Dynamic Fair Bandwidth Allocation for DiffServ Classes

Hideyuki Shimonishi† Ichinoshin Maki‡ Tutomu Murase† Masayuki Murata‡
† Networking Research Labs, NEC Corporation
‡ Graduate School of Engineering Science, Osaka University
simonisi@ccm.cl.nec.co.jp

Abstract

The Assured Forwarding Per Hop Behavior standardized by the IETF Differentiated Services working group provides four class-based differentiated IP services. In this service, however, unexpected service degradation may occur and differentiation among classes may be disordered if the network is designed to minimize over-provisioning or is under-provisioned. We therefore developed a packet scheduling scheme that dynamically allocates bandwidth to each class queue to guarantee the differentiation among classes under any traffic conditions. The scheme estimates the sum of CIRs (Committed Information Rates), i.e. rate of the packets having lowest drop preference, of active flows in each class and initially allocates the link bandwidth according to the sum of CIRs. It allocates the excess bandwidth by using a combination of CIR-proportional allocation and equal-share allocation. The equal share part enables that the flows in best effort class or the flows having zero CIRs can utilize minimum share of the bandwidth. Our scheme also introduces a scalable scheduling technique to improve fairness among flows in the same class. We evaluate the proposed scheme and show that it makes DiffServ operations fairer under any traffic conditions.

1 Introduction

The diverse service requirements of emerging Internet applications increase the need for flexible and scalable IP QoS (Quality of Service) schemes. The Differentiated Services (DiffServ) architecture [1] has a scalable QoS mechanism providing different service levels in the backbone networks. This architecture provides three classes of service: EF (Expedited Forwarding), AF (Assured Forwarding) [2], and BE (Best Effort). The EF class provides a QoS guarantee as tightly as the QoS obtained using private lines. Flows in the EF class are provisioned so well that their packets are never overloaded, and bandwidth can thus be guaranteed with minimum queuing delays. The AF class has four subclasses, AF1-AF4, each with its own forwarding priority and three levels of drop precedence. It provides a minimum bandwidth guarantee as well as efficient utilization of excess bandwidth.

The DiffServ architecture puts complicated functionality on the edge of the network domain and keeps the network core simple. Edge devices maintain all user traffic profiles and monitor all incoming packets to ensure that the traffic of individual users conforms to their profiles. In the AF service, to make effective use of excess bandwidth, edge devices allow more packets than specified in the profile to be injected as long as they are labeled with a mark specifying high drop preference. Core devices are responsible for forwarding in-profile packets, but the forwarding of out-of profile packets depends on the availability of resources. Core devices provide class-based aggregated flow treatment and maintain separate queues for each class so that a different forwarding priority can be assigned to each queue.

There are many policies that can be used in providing the AF service. One is to over-provision the network so that service quality can be guaranteed under any traffic conditions. Since network resources can be properly allocated according to the results of theoretical analyses, a CIR (Committed Information Rate), rate of the packets having lowest drop preference, can be guaranteed and end-to-end queuing delay can be predicted accurately. As well as the excessive resource allocation, the policy also needs route fixing techniques such as static routing or MPLS (Multi-Protocol Label Switching).

Another policy yields a more statistical AF service that maximizes statistical multiplexing gain. Resources needed to meet anticipated traffic demands are provided but over-provisioning is minimized so that network resources are utilized efficiently: the network may even be under-provisioned so that network cost thereby minimized. Some papers have pointed out, however, that this policy makes it possible for aggressive users to severely degrade the services for other users [4, 5]. Aggressive flows, such as non-TCP-friendly UDP flows, can obtain large amounts of excess bandwidth while the CIRs of other flows fail to be guaranteed. These papers also pointed out that this policy can also result in excess bandwidth being used unfairly when the network is over-provisioned. Non-TCP-friendly UDP flows can occupy all the excess bandwidth while TCP flows occupy none of it. And TCP flows with longer RTTs (Round Trip Times) will tend to use less excess bandwidth than those with shorter RTTs.

In this paper we focus on the fairness of the second AF service policy described above. We investigate the unfairness among flows in the same class as well as the unfair service differentiation among flows in different classes. For example, we investigate a case in which the number of AF4 flows temporarily increases beyond the number for which resources were allocated and the number of AF1 flows is less than the number for which resources were allocated. In this case, the CIRs of flows in AF4 will not be guaranteed while flows in AF1 can obtain large excess bandwidth. We therefore describe a new dynamic bandwidth allocation scheme for DiffServ compliant schedulers, one that guarantees the service differentiation among AF classes and the BE class in any traffic condition. Our scheme aims to allocate bandwidth in a way that the CIRs of all *active* flows are satisfied as much as possible, the excess bandwidth of a class is not reused by that class but is allocated to other classes that would otherwise be unable to guarantee their CIRs. The excess bandwidth is the bandwidth left after the CIRs of all flows are satisfied. Only when the CIRs of all flows are satisfied, the excess

bandwidth is shared using a combination of CIR-proportional allocation and equal-share allocation. The equal-share allocation intends that the flows in the BE class or AF flows having zero CIRs can capture the least bandwidth. The fairness for the excess bandwidth allocation can be weighted so that the flows in the higher classes can obtain more bandwidth.

This new scheme also employs a scalable scheduling technique [9] to improve fairness among flows in the same class. To reduce the interaction among flows in the same class, multiple queues are assigned to each class. The bandwidth allocated to the class is shared by these queues according to the sum of the CIRs and the numbers of flows in these queues. In the relatively slow edge routers, a lot of queues can be assigned to each class, and even high-speed backbone routers will be able to have several queues in a class. Note that the number of queues is not dependent on the number of flows, classes, ports, and so on but it is a design choice and depends on the queue management capability of a router. This means that flow aggregation level is controllable and thus fairness among flows is also controllable. To further improve fairness among flows aggregated in the same queue, the scheme identifies the flows having higher arrival rates and increases the drop preference for these flows.

2 Marker and scheduler

A marker implemented in DiffServ edge devices ensures that individual user flows conform to their profiles by setting a drop preference of each incoming packet. A TRTCM (Two Rate Three Color Marker) [7] setting three drop preference (DP) levels in AF service, it meters individual flows and marks their packets either green (DP=0), yellow (DP=1), or red (DP=2). A packet is marked red if it exceeds the PIR (Peak Information Rate) and is marked yellow if it exceeds the CIR but does not exceed the PIR. Otherwise, the packet is marked green.

A scheduler (or shaper) is implemented in all DiffServ-compliant devices to control packet discarding and forwarding. Generally, it is implemented as shown in Fig. 1. It maintains separate queues for each class, and a fixed amount of bandwidth is allocated to each class. Most implementations of AF Per Hop Behavior (PHB) use active queue management techniques based on RED (Random Early Detection) [3] to detect and respond to congestion early. RED uses four parameters: $min_{th}, max_{th}, max_p$, and w_q. Incoming packets are not dropped if the average queue length is less than min_{th}, and these packets are dropped with probability max_p if the average queue length is greater than max_{th}. Otherwise, packets are dropped with a probability proportional to the average queue length. The average queue length $\overline{Q_t}$ at time t is calculated by $\overline{Q_t} = w_q\overline{Q_{t-1}} + (1 - w_q)Q_t$, where Q_t is the instantaneous queue length at time t.

Since we used the TRTCM scheme, we used Multi-level RED (MRED) to deal with three drop preference levels. It is an extension of RED and has three parameter sets for three different drop preferences. Note that $DP2min_{th} < DP2max_{th} = DP1min_{th} < DP1max_{th} = DP0min_{th} < DP0max_{th}$ and $DP2max_p \geq DP1max_p \geq DP0max_p$ (Fig. 2).

3 Dynamic bandwidth allocation

3.1 Outline

The proposed scheme is illustrated in Fig. 3. After incoming packets are classified, the CIRs of active flows and the he number of active flows aggregated in one queue are estimated. Then,

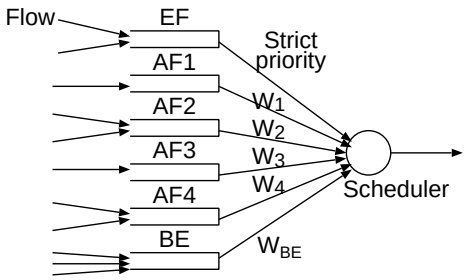

Figure 1: Example of a DiffServ scheduler.

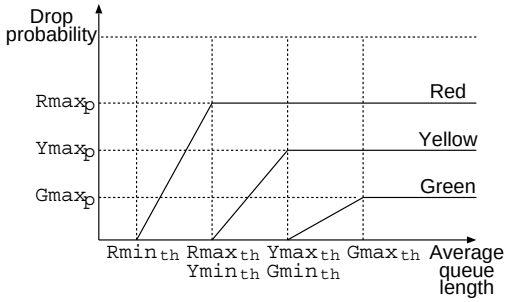

Figure 2: MRED scheme.

bandwidth is accordingly allocated as follows.

The link bandwidth is initially allocated to each class according to the estimated CIRs. When the sum of CIRs exceeds the link bandwidth (i.e., when the network is under-provisioned and therefore congested), the CIRs of all active flows cannot be guaranteed. In this case, rather than bandwidth being allocated in a way that satisfies the CIRs of higher classes at the cost of the lower classes, it is allocated so that all classes are penalized fairly. The allocation scheme is a simple design choice and both schemes can be easily implemented. If there is excess bandwidth, some is allocated to flows in proportion to their CIRs and the rest is allocated to each class in proportion to the number of flows in the class. The latter allocation ensures that the least bandwidth is provided for flows in the BE class and flows with a CIR of zero.

Multiple queues can be assigned to each class in order to improve the fairness among flows in the same class. The number of queues in a class is a design choice and depends on the queue management capability of the router.

3.2 Classification

When a class has just one queue (*single-queue case*), the class queue to which the packet belongs will be determined by exam-

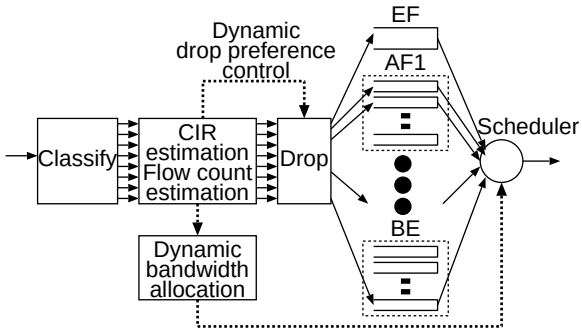

Figure 3: Proposed DiffServ scheduler.

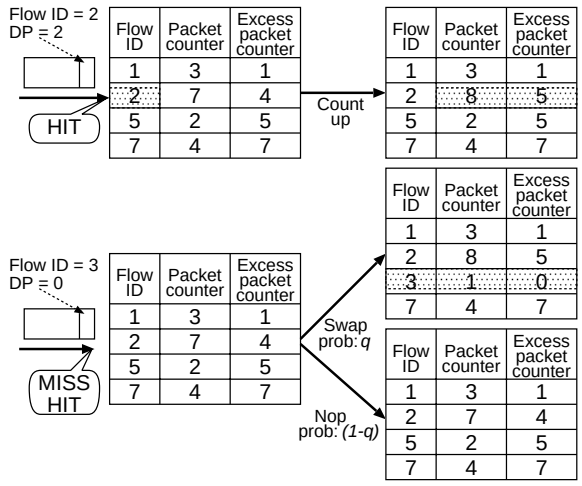

Figure 4: Example of zombie list behavior.

ining the TOS (Type Of Service) field in the IP header. When a class has multiple queues (*multiple-queue case*), the IP and TCP/UDP headers are further examined to determine a queue. Since the second classification is to guarantee that packets in the same flow are stored in the same queue, flow identification is not required and simple hashing schemes can be used.

3.3 Estimation of CIR and the number of flows

Since a packet is marked DP=0 when it conforms its CIR, the sum of the CIRs of the active flows is estimated by counting the number of packets having lowest drop precedence (DP=0). The estimation of the number of flows aggregated in a queue is based on the technique proposed by us [9]. It uses *zombie lists* (Fig. 4), which are prepared for each queue and are short histories of newly arrived flows. Each record in one of these lists consists of a flow ID, an arrived packet counter, and an excess packet counter. The arrived packet counter is the number of packets that arrived after the flow was registered in the zombie list, and the excess packet counter is the number of arrived packets having a high drop preference (DP=1,2,...). When a packet arrives at queue i, the zombie list Z_i corresponding to that queue is updated as follows.

- Use the flow ID of the input packet and search all entries in Z_i
  - If the flow ID of entry j matches that of the input packet, update the packet counter of entry j. Also update the excess packet counter of entry j unless the packet has the lowest drop precedence.
  - Otherwise, arbitrarily choose an entry j and, with probability q, swap the flow ID of entry j for that of the input packet and reset the counters of entry j.

The packet counter of entry j becomes maximum when the entry is replaced and the maximum counter value is proportional to the arrival rate. Therefore, the arrival rate of the replaced flow k in entry j is estimated by the following equation [9]:

$$R_k = (1-p)\frac{q}{M}(E_j - 1) \tag{1}$$

where R_k is the portion of the packets of flow k among all packets belonging to queue i, p is the probability that an input packet matches one of the entries in Z_i, and M is the number of entries in Z_i. E_j is the value in the arrived packet counter of entry j when the entry is replaced. Then the moving average of R_k is updated by the following equation:

$$R_{avg} = \{1 - \beta(\frac{E_j}{R_k})\}R_{avg} + \{\beta(\frac{E_j}{R_k})\}R_k \tag{2}$$

where $0 < \beta < 1$. The term R_k in this equation is weighted by $(\frac{E_j}{R_k})$ because flows with higher arrival rates are to be registered in a zombie list more frequently than other flows. Then number of flows N_i in queue i can be estimated as follows because $R_{avg} = \sum_k R_k/N_i$ ($\sum_k R_k = 1$):

$$N_i = \frac{1}{R_{avg}} \tag{3}$$

3.4 Bandwidth allocation

In the single-queue case, the link bandwidth is initially allocated to each class according to the sum of the estimated CIR. Then part of the excess bandwidth is allocated to each class according to the estimated number of flows and the rest of the bandwidth is allocated according to the CIRs. The higher classes should be allocated more of the excess bandwidth, so the scheduler uses weight parameters C_l ($C_0 \geq C_1 \geq C_2 \geq C_3 \geq 1$) for class AF$l$ and uses a weight parameter C_{BE} for the best effort class. The weight allocation in the scheduler is thus determined by the following equations:

$$B_{excess} = B_{link} - \sum_l CIR_l \tag{4}$$

$$W_l = \begin{cases} CIR_l + \alpha\frac{N_l C_l}{\sum_k N_k C_k} B_{excess}, & \text{if } l = AF1, \ldots, 4 \\ \alpha\frac{N_l C_l}{\sum_k N_k C_k} B_{excess}, & \text{if } l = BE \end{cases} \tag{5}$$

where $0 \leq \alpha \leq 1$. All the excess bandwidth is shared equally by the flows if α is 1 and is allocated in proportion to CIRs if α is 0.

In the multiple-queue case, these equations can be used to calculate the bandwidth allocation for each queue.

3.5 Packet-dropping

The scheme dynamically changes the drop preferences of the input packets in order to improve per-flow fairness among flows aggregated in the same queue. If the drop preference of the packet is not the lowest and the excess packet counter of the flow is more than the average of the counters in the same queue, the drop preference of the packet is increased by one unit.

4 Performance evaluation

4.1 Evaluation model

The performance was evaluated using the single-bottleneck link topology illustrated in Fig. 5. The capacity of all links are 150 Mbps, and the propagation delays of the bottleneck link and other links are respectively 1 ms and 0.1 ms. It was assumed that the edge routers are not congested and that they thus are responsible for marking but not for dropping. All terminals have an infinite amount of data to transmit and they use either TCP (RENO) or UDP (3 Mbps constant bit rate). The packet length is fixed to 500 bytes.

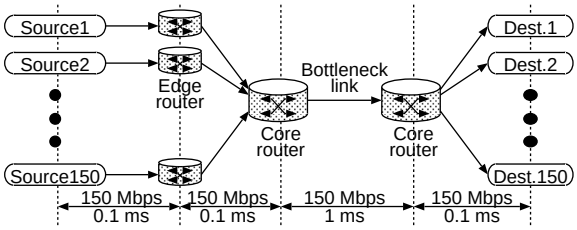

Figure 5: Network model.

Drop precedence	max_p	min_{th} [pkts]	max_{th} [pkts]
0	0.02	40	80
1	0.10	20	40
2	0.20	10	20
3	0.40	5	10

Table 1: MRED parameters.

The parameters setting for MRED are listed in Table 1. We introduce DP=3 for the packets whose DP is increased from DP=2. W_q is 0.0002 and the size of all queues is 200 packets. For all flows the CIR, CBS (Committed Burst Size), PIR, and PBS (Peak Burst Size) are respectively 1 Mbps, 16 packets, 1.2 Mbps, and 8 packets.

We used a Network Simulator 2.1b8a [8] in this evaluation and assumed that the network is provisioned so that there is a total of 30 flows in each class in the bottleneck link (all four AF classes and the BE class together). Therefore, in the fixed bandwidth allocation scheme, the allocated bandwidth for all classes is 30 Mbps, and all the excess bandwidth is used by AF4 flows and lower classes use excess bandwidth only when the AF4 flows cannot use all of it. In the proposed scheme, weights for excess bandwidth allocation for the AF4-1 and BE classes are respectively 1.3, 1.2, 1.1, 1.0, and 2. Half of the excess bandwidth is allocated in proportion to CIRs and the rest is allocated equally. That is, $\alpha = 0.5$.

4.2 Single-queue case

The average throughput for classes AF1-4 and BE in the single-queue case is shown for the fixed allocation in Figure 6 and is shown for the proposed scheme in Figure 8. To investigate the case where the actual traffic condition is different from the provisioned traffic condition, the number of AF1 flows is 10 and the number of AF4 flows varies from 10 to 100. The number of flows in other classes is the same as it is provisioned, i.e., 30 flows. In these figures, all flows use TCP.

When the fixed allocation scheme is used (Fig. 6), flows in class AF1 always use a large amount of bandwidth because the class is over-provisioned. Flows in class AF4 also use a large amount of bandwidth if the number of flows in this class is less than 30, otherwise these flows fails to achieve CIRs. This result shows that if the network is not accurately provisioned, there could be a case where a higher class fails to achieve CIRs while lower classes use excess bandwidth. On the other hand, when the proposed scheme is used (Fig. 8), the link bandwidth is allocated fairly according to the sum of the CIRs of active flows. When the number of AF4 flows is less than 80 (i.e., the sum of the CIRs of all classes is less than the link capacity), the CIRs of all classes are satisfied. Also, half of the excess bandwidth is shared in proportional to CIRs and the rest is shared in proportional to the weight, i.e., 1.3, 1.2, 1.1, 1.0, and 2. When the number of AF4 flows exceeds 80, all AF classes are equally penalized rather than rather than specific class is heavily penalized. In this case, BE can obtain no bandwidth since there is no excess bandwidth.

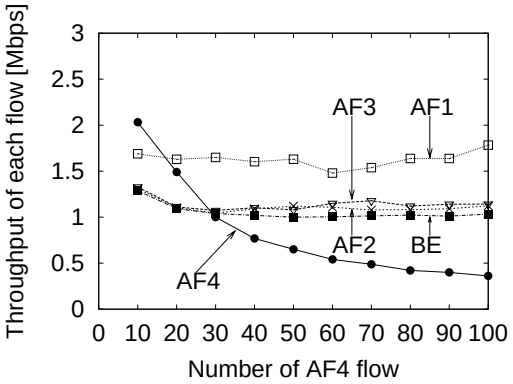

Figure 6: Average throughput evaluation for fixed allocation in the single-queue case: number of AF4-AF1, BE flows=10-100, 30, 30, 10, and 30. No UDP flows.

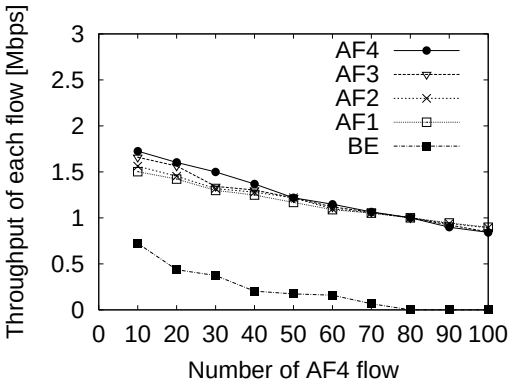

Figure 7: Average throughput evaluation for the proposed scheme in the single-queue case: number of AF4-AF1, BE flows=10-100, 30, 30, 10, and 30. No UDP flows.

To investigate unfairness between TCP flows and UDP flows aggregated into the same class, the average throughput of TCP flows and UDP flows is shown in Figure 7 for the fixed allocation scheme and is shown in Figure 9 for the proposed scheme. In these figures, the number of AF4 flows varies from 10 to 100 and all other classes have 30 flows. Half of the flows use TCP and other flows use UDP. In the fixed allocation scheme, UDP flows use most of the excess bandwidth and TCP flows are penalized more than UDP flows. Also, TCP flows in AF4 are severely penalized. The proposed scheme can improve the fairness between TCP flows and UDP flows. The CIRs of all flows can be satisfied when the number of AF4 flows is less than 60, and the excess bandwidth usage of UDP flows and the penalty for TCP flows are minimized when the number of AF4 flows is greater than 60.

4.3 Multiple-queue case

The proposed scheme was compared with the fixed allocation scheme in a multiple-queue case where there are four queues for each class (thus the total number of the queues is 20). Flows in a same class are assigned to one of the four queues by using 5-tuple identifier and a CRC-16 hash function. To investigate the fairness improvement by the multiple queue allocation, the average throughput of TCP flows and UDP flows are compared in Figure 10 and Figure 11. The number of AF4 flows is 10 in

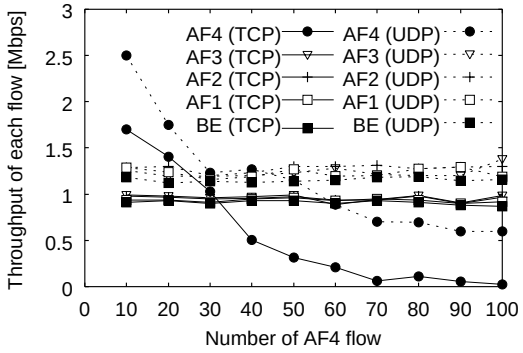

Figure 8: Throughput comparison of TCP flows and UDP flows for fixed allocation in the single-queue case: number of AF4-AF1, BE flows=10-100[5-50UDPs], 30[15UDPs], 30[15UDPs], 30[15UDPs], and 30[15UDPs].

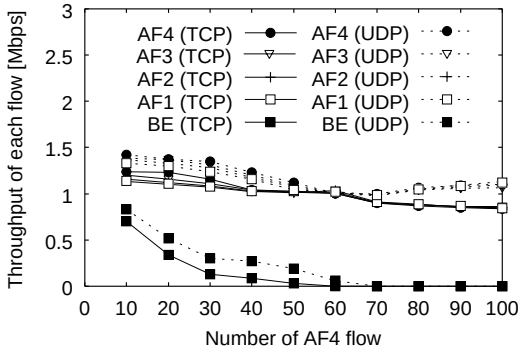

Figure 9: Throughput comparison of TCP flows and UDP flows for the proposed scheme in the single-queue case: number of AF4-AF1, BE flows=10-100[5-50UDPs], 30[15UDPs], 30[15UDPs], 30[15UDPs], and 30[15UDPs].

Fig. 11 and 90 in Fig. 11, and all other classes have 30 flows.

When the number of AF4 flows is 10, in the fixed allocation, all the excess bandwidth is captured by AF4 class and TCP flows in other classes fails to achieve their CIRs, while UDP flows can use the bandwidth more than CIR. The proposed scheme, however, improves the fairness between TCP flows and UDP flows, and all TCP flows satisfy their CIRs. In the multiple-queue case, the fairness between TCP and UDP is further improved. When the number of AF4 flows is 90, the sum of the CIRs exceeds the line bandwidth. In the fixed allocation, flows in AF4 class are severely penalized while UDP flows in other classes can capture large bandwidth. The proposed scheme improves both fairness among classes and fairness among flows.

5 Conclusion

This paper described a new packet scheduling scheme for the DiffServ AF classes and BE class. It improves both service differentiation among classes and fairness among flows.

We showed that unexpected service degradation may occur and differentiation among classes may be disordered if the network is designed to minimize over-provisioning or is under-provisioned. The proposed packet scheduling scheme estimates the sum of CIRs and the number of flows in a queue, then dynamically allocates bandwidth on the basis of CIR-proportional allocation and equal-share allocation. The scheme also introduces an aggregated-flow-based scalable packet scheduling technique to improve fairness among flows in a same class. We

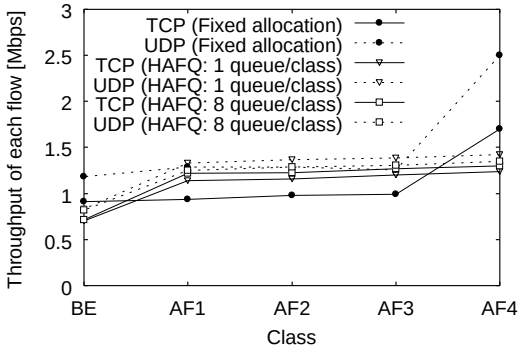

Figure 10: Throughput comparison of TCP flows and UDP flows: number of AF4-AF1, BE flows=10[5UDPs], 30[15UDPs], 30[15UDPs], 30[15UDPs], and 30[15UDPs].

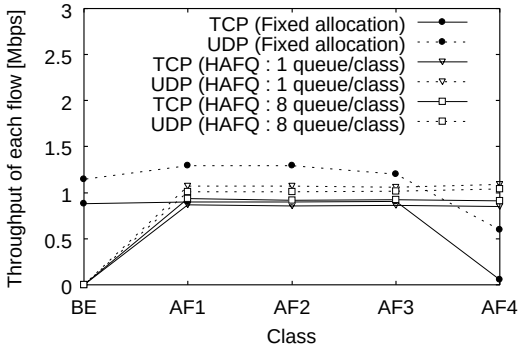

Figure 11: Throughput comparison of TCP flows and UDP flows: number of AF4-AF1, BE flows=90[45UDPs], 30[15UDPs], 30[15UDPs], 30[15UDPs], and 30[15UDPs].

showed numerical examples demonstrating that the proposed scheme can provide fair service among classes according to the estimated CIRs of active flows and can also share the excess bandwidth fairly. We also showed that the proposed scheme can improve fairness among TCP and UDP flows served in the same class.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", IETF RFC 2475, Dec. 1998.

[2] J. Heinanen, F. Baker, W. Weiss, and J. Wroclawski, "Assured Forwarding PHB Group", IEEE RFC 2597, June 1999.

[3] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Trans. on. Networking, Vol.1, Aug. 1992.9.

[4] N. Seddigh, B. Nandy, and P. Pieda, "Bandwidth Assurance Issues fot TCP flows in a Differentiated Service Network", *in Proc. of* IEEE GLOBECOM'99, Dec.1999.

[5] A. Kolarov, "Study of the TCP/UDP Fairness Issue for the Assured Forwarding Per Hop Behavior in Differentiated Services Networks", IEEE HPSR'2001, May 2001.

[6] M. Goyal, A. Durresi, P. Misra, C. Liu, and R. Jain, "Effect of Number of Drop Precedences in Assured Forwarding", in Proc. of IEEE GLOBECOM'99, Dec. 1999.

[7] J. Heinanen and R. Guerin, "A Two Rate Three Marker", IETF RFC-2698, Dec. 1999.

[8] "NS Simulator, version ns-2.1b8a", Available from http://www-mash.cs.berkeley.edu/ns

[9] I. Maki, H. Shimonishi, T. Murase, M. Murata, and H. Miyahara, "Hierarchically Aggregated Fair Queuing (HAFQ) for Per-flow bandwidth Allocation in High-speed Networks", *Submitted to ICC2001*.