# Adaptive Token Bucket Algorithm for Fair Bandwidth Allocation in DiffServ Networks

Eun-Chan Park and Chong-Ho Choi
School of Electrical Engineering and Computer Science,
Seoul National University, Seoul, KOREA
{ecpark|chchoi}@csl.snu.ac.kr

*Abstract*— **We propose an adaptive token bucket algorithm for achieving proportional sharing of bandwidth among aggregate flows in differentiated service (DiffServ) networks. By observing the simulation results obtained in a study of the throughput of TCP flows in a DiffServ network, we note that the aggregate flow with a lower target rate occupies more bandwidth than its fair share, while the aggregate flow with a higher target rate gets less than its fair share. The proposed algorithm solves this unfairness problem by adjusting the target rate according to the edge-to-edge feedback information. This algorithm does not require any additional signaling protocol or measurement of per-flow states, since it can be implemented in a distributed manner using only two-bit feedback information carried in the TCP acknowledgement. Using *ns-2* simulations, we show that the proposed algorithm provides fair bandwidth sharing under various network conditions.**

## I. Introduction

Differentiated service (DiffServ) architecture has been proposed as a solution for providing different levels of service to satisfy different service requirements in a scalable manner [1]. In DiffServ architecture, IP flows are classified and aggregated into different forwarding classes. These flows are marked with different levels of priority at the edge of a network and are dropped with different dropping mechanisms at the core of a network. Therefore, DiffServ networks can provide Quality-of-Service (QoS) over and above the current *best-effort* service. The most prevalent forwarding mechanisms standardized by the Internet Engineering Task Force (IETF) are *Expedited Forwarding Per-Hop Behavior (EF PHB)* [2] and *Assured Forwarding Per-Hop Behavior (AF PHB)* [3]. The former is intended to support traffic flows requiring a short delay and the latter is intended to assure a minimum level of throughput. To assure this minimum throughput, which is referred to as the *target rate* or *committed information rate (CIR)*, AF PHB introduces two components; a packet marking mechanism administrated by *profile meters* or *traffic conditioners* at edge routers and a queue management mechanism at core routers. The packet marking mechanism monitors and marks packets according to the service profile at the edge of a network. If the measured flow conforms to the service profile, the packets belonging to this flow are marked with a high priority and receive better service. Otherwise, the packets belonging to the non-conformant part of a flow are marked with a low priority and receive best effort service. The queue management mechanism, deployed at core routers, provides preferential treatment for packets that have high priority. During times of congestion, high priority packets are forwarded preferentially and low priority packets are dropped with higher probability.

The most prevalent profile meters are the Token Bucket (TB) based marker [4]-[6] and the average rate estimator based (Time Sliding Widow (TSW)) marker [7]-[11]. In DiffServ networks, the most widely deployed queue management algorithm is the RED-based algorithm (RED with In/Out (RIO)) [4]. Many mechanisms have been proposed to assure the target rate [5], [6], [7]. Moreover, considerable research has been done on modeling TCP behavior in DiffServ networks [8], [9] and on the conditions required for achieving the target rate [10].

In this paper, we focus on the fairness problem among aggregates, especially when surplus bandwidth is available to be distributed among them or, conversely, when the demand made by the aggregates exceeds the available capacity. The bandwidth should be distributed fairly among all the aggregates sharing the common bottleneck link. Considering the fact that the target rate is determined by the terms of the Service Level Agreement (SLA), which depends on the price that a customer pays, the bandwidth should be distributed in proportion to the target rate. We refer this policy as "proportional sharing of bandwidth".

For proportional sharing [1] of bandwidth, we propose an adaptive token bucket algorithm. This algorithm differs from those proposed in previous studies in the following respects. Firstly, compared with the studies referred to in [5], [6], and [7], our algorithm not only achieves the target rate, but also attempts to provide each aggregate with its fair share of the bandwidth, in proportion to the target rate. Secondly, the study referred to in [12] aims to achieve proportional sharing of excess bandwidth among aggregates, however, the algorithm [12] works well for a restricted range of subscription-levels (20%$\sim$70%) only, while our scheme is designed to work well, regardless of the subscription-level, including in the case where the network is over-subscribed. Thirdly, unlike in the previous studies referred to in [6], [13] and [14], our approach to solve this unfairness problem is based on feedback information, which does not require any additional signaling protocol or measurement of per-flow states. The proposed algorithm adapts the target rate according to the congestion level of the network. It utilizes edge-to-edge feedback information and the feedback information is conveyed using TCP acknowledgements (ACK).

The remainder of the paper is organized as follows. The preliminary simulation results in Section II provide the motivation for this paper. We study the unfairness problem by observing the TCP throughput of aggregate under several different conditions. Based on these observations, in Section III we propose

[1]Note that the notion of *proportional fairness* [11] is different from that of *proportional sharing* of bandwidth. We use the term "proportional sharing of bandwidth" to denote that each aggregate flow occupies bandwidth proportional to its target rate.

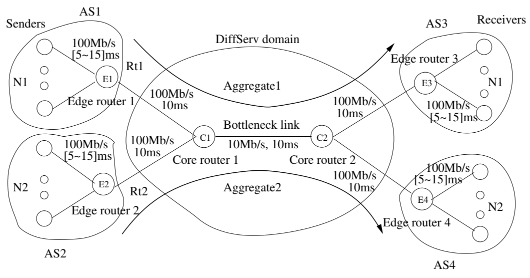
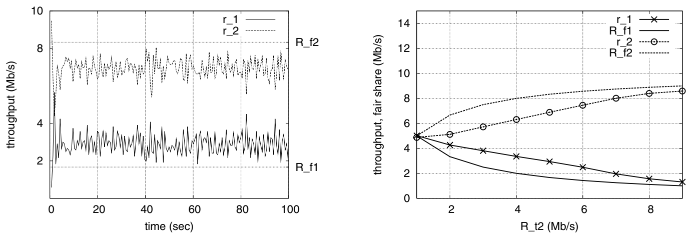
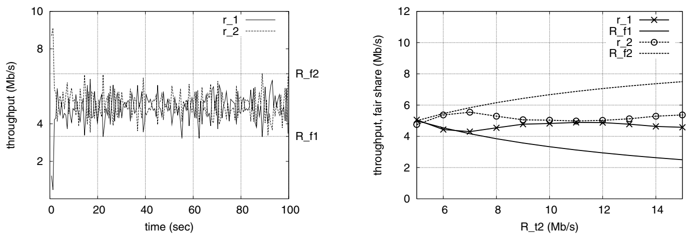


Fig. 1. Network configuration used in the simulation



(a) $R_{t,1}$=1Mb/s and $R_{t,2}$=5Mb/s (b) $R_{t,1}$=1Mb/s and $R_{t,2}$=1∼9Mb/s

Fig. 2. Unfair bandwidth allocation in the under-subscription case with the token bucket algorithm: $r_1 > R_{f,1}, r_2 < R_{f,2}$



(a) $R_{t,1}$=5Mb/s and $R_{t,2}$=10Mb/s (b) $R_{t,1}$=5Mb/s and $R_{t,2}$=5∼15Mb/s

Fig. 3. Unfair bandwidth allocation in the over-subscription case with the token bucket algorithm: $r_1 > R_{f,1}, r_2 < R_{f,2}$

an adaptive token bucket algorithm for proportional bandwidth sharing, and present its architecture and the algorithm used for its implementation. Section IV presents the simulation results, in order to show the effectiveness of the proposed algorithm using *ns-2*. Section V concludes this paper.

## II. Motivation

This section provides the motivation for this study. The *ns-2* simulation study in this section shows that the use of the current Token Bucket (TB) algorithm in DiffServ networks causes unfairness among aggregates that have different target rates. Our study shows that if the aggregate has a relatively lower/higher target rate, it occupies more/less bandwidth than its fair share.

For simplicity, we consider the case in which two identical TCP aggregates, denoted by $Agg_1$ and $Agg_2$, with different target rates $R_{t,1}$[packets/s] and $R_{t,2}$[packets/s] ($R_{t,1} < R_{t,2}$) compete with each other for the bandwidth of a common bottleneck link whose capacity is $C$ [packets/s]. Figure 1 shows the simple network configuration used for the simulation, which is nevertheless sufficient to reveal the unfairness. Further details about the simulation configuration are provided in Section IV. At the ingress edge routers, E1 and E2, the TB algorithm marks the packets as either *IN* or *OUT*, by monitoring the characteristics of the incoming aggregate flows and comparing them with the service profile. Those packets that follow this profile, which is specified in terms of target rate [2] and bucket size, are marked as *IN*, while those packets that do not are marked as *OUT*. Edge router queues implement the drop-tail policy. When packets arrive at core router C1, they are handled differently according to their marking status. Packets marked as *IN* will be forwarded preferentially; *OUT* packets are dropped first on congestion and *IN* packets will not be dropped before *OUT* packets are dropped. Core router queues are managed by the non-overlapping RIO active queue management scheme [4].

We consider that a network is *under-subscribed* or *over-provisioned* if $\sum_i R_{t,i} < C$, and that a network is *over-subscribed* or *under-provisioned* if $\sum_i R_{t,i} > C$. Let us define the *Fair Share* of the $i$th aggregate, $R_{f,i}$ as

$$R_{f,i} = R_{t,i} + (C - \sum_i R_{t,i}) \frac{R_{t,i}}{\sum_i R_{t,i}} = \frac{R_{t,i}}{\sum_i R_{t,i}} C. \quad (1)$$

In order to quantify the discrepancy between the actual throughput $r_i$ and $R_{f,i}$, we define the *Relative Gain* of the $i$th aggregate, $G_i$ as $G_i = r_i / R_{f,i}$.

The following simulation study demonstrates the unfairness problem when the network is under-subscribed or over-subscribed.

[2]The target rate in the TB algorithm is equivalent to the token generation rate.

### A. Under-subscription case

Figure 2 shows the unfairness problem when the network is under-subscribed. In Fig. 2(a), i.e., $R_{t,1}$=1Mb/s, $R_{t,2}$=5Mb/s, and $C$=10Mb/s, we can see that $Agg_1$ gets more bandwidth than $R_{f,1}$=1.67Mb/s, while $Agg_2$ gets less bandwidth than $R_{f,2}$=8.33Mb/s. In order to confirm this unfairness with various sets of target rates, we repeated the simulation; with a fixed value of $R_{t,1}$=1Mb/s and $R_{t,2}$ changed from 1Mb/s to 9Mb/s. As shown in Fig. 2(b), $Agg_1$ which has a lower target rate still occupies more bandwidth than its fair share, i.e., $r_1 > R_{f,1}$, while $Agg_2$ occupies less than $R_{f,2}$. It is worthwhile to note that so long as the network is under-subscribed, the surplus bandwidth, $(C - \sum_i R_{t,i})$, is distributed near **equally**, **not proportionally**, among the aggregates participating in the contention. This explanation is supported by Fig. 2(b), where $r_1$ decreases linearly, while $r_2$ increases linearly.

**Remark 1:** If the target rates of two aggregates are the same, the surplus bandwidth is evenly shared by these two aggregates. Moreover, as the difference between the total target rate and the bottleneck link capacity decreases, the unfairness of the bandwidth sharing also decreases.

### B. Over-subscription case

In order to examine the unfairness problem for over-subscription cases, we performed similar simulations to those conducted for the under-subscription cases. Fig. 3 shows the throughput of the two aggregates and their corresponding fair shares when $R_{t,1}$=5Mb/s and $R_{t,2}$=10Mb/s (Fig. 3(a)) and when $R_{t,1}$ is fixed at 5Mb/s and $R_{t,2}$ varies from 5Mb/s to 15Mb/s (Fig. 3(b)). Figure 3(a) shows that $r_1$ and $r_2$ are nearly the same, although $R_{t,2}$ is twice that of $R_{t,1}$. This unfairness can also be observed in Fig. 3(b), which shows that no matter how large $R_{t,2}$ is, the bandwidth occupied by $Agg_2$ is almost the same as that occupied by $Agg_1$. Figure 3 reconfirms the fact the unfairness problem occurs when the network is over-subscribed.

**Remark 2:** As the total target rate exceeds link capacity and increases, the unfairness among aggregates with different target rates also increases. Conversely, as the difference between the total target rate and the link capacity, $(\sum_i R_{t,i} - C)$, decreases, the throughput of each aggregate approaches to its fair share and the unfairness problem among aggregates is alleviated.

## III. Adaptive Token Bucket algorithm

### A. Design rationale

Remark 1 and Remark 2 in Section II give a clue to the problem of unfair bandwidth sharing; they show that if a network is well-provisioned, there is no bias in favor of the aggregate that has the lower target rate. By taking this fact as a starting point, we argue that the unfairness problem can be alleviated by making a network well-provisioned, whether the network is actually under-subscribed or over-subscribed. We adjust the target rates, so that the sum of the target rates matches the bottleneck link capacity, while keeping the ratio of their original values fixed, i.e,

$$R'_{t,i} = (1 \pm \delta) R_{t,i} \text{ such that} \sum_i R'_{t,i} = C. \quad (2)$$

Here $\delta$ $(> 0)$ is an adjustment factor. If a network is under-subscribed/over-subscribed, we increase/decrease the target rates in proportion to their original values.

In order to accomplish proportional bandwidth allocation, we need to know whether the network is under-subscribed or over-subscribed, so that we can adjust the target rates accordingly. We believe that the solution to this problem should be consistent with the philosophy of DiffServ, i.e., "moving complexity to the edges of the network". The solution adopted should not require any per-flow state at the core routers, for the sake of **scalability**, or any critical changes either in the edge routers or the current transport-layer protocol, for the sake of **compatibility**. A solution that allocates the bandwidth proportionally should have the following two properties.

- The target rates should be adjusted so that their sum matches the bottleneck link capacity as closely as possible.
- The adjustment of the target rates should be performed at the edge routers in a distributed manner, without maintaining any per-flow state.

### B. Architecture and algorithm

The preferential dropping at the core routers provides a good indication of the state of congestion. If the network is far from being congested, *IN* packets will rarely be dropped and the dropping probability for the *IN* packets, $p^{in}$, will be insignificant. Otherwise if the network is heavily congested, almost all of the *OUT* packets will be dropped. Also, in this case, a certain proportion of the *IN* packets will be dropped and $p^{in}$ will not be negligible. Thus, by estimating $p^{in}$ we can infer the state of congestion and determine whether we should increase or decrease the target rate. The egress edge router is in charge of estimating $p^{in}$ and generating feedback information for adjusting the target rate. Then, the feedback information can be carried in a two-bit flag in a packet header via TCP receivers and TCP senders, and finally it is utilized at the ingress edge router when adjusting the target rate. The functionality of each component can be summarized as follows:

- Core router: preferential dropping
- Egress edge router: generating feedback information
- TCP receiver and TCP sender: conveying the feedback information
- Ingress edge router: adjusting the target rate based on the feedback information

The egress edge router estimates $p^{in}$ and makes decisions about changing the target rate. Here, we assume that the networks support the Explicit Congestion Notification (ECN) mechanism [15], which has been proposed as a solution for conveying the congestion signaling rapidly and explicitly to TCP senders. We utilize the ECN mechanism for estimating $p^{in}$. Because the ECN mechanism marks packets instead of dropping [3] them as a means of signaling congestion, we can make use of this information to estimate $p^{in}$ at the egress edge router. Let us denote $\bar{p}^{in}$ and $\hat{p}^{in}$ as the moving average and the estimate of $p^{in}$, respectively. First, we calculate $\bar{p}^{in}$ as the fraction of ECN-marked packets in the recently arrived $N$ *IN* packets. Next, we obtain $\hat{p}^{in}$ as the weighted average of $\bar{p}^{in}$, in order to reduce the bursty nature of TCP, i.e., $\hat{p}^{in} = (1-w)\hat{p}^{in} + w\bar{p}^{in}$. Here, we set the window size, $N$, to 10 and the weight, $w$, to 0.1. Using $\hat{p}^{in}$, the edge router makes a decision about changing the target rate. If $\hat{p}^{in}$ is smaller than a given threshold close to zero, $p^{th}_{min}$, then the edge router sets the `ITR` (Increase Target Rate) bit in the IP header of a packet, which can be used to indicate the need to increase the target rate. Similarly, if $\hat{p}^{in}$ is larger than a certain threshold $p^{th}_{max}$ that is close to the maximum value of $p^{in}$ $(p^{in}_{max})$ in the RIO algorithm, then the edge router sets the `DTR` (Decrease Target Rate) bit in packet's header, i.e.,

$$\begin{aligned} &\text{if } (\hat{p}^{in} < p^{th}_{min}) \longrightarrow \text{Set } \mathtt{ITR} \text{ bit in IP header}, \\ &\text{else if } (\hat{p}^{in} > p^{th}_{max}) \longrightarrow \text{Set } \mathtt{DTR} \text{ bit in IP header}. \end{aligned} \quad (3)$$

When assigning the `ITR` and `DTR` bits in the IP header of a packet, we can make use of the currently unused two-bit subfield in the IPv4 Type-Of-Service (TOS) field or IPv6 Traffic Class (TC) field.

Ideally, this feedback information should be conveyed from the egress edge router where the information is generated to the ingress edge router where it is utilized. However, it is impossible to communicate directly with these edge routers without the aid of any additional signaling protocol, because current IP networks do not have any signaling architecture for this feedback information. Hence, we have to find a way to convey the information to the ingress edge router. The TCP ACK packet can serve as a good transporter for this purpose. If TCP receivers receive a packet whose `ITR` or `DTR` bit is set, they simply extract these flags from the IP header and copy them into the unused field in the TCP header in order for them to be fed back to the TCP senders. Similarly, the TCP senders have the role of conveying information to the ingress edge router. Because the feedback information consists of only two one-bit flags, this does not create a great deal of overhead.

The ingress edge routers are in charge of adjusting the target rates. When the feedback information is conveyed in packet headers, the rate at which the information is transported to each ingress router is not identical. As the sender transmits more packets, the ingress edge router updates its target rate more

[3]Although the packets are marked rather than dropped in ECN-capable networks, we use the term "drop" and "dropping probability" to avoid confusion between **ECN marking** and **priority (*IN/OUT*) marking**.

frequently. In order to resolve the possible imbalance in the update rates among the ingress edge routers, we introduce a timer whose interval is $T_s$. When the timer expires, the target rate is updated. The timer resides on each ingress router, and does not need to be synchronized. We introduce a variable $numATR$, which is used to determine whether to increase or decrease the target rate. It is initialized on expiration of the timer and is increased/decreased by one upon the receipt of a packet whose ITR/DTR bit is set. At each expiration of the timer, if $numATR$ is positive/negative then the target rate is increased/decreased by $\delta$, i.e., $R_t^{'} = (1 \pm \delta)R_t$. There is a trade off to take into account when setting the values of $T_s$ and $\delta$. If $T_s$ is too small or $\delta$ is too big, then the target rate will fluctuate and will not converge toward a level which corresponds to a fair share of the bandwidth. In the opposite case, the response to changes in the network will be slow.

## IV. SIMULATION

### A. Simulation setup

The network used for the simulation consists of four autonomous systems (AS), each of which has an edge router, and two core routers that are connected to edge routers as shown in Fig. 1. We consider two one-directional aggregate flows sharing a common bottleneck link, which is the link between two core routers and has capacity of $C$ [packets/sec]. Each AS contains many TCP senders/receivers ($N_i$=50) and one UDP sender/receiver.

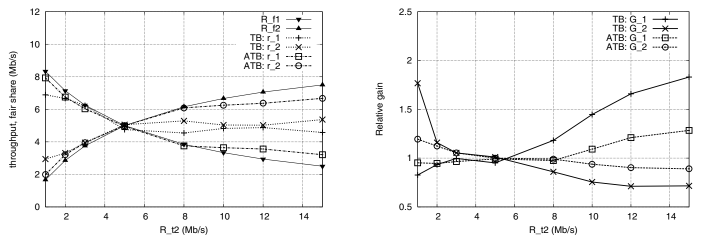
We use greedy FTP applications over a TCP connection and CBR (Constant Bit Rate) applications over UDP connections, whose sending rate is one tenth of the target rate, i.e., $0.1R_{t,i}$. The parameters for RIO are set to $(q_{min}^{out}, q_{max}^{out}, p_{max}^{out})$ = (10, 50, 0.1) for *OUT* packets and $(q_{min}^{in}, q_{max}^{in}, p_{max}^{in})$ = (50, 80, 0.02) for *IN* packets. Here, $q_{min}^{in/out}$ and $q_{max}^{in/out}$ denote the minimum and maximum thresholds for the *IN* and *OUT* packets, respectively, and $p_{max}^{in/out}$ is the maximum dropping probability for *IN*/*OUT* packets. Note that $p^{in}$ and $p^{out}$ do *not overlap*, because $q_{max}^{out} \leq q_{min}^{in}$. The propagation delays and the capacities of the links are shown in Fig. 1.

We set the thresholds $p_{min}^{th}$ and $p_{max}^{th}$ in (3), to 0.005 and 0.02, respectively. We set the interval of the timer that is used to update the target rate, $T_s$, to 20ms and the adjustment parameter in (2), $\delta$, to 0.0001.

### B. Simulation 1: Performance comparison with token bucket algorithm

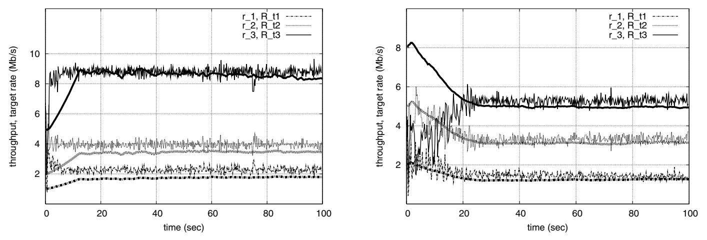
The first simulation is designed to show that the proposed algorithm can solve the problem of unfair bandwidth allocation. In the simulation, $R_{t,1}$ is fixed at 5Mb/s and $R_{t,2}$ varies from 1Mb/s to 15Mb/s.

Figure 4 compares the throughputs and relative gains of the TB algorithm and the adaptive token bucket (ATB) algorithm. As shown in Fig. 4(a), the throughput in the case of the ATB algorithm, $r_{i,ATB}$, is close to its fair share, whether the network is under-subscribed ($R_{t,2}$ <5Mb/s) or over-subscribed ($R_{t,2}$ >5Mb/s). However, if the TB algorithm is used, the performance is degraded significantly when the difference between $R_{t,1}$ and $R_{t,2}$ is large. Once $R_{t,2}$ exceeds 5Mb/s, $r_{1,TB}$ and $r_{2,TB}$ tends to share the bandwidth almost equally, even though $R_{t,2}$ is higher than $R_{t,1}$ by 10Mb/s, as shown in Fig. 4(a).



(a) Average throughput (b) Relative gain

Fig. 4. Performance comparison of the TB algorithm and the ATB algorithm: $R_{t,1}$ is fixed to 5Mb/s and $R_{t,2}$ is varying from 1Mb/s to 15Mb/s.



(a) under-subscription case (b) over-subscription case

Fig. 5. Target rates and throughputs of three aggregates when the ATB algorithm is used for marking scheme: bold lines and normal lines indicate the target rates and the throughputs, respectively.

On the other hand, when the ATB algorithm is used, $r_{1,ATB}$ and $r_{2,ATB}$ lie within 0.5Mb/s of their fair shares. Figure 4(b) shows that if the ATB algorithm is adopted, the relative gains of the two aggregates, $G_{1,ATB}$ and $G_{2,ATB}$, do not exceed 1.3 and do not fall below 0.85 for the entire range of $R_{t,2}$. In contrast to ATB, if TB is used, $G_{2,TB}$ is about 1.8 when $R_{t,2}$=1Mb/s, and $G_{1,TB}$ and $G_{2,TB}$ are about 1.8 and 0.7, respectively, when $R_{t,2}$=15Mb/s.

These simulation results confirm that the ATB algorithm greatly alleviates the problem of unfair bandwidth allocation.

### C. Simulation 2: Performance evaluation of the ATB algorithm

This simulation was performed in order to explain how the ATB algorithm adjusts its target rate according to the condition of network, and makes the throughput approach to its target rate. We consider a simulation scenario wherein three aggregates share a common bottleneck link. For the under-subscription case, the target rates are set to $\{R_{t,i}\}$=(1,2,5) Mb/s and $C$=15Mb/s. Similarly, for the over-subscription case, $\{R_{t,i}\}$ are set to (2,5,8) Mb/s and $C$=10Mb/s. Note that $\{R_{f,i}\}$=(1.875, 3.75, 9.375) Mb/s for the under-subscription case and $\{R_{f,i}\}$=(1.333, 3.333, 5.333) Mb/s for the over-subscription case.

Figure 5 shows the target rates and the throughputs of the three aggregates when the network is under-subscribed (Fig. 5(a)) and over-subscribed (Fig. 5(b)). These figures show that the ATB algorithm increases or decreases the target rates, in order that they approach the values corresponding to fair share of bandwidth allocation. Therefore, the ATB algorithm can allocate the bandwidth **proportionally** to the original target rates. Note that the target rates are adjusted using only a two-bit feedback signal, however this is not sufficient to match them perfectly to their fair shares. This limitation on the feedback information makes the target rate slightly smaller than the corresponding fair share, as shown in Fig. 5.

TABLE I

EFFECT OF ROUND-TRIP TIME AND THE NUMBER OF CONNECTIONS

Under-subscription case: $\{R_{t,i}\}$=(1,2,5) Mb/s, $C$=15Mb/s, and $\{R_{f,i}\}$=(1.875,3.75,9.375) Mb/s

| $\overline{T}_i$ [ms] | $N_i$ | $r_{i,TB}$ [Mb/s] | $r_{i,ATB}$ [Mb/s] | $\sum_i r_{i,TB}$ | $\sum_i r_{i,ATB}$ | $G_{i,TB}$ | $G_{i,ATB}$ |
|---|---|---|---|---|---|---|---|
| (100,100,100) | (50,50,50) | (3.03, 4.13, 7.48) | (2.22, 3.97, 8.74) | 14.64 | **14.93** | (1.62, 1.10, 0.80) | **(1.18, 1.06, 0.93)** |
| (100,100,100) | (30,50,100) | (2.22, 3.95, 8.53) | (2.00, 3.85, 9.06) | 14.70 | **14.92** | (1.18, 1.05, 0.91) | **(1.07, 1.03, 0.97)** |
| (100,100,100) | (100,50,30) | (4.21, 4.11, 6.31) | (2.49, 4.14, 8.30) | 14.63 | **14.93** | (2.25, 1.10, 0.67) | **(1.33, 1.11, 0.89)** |
| (80,100,150) | (50,50,50) | (3.05, 4.31, 7.26) | (2.26, 3.97, 8.73) | 14.61 | **14.94** | (1.63, 1.15, 0.77) | **(1.21, 1.06, 0.93)** |
| (150,100,80) | (50,50,50) | (2.90, 4.19, 7.66) | (2.20, 3.94, 8.79) | 14.75 | **14.95** | (1.55, 1.12, 0.82) | **(1.17, 1.05, 0.94)** |

Over-subscription case: $\{R_{t,i}\}$=(2,5,8) Mb/s, $C$=10Mb/s, and $\{R_{f,i}\}$=(1.333,3.333,5.333) Mb/s

| $\overline{T}_i$ [ms] | $N_i$ | $r_{i,TB}$ [Mb/s] | $r_{i,ATB}$ [Mb/s] | $\sum_i r_{i,TB}$ | $\sum_i r_{i,ATB}$ | $G_{i,TB}$ | $G_{i,ATB}$ |
|---|---|---|---|---|---|---|---|
| (100,100,100) | (50,50,50) | (2.04, 3.79, 3.97) | (1.61, 3.38, 4.95) | 9.79 | **9.94** | (1.53, 1.14, 0.74) | **(1.21, 1.02, 0.93)** |
| (100,100,100) | (30,50,100) | (1.65, 3.63, 4.43) | (1.53, 3.29, 5.09) | 9.71 | **9.91** | (1.24, 1.09, 0.83) | **(1.15, 0.99, 0.95)** |
| (100,100,100) | (100,50,30) | (2.43, 4.50, 2.95) | (1.81, 3.76, 4.39) | 9.88 | **9.96** | (1.83, 1.35, 0.55) | **(1.36, 1.13, 0.82)** |
| (80,100,150) | (50,50,50) | (2.29, 4.39, 3.14) | (1.78, 3.61, 4.57) | 9.82 | **9.96** | (1.72, 1.32, 0.59) | **(1.33, 1.08, 0.86)** |
| (150,100,80) | (50,50,50) | (1.88, 2.95, 5.04) | (1.58, 3.51, 4.87) | 9.88 | **9.96** | (1.41, 0.89, 0.95) | **(1.19, 1.05, 0.91)** |

### *D. Simulation 3: Effect of system parameters on the performance*

Up to now, we have focused on simulation scenarios wherein all aggregates are identical except for the target rates. However, the throughput of aggregates in DiffServ networks is influenced by RTT and by the number of flows within these aggregates, as well as by the target rates [16]. Let us denote $\overline{T}_i$ as the average round-trip time of flows belonging to the $i$th aggregate. In this simulation, by setting $\overline{T}_i$ and $N_i$ differently for each aggregate, we study the effect of system parameters such as $\overline{T}_i$ and $N_i$ on the performance of the ATB algorithm.

Table I lists the performance indexes, $r_i$ and $G_i$, and compares the performances of the TB algorithm and the ATB algorithm when $\overline{T}_i$ and $N_i$ change. We expect that when $N_1$ is larger than $N_3$, or when $\overline{T}_1$ is shorter than $\overline{T}_3$, $Agg_1$, which has a lower target rate, becomes more aggressive and gets much more bandwidth than its fair share. In Table I, when $\{N_i\}$=(100, 50, 30) for the under-subscription case, $G_{1,TB}$ is bigger than two and $G_{3,TB}$ is about two-thirds, while $G_{1,ATB}$ is around 1.3 and $G_{3,ATB}$ is around 0.9. Also, when $\{N_i\}$=(100, 50, 30) for the over-subscription case, $G_{3,TB}$ is about one half, which means that the third aggregate can only occupy one half of its fair share, however, this unfairness is significantly improved when the ATB algorithm is used, i.e., $G_{3,ATB}$ is larger than 0.8. Table I also shows that the TB algorithm is biased toward aggregates with a shorter round-trip time and that this bias is alleviated by the ATB algorithm. Moreover, we note that the total throughput of the three aggregates is increased slightly more in the case where the ATB algorithm is used than in the case where the TB algorithm is used.

## V. CONCLUSION

We studied the unfairness problem associated with bandwidth allocation in DiffServ networks by observing the throughput of TCP flows during simulations. Based on these observations, we proposed an adaptive token bucket algorithm to solve this unfairness problem. The main idea of the algorithm is to adjust the target rates according to feedback information, so that the network is close to being well-provisioned. We implemented this algorithm in a distributed manner using only two-bit feedback information conveyed in a packet header. Hence, the proposed algorithm does not require any measurements based on per-flow states. In this way, the aggregates can obtain their fair share of the bandwidth. The simulation results confirmed that the adaptive token bucket algorithm can solve the problem of unfair bandwidth allocation under various network conditions.

In future studies, we will analyze the cause of the unfairness problem in the conventional profile meter such as the token bucket algorithm, and we will study the performance of the ATB algorithm using multiple bottleneck scenarios.

## REFERENCES

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An architecture for differentiated services," *RFC 2475*, December 1998.

[2] V. Jacobson, K. Nichols, and K. Poduri, "An expedited forwarding phb," *RFC 2598*, June 1999.

[3] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured forwarding phb group," *RFC 2597*, June 1999.

[4] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service," *IEEE/ACM Trans. on Networking*, vol. 6, no. 4, pp. 362–373, 1998.

[5] Y. Chait, C. Hollot, V. Misra, D. Towsley, and H. Zhang, "Providing throughput differentiation for tcp flows using adaptive two color marking and multi-level AQM," in *Proceedings of IEEE INFOCOM*, 2001.

[6] W. Lin, R. Zheng, and J. Hou, "How to make assured service more assured," in *Proceedings of IEEE ICNP*, 1999.

[7] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Adaptive packet marking for maintaining end-to-end throughput in a differentiated-services internet," *IEEE/ACM Trans. on Networking*, vol. 7, no. 5, pp. 685–697, 1999.

[8] I. Yeom and A. L. N. Reddy, "Modeling TCP behavior in a differentiated services network," *IEEE/ACM Trans. on Networking*, vol. 9, no. 1, pp. 31–46, 2001.

[9] N. Malouch and Z. Liu, "Performance analysis of TCP with RIO routers," in *Proceedings of IEEE Globecom*, p. Nov., 2002.

[10] S.Sahu, P. Nain, C. Diot, V. Firoiu, and D. Towsley, "On achievable service differentiation with token bucket marking for TCP," in *Proceedings of ACM SIGMETRICS*, pp. 23–33, 2000.

[11] F. P. Kelly, A. Maulloo, and D. Tan, "Rate control for communication networks: Shadow prices, proportional fairness and stability," *Journal of Operations Research Society*, vol. 49, no. 3, pp. 237–252, 1998.

[12] H. Su and M. Atiquzzaman, "Itswtcm: A new aggregate marker to improve fairness in diffserv," in *Proceedings of IEEE Globecom*, pp. 1841–1846, 2001.

[13] M. Zhang, R. Wang, L. Peterson, and A. Krishnamurthy, "Probabilistic packet scheduling: Achieving proportional share bandwidth allocation for TCP flows," in *Proceedings of IEEE INFOCOM,*, June 2002.

[14] N. S. B. Nandy, P. Pieda, and J. Ethridge, "Intelligent traffic conditioners for assured forwarding based differentiated services networks," in *Proceedings of IFIP High Performance Networking HPN*, June 2000.

[15] K. Ramakrishnan and S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP." RFC 2481, 1999.

[16] J. Ibanez and K. Nichols, "Preliminary simulation evaluation of an assured service," *IETF Internet Draft*, August 1998.