Regular Paper

A New Round Robin Scheduling Algorithm for Multimedia Traffic and Differentiated Services Router Implementation

Hidetoshi Yokota,† Toshihiko Kato†
and Hideyoshi Tominaga††

The Internet has to handle increasing multimedia traffic such as audio and video, and there is a growing demand for it to supply Quality of Service (QoS) to enhance this multimedia communication. One potential candidate to support multimedia is Differentiated Services (Diffserv), which defines not only service classes but also functional modules to realize required services. The scheduling function plays a crucial role in QoS support, and various algorithms have been proposed and analyzed. These analyses basically assume a backlogged situation in all queues, but multimedia traffic is bursty in nature, and the fairness of bursty traffic relative to continuous traffic has not been fully studied yet. In this paper, we discuss the potential unfairness that bursty traffic may be subject to, and propose a new frame-based packet scheduling algorithm, called the "Elastic Weighted Round Robin," for frame-based packet scheduling. We then implement this proposed algorithm as in a Diffserv router on FreeBSD and show its performance in terms of its bandwidth assignment, delay, and packet loss properties in this implementation. We also evaluate its jitter performance on bursty traffic and discuss the effectiveness of the proposed algorithm.

1. Introduction

As broadband networks are being made available to more people, conventionally separated network services such as data, telephony, and video broadcasting are converging on the Internet. Transmission of multimedia streams is particularly more sensitive to bandwidth, delay, and jitter requirements, and there is a growing demand for provision of QoS-oriented services. Intserv/RSVP were developed for this purpose, but they focused on service definitions[1),2)] and a signaling protocol[3)], and QoS-enforcing functions such as traffic control were somewhat outside the scope. It became clear that traffic conditioning functions must be considered alongside the service definition, and when Differentiated Services (Diffserv) was proposed, QoS mechanisms were discussed at an earlier stage. In Diffserv, the forwarding discipline for each class is determined by Per-Hop Behavior (PHB), and many traffic streams can be aggregated into one of a small number of behavior aggregates (BAs), which are each forwarded by using the same PHB at the router. The scheduling function for controlling the serving order of multiple queues plays an important role in controlling the QoS according to the PHB.

According to the classification of schedulers in Zhang and Keshav[5)], there are two main architectures: sorted-priority and frame-based. In a sorted-priority scheduler a timestamp is associated with each packet in the system, and packets are sorted on the basis of their timestamps and transmitted in that order. Weighted Fair Queueing[6)] and its variations such as Self-Clocked Fair Queueing[7)] are examples of this type of scheduler. On the other hand, in a frame-based scheduler, time is split into frames of variable length, and packets are served in a predetermined order. As an example of this type of scheduler, Deficit Round Robin (DRR) is proposed and analyzed in Shreedhar and Varghese[8)].

These algorithms are systematically analyzed in Refs. 8) and 9), and their advantages and disadvantages are described in terms of fairness or complexity. These analyses assume a so-called "backlogged" situation where packets arrive continuously. However, multimedia traffic such as audio and video is bursty in nature and this backlogged situation cannot always be assumed in such traffic patterns. Yet, the fairness of continuous traffic versus bursty traffic has not been fully studied and discussed. We propose a new frame-based scheduling discipline providing low jitter for bursty traffic, which is a crucial property in audio and video commu-

† KDDI R&D Laboratories, Inc.
†† Department of Electronics, Information and Communication Engineering, Waseda University

nications. We also implemented the proposed algorithm as a scheduler of a Diffserv router to verify and validate the algorithm. We briefly describe the configuration and further evaluate its fairness and jitter performance in bursty traffic.

The remainder of this paper is organized as follows. In Section 2, we briefly describe the DRR scheduling algorithm as related work, and we subsequently propose the new Elastic Weighted Round Robin algorithm in Section 3. In Section 4, we show how our algorithm is implemented in the scheduling function of a Diffserv router. Section 5.1 evaluates the throughput, delay, and packet loss rate performance under QoS control, thereby verifying the proposed algorithm and the implementation method in an actual environment. Section 5.2 presents the jitter performance in bursty traffic, and we discuss the advantages of EWRR using results from these evaluations in Section 6. Finally, in Section 7, we offer some closing remarks.

2. Related Work

The fairness parameter of a scheduling algorithm is defined in Stiliadis and Verma[9] as the maximum difference in the service normalized by the allocated bandwidth offered to any two connections over an interval in which both are continuously backlogged. To provide fair handling among queues, this fairness parameter needs to be bounded. In this section, we consider DRR as an example of a frame-based scheduling algorithms, and investigate its scheduling discipline. The DRR algorithm is based on the Weighted Round Robin, but it allocates a weight to each queue not according to the number of packets but according to the allocated bandwidth share, which is called a quantum. Note that packet length is variable, and that a packet is transmitted only when the quantum is greater than the packet size. When the quantum is smaller, it is carried over to the next cycle, and the scheduler attempts to send the packet again. If there is no traffic, the quantum is initialized to zero, and the quantum is assigned when a packet arrives at the queue. In the example shown in **Fig. 1**[8], the first packet (200 bytes) in queue 1 can be transmitted, but the next packet (750 bytes) exceeds the quantum (300) at the point, so it is not transmitted in this cycle, and the next queue is serviced.

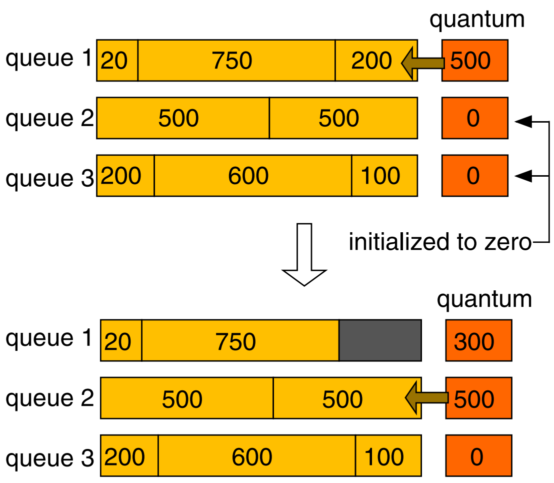


Fig. 1 DRR algorithm.

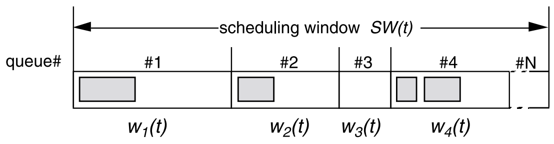


Fig. 2 Scheduling window.

3. EWRR Algorithm

In DRR, when there is no packet to send, or in other words when the connection is not backlogged, the allocated share is initialized to zero. However, if intermittent or bursty traffic arrives, the previously unused share is not taken into account, and therefore such traffic may be treated unfairly in comparison with continuous traffic even when two traffic sources are transmitted at the same rate. We propose an improved scheduling algorithm, the Elastic Weighted Round Robin (EWRR), which attempts to handle bursty traffic as fairly as possible. The baseline mechanism of EWRR is explained below.

A weight, W_i, is allocated to each queue according to the required bandwidth. We define a scheduling window (SW) as a means of controlling the dequeueing of packets from each queue. When a queue is serviced in an SW, packets are transmitted according to the amount of the current weight (**Fig. 2**). Note that packets are of variable length, and that the length may not match the current weight. In such cases, the first packet that exceeds the current weight is transmitted, and in the next SW, the difference is subtracted from the weight (**Fig. 3** (a)). We denote W_i as the weight allocated to the

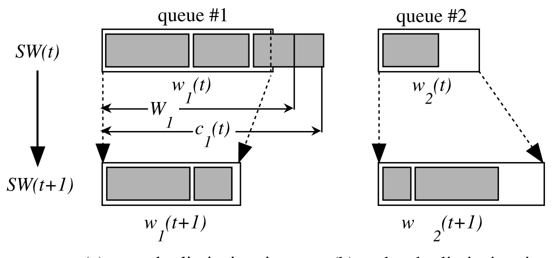


Fig. 3 Scheduling discipline of the EWRR algorithm.

queue i at the beginning. When we define $w_i(t)$ and $c_i(t)$, respectively, as the weight and total length of transmitted packets in the t-th scheduling window $SW(t)$, then the weight in $SW(t+1)$ is written as

$$w_i(t+1) = w_i(t) + (W_i - c_i(t)). \quad (1)$$

If there is no packet to send in the SW or the total length of transmitted packets is below $w_i(t)$, then the remaining weight is accumulated in the next SW (Fig. 3 (b)). Note that W_i^{max} is present to satisfy the following inequality:

$$w_i(t) \le W_i^{max}. \quad (2)$$

Those queues that do not have packets to send are extracted from the SW, and when all queues have been removed from the SW, $w(i)$ is recalculated from Eqs. (1) and (2), and the process moves on to the next SW. Thus, the length of the SW changes at each cycle according to whether there are packets to send and the value of $w(i)$. Note that bursty traffic is not more advantageous than continuous traffic in the EWRR algorithm, since W_i^{max} by itself is just the partial sum of the unused weights. Although bursty traffic is transmitted up to the weight that has been accumulated by the time, this weight is not taken from those allocated to other classes, so EWRR works without sacrificing other classes.

The DRR described in Section 2 does not hold an unused weight in the past cycles, so its maximum weight is the allocated bandwidth share. Thus, DRR could be considered as the case that $W_i^{max} = W_i$ in EWRR. Although DRR does not allow the sending of a packet whose length exceeds the current quantum, unlike EWRR, when we consider subsequent cycles, we see that DRR waits until the total quantum reaches the packet length for sending, and EWRR waits for the current weight to become large enough to send the next packet. Hence in a backlogged situation, the two algorithms will show an equivalent performance over time. The relationship between an allocated weight and an average transmission rate is shown later.

In Eq. (1), after summing from $t = 1$ to T, and dividing by T, we obtain the following equation:

$$\frac{1}{T}(w_i(T+1) - w_i(1)) = W_i - \frac{1}{T}\sum_{t=1}^{T} c_i(t). \quad (3)$$

The upper bound of $w_i(t)$ is provided by Eq. (2), and a lower bound is present when the transmission rate is finite. Hence under a backlogged condition with a sufficiently large T, we obtain the following equation from Eq. (3):

$$\lim_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T} c_i(t) = W_i. \quad (4)$$

Also, the service time $\tau(t)$ for $SW(t)$ is

$$\tau(t) = \frac{1}{r}\sum_{i=1}^{N} c_i(t), \quad (5)$$

where r is the output interface rate and N is the number of queues. Accordingly, from Eqs. (4) and (5), the average transmission rate $\overline{r_i}$ from $t = 1$ to a sufficiently large T can be written as:

$$\begin{aligned}\overline{r_i} &= \lim_{T\to\infty} \frac{\sum_{t=1}^{T} c_i(t)}{\sum_{t=1}^{T} \tau(t)} \\ &= \frac{\lim_{T\to\infty} \sum_{t=1}^{T} c_i(t)}{\sum_{i=1}^{N} \lim_{T\to\infty} \sum_{t=1}^{T} c_i(t)} r \\ &= \frac{W_i}{\sum_{i=1}^{N} W_i} r \end{aligned} \quad (6)$$

From the equation above, the bandwidth allocated to each class is provided by the product of the ratio of class i's weight W_i to the sum of all weights, and the outgoing interface rate. As long as the above ratio is consistent, the allocated bandwidth is not dependent on its absolute value.

4. Implementation of the Proposed Algorithm in a Diffserv Router

To evaluate and validate the proposed algorithm, we implemented it as the scheduling part of a Diffserv router. A conceptual model of a Diffserv router is proposed in Bernet, et al.[10], where four components are defined:

- classifier elements,
- meter elements,
- action elements, and
- queueing elements.

A combination of the above elements is also defined as the Traffic Conditioning Block. For classifier elements, we implemented a Behavior Aggregate (BA) Classifier and a Multi Field (MF) Classifier. The former classifies packets only with DSCP, and the latter also considers source and destination addresses, ports, and the protocol used. As a meter element, we adopted a two-parameter token bucket (TB) meter, and for out-of-profile packets, an absolute dropper, which simply discards packets, is applied as an action element. A queueing element is further decomposed into queues, schedulers, and algorithmic droppers. We used a FIFO queue, EWRR as a scheduler, and a threshold-dropper, which simply discards packets when the queue length exceeds a predefined threshold, as an algorithmic dropper. As we see above, various components are involved in a Diffserv router, and one component may affect another. While our implementation supports all four elements, we focus in later sections on evaluating the queueing element, especially the performance of the EWRR scheduler implementation.

The hardware specification and the operating system on which our algorithm was implemented were a Pentium II processor (clock rate: 333 MHz, main memory: 128 MB) and FreeBSD3.3-RELEASE. In almost all operating systems, packet enqueueing and dequeueing are conducted in the kernel for performance reasons, and thus a natural way of implementation would be to put the queueing element into the kernel. On the other hand, the current version of FreeBSD supports two special functions called `ipfw` and `divert`, which allow user applications to handle packets quite easily. The `ipfw` function filters incoming packets that match user-defined rules based on source/destination addresses/ports or a protocol number, and the `divert` function directs those packets to a specific port where a user application is waiting. For instance, the following command creates a rule numbered 10 to direct all IP packets heading for interface `if1` to port number 1000:

```
ipfw add 10 divert 1000 ip from any to
any out xmit if1
```

After the application has processed those packets, they are placed back in the kernel. We leveraged these two functions and made a scheduler in the user space. The configuration is shown in **Fig. 4**. Incoming packets are handled in the following way:

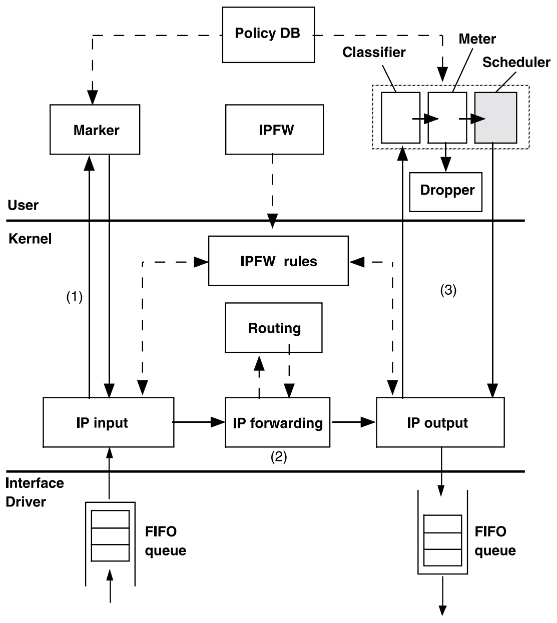


Fig. 4 Configuration for a Diffserv router.

(1) Those packets that satisfy the rules for `ipfw` are passed from the kernel space to the user space, and are placed back in the kernel space after they have been marked according to the packet types defined in the Policy Database (DB).

(2) The router forwards these packets to the output interface for their destination by referring to the routing table.

(3) At the output interface, the router passes these packets to the user space again, classifies them according to the DSCP (Differentiated Services Code Point), passes them through the TB Meter, and places them back in the kernel space after scheduling the forwarding order by means of the EWRR scheduler. Packets not matching the profile in the TB Meter are dropped by the dropper.

Class Based Queueing (CBQ)[11)] and Dummynet[12)] were also implemented on FreeBSD, but the queueing function is embedded in the kernel, while our approach does not need to alter the kernel code for packet scheduling. As a result of implementing these Diffserv modules as user applications, data copying occurs between the user space and the kernel space; however, portability between versions is gained.

This is quite convenient when implementing such modules on rapidly evolving operating systems. In later sections, we evaluate the implementation to determine whether it fulfills the required scheduling function.

5. Performance Evaluation

5.1 Bandwidth Utilization of the Proposed Algorithm

The experimental environment for evaluating the algorithm's performance is shown in **Fig. 5**. We defined three traffic classes and two input traffic sources in the experiment. Each node is connected via a 10 Mbps Ethernet, and thus the maximum incoming rate is 20 Mbps and the outgoing rate is 10 Mbps.

IP packets classified by the DSCP are contained in Ethernet frames, and the frame size of each class is as follows: class 1, 256 bytes; class 2, 512 bytes; and class 3, 1,024 bytes. Frames for the three classes are generated in an orderly manner as shown in **Fig. 6**, and 10,000 frames are transmitted per class (5,000 frames per node) at the same frame transmission rate by a traffic generator. The InterFrameGap, which is a minimum interframe spacing between two successive frames, is set to 9.6 microseconds for the 10 Mbps Ethernet, as defined in Ref. 13). The buffer size of each class was set at 8192 K bytes.

Figures 7 and **8** show the throughput of each class in a case where there is no QoS control and in a case with QoS control by EWRR, respectively. For EWRR, we assigned equal bandwidth shares to all classes, and selected three parameter sets for W_i and W_i^{max} $(i = 1, 2, 3)$: (a) W_i = 1,500 bytes, W_i^{max} = 3,000 bytes, (b) W_i = 1,500 bytes, W_i^{max} = 150 K bytes, and (c) $W_i = W_i^{max}$ = 75 K bytes.

Figure 9 shows the throughput of three classes in a case where DRR is used. The bandwidth shares for the three classes are set to be equal, and two sizes of weights or quanta are investigated: (a) W_i = 1,500 bytes and (b) W_i = 75 K bytes. Throughout these experiments, no packet loss was experienced in any class, and the average outgoing transmission rate was about 9.78 Mbps. We can therefore say that data copying between the kernel space and the user space does not significantly affect the performance at this interface speed. For faster speeds such as Fast Ethernet, which is more widely available nowadays, further investigation is needed. It is to be expected, however, from the general principle guiding the motivation of Diffserv [10)], that edge routers rather than core routers will implement such complex functionalities as described in Section 4, and for those edge routers, precise traffic conditioning based on the service level agreement is more essential than fast forwarding.

In **Figs. 10** and **11**, the packet loss rate and the average delay time of EWRR, respectively, are shown when the buffer size of each queue is changed from 64 K bytes to 8,192 K bytes. For all classes, W_i and W_i^{max} are 1,500 bytes and 3,000 bytes, respectively, which are the same as EWRR parameter set (a).

From Fig. 8 (a) and Fig. 9 (a), we can say that the two algorithms perform in the same way for the back-to-back traffic loaded in this experi-

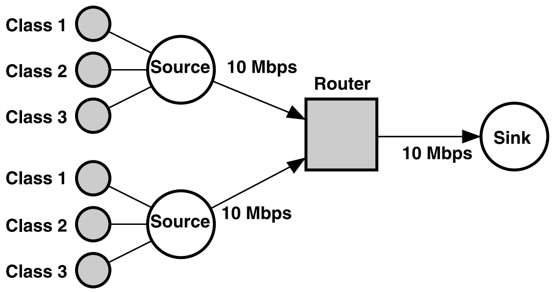


Fig. 5 Experiment environment.

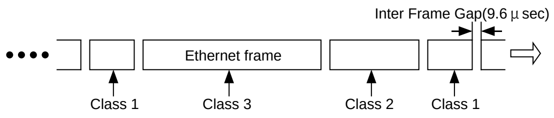


Fig. 6 Generated frame sequence.

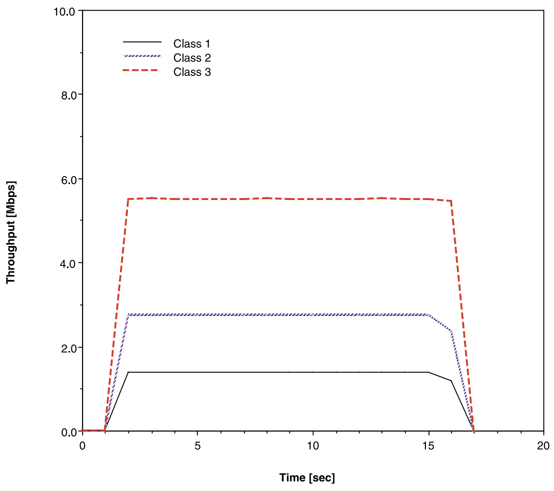
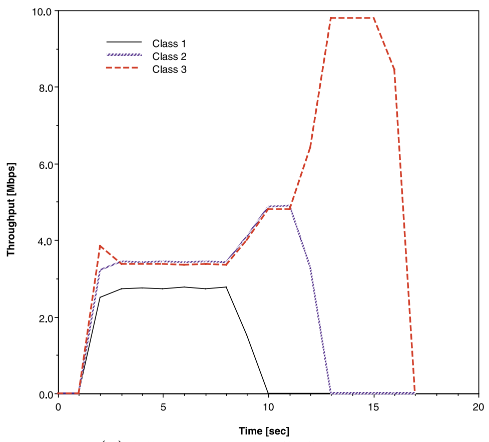
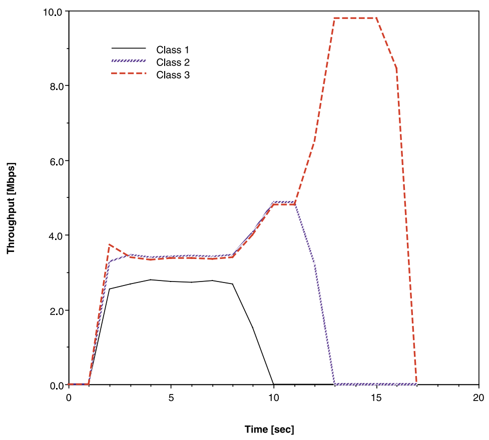


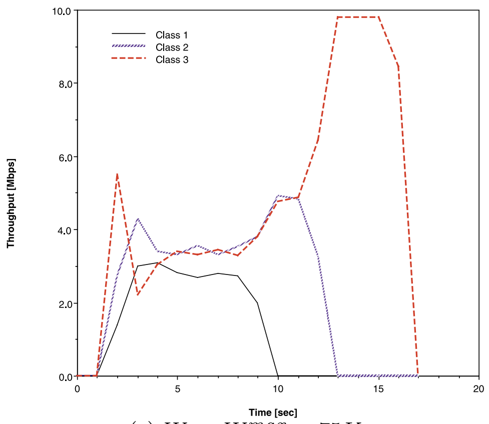
Fig. 7 Temporal change in throughput without QoS control.



(a) $W_i = 1500$, $W_i^{max} = 3000$



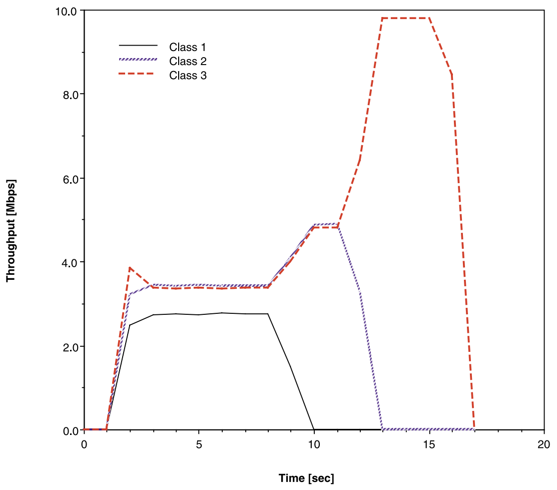
(b) $W_i = 1500$, $W_i^{max} = 150K$



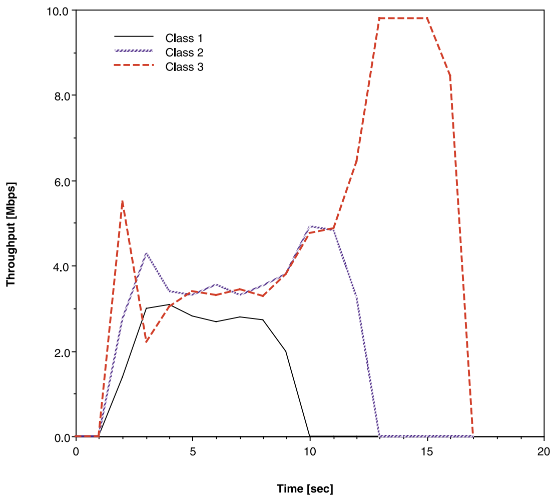
(c) $W_i = W_i^{max} = 75K$

Fig. 8 Temporal change in throughput with EWRR.

ment. However, when we look at the beginning of the transmission in Fig. 8 (c) for EWRR and in Fig. 9 (b) for DRR, we can see that the throughput of all classes fluctuates. The assigned weight W_i used in this experiment is larger than that used in the previous experi-



(a) $W_i = 1500$



(b) $W_i = 75K$

Fig. 9 Temporal change in throughput with DRR.

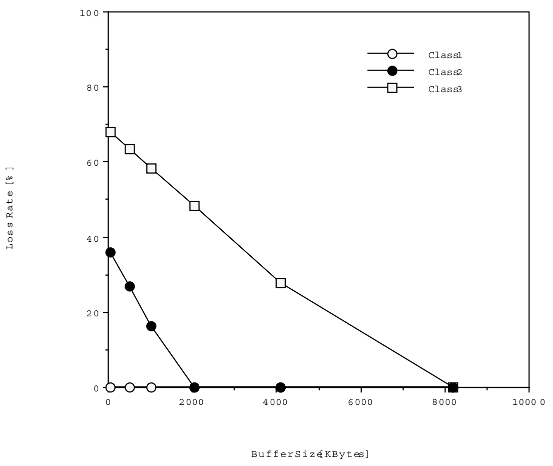


Fig. 10 Relationship between buffer size and packet loss rate.

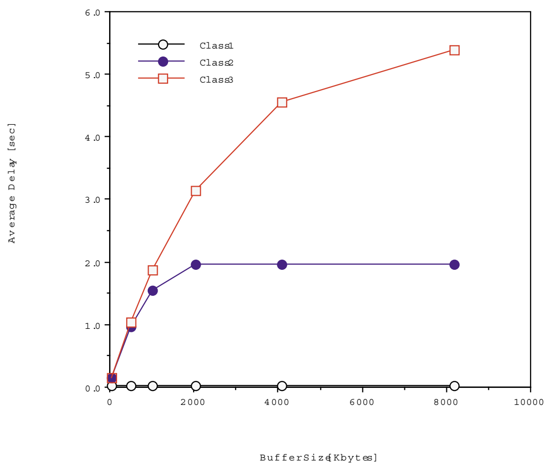


Fig. 11 Relationship between buffer size and average delay time.

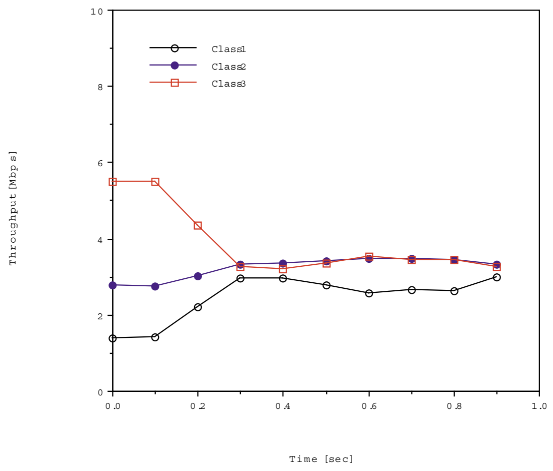


Fig. 13 Transition of throughput ($m = 10.0$).

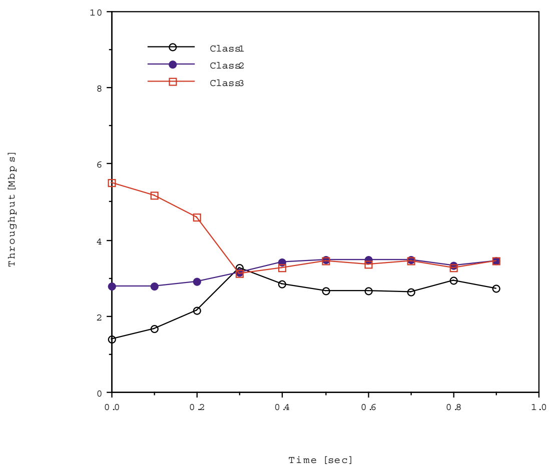


Fig. 12 Transition of throughput ($m = 1.0$).

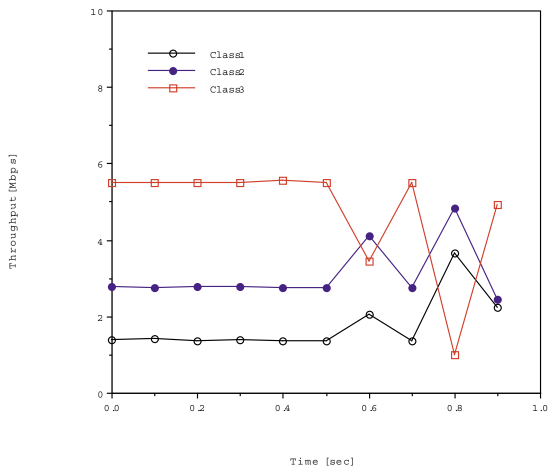


Fig. 14 Transition of throughput ($m = 100.0$).

ment, and this parameter setting is considered to affect the bandwidth assignment at the start of the transmission. To study this throughput fluctuation, we conducted a further evaluation focusing on the packet sizes and W_i. We used the same environment and input sources as in Fig. 5, and showed the change in throughput every 0.1 seconds from 0 to 1 second by varying W_i in **Figs. 12**, **13**, **14**. To show that the values of W_i relative to the length of transmitted packets rather than its absolute values affect the throughput performance, we normalized W_i by defining m, which satisfies $W_i = mL^{max}$, where L^{max} is the maximum packet length ($\max_{i,t}\{L_i(t)\}$). In this experiment, we set $W_i^{max} = W_i$, and assigned the same W_i to all classes. These results show that in cases where $m = 1.0$ and 10.0, the throughput converges to the allocated bandwidth within 0.3 second, but when $m = 100.0$, it takes about 0.6 second, and we can see that it also fluctuates widely thereafter.

5.2 Jitter Performance in Bursty Traffic

Recall that in the DRR algorithm, the quantum returns to zero when there is no packet to send, and therefore in the case of bursty traffic, the quantum, or the weight, during off-periods is not considered, and that this could lead to unfair bandwidth utilization compared with continuous traffic. On the other hand, the EWRR retains an unused weight up to W^{max} when there is no traffic. These two algorithms show equivalent performance in a backlogged situation; however, at times when incoming traffic switches from an idle state to a busy

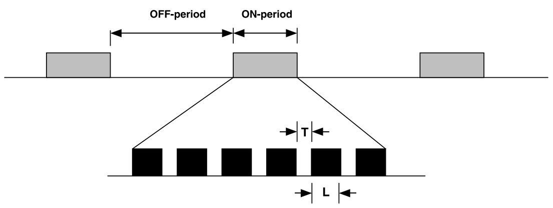


Fig. 15 Burst traffic model.

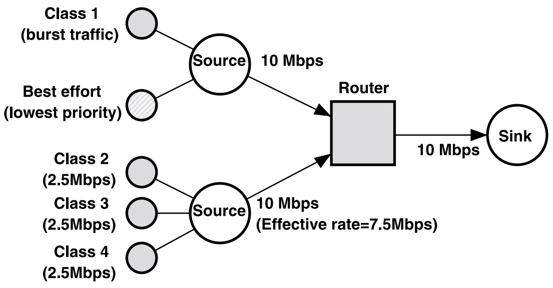


Fig. 16 Input traffic sources.

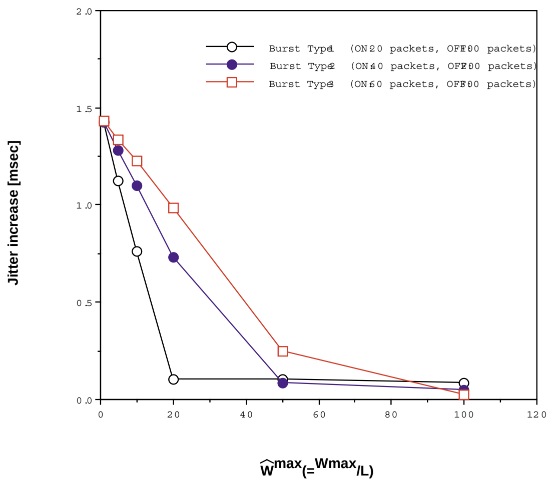


Fig. 17 Relationship between $\widehat{W}_i^{max}$ and jitter increase.

state, they differ in how the packets are handled and this primarily affects the jitter performance. In this section, we evaluate the jitter property of EWRR and DRR in situations where regulated continuous traffic and bursty traffic are mixed together, and we then verify the effectiveness of the proposed algorithm.

In this experiment, we assume four priority classes which are targeted for QoS control and the best-effort (BE) traffic with the lowest priority. Class 1 traffic has burstiness as shown in **Fig. 15**, and the other classes are assumed to be continuous.

In this paper, we define jitter as the difference in arrival times between two consecutive packets within the same class. We define a jitter increase as the increment or increase in jitter obtained when only the target traffic is injected versus the jitter obtained when other traffic is also injected. To evaluate the change in jitter for bursty traffic, we consider the input sources shown in **Fig. 16**. In class 1, the packet length $L = 512$ bytes (fixed), where L is the same as in Fig. 15 and the packet arrival interval $T = \mathit{InterFrameGap}$ (fixed). We also define k as the ratio of the ON-period to the OFF-period of the bursty traffic, and we change the ON-period of the traffic to control the input traffic. In this experiment, we set $k = 0.2$, and prepared three types of burst traffic by changing the number of packets in the ON-period: type 1 (ON-period: 20 packets), type 2 (ON-period: 40 packets), and type 3 (ON-period: 60 packets). The packet length of classes 2 to 4 is also 512 bytes ($= L$), and we set these classes as continuous traffic at a speed of 2.5 Mbps, which is a fourth of the outgoing interface speed. The BE traffic, whose packet length is also 512 bytes, arrives whenever there is no class 1 traffic, that is, during the OFF-period, and is served only when there is no priority traffic. Those W_i where $i = 2$ to 4 are all the same as $W_i = W_i^{max} = L$. As for class 1, $W_1 = L$, and we changed W_1^{max} to adapt to the burst traffic. To make the relationship between W_i^{max} and the length of transmitted packets distinctive, we define $\widehat{W}_i^{max}$ as W_i^{max}/L by normalizing W_i^{max} with the packet length. The relationship between $\widehat{W}_i^{max}$ and a jitter increase is shown in **Fig. 17**, where we changed $\widehat{W}_i^{max}$ within the range of 1.0 to 100.0 to determine the influence in the order of magnitude. Each queue had a sufficient buffer and no packet loss occurred throughout the experiment. For comparison, we show the jitter increase of DRR and that of EWRR with $\widehat{W}_i^{max} = 1$ in **Table 1**. The tabulated results are the averages over five trials.

Table 1 Comparison of jitter increase in DRR and EWRR.

Burst type	DRR	EWRR ($\widehat{W}_i^{max}$=1)
Type 1	1.458 [msec]	1.463 [msec]
Type 2	1.418 [msec]	1.422 [msec]
Type 3	1.396 [msec]	1.420 [msec]

6. Discussion

6.1 Throughput, Delay, and Packet Loss Rate

In the experiment described in Section 5.1, the ratio of the frame size in the three classes is 1 : 2 : 4, and the frame transmission rates are the same for all classes. The bandwidths occupied by the input traffic of each class are therefore 2.9 Mbps, 5.7 Mbps, and 11.4 Mbps, respectively. Hence, in the case of no QoS control, as shown in Fig. 7, the throughput of classes 1, 2, and 3 becomes 1 : 2 : 4. This means that the throughput is dependent on the amount of input traffic, and that classes which have a large volume of traffic occupy a large bandwidth; thus, fairness in bandwidth utilization is not maintained.

On the other hand, when QoS is controlled by the EWRR scheduler and equal weights are assigned to all classes, each class is assigned 3.3 Mbps versus the 10 Mbps of the output interface rate. Therefore, as shown in Fig. 8 (a), when all classes transmit frames, the input rate of class 1 is less than the allocated weight, and therefore the input rate equals the output rate. For the other classes, the input rate is greater than the allocated weights, so they share the rest of the bandwidth in a ratio of 1 to 1. After the transmission of class 1 is finished, frames of class 2 and class 3 still remain, and these frames share the entire bandwidth in the ratio of 1 to 1. After the transmission of class 2, only class 3 frames remain, and these frames may occupy the entire bandwidth until the end of the transmission. Since EWRR and DRR are work-conserving, the unused bandwidth is used by classes with frames to transmit. This can be seen for DRR in Fig. 9 (a). In addition to this experiment, we conducted another that changes the frame rate of each class under the condition that the frame sizes of all classes are equal, and obtained results showing the same property.

With regard to packet loss rate, packets from class 2 and class 3 were transmitted in more than their allocated bandwidth, and the input rate of class 3 was the highest; thus the delay time of class 3 was highest and that of class 1 was lowest (18.9 msec on average), as shown in Fig. 10. By controlling the QoS, the packets are scheduled according to their weights; thus the delay time and the packet loss rate of classes that transmit packets exceeding their allocated bandwidth are degraded.

6.2 Fluctuation in Throughput

From Figs. 8 (a) and (b), we can see that the throughput of each class is dependent only on the assigned weight W_i, and that W_i^{max} does not affect the throughput. This property is compliant with Eq. (6), where the average transmission rate is dependent only on W_i and r. Figure 9 (a) also suggests that this applies to DRR as well. On the other hand, Fig. 8 (c) for EWRR and Fig. 9 (b) for DRR show that the throughput of all classes fluctuates, and does not seem to follow the equation at the beginning of the transmission. The assigned weight W_i used in these experiments is fifty times larger than that in previous experiments. In fact, Eq. (6) holds when transmission is observed for a sufficiently long time, and does not tell us about the instantaneous throughput.

The cause of the fluctuation in throughput at the beginning of service resides in the relationship between the size of an assigned weight and that of transmitted packets. Since each class can transmit packets up to the allocated weight, the throughput is proportional to the number of transmitted packets at the time just after service begins. With the results from Fig. 12 to Fig. 14, the larger the weight allocated to the maximum packet size, the longer it takes for the throughput to converge to the allocated bandwidth. We can also see that the throughput fluctuates greatly after convergence. This can be regarded as a common property not only for EWRR or DRR but also for all algorithms based on frame-based scheduling.

6.3 Alleviating Jitter in Bursty Traffic

From Table 1, the jitter increase of bursty traffic (Class 1) with EWRR shows almost the same result as the case with DRR when we set $\widehat{W}_1^{max} = 1$, that is, $W_1^{max} = L$. However, we can see that the larger the W^{max} we take compared to the packet length, the more the jitter increase can be reduced. From Fig. 17, W^{max} should be set to at least the burst size, to cancel the jitter increase attributed to burstiness; that is, $\widehat{W}^{max}$ requires 20 for burst type 1, 40 for burst type 2, and 60 for burst type 3. Notice that if the OFF-period is too short compared to the ON-period, W^{max} may not become large enough to completely absorb burst traffic. The ratio of the ON-period to the OFF-period k determines the total input traffic load, and should be carefully regulated by the source according to its bandwidth share W_i; otherwise, the TB Meter in the Diffserv router regulates the input traffic.

Some may argue that burstiness could be absorbed by allocating a large weight relative to the packet size. According to the results in Fig. 12 to Fig. 14, however, this would require a longer time to converge to the allocated bandwidth, and larger variance in the throughput after convergence would be observed. In the case of EWRR, however, by setting W_i^{max} larger than W_i, we can reduce the jitter in bursty traffic without increasing its weight relative to the transmitted packets. Since the maximum length of packets L^{max} can be obtained from the MTU (Maximum Transmission Unit), and the burst size can be obtained from the bucket size b of the TB Meter, we can provide an appropriate W^{max} by setting it to larger than $L^{max} \times b$. Notice that a large W^{max} will not affect the throughput performance in a backlogged situation, as shown in Fig. 8 (b).

7. Conclusion

In this paper, we have pointed out possible unfairness in non-backlogged situations that often occur in bursty traffic. Such traffic patterns are typical of multimedia communications and also usually require tighter delay and jitter bounds. We proposed a new frame-based scheduling algorithm, Elastic Weighted Round Robin, and showed a configuration in which we implemented the proposed algorithm as a scheduler for a Diffserv router. In this configuration, we presented the delay time, packet loss rate, and fairness of throughput properties for each class when QoS control is imposed. We further investigated fluctuations in the throughput with regard to allocated weights and packet sizes, and demonstrated that by using EWRR we can lower the jitter increase in the bursty traffic while reducing fluctuations in throughput compared with the DRR (Deficit Round Robin) algorithm by choosing appropriate W^{max} settings.

References

1) Wroclawski,W.: *Specification of the Controlled-Load Network Element Services*, RFC2211, IETF (Sep. 1997).
2) Shenker, S., et al.: *Specification of the Guaranteed Quality of Service*, RFC2212, IETF (Sep. 1997).
3) Braden, R., et al.: *Resource Reservation Protocol (RSVP) — Version 1 Functional Specification*, RFC2205, IETF (Sep. 1997).
4) Blake, S., et al.: *An Architecture for Differentiated Services*, RFC2475, IETF (Dec. 1998).
5) Zhang, H. and Keshav, S.: Comparison of rate based service disciplines, *Proc. SIGCOMM '91*, pp.113–121 (Aug. 1991).
6) Demers, A., Keshav, S. and Shenker, S.: Analysis and Simulation of a Fair Queueing Algorithm, *Proc. SIGCOMM '89*, pp.1–12 (Sep. 1989).
7) Golestani, S.J.: A self-clocked fair queueing scheme for broadband applications, *Proc. INFOCOM '94*, pp.636–646 (1994).
8) Shreedhar, M. and Varghese, G.: *Efficient Fair Queueing using Deficit Round Robin*, Washington University (Oct. 1995).
9) Stiliadis, D. and Varma, A.: Latency-rate servers: A general model for analysis of traffic scheduling algorithms, *Proc. INFOCOM '96*, pp.111–119 (1996).
10) Bernet, Y., et al.: *An Informal Management Model for Diffserv Routers*, draft-ietf-diffserv-model-06, IETF (Feb. 2001).
11) Floyd, S. and Jacobson, V.: Link Sharing and Resource Management Models for Packet Networks, *IEEE/ACM Trans. Networking*, Vol.3, No.4, pp.365–386 (Aug. 1995).
12) Rizzo, L.: *Dummynet*: A simple approach to the evaluation of network protocols, *Computer Communication Review*, 27(1), pp.31–41 (Apr. 1997).
13) International Standard ISO/IEC 8802-3:1996 [ANSI/IEEE Std 802.3, 1996 Edition], *IEEE* (1996).

(Received June 6, 2001)
(Accepted November 14, 2001)

Hidetoshi Yokota received the B.E. and M.E. degrees in electrical engineering from Waseda University, Tokyo, in 1990 and 1992, respectively. He joined KDDI R&D Laboratories, Inc., Japan, in 1992. From 1995 to 1996 he was with SRI International, Menlo Park, CA as an International Fellow. He received the Young Engineer Award in 1998 from IEICE. His current research interests include admission control, packet scheduling and mobile communications.

Toshihiko Kato Received the B.E., M.E., and Dr.Eng. degrees of electrical engineering from the University of Tokyo, in 1978, 1980 and 1983 respectively. Since joining KDD in 1983, he has been working in the field of OSI protocol implementation, conformance testing of communication protocols, distributed systems, high speed protocol, and mobile Internet protocol. From 1987 and 1988, he was a visiting researcher at Carnegie Melon University. He is currently the senior manager of Mobile IP Network Laboratory in KDDI R&D Laboratories, Inc. Since 1993, he has been a Guest Associate Professor of Graduate School of Information Systems in the University of Electro-Communications. He received Motooka Award in 1990.

Hideyoshi Tominaga received the B.E., M.E., and D.E. degrees from Waseda University, Tokyo, Japan, in 1962, 1964 and 1974, respectively. From 1964 to 1971, he worked for NTT Electric Communications Laboratories. Since 1971 he has been with Waseda University. Currently he is a professor of the Department of Electronics, Information and Communication Engineering. His research areas include picture coding, switching systems, and information security. He has been involved in standardization activities, such as MPEG, DAVIC, ITU-T, and ISO-SGs. He founded the Global Information and Telecommunication Institute (GITI), Waseda University in 1996, and also Graduate School of Global Information and Telecommunication Studies, Waseda University in 2000. He received the IEICE Achievements Award in 1989, etc. He is a senior member of IEEE.