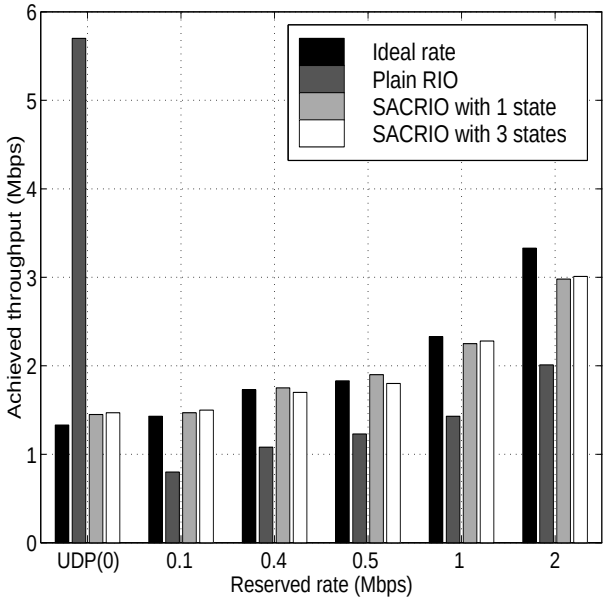

Figure 16: Performance of SACRIO in complex topology

How effective is SACRIO likely to be in real-world when a router may observe thousands of flows? How big a cache is likely to be required for SACRIO to be effective? How scalable is SACRIO?

To answer these questions, we have looked at traces of real-world networks. We studied traces available publicly from NLANR [14] to gain an understanding of the scalability of SACRIO. We collected six traces representing traffic at different times on different days. We reran our simulation with the traces to see the effectiveness of a small amount of state. The traces are all collected over a period of about 95 seconds of wall clock time. The traces range from 43MBs of data to 407MBs of data over the trace period. The number of flows ranges from 4,245 to 28,585 and the number of packets ranges from 131,044 to 835,526.

The results of caching with a 100-flow cache are shown in Table 1. The results show that a small cache of 100 entries can monitor slightly larger number of flows (189 to 327) over the simulation time since the captured flows are not active during the entire 95 second period of the trace. Even though this represents only about 0.66% to 7.70% of the flows, the packet monitoring rate (or packet hit rate) is much higher, ranging from 26.42% to 52.52%. Similarly the byte monitoring rate (or byte hit rate) is much higher, ranging from 23.45% to 56.25%. This shows that SACRIO's cache policy resulted in most of the higher-rate flows being captured and monitored in the cache. This is a direct result of the nature of the network trace where a few flows contribute to most of the packets and bytes transfered through the link. This heavy-tailed property has been extensively reported in the literature [15, 16].

We ran a second experiment with Trace5 to see the impact of the cache size on the performance of SACRIO. The results are shown in Table 2. As the cache is varied from 50 flows to 1000 flows, the packet hit ratio in the cache goes up from 21.63% to 54.31% and the byte hit ratio goes up from 31.16% to 59.04%. This shows that with modest size caches (1000 flows), it is possible to observe and monitor a significant (about 60%) fraction of the traffic. As the hardware improves, more state may be deployed, and thus allowing greater service from this approach.

These results, coupled with the results from more controlled simulations of earlier sections, lead us to believe that partial state in the form of small caches can be highly effective in managing link bandwidth. The results show that as the cache size increases, more traffic can be monitored and controlled without requiring the network elements to establish complete state for all the flows through the link.

6. CONCLUSION

In this paper, we proposed a novel approach for active resource management in a differentiated services network. The proposed approach, SACRIO, employed: (a) active resource allocation through localized remarking of packets at a router, (b) sampling and caching (partial state) to identify and maintain state for a select few flows, and (c) active buffer management for realizing performance goals. Results from both ns-based simulations and real-world traces were presented to show the effectivenss of SACRIO. It was shown that SACRIO (a) can be very effective in containing non-responsive flows, and (b) can reduce the RTT bias among responsive TCP flows. It was also shown that SACRIO is scalable i.e., service is improved (a) as more routers deploy

Table 1: Performance of a 100-flow cache on different traces

	Flows			Packets			Bytes		
Trace	Total Flows	Cached Flows	Hit Ratio	Total Packets	Cached Packets	Hit Ratio	Total Bytes(MB)	Cached Bytes(MB)	Hit Ratio
Trace1	4,245	327	7.70	131,044	61,794	47.16	43.58	10.22	23.45
Trace2	5,870	218	3.71	442,310	167,174	37.80	174.79	64.72	37.03
Trace3	9,059	207	2.29	458,691	240,922	52.52	185.86	104.54	56.25
Trace4	16,112	222	1.38	573,382	224,851	39.21	287.31	127.40	44.34
Trace5	21,348	257	1.20	557,001	147,179	26.42	236.19	80.71	34.17
Trace6	28,585	189	0.66	835,526	365,505	43.75	407.36	215.10	52.81

Table 2: Impact of cache size on Trace5

	Flows			Packets			Bytes		
Cache Size	Total Flows	Cached Flows	Hit Ratio	Total Packets	Cached Packets	Hit Ratio	Total Bytes(MB)	Cached Bytes(MB)	Hit Ratio
50	21,348	120	0.56	557,001	120,491	21.63	236.19	73.60	31.16
100	21,348	257	1.20	557,001	147,179	26.42	236.19	80.71	34.17
200	21,348	557	2.61	557,001	204,740	36.76	236.19	107.53	45.53
500	21,348	1,290	6.04	557,001	248,742	44.66	236.19	119.00	50.38
1000	21,348	2,175	10.19	557,001	302,496	54.31	236.19	139.44	59.04

SACRIO and (b) as more (partial) state is employed within a router. It was shown that the packet handling cost remains O(1) in SACRIO.

7. ACKNOWLEDGEMENTS

Adnan Khaleel carried out the analysis of Internet traffic presented in section 5.

8. REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentited Services," RFC2475, IETF, Dec. 1998.

[2] D. Clark, W. Fang, *Explicit Allocation of Best-Effort Packet Delivery Services*, *IEEE/ACM Transactions on Networking*, August 1998.

[3] V. Jacobson, K. Nichols and K. Poduri, "An expedited Forwarding PHB," RFC2598, IETF, June 1999.

[4] J. Ibanez and K. Nichols, "Preliminary Simulation Evaluation of an Assured Service," draft-ibanez-diffserv-assured-eval-00.txt, INTERNET DRAFT, August, 1998.

[5] D. Tong and A.L.Narasimha Reddy, *QoS enhancement with partial state*, International Workshop on QOS '99.

[6] L. Peterson and B.Davie, *Computer Networks: A Systems Approach*, Morgan-Kaufman Publishers '99.

[7] K.C. Claffy, G.C. Polyzos and H-W. Braun, *Application of sampling methodologies to network traffic characterization*, ACM SIGCOM '94.

[8] I. Stoica, S. Shenker, H. Zhang, *Core-stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks*, SIGCOMM '98.

[9] J Heinanen, T. Finland, R. Guerin, *A Three Color Marker*, draft-heinanen-diffserv-tcm00.txt, INTERNET DRAFT, Feb '99.

[10] J Heinanen, T. Finland, R. Guerin, *A Two Rate Three Color Marker*, draft heinanen-diffserv-trtcm-00.txt, INTERNET DRAFT, Mar '99.

[11] S. Floyd and V. Jacobson, *Random Early Detection Gateways for Congestion Avoidance*, IEEE/ACM Transactions on Networking '93.

[12] I. Yeom and A.L.Narasimha Reddy, *Realizing throughput guarantees in a differentiated services network*, IEEE Int. Conf. on Multimedia Computing and Systems '99.

[13] I. Yeom and A.L.Narasimha Reddy, *Modeling TCP behavior in a Differentiated Services Network*, ACM/IEEE Trans. on Networking, Feb. 2001.

[14] National Laboratory for Applied Network Research (NLANR), *Wide area Network Traces*, available from http://moat.nlanr.net/Traces, June 2000.

[15] W. Willinger, M. Taqqu, R. Sherman and D. Wilson, *Self-similarity through high-variability: Statistical analysis of Ethernet LAN traffic at the source level*, Proc. ACM SIGCOMM '95, 1995.

[16] M. Crovella and A. Bestavros, *Self-Similarity in World Wide Web Traffic: Evidence and Possible Causes*, Proc. ACM SIGMETRICS '96, May 1996.

[17] S. Sahu, P. Nain, D. Towsley, C. Diot, and V. Firoiu, "On Achievable Service Differentiation with Token Bucket Marking for TCP," Tech. Report 99-72, University of Massachusetts.

[18] T. J. ott, T. V. Lakshman, L. H. Wong: *SRED: Stabilized RED*, IEEE Infocom'99.

[19] Saikrishnan Gopalakrishnan, *TRIO: An Active Buffer Management Scheme for Differentiated Services Networks*, Master's thesis, Texas A & M University, May 2000.