Providing Enhanced Differentiated Services for Real-Time Traffic in the Internet

Tamrat Bayle[1], Reiji Aibara[2], and Kouji Nishimura[2]

[1] Department of Information Engineering,
Graduate School of Engineering, Hiroshima University,
1-4-1 Kagamiyama, Higashi-Hiroshima, 739-8527, Japan
tamrat@hiroshima-u.ac.jp
[2] Information Media Center, Hiroshima University,
1-4-2 Kagamiyama, Higashi-Hiroshima, 739-8511, Japan
{ray, kouji}@hiroshima-u.ac.jp

Abstract. The Differentiated Services (DiffServ) architecture offers a scalable alternative to provide Quality of Service (QoS) guarantees for performance-sensitive applications in the Internet. Within the DiffServ framework, efficient traffic scheduling mechanism is a key component to ensure such QoS guarantees. In this paper, scheduling algorithm called Enhanced Weighted Fair Queueing (EWFQ) is proposed that enables fair bandwidth sharing while supporting tight bounds on end-to-end delay for real-time traffic such as voice over IP (VoIP) in DiffServ networks. EWFQ allows to create service classes and assign proportional weights to such classes efficiently according to their resource requirements. The results from the simulation studies show that the mechanism is able to ensure both the required end-to-end delay bounds and bandwidth fairness based on the specified service weights. Besides, our scheme has lower implementation complexity, along with scalability to accommodate the growing traffic flows in the Internet backbone.

Keywords: Internet QoS, DiffServ, Scheduling Algorithm, VoIP

1 Introduction

The current Internet architecture provides only best effort service. Such best effort service is adequate for traditional Internet applications like e-mail, web browsing or file transfers. However, the new emerging real-time applications, such as voice over IP (VoIP), and multimedia conferencing, are sensitive to delay and delay variation, and require bandwidth guarantees. Consequently, the need to equip the Internet infrastructure with mechanisms to enable Quality of Service (QoS) is critical.

The research efforts by the Internet Engineering Task Force (IETF) to enable end-to-end QoS over IP networks have led to the design of two different architectures: the *Integrated Services* (IntServ) architecture [1] and more recently, the *Differentiated Services* (DiffServ) architecture [2], which although different,

A. Marshall and N. Agoulmine: MMNS 2003, LNCS 2839, pp. 272–285, 2003.

support services that go beyond the best effort service. Furthermore, due to the scalability limitations of the IntServ model for deployment in network backbones, the DiffServ architecture defines a scalable framework for providing QoS in the Internet. The DiffServ approach addresses the scaling concerns by reducing all traffic flows only into a small number of traffic aggregations, each with a different set of QoS requirements. [3] describes a DiffServ *per-hop behavior* (PHB) called *expedited forwarding* (EF) intended for use in building a scalable, low loss, low latency, low jitter, assured bandwidth, end-to-end service that appears to the endpoints like an unshared, point-to-point connection. Typically real-time, and mission-critical applications require this service. On the other hand, *assured forwarding* (AF) PHB [4] is suggested for applications that require a better reliability than the best-effort service. However, as a QoS control technology, DiffServ involves different traffic management mechanisms to provide the required multiple levels of services [5]. In this context, therefore, the packet scheduler is one of the key components of DiffServ networks, which plays important roles in service isolation because it actually gives different services to different traffic classes.

We propose a simple and efficient scheduling mechanism for DiffServ based Internet that enables fair bandwidth sharing while supporting better bounds on end-to-end delay for QoS-sensitive applications such as VoIP. It is based on our previous work [6] and incorporates the best characteristics of some existing approaches, notably the rate-based packet fair queueing algorithms [7,13, 14]. However, our scheme considers only serving a greatly reduced number of service classes rather than potentially a huge number of flows, and so lowers significantly its implementation complexity, while maintaining the robustness properties, required for end-to-end delay bounds and bandwidth fairness, of previously proposed alternatives. We call this algorithm *Enhanced WFQ* (EWFQ), as it depicts the capacity to adapt efficiently to Differentiated services environment.

The rest of the paper is organized as follows. Section 2 briefly reviews existing alternatives, and explains both their strengths and limitations. Section 3 discusses a new scheduling mechanism that lowers the implementation complexity, and ensures tight delay bounds for real-time packets within EF service class and bandwidth fairness among all traffic classes. Section 4 analyzes the performance of the proposed mechanism using simulations. The same Section, first describes the network topology and traffic models considered and some of the assumptions made for evaluation of the mechanism, and then presents the simulation results and discusses their implications. Finally, Section 5 concludes the paper with a brief summary and outlines directions for future work.

2 Background and Related Work

Scheduling disciplines are the key to fairly share a limited amount of network resources and provide QoS for performance-sensitive applications. A queue scheduling discipline allows to manage access to such a fixed amount of output link bandwidth by selecting the next packet that is transmitted on output port.

So, when there is congestion, then scheduling is the mechanism used to differentiate traffic and provide the required QoS. But the key challenge is to find an appropriate scheduling algorithm, especially with desirable properties of *low end-to-end delay bound*, *efficiency (low complexity of implementation)*, *fairness* and *scalability*.

There have been many research work done regarding this issue, each attempting to find the right balance among complexity, control, scalability, and fairness. For instance, priority queuing (PQ) [8] is the basis for a class of queue scheduling algorithms that are designed to provide a relatively simple method of supporting differentiated service classes. There is, however, a potential undesirable side effect with this scheme. Traffic with low-priority can become starved if there are a large number of high-priority classes. Moreover, PQ provides just a delay guarantee but no bandwidth guarantee. On the other hand, deficit round-robin (DRR) scheduling algorithm [10] addresses the limitations of the PQ mechanism by accurately supporting the weighted fair distribution of bandwidth when servicing queues that contain variable-length packets. DRR, though, does not provide the required good end-to-end delay bounds as other queue scheduling disciplines, which are described below. DRR provides a bandwidth guarantee but no delay guarantee.

WFQ [7] also known as Packetized Generalized Processor Sharing (PGPS) is more appropriate queue scheduling approach for applications with variable-size packets. WFQ supports bounded delay and fair bandwidth distribution for variable-length packets by approximating the ideal generalized processor sharing (GPS) system [9]. It does so by time-stamping each arriving packet with a *finish time*, the expected completion time of the packet if it were scheduled under the ideal GPS scheduler. However, WFQ implements a complex algorithm that requires the maintenance of a significant amount of per-packet state and iterative scans of state on each packet arrival and departure. Such computational complexity impacts the scalability of WFQ when attempting to support a very large number of flows on high-speed networks, which simply turns out not to be practical to implement it. As a result, many variants of WFQ [11,12,13,14] have been proposed with different trade-offs. Among the well-known variants, the worst case fair weighted fair queueing (WF^2Q) [13], and worst-case fair weighted fair queueing plus (WF^2Q+) [14] achieve tight delay bounds and worst-case fairness properties. While WFQ uses only *finish times* of packets in the GPS system, WF^2Q uses both the *start* and *finish times* of packets to achieve a more accurate emulation of a GPS system to provide improved worst-case fairness. However, WF^2Q has the same computational complexity as WFQ. On the other hand, WF^2Q+ is an enhancement to WF^2Q, and implements a new virtual time function that results in lower complexity. Even though both schemes are fair in the *worst-case* sense and tend to have low delay, they were not designed to provide service differentiation among classes in the context of DiffServ networks.

Non of the scheduling disciplines described above, however, avoid the maintenance of either per-packet or per-flow state somewhere in the network or oth-

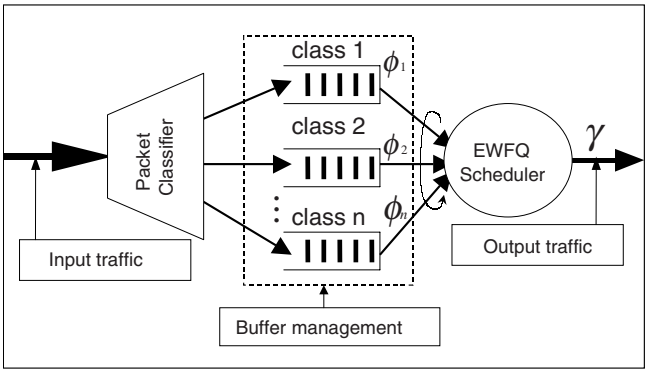

Fig. 1. Scheduler based on EWFQ

erwise dealing only with flows instead of with differentiated service classes. They significantly affect the scalability of the Internet backbone.

Therefore, by exploiting the DiffServ architecture, and use of the *start* and *finish* times for a packet only at the head of each of the active service class queues, the approach used in this paper goes a step further to reduce the size of huge traffic flows into very few service classes, as well as reducing the computational complexity, with respect to system virtual time computation, required for its implementation in high-speed backbone routers when compared with the above approaches.

3 Proposed Scheduler: EWFQ

In this section, we propose a scheduling mechanism called EWFQ for real-time IP traffic in DiffServ Networks. EWFQ is an improved mechanism in such a way that not only controlled bandwidth sharing and tight delay bounds are supported but also simplifications of implementation complexity are offered in the context of DiffServ environment.

EWFQ algorithm requires the maintenance of only aggregate state of very few traffic classes in the DiffServ networks, instead of a huge number of flows. In our approach, flows are aggregated and such aggregated flows are mapped into separate queues with different weights corresponding to service classes. Then the service time for a packet only at the head of the queue of active service class is calculated. Also, the sorting to transmit the next packet is done only among the head of very few active QoS classes. This implies that the complexities associated with EWFQ scheduler both for computing the system virtual time (for tagging with virtual finish time) and maintaining the set of eligible classes sorted by virtual finish times depend only on the number of supported service classes, which typically are much smaller than the number of sessions or flows. This simplification with our approach greatly reduces the computational complexity that is attached inherently with other approaches.

EWFQ Algorithm

```
/* a new packet P of class i arrives */
FOR i ← index of class i's queue that will hold new P
DO
BEGIN Enqueue(i, P)
   if (queue[i] == empty) {
     enqueue(P, queue[i]);
     /* compute starting/finishing times for */
     /* packet at the head of class i queue */
     S[i] ← max(V, F[i]);
     F[i] ← S[i] + L_i^P / φ_i;
     /* update system virtual time */
     V ← max(min(S[j])_{j∈B}, V);
   } else
     /* append packet to end of class i queue */
     enqueue(P, queue[i]);
END Enqueue

BEGIN Dequeue(i)
     /* dequeue and transmit (tx) the head packet */
     P_tx ← dequeue(queue[i]);
     send(P_tx);
     /* get the next head packet from the same class i */
     /* queue and compute starting and finishing times */
     P_next ← getfromhead(queue[i], head);
     if (P_next) {
       S[i] ← F[i];
       F[i] ← S[i] + L_i^{P_next} / φ_i;
     } else
     /* update system virtual time */
     V ← max(min(S[j])_{j∈B}, (V + L_i^{P_tx} / Σ_{j∈B} φ_j));
END Dequeue
```

Fig. 2. Enqueue and Dequeue Pseudocode for EWFQ

Consider n traffic classes, each class i, $i = 1, ..., n$, assigned to a *separate queue*, is associated with weight ϕ_i, and the link capacity is shared among all active classes in direct proportion to their weights, such that the sum of the weights of all classes is no larger than a predefined value γ. That means, if we consider Fig. 1, in which there are n service classes, then

$$\phi_1 + \phi_2 + ... + \phi_n \leq \gamma \ , i = 1, ..., N \ . \tag{1}$$

Here, the weight of the class specifies a relative share of how much of the capacity of the output link the class is entitled to receive. Furthermore, each

class i is associated with two variables $S[i]$ and $F[i]$ that represent, respectively, the starting and the finishing times corresponding to the packet at the head of the queue for a particular class i. Finally, a global variable V, called system virtual time, is associated to the system. The EWFQ can be then described briefly as follows: First initialize variables $S[i]$, and $F[i]$ for all classes i, and V to *0*, and other variables, such as ϕ_i and n to their respective values. Then proceed with *Enqueue*, and *Dequeue* operations as described below.

Enqueue: The *Enqueue* operation is called whenever a new packet of class i arrives. According to Fig. 2, the scheduler visits each queue and when a packet P of length L for class i arrives at its queue i, it executes the first part of the pseudocode. The function first checks whether class i just becomes backlogged[1] from the idle state. If the case is a transition from the idle state, it first places the newly arriving packet into the head of its corresponding class queue. Then it involves computing the starting $S[i]$ and finishing $F[i]$ times, as well as updating the system virtual time V for packet at the head of queue of class i, in this case, which is the newly arriving packet. Otherwise, if class i queue were previously backlogged, i.e., non-empty, it only appends the newly arriving packet to the end of queue of class i. In updating the system virtual time, $min(S[j])_{j \in B}$ represents the minimum of starting times among all backlogged classes.

Dequeue: The Dequeue function is the core of the algorithm that schedules packets from the queues corresponding to different service classes. Consider the set of all backlogged traffic classes B, such that their *starting times* are no larger than the system virtual time V, i.e., $S[j] \leq V$, for any class j in B, and form a set of *eligible* classes E for service. The algorithm then selects class i among the set of classes in E that has the *smallest finishing time*. Accordingly, the algorithm dequeues the *head packet* from this class queue and denotes it by P_{tx} for transmission. Finally, it proceeds to execute the remaining second part of the pseudocode in Fig. 2. The $getfromhead()$ function gets the next available packet P_{next} from the head of the same class i's queue whose head is passed as a parameter to the function. Then $S[i]$ and $F[i]$ are computed for the class. Otherwise, if $P_{next} \equiv NULL$, the system virtual time V will be updated.

In Eq. (1), if $\gamma = C$, where C represents the link capacity, then the weight of a class represents the minimum bandwidth that the class is guaranteed to receive. When the traffic class i is constrained by a token bucket (ρ_i, σ_i), where ρ_i is the average token generation rate, and σ_i is the token bucket depth, then class i is guaranteed to obtain a minimum fair service rate shown in Eq. (2).

$$r_i = \begin{cases} \left[\frac{\phi_i}{\sum_{j \in B} \phi_j} \right] \gamma & \forall i \in B \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

[1] A class is called backlogged if it has at least one packet in the queue waiting for transmission on the output link, or idle otherwise.

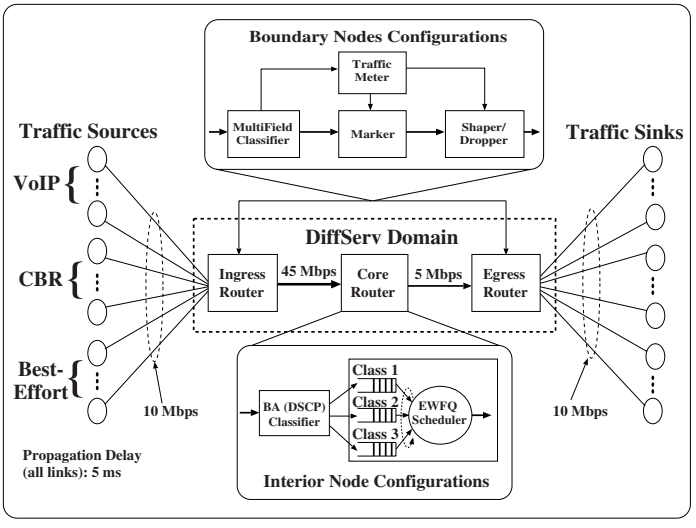

Fig. 3. DiffServ Network Model for Simulations

4 Performance Evaluation

The EWFQ algorithm is evaluated using a network simulation test-bed. We implemented the algorithm in the most popular ns-2 network simulator [16], and then validate its functionality and effectiveness with several simulation results. The key aspect of the experiment is to evaluate EWFQ scheme on its assurance of bounds on delay and jitter, as well as the minimum guaranteed bandwidth for the class that was given high priority, while equally observing its fair allocation of link bandwidth to other low priority service classes. These performance characteristics enable to determine whether the suggested scheme is fair and efficient, and can support VoIP applications in DiffServ networks in order to achieve an acceptable voice quality.

Note that the implementation of different PHBs is achieved through a combination of a variety of packet classification, packet marking, traffic conditioning, and buffer management mechanisms. These are on top of our packet scheduling mechanism, and thus, we make use of all the DiffServ components in our modified ns-2 to implement both EF and AF PHBs. Also note that the scheduling mechanism analyzed in this section assumes that the input traffic for both EF and AF is rate regulated [2] at the ingress hop using traffic policing technique. Thus, the proposed scheduler does not provide performance guarantees if the input traffic is not conditioned according to some service level agreements (SLAs). Another assumption is that the effect on high priority traffic should be minimal or negligible as the medium (AF) and low priority traffic classes are varied.

[2] This rate-limiting functionality is achieved using a token bucket scheme that regulates class i traffic with (σ_i, ρ_i) model in the DiffServ network we consider.

Table 1. Classifying traffic packets into their service Classes (1^{st} scenario)

Traffic Types	Mapped Classes	No. Flows per Class	Service (PHBs)	Relative Weights
VoIP	Class 1	100	EF	$\phi_1 = 60\%$
CBR (data)	Class 2	5	AF	$\phi_2 = 30\%$
FTP	Class 3	5	BE	$\phi_3 = 10\%$

4.1 Simulation Network Model

To evaluate the effectiveness of the algorithm through simulations, we use the simple network model as depicted in Fig. 3. In this model, we consider a network connection between six computers over a single-domain route in which the network connection passes through three DiffServ routers.

4.2 Traffic Model

In order to perform more realistic simulations with a variety of traffic behaviors, we consider three types of traffic mixes as follows: voice traffic, constant-bit-rate (CBR) data over UDP, and FTP data over TCP. The voice traffic is assumed to be an exponential distributed on-off traffic of two states, speaking and silence [15], and consists of 100 flows; each characterized by a packet size of 84 bytes, burst time 350 ms, idle time 650 ms, and peak rate of 64 kbps during on period. Therefore, the average sending rate for each voice source is 22.4 kbps, which comprises the total voice traffic rate about 2.24 Mbps. Our aim is to emulate a voice traffic over differentiated IP networks so that we can use this as VoIP traffic model to be treated with DiffServ EF PHB. In fact, the characterization depends on the coding system utilized, but for our simulations purpose we use such a simple voice traffic model to simulate VoIP. To describe both the second, and third traffic classes, we consider two different scenarios based on different changes in the traffic types and their sending rate behaviors.

Scenario 1. In the first scenario, we imagine the CBR traffic is sending at a rate more than its subscription rate (oversubscription case), and the background traffic class comprises 5 FTP sessions. And thus, the second class of traffic consists of 5 CBR sessions, which can be treated with AF PHB, and the third class, which represents best effort (BE) background traffic, consists of a set of 5 FTP connections. For FTP traffic transportation, we consider TCP Reno [17], which is widely used in today's Internet. Here, the main objective is to observe the ability of EWFQ mechanism how it can satisfy the delay, and average bandwidth requirements of the traffic in the first class (voice traffic) by protecting it from other misbehaving traffic classes. In this case, the CBR source is sending traffic at an average rate of 4 Mbps to be served with AF treatment. Imagine the bandwidth of the bottleneck link is 5 Mbps.

Furthermore, packets from these classes of traffic are scheduled using EWFQ policy with relative weights of $\phi_1 = 60\%$, $\phi_2 = 30\%$, and $\phi_3 = 10\%$, to represent

Table 2. Classifying traffic packets into their service Classes (2^{nd} scenario)

Traffic Types	Mapped Classes	No. Flows per Class	Service (PHBs)	Relative Weights
VoIP	Class 1	100	EF	$\phi_1 = 60\%$
CBR (data)	Class 2	5	AF	$\phi_2 = 30\%$
Self-Similar	Class 3	30	BE	$\phi_3 = 10\%$

DiffServ PHBs of EF, AF, and BE, respectively. Table 1 shows the summary of the traffic mapping into their respective classes, and the corresponding PHBs treatments for the first scenario. EWFQ was setup at the hot-spot link between the Core and Egress routers, according to Fig. 3. That means, the three service classes share the same bottleneck link of capacity 5 Mbps. To isolate the three types of service classes, each router uses three separate physical output queues of each size of 100 packets scheduled with EWFQ policy. Then, with this scenario, we carried out several tests, and measured the results of throughput, latency, jitter and packet loss for all traffic classes to ensure that the oversubscription by medium priority (AF) class should not affect the high priority class but only the low priority class.

Scenario 2. Table 2 shows the summary of the traffic mapping into their respective classes, and PHBs treatments for the second case. In this scenario, as a background best-effort traffic, we use 30 Pareto On/Off sources, each with an average rate of 100 kbps to generate an aggregate traffic rate of 3 Mbps. Traffic characterization according to a Pareto ON/OFF distribution has been proved to be self-similar in nature [18]. Such traffic modelling is widely considered to describe well the selfsimilarity (burstiness) nature of the Internet traffic. Moreover, the total CBR traffic rate was reduced from 4 Mbps into 1.5 Mbps (still 5 CBR sessions). Also, in order to reduce the delay experienced by all packets because of the queue depth, the three queue sizes are set to be 50 packets.

4.3 Simulation Results and Discussion

In this section, we discuss the results of the simulation described above by showing the benefits of the proposed scheduler in DiffServ network environments. We record the packet loss, throughput, latency and jitter for each class, as these are vital performance metrics for supporting real-time voice traffic over IP networks.

Table 3 depicts the summary of packets statistics for each DSCP $\rightarrow$ Class match for the first simulation scenario. Here, the packets for each class are identified by two differentiated services code points (DSCPs), signifying in profile, and out of profile packets, according to the predefined SLAs. So, this enables, first to classify marked packets into their corresponding classes, and second, within each service class, to identify the drop precedences for the buffer management mechanisms.

Table 3. Summary of Packets Statistics for each DSCP → Class match (1^{st} scenario)

DSCP Mapping	Total Pkts Received	Total Pkts Sent	Total Pkts Drops	Pkts Loss Rate (%)
00 → BE	5415	5101	314	6
01 → BE	86	65	21	24
10 → EF	197300	197300	0	0
11 → EF	0	0	0	0
20 → AF	29493	15493	14000	48
21 → AF	0	0	0	0

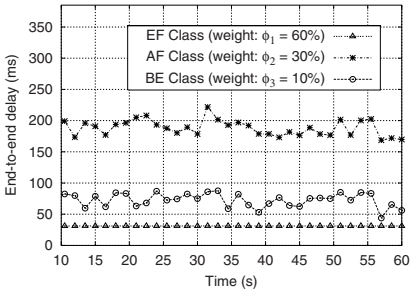

Fig. 4. Average end-to-end delay (1^{st} scenario)

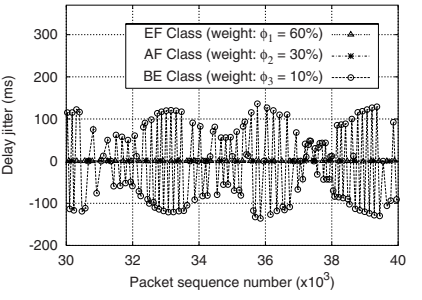

Fig. 5. Inter-packet delay jitter (1^{st} scenario)

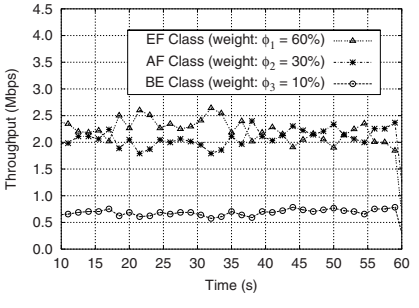

Fig. 6. Bottleneck link bandwidth (1^{st} scenario)

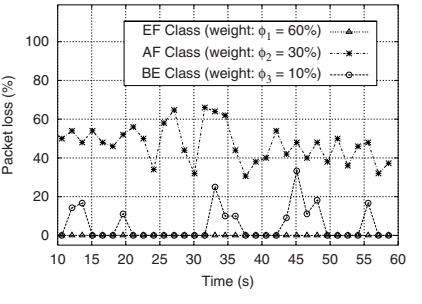

Fig. 7. Packet loss at the bottleneck (1^{st} scenario)

For the first scenario, the results for different performance metrics are shown in Fig. 4 through Fig. 7. Figure 4 and Fig. 5 present the end-to-end packet delay and jitter of each class, respectively. From these simulation results, we observe that voice traffic, which is classified as high priority class (Class 1) for EF PHB treatment with EWFQ, receives much tight delay bound, while its jitter is almost insignificantly small. Its request for bandwidth is also fully satisfied compared with medium (Class 2) and low priority (Class 3) traffic classes. This

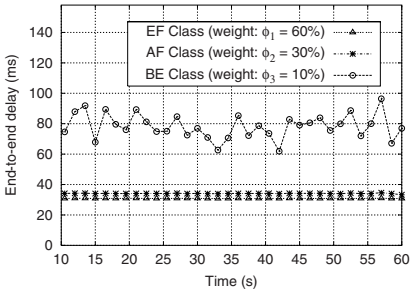

Fig. 8. Average end-to-end delay (2^{nd} scenario)

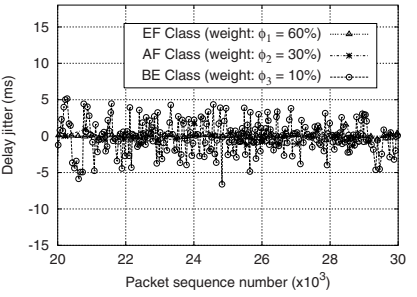

Fig. 9. Inter-packet delay jitter (2^{nd} scenario)

is true even when the rates of the background best effort traffic is increased. For example, if the weight ϕ_1 is set to be more bigger, and at the same time the voice traffic class has many more sessions to establish, then the performance of other classes is degraded significantly due to the delay in their respective queues. This is because the scheduler visits much often the queue for high priority class for it has a big weight assigned by EWFQ.

As one can see clearly from Fig. 4 and Fig. 7, the delay and packet loss for AF class are rather high compared to the best-effort class. There are two obvious factors that can be causing such behavior. First, the background traffic consists of a set of TCP flows, and thus react to congestion by shrinking their flow-control windows and sending less packets to reduce the packet loss. In fact, as it can been seen from Fig. 6, and Fig. 4, this behavior of TCP traffic results in lower bandwidth and poor response times, respectively. Second, and more important is that the traffic destined to AF class is sending with a rate much higher than the subscribed rate, 4 Mbps. As a result, packets from this class suffer much from the combination of both packet loss and longer delay in the queue.

On the other hand, the real-time packets marked for EF treatment are forwarded with very low end-to-end delay (about 30 ms), minimal jitter and no packet loss compared with packets marked for other service classes. This holds true as long as it complies with the SLA of the network. We also observe that it is difficult to create a wide range of influence on the targeted service classes when the background best-effort traffic consists of a set of TCP flows. This is, as described above, partly because of the TCP behavior itself, and partly there are several other parameters needed to be tuned, especially with RIO (RED with In and Out) buffer management mechanism that makes it extremely difficult to generate the desired effect.

For the second scenario, Fig. 8 through Fig. 11 present the end-to-end delay, jitter, throughput, and packet loss results, respectively, using EWFQ scheme. These results are supplemented with the packet loss results in Table 4. Note that we use self-similar traffic as a background best-effort traffic. Even in this case, the performance of voice traffic is not affected by the self-similarity nature

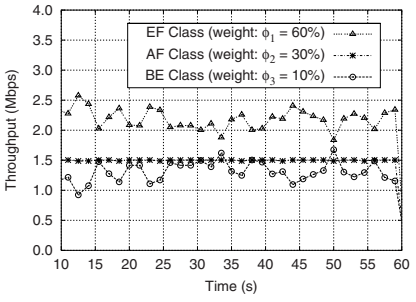

Fig. 10. Bottleneck link bandwidth (2^{nd} scenario)

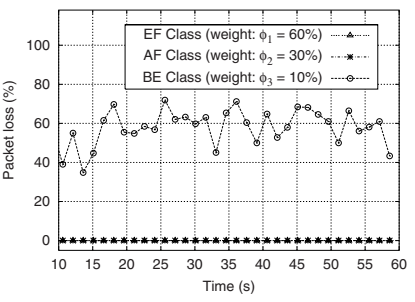

Fig. 11. Packet loss at the bottleneck (2^{nd} scenario)

Table 4. Summary of Packets Statistics for each DSCP → Class match (2^{nd} scenario)

DSCP Mapping	Total Pkts Received	Total Pkts Sent	Total Pkts Drops	Pkts Loss Rate (%)
00 → BE	59486	29666	29820	50
01 → BE	32877	8373	24504	75
10 → EF	197429	197429	0	0
11 → EF	23	23	0	0
20 → AF	11154	11154	0	0
21 → AF	0	0	0	0

of the traffic. This time, what one can see from Fig. 8, and Fig. 9 is that for both EF and AF classes the delay is minimal, and almost no jitter, respectively. And from Fig. 11, and Table 4, we also observe no packet loss for the two classes except best-effort class, which is expected. This is mainly because that our scheduler also bounds the delay and packet loss for AF class, as long as the traffic is within its allowed rate.

As for the bandwidth, both EF and AF traffic are also guaranteed with their requirements. This can be seen clearly in Fig. 10. And since the voice traffic consists of on/off traffic sources, during the on period it is ensured all its QoS requirement, but during its off time, the bandwidth which is not used by it is consumed by the best effort traffic. So, whenever excess bandwidth is available, EWFQ distributes this extra bandwidth among all the classes proportionally, i.e., according to their relative weights.

With EWFQ, all results in the above Figures show clearly how all voice packets are served with the highest assurance of delay bounds, minimal jitter, bandwidth guarantees, and packet loss regardless of the rates of medium and low priority classes. For other classes, the delay and delay variations are correlated largely with the increment of their average queue sizes during congestion, and their corresponding weights, as well. This is mainly due to the 60% of the service time of the scheduler is spent for serving the voice class. As a result, the more the queue is in congestion, the more the delay for packets to reach the destination

is increased. The delay and jitter problems will be more aggravated if the queue size is getting large. On the other hand, the queue size should be set large enough to avoid packet loss. So, one has to face a tradeoff between packet loss and delay for medium and low classes during times of high congestion.

5 Conclusion and Future Work

In this paper, we study whether it is feasible to achieve fair and scalable bandwidth sharing while supporting better bounds on end-to-end delay for real-time IP traffic classes within the DiffServ framework. To this end, we propose a scheduling mechanism called EWFQ. The proposed scheme is effective in providing efficient and enhanced services for real-time multimedia traffic class, such as VoIP, over other non-real-time traffic classes while providing the additional benefits of fair bandwidth sharing among all the classes proportionally. With EWFQ, it is possible to maintain tight delay bound for high priority class, and distribute the bandwidth according to the predetermined weights for all service classes. The results demonstrate that the mechanism has the capability for providing service isolation needed among different traffic classes in the network by protecting high priority class from other misbehaving traffic classes, which is vital in order to support real-time multimedia applications in the Internet. Another important aspect of EWFQ, in the same context, is that the scheme lowers significantly its implementation complexity, while maintaining the required end-to-end delay bounds and bandwidth fairness, along with scalability.

Further work on testing the algorithm in the realistic Internet environments is needed to see the impacts of extreme events. Other future work to consider, which would also most likely be important in this regard is to provide mathematical analysis on the end-to-end delay bound of EWFQ algorithm to show further its strengths and limitations in comparison with other approaches.

Acknowledgement. This work was supported in part by Telecommunication Advancement Organization Grant for Application Research and Development on Japan Gigabit Network (JGN-P341005).

References

1. R. Braden, D. Clark, S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633, June 1994.
2. S. Black, *et al.*, "An Architecture for Differentiated Services," RFC2475, Dec. 1998.
3. V. Jacobson, K. Nichols, K. Poduri, "An Expedited Forwarding PHB," RFC2598, June 1999.
4. J. Heinanen, *et al.*, "Assured Forwarding PHB Group," RFC2597, June 1999.
5. Zheng Wang, "Internet QoS: Architectures and Mechanisms for Quality of Service," Morgan Kufmann, March 2001. (ISBN: 1558606084)
6. T. Bayle, R. Aibara and K. Nishimura, "Scheduling IP Traffic for Enhanced Services in DiffServ Networks," Proc. of APCC2002, September 2002.

7. A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair-Queueing Algorithm," Proc. ACM SIGCOMM '89, September 1989.
8. H. Zhang and D. Ferrari, "Rate Controlled Static Priority Queueing," Proceedings of IEEE INFOCOMM 1993.
9. A. K. Parekh and R. G. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single-Node Case," IEEE/ACM Transactions on Networking, Vol. 1, No. 3, June 1993.
10. M. Shreedhar and G. Varghese, "Efficient Fair Queueing using Deficit Round Robin," Proc. ACM SIGCOMM '95, Vol. 25, No. 4, October 1995.
11. S. J. Golestani, "A Self-Clocked Fair Queueing Scheme for Broadband Applications," Proc. INFOCOM '94, Apr. 1994. April 1994.
12. P. Goyal, Harrick M. Vin, H. Cheng, "Start-time Fair Queueing: A Scheduling Algorithm for Integrated Services Packet Switching Networks," IEEE/ACM Transactions on Networking, Vol. 5, No. 5, October 1997.
13. Bennett, J. and Zhang, H. "WF^2Q: Worst-case Fair Weighted Fair Queueing," Proceedings of IEEE INFOCOM '96, March 1996.
14. Bennett, J. and Zhang, H. "Hierarchical Packet Fair Queueing Algorithms," Proc. ACM SIGCOMM '96, August 1996.
15. J. G. Gruber, "A Comparison of Measured and Calculated Speech Temporal Parameters Relevant to Speech Activity Detection", *IEEE Trans. Communs.*, Vol. COM–30, No. 4, pp 728–738, 1982.
16. "The Network Simulator, NS-2," http://www.isi.edu/nsnam/ns/.
17. J. Padhye *et al.*, "Modeling TCP Reno performance: a simple model and its empirical validation," IEEE/ACM Transactions on Networking 8, April 2000.
18. W. E. Leland, *et al.*, "On the Self-Similar Nature of Ethernet Traffic," Proc. SIGCOM93, 1993, San Francisco, California, pp. 183–193.