Adaptive Input Traffic Prediction Scheme for Absolute and Proportional Delay Differentiated Services in Broadband Convergence Network

Junghoon Paik, Jeong-dong Ryoo, and Bheom-Soon Joo

In this paper, an algorithm that provides absolute and proportional differentiation of packet delays is proposed with the objective of enhancing quality of service in future packet networks. It features an adaptive scheme that adjusts the target delay for every time slot to compensate the deviation from the target delay, which is caused by prediction error on the traffic to arrive at the next time slot. It predicts the traffic to arrive at the beginning of a time slot and measures the actual arrived traffic at the end of the time slot. The difference between them is utilized by the delay control operation for the next time slot to offset it. Because the proposed algorithm compensates the prediction error continuously, it shows superior adaptability to bursty traffic and exponential traffic. Through simulations we demonstrate that the algorithm meets the quantitative delay bounds and is robust to traffic fluctuation in comparison with the conventional non-adaptive mechanism. The algorithm is implemented with VHDL on a Xilinx Spartan XC3S1500 FPGA, and the performance is verified under the test board based on the XPC860P CPU.

Keywords: Absolute delay, proportional delay, DiffServ, BcN.

Manuscript received Aug. 14, 2007; revised Feb. 25, 2008.

This work was supported by the IT R&D program of MIC/IITA, Rep. of Korea [2007-S-102-02, Carrier Class Ethernet Technology].

Junghoon Paik (phone: +82 31 670 6734, email: jhpaik@dima.ac.kr) is with the Department of Broadcasting Communication, Dong-Ah Institute of Media and Arts, Anseong, Gyeonggi-do, Rep. of Korea.

Jeong-dong Ryoo (email: ryoo@etri.re.kr) and Bheom-Soon Joo (email: bsjoo@etri.re.kr) are with Broadcasting and Telecommunications Convergence Research Laboratory, ETRI, Daejeon, Rep. of Korea.

I. Introduction

Two broad paradigms for quality-of-service (QoS) in the Internet have emerged, namely Integrated Services (IntServ) and Differentiated Services (DiffServ) [1], [2]. The IntServ model, which aims to provide hard end-to-end QoS guarantees to each individual data flow, requires per-flow-based resource allocation and service provisioning; thus, it suffers from scalability and manageability problems due to the huge number of data flows.

This lack of scalability is, to a large extent, being addressed within the DiffServ architecture. In the DiffServ model, traffic is aggregated into a finite number of service classes, which receive different forwarding treatment. It achieves scalability and manageability by providing quality per traffic aggregate, not per application flow. However, it is difficult to contrive efficient resource allocation mechanisms to guarantee end-to-end QoS for each individual data flow.

With superiority in terms of scalability and manageability, DiffServ is gaining more popularity as the QoS paradigm for the future Internet. Several schemes have been devised to realize the DiffServ philosophy. At one end of the spectrum, absolute DiffServ seeks to provide end-to-end absolute performance measures without per-flow state in the network core [3]. At the other end of the spectrum, relative DiffServ seeks to provide per-class relative services [4]. In the relative DiffServ model, the traffic from a higher priority class will receive service that is no worse than that received by traffic from a lower priority class.

In our view, absolute DiffServ is essential for handling a real-

time application that requires guaranteed QoS measures for future Internet. In addition, proportional DiffServ is needed to handle soft real-time service which is tolerant of occasional delay violations, and hence, does not require strict delay bounds. Consequently, QoS architectures providing any mix of absolute and relative differentiated schemes under the DiffServ paradigm are the most suitable service architectures for future Internet.

The common approach for delay differentiation is to use scheduling algorithms. In general, schedulers can be characterized as work-conserving or non-work-conserving. A scheduler is work-conserving if the server is never idle when a packet is buffered in a system. A non-work-conserving server may remain idle even if there are available packets.

Work-conserving servers always have lower average delays than non-work-conserving servers. Examples of work-conserving schedulers include generalized processor sharing [5], weighted fair queueing (WFQ) [6], virtual clock [7], weighted round robin [8], deficit round robin [9], waiting-time priority (WTP) [4], and mean-delay proportional (MDP) [10] scheduler.

The BPR schedulers dynamically adjust service rate allocations of classes to meet relative QoS requirements. The service rate allocation is based upon the backlog of classes at the scheduler. The WTP scheduler implements a well-known scheduling algorithm with dynamic time-dependent priorities. Each packet is assigned a time-dependent priority and is transmitted in order of priority. The MDP scheduler has a dynamic priority mechanism, but it uses estimates of the average delay of a class to determine the priority of that class.

Although the WFQ scheme is one of the most effective schedulers in providing delay differentiation, it has a limitation in that the burstiness of the traffic should be bounded with a leaky bucket shaper. Several schedulers have been devised to overcome this limitation. Our study also focuses on this point. We have designed a scheduler which guarantees the delay bound under bursty traffic environments without a traffic shaper as in WFQ.

As a solution to this limitation, the joint buffer management and scheduling (JoBS) scheme was proposed in [11]. It makes predictions on the delays of backlogged traffic and uses the predictions to update the service rate of classes and the amount of traffic to be dropped. Our approach is similar to that of [11], however, the main difference is whether the prediction error that inevitably occurs is applied to future control operations.

While most conventional schemes do not reflect the prediction error, our algorithm makes use of the deviation to improve the QoS. More specifically, it predicts traffic to arrive at the beginning of a time slot and measures the actual arrived traffic at the end of a time slot. The prediction deviation is derived at the beginning of the next time slot and is quantified to be reflected in the delay control mechanism for the next time slot. The target delay is adjusted to some extent, which is determined by the prediction error at every time slot. As the suggested algorithm continually compensates the prediction error for every time slot, it shows superior adaptability to bursty traffic and exponential traffic as compared with conventional approaches.

The remainder of this paper is organized as follows. In section II, an algorithm that provides quantitative DiffServ is developed. In section III, simulations to demonstrate the performance of the scheme are presented. In section IV, the details of the implementation of the algorithm are given. Finally, in section V, some concluding remarks are presented.

II. Adaptive Delay Differentiation Model

1. Service Differentiation Objective

It is assumed that there are M service classes, and class i+1 is better than class i in terms of service metrics. With this convention, the service guarantees for the classes can be expressed. An absolute delay guarantee for class i in a node is specified as

$$D_i <= D_i^*, \qquad \forall i \in \{1, \cdots, K\}, \tag{1}$$

where D_i^* is a desired delay for class i in a node. The proportional delay guarantee between class i and class i+1 is defined as

$$\frac{D_{i+1}}{D_i} = \alpha_i^*, \qquad \forall i \in \{K+1, \cdots, M-1\}, \tag{2}$$

where α_i^* is a constant that quantifies the proportional differentiation, $0 < \alpha_i^* < 1$.

2. Absolute Service Differentiation

As shown in Fig. 1, a time axis is slotted with interval T, and time slot n spans the time interval $[t_{n-1}, t_n]$.

The input rate $\tilde{\lambda}_i(n)$ of class i for time slot n is predicted at the beginning of time slot n, that is, at the time t_{n-1}, with the exponentially weighted moving average (EWMA) scheme like (3) with $\rho = 0.9$. Predicted values are indicated by a tilde (~).

$$\tilde{\lambda}_i(n) = (1-\rho)\frac{\sum_{k=n-N}^{n-2} \lambda_i(k)}{N-1} + \rho\lambda_i(n-1) \tag{3}$$

In general, traffic prediction is categorized into both formula-based and history-based schemes. The EWMA scheme is a

Fig. 1. Time axis notation.

typical history-based prediction scheme. In [12], it is concluded that history-based schemes are much more accurate in predicting TCP flows than formula-based ones. We chose 0.9 as a value of ρ in (3) based on our own simulation experience and that of others. The value of 0.9 was chosen to predict wireless LAN traffic through simulation in [13]. In (3), N is the number of the previous time slots that are taken to calculate the average input rate, and $\lambda_i(n)$ is the actual input rate of class i during the time slot n. It is obtained at the end of time slot n, that is, at the time t_n. The backlog $B_i(t)$ of class i at time t is derived from $R_i^{in}(t)$ and $R_i^{out}(t)$ as in (4), where $R_i^{in}(t)$ is the arrived traffic at the class i buffer, and $R_i^{out}(t)$ is the serviced traffic from the class i buffer in the interval $[0, t]$.

$$B_i(t) = R_i^{in}(t) - R_i^{out}(t). \tag{4}$$

Now, other parameters related to class i should be predicted to derive the service rate for the next time slot n. With the predicted input rate for the next time slot n of (3), the amount of class i traffic that is predicted to arrive during time slot n, $\tilde{R}_i^{in}(t; t \in [t_{n-1}, t_n])$, is given by

$$\tilde{R}_i^{in}(t; t \in [t_{n-1}, t_n]) = \tilde{\lambda}_i(n) \times (t - t_{n-1}). \tag{5}$$

Similarly, with the definition of the service rate $\gamma_i(n)$ of class i for the time slot n, the amount of class i traffic that is expected to be serviced at maximum during time slot n, $\tilde{R}_i^{out}(t; t \in [t_{n-1}, t_n])$, is given by

$$\tilde{R}_i^{out}(t, t \in [t_{n-1}, t_n]) = \gamma_i(n) \times (t - t_{n-1}). \tag{6}$$

With (5) and (6), the predicted backlog $\tilde{B}_i(t, t \in [t_{n-1}, t_n])$ of class buffer i for time slot n is derived as

$$\tilde{B}_i(t; t \in [t_{n-1}, t_n]) = B_i(t_{n-1}) + (\tilde{\lambda}_i(n) - \gamma_i(n)) \times (t - t_{n-1}). \tag{7}$$

Since the backlog is always positive, the service rate is constrained by

$$\gamma_i(n) \le \tilde{\lambda}_i(n) + \frac{B_i(t_{n-1})}{t - t_{n-1}}. \tag{8}$$

When the variable, $t - t_{n-1}$, is T, the maximum value of $\gamma_i(n)$ is obtained for the interval $[t_{n-1}, t_n]$. Therefore, the upper bound of the service rate constrained by the backlog is given by

$$\gamma_{i,\max,\text{backlog}}(n) = \tilde{\lambda}_i(n) + \frac{B_i(t_{n-1})}{T}. \tag{9}$$

The predicted delay $\tilde{D}_i(t; t \in [t_{n-1}, t_n])$ of a class i input packet arriving at buffer i at time t, $t \in [t_{n-1}, t_n]$, is described as

$$\begin{aligned}\tilde{D}_i(t; t \in [t_{n-1}, t_n]) &= \frac{\tilde{B}_i(t; t \in [t_{n-1}, t_n])}{\gamma_i(n)} + d_i(t) \\ &= \frac{B_i(t_{n-1}) + (\tilde{\lambda}_i(n) - \gamma_i(n)) \times (t - t_{n-1})}{\gamma_i(n)} + d_i(t).\end{aligned} \tag{10}$$

In (10), $d_i(t)$ is the residual service time of the packet being serviced when a class i packet arrives at the buffer at time t and is upper bounded by L/C, where L is the maximum packet length, and C is the output link capacity. The maximum predicted delay at time slot n, $\tilde{D}_{i,\max}(n)$, is obtained when $t - t_{n-1} = T$.

$$\tilde{D}_{i,\max}(n) = \frac{1}{\gamma_i(n)} \left\{ B_i(t_{n-1}) + T\left[\tilde{\lambda}_i(n) - \gamma_i(n) \right] \right\} + \frac{L}{C}. \tag{11}$$

Now with $\tilde{D}_{i,\max}(n) \le D_i^*$ for the absolute delay guarantee, and in (11), the lower bound for the service rate is obtained as

$$\gamma_i(n) \ge \frac{B_i(t_{n-1}) + T\tilde{\lambda}_i(n)}{D_i^* + T - \frac{L}{C}}. \tag{12}$$

As the service rate is always positive, the condition for T is derived as

$$T > \frac{L}{C} - D_i^*. \tag{13}$$

The predicted delay in (10) is for the fixed service rate, whose value is determined at the beginning of time slot n, and the predicted input rate for time slot n. To make the delay less than the desired absolute delay D_i^*, the lower bound of service rate is obtained. As packets of the same class require the same amount of absolute delay but the input rate may vary, the algorithm allows the service rate to be changed at every time slot. The resulting delay can be bigger than any number of time slots, but less than the desired absolute delay.

As we previously mentioned, it is a feature of our algorithm that the prediction error of the input rates over the current time slot is reflected in the derivation of the service rates for the next time slot. That is, as the actual input rates can be calculated at the end of time slot n, the delay prediction error can be derived. With the derived delay error at the current time slot, the target delay at the next time slot is changed to reflect the error. In

order to reflect the prediction error in the derivation of the service rates, the error $\Delta\lambda_i(n)$ between the measured input rates $\lambda_i(n)$ and the predicted input rates $\tilde{\lambda}_i(n)$ is defined as

$$\Delta\lambda_i(n) = \lambda_i(n) - \tilde{\lambda}_i(n). \quad (14)$$

With the definition of (14), the delay difference $\Delta D_{i,\Delta\lambda_i}(n)$ caused by the prediction error $\Delta\lambda_i$ on input rates is derived from (11) and given by

$$\Delta D_{i,\Delta\lambda_i}(n) = T\frac{\Delta\lambda_i(n)}{\gamma_i(n)}. \quad (15)$$

The actual maximum delay $D_{i,max}(n)$ over time slot n is adjusted to the extent of (15) and expressed as

$$\begin{aligned} D_{i,\max}(n) &= \left[\tilde{D}_{i,\max}(n) + \Delta D_{i,\Delta\lambda}(n)\right]^+ \\ &= \left[\frac{1}{\gamma_i(n)}\left\{B_i(t_{n-1}) + T\left[\tilde{\lambda}_i(n) + \Delta\lambda_i(n) - \gamma_i(n)\right]\right\} + \frac{L}{C}\right]^+, \end{aligned} \quad (16)$$

where $[z]^+ = \max\{z, 0\}$. As one of our objective is to find the appropriate value of service rate γ_i so that $D_{i,\max}(n) \le D_i^*$, the service rate for the time slot n+1 is derived from (16) as

$$\gamma_i(n+1) \ge \frac{B_i(t_n) + T\left\{\tilde{\lambda}_i(n+1) + \Delta\lambda_i(n)\right\}}{D_i^* + T - \frac{L}{C}}. \quad (17)$$

The service rate that can be allocated to class i is upper bounded by the output link capacity minus the minimum service rates needed by the other classes, that is,

$$\gamma_{i,\max,capacity}(n+1) = C - \sum_{j\neq i}\gamma_{j,\min}(n+1). \quad (18)$$

Therefore, the maximum service rate is given as

$$\gamma_{i,\max}(n+1) = \min\left\{\gamma_{i,\max,capacity}(n+1), \gamma_{i.\max,\text{back log}}(n+1)\right\}. \quad (19)$$

Therefore, the service rate can take any value $\gamma_i(n+1)$ with $\gamma_{i,\min}(n+1) \le \gamma_i(n+1) \le \gamma_{i,\max}(n+1)$ subject to the constraint $\sum_i \gamma_i(n+1) \le C$.

As the derivation of the service rate reflects the prediction error $\Delta\lambda_i(n)$, this scheme can follow the fluctuation of the input traffic more quickly than other conventional schemes that do not apply the error. Its expected result is a smaller number of packets that violate the delay bound. We define our algorithm as adaptive because the prediction error is adapted. We define the traditional algorithm as non-adaptive because it only decides the service rate based on the history of input traffic. Since the decision of the service rate is based on the prediction of the input traffic, the adaptive algorithm can be worse than the conventional algorithm when the increase of the arrival rate jumps from a positive value to a negative value or vice versa. Therefore, some schemes for smoothing drastic changes in the service rates caused by the abrupt alteration of the input rates are needed. For this issue, the minimum rates $\gamma_{i,\min}(n+1)$ are given from

$$\gamma_i(n+1) \ge \begin{cases} \dfrac{B_i(t_n) + T\tilde{\lambda}_i(n+1) + \Delta\lambda_i(n)}{D_i^* + T - \frac{L}{C}}, & \Delta\lambda_i(n) \ge 0, \\ \dfrac{B_i(t_n) + T\tilde{\lambda}_i(n+1)}{D_i^* + T - \frac{L}{C}}, & \Delta\lambda_i(n) < 0, \end{cases} \quad (20)$$

$$\gamma_i(n+1) \ge \begin{cases} \dfrac{B_i(t_n) + T\tilde{\lambda}_i(n+1) + \Delta\lambda_i(n)}{D_i^* + T - \frac{L}{C}}, & \Delta\lambda_i(n) \ge 0, \\ \dfrac{B_i(t_n) + T\left\{\tilde{\lambda}_i(n+1) + \Delta\lambda_i(n)\right\}}{D_i^* + T - \frac{L}{C}}, & \Delta\lambda_i(n) < 0. \end{cases} \quad (21)$$

Compared to (17), when the prediction error increases, both (20) and (21) conservatively increase the service rate. However, when the prediction error decreases, they show different behavior.

Equation (20) does not adapt the prediction error; rather, it uses the average input rate for the next period, while (21) adapts the average prediction error when the prediction error decreases. We call (20) an adaptive enriched increase non-adaptive decrease (AEIND) algorithm and (21) an adaptive enriched increase adaptive decrease (AEIAD) algorithm. It is easy to expect that the more conservative algorithm for decreasing the service rate (AEIND) can have better performance than the other one. However, the average service rate would increase to meet the absolute delay bound, and thus, reduce the resource utilization efficiency. Also, the delay variation could increase because the adaptive increase of the service rate sometimes reduces the packet delay so much that their delays are relatively much lower than the target delay. More detailed advantages and disadvantages are discussed in section III with the simulation.

3. Proportional Service Differentiation

In proportional DiffServe, the delay ratio between two adjacent classes should be fixed such that (2) is satisfied. As (16) indicates the actual maximum delay for time slot n, the

actual relative ratio of the delay for the service classes i and i+1 at time slot n is described as

$$\frac{D_{i+1}^{avg}(n)}{D_i^{avg}(n)} = \frac{\frac{1}{\gamma_{i+1}(n)}\left\{B_{i+1}(t_{n-1}) + T\left[\tilde{\lambda}_{i+1}(n) + \Delta\lambda_{i+1}(n) - \gamma_{i+1}(n)\right]\right\} + \frac{L}{C}}{\frac{1}{\gamma_i(n)}\left\{B_i(t_{n-1}) + T\left[\tilde{\lambda}_i(n) + \Delta\lambda_i(n) - \gamma_i(n)\right]\right\} + \frac{L}{C}} = \alpha_i(n). \tag{22}$$

As there is the possibility that $\alpha_i(n)$ in (22) deviates from the target value α_i^*, the delay ratio difference $\Delta\alpha_i(n) = \alpha_i^* - \alpha_i(n)$ is applied to the updated target value at the next time slot n+1 to compensate the deviation at the previous time slot n:

$$\alpha_i(n+1) = \alpha_i^* + \Delta\alpha_i(n). \tag{23}$$

From (22) and (23), the delay relation between time slots n and n+1 is given as

$$\frac{D_{i+1}(n+1)}{D_i(n+1)} = K \cdot \frac{D_{i+1}^{avg}(n)}{D_i^{avg}(n)}, \quad \text{where} \quad K = \frac{2\alpha_i^*}{\alpha_i(n)} - 1. \tag{24}$$

By converting K in (24) into y/x, (24) is expressed as

$$\begin{aligned} D_{i+1}(n+1) &= y \cdot D_{i+1}^{avg}(n), \\ D_i(n+1) &= x \cdot D_i^{avg}(n). \end{aligned} \tag{25}$$

To determine the service rate for time slot n+1, the case of $\Delta\alpha_i(n) > 0$ is considered first. As this case indicates that the delay of the higher class has been shortened and/or that of the lower class has been lengthened, the delay of the higher class and that of lower class for the next time slot should be decreased and increased respectively. That is, x and y should be $0 \le x \le 1$ and $1 \le y \le K$. We choose x and y values as $1/K$ and 1 that can reduce the delay of the class whose relative delay is higher than that of the other class.

Next, the case of $\Delta\alpha_i(n) < 0$ is considered. There are two possible ways to satisfy the condition. As the delay ratio should not be negative, two cases, $|\Delta\alpha_i(n)| < \alpha_i^*$ and $|\Delta\alpha_i(n)| \ge \alpha_i^*$, have different solution procedures.

First, we consider the case of $|\Delta\alpha_i(n)| < \alpha_i^*$, where delay ratio is not negative. Applying the same logic as in the case of $\Delta\alpha_i(n) > 0$, the area of x and y is $1 \le x \le 1/K$ and $0 \le y \le 1$. The values of x and y are determined to be 1 and K.

Finally, the case of $|\Delta\alpha_i(n)| \ge \alpha_i^*$ is considered. In this case, the higher class has much higher delay than the lower class. Since K becomes negative in this condition, x is negative, and y is positive. Without violating the relative ratio, K can be recalculated as $K = x/(x-y)$, and K always satisfies the bound $K < 1$. The other steps are the same as in the case of $|\Delta\alpha_i(n)| < \alpha_i^*$.

Descriptions of all cases are summarized as follows:

(Case 1) $\Delta\alpha_i(n) > 0$:

$$\begin{aligned} D_{i+1}(n+1) &= D_{i+1}(n), \\ D_i(n+1) &= K \cdot D_i(n). \end{aligned}$$

(Case 2) $\Delta\alpha_i(n) < 0$: (26)

(Case 2-1) $|\Delta\alpha_i(n)| < \alpha_i^*$:

$$\begin{aligned} D_{i+1}(n+1) &= K \cdot D_{i+1}(n), \\ D_i(n+1) &= D_i(n). \end{aligned}$$

(Case 2-2) $|\Delta\alpha_i(n)| \ge \alpha_i^*$:

$$\begin{aligned} K &= x/(x-y), \qquad x < 0, y > 0, \\ D_{i+1}(n+1) &= K \cdot D_{i+1}(n), \\ D_i(n+1) &= D_i(n). \end{aligned}$$

With the values $D_{i+1}(n+1)$ and $D_i(n+1)$, we can derive the service rate which is given as

$$r_k(n+1) \ge \frac{B_k(t_n) + T\tilde{\lambda}_k(n+1)}{D_k(n+1) + T}, \qquad k = i,\ i+1. \tag{27}$$

The service rate that can be allocated to class i is upper bounded by the output link capacity minus the minimum service rates needed by the other classes as in (18). Therefore, the maximum service rate is given in (19). Thus, the service rate can take any value $\gamma_i(n+1)$ with $\gamma_{i,\min}(n+1) \le \gamma_i(n+1) \le \gamma_{i,\max}(n+1)$ subject to the constraint $\sum_i \gamma_i(n+1) \le C$.

III. Evaluation

Simulations for the examination of the efficiency of the three proposed algorithms and for comparison with the conventional algorithm are presented in this section. Using the OPNET simulator, we simulate four algorithms with the simple network topology illustrated in Fig. 2. Our focus is mainly the service delays and service rates of the four algorithms as well as their distributions. The important simulation parameter is the length of the time period T, which is the time interval adopted in the discrete packet prediction model. It is also the service rate adjustment period. Intuitively, if T is large, it will introduce more distortions into the packet arrival prediction scheme and the updates of the service rates are infrequent. Thus, the delay time will become more uncontrollable. On the other hand, if T is small and the service rate refreshes frequently, the delay and delay differentiation become more predictable and controllable.

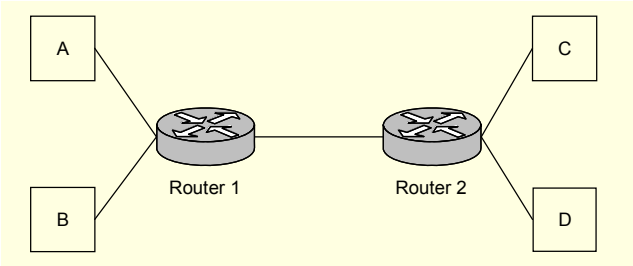


Fig. 2. Simulated network topology.

That is, it is already proven that a shorter prediction time period yields a more accurate result but a heavier overhead for the router than a shorter period [14]. For this reason, we fix the same time period T of the prediction for every simulation. The value N is also set to 5.

In Fig. 2, nodes A and B generate packets, and the packets arrive at nodes C and D after traveling through routers 1 and 2. Each source node generates a number of traffic flows, whose inter-arrival time and packet size are exponentially distributed with means of 0.001 seconds and 1,000 bits, respectively. In nodes A and B, we create two absolute service classes, classes 1 and 2, and two proportional classes, classes 3 and 4, where class 4 is higher priority class than class 3. The delay requirements are set to 20 ms and 40 ms respectively for the absolute delay, and $\alpha_3^* = 0.5$ for the proportional delay. Traffic load distribution is set to 30% for classes 1 and 3 and 20% for classes 2 and 4. Link capacity is set to 100 Mbps and the link propagation delay is assumed to be negligible.

1. Absolute Delay Differentiation

We uniformly collect 1,000 statistics that have the maximum packet delay throughout the absolute delay simulation.

Figure 3 shows the maximum queueing delay of class 1 for exponential traffic. Since the non-adaptive algorithm shown in Fig. 3(a) is based on the average input rate of traffic, it frequently exceeds the delay target, while AEIND shown in Fig. 3(c) has the lowest number of packets that exceed the delay target. The cumulative distribution function (CDF) of the maximum packet delay shown in Fig. 4 demonstrates the performance of each algorithm more clearly. Only 5% of packets exceed the delay bound when the adaptive AEIND algorithm is used.

Since current Internet traffic is not exponential, more realistic traffic that reflects the burstiness characteristic is required. For bursty traffic, we create hundreds of flows whose inter-arrival time and flow durations follow Pareto distribution with a shaper value of 1.9. The packet size is exponentially distributed with the mean value of 1,000 bits. Using this traffic, we simulate four algorithms and draw CDFs of the maximum delay for each algorithm as shown in Fig. 5.

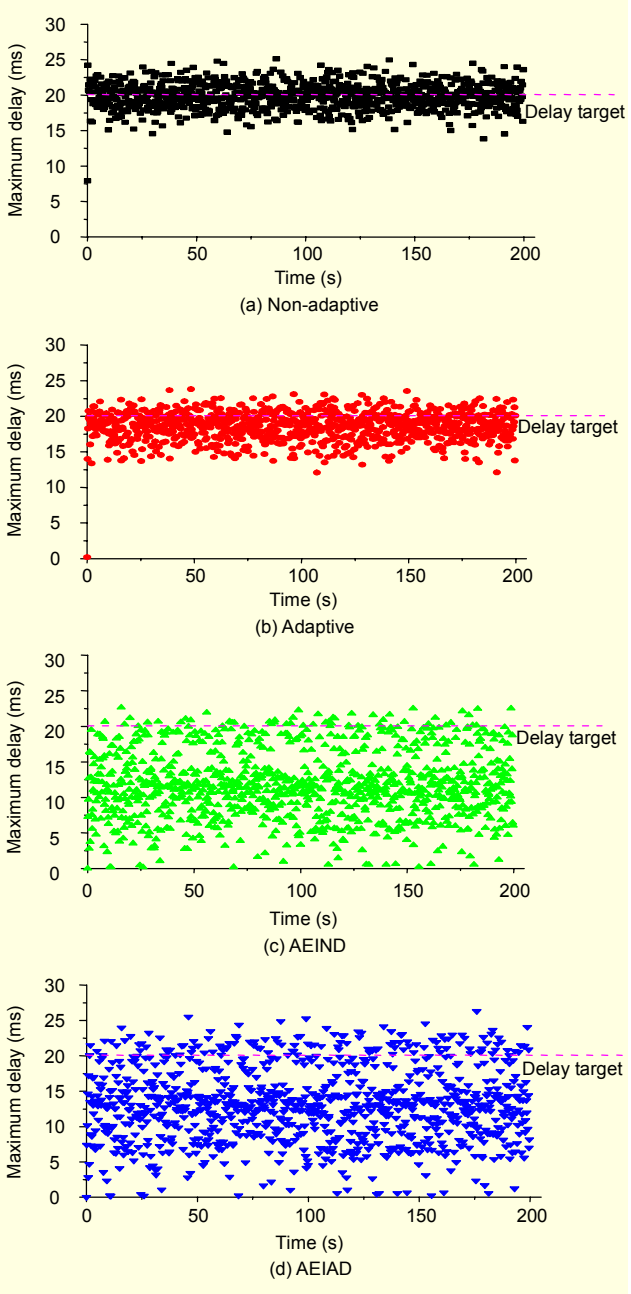


Fig. 3. Delay of class 1 with exponential traffic load.

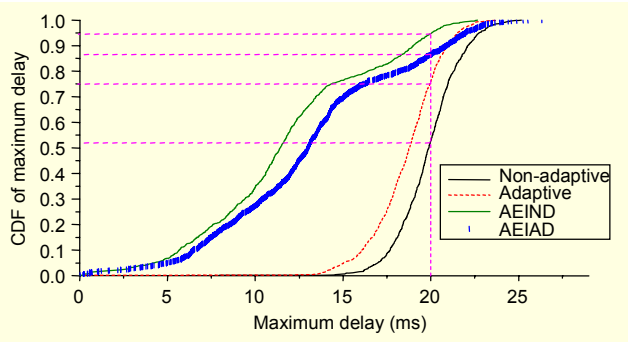


Fig. 4. Delay of class 1 with exponential traffic load.

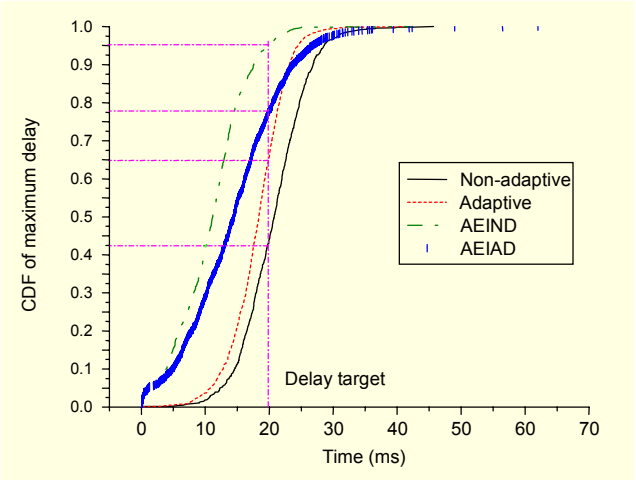


Fig. 5. Delay of class 1 with Pareto traffic load.

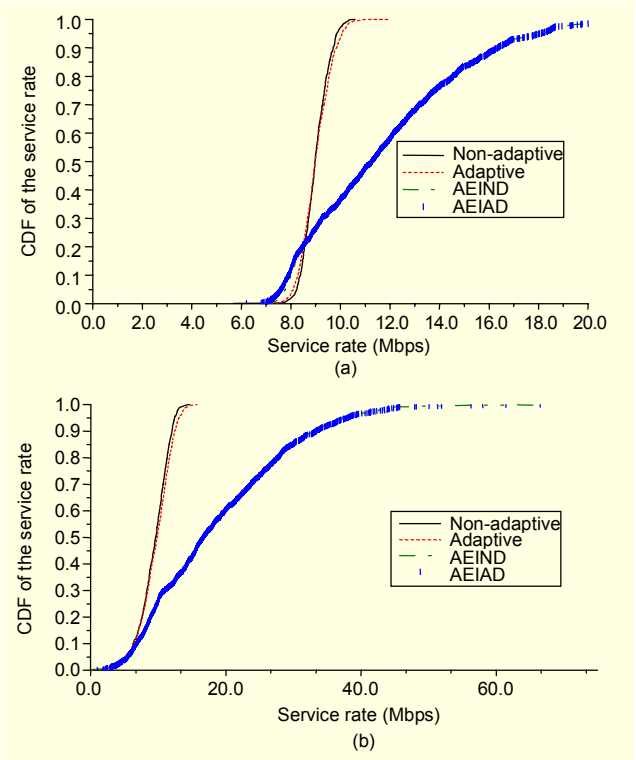


Fig. 6. CDF of the service rate for exponential and bursty traffic: (a) traffic with exponential distribution and (b) traffic with Pareto distribution.

The delay performance is more obvious when running a simulation with bursty traffic. Less than 45% of the total packets in the non-adaptive algorithm meet the delay requirement while other algorithms satisfy the delay requirement up to 65%, 77%, and 95%. This shows that the AEIND algorithm guarantees better delay performance with both exponential and Pareto traffic. This is because AEIND and AEIAD exponentially increase the service rate when they perceive the traffic increase from the prediction error. AEIND shows better performance than AEIAD. Moreover, AEIAD has the highest packet delay of all the algorithms, which is over 60 ms. Reducing the service rate according to the prediction error can lead to adverse effects and can cause higher delay than the non-adaptive algorithm. Therefore, reducing the service rate conservatively is better than adapting the prediction error when the rate of input traffic decreases.

From the resource utilization point of view, the AEIND algorithm is not always better than other algorithms.

Figure 6 shows CDFs of the service rates of four schemes. The average service rates of AEIND and AEIAD are 11.78 Mbps and 11.82 Mbps, respectively, while those of the non-adaptive and adaptive algorithms are 9.0 Mbps and 9.02 Mbps. When input traffic is Pareto distributed, the average service rates for AEIND and AEIAD increase up to 18.5 Mbps and 18.6 Mbps, respectively. However, for the non-adaptive and adaptive algorithms the average rates are only 9.32 and 9.56 Mbps, almost half those of the other two algorithms. Therefore, the tradeoff between service rate and delay is clearly derived. The choice of algorithm depends on the delay sensitivity and the cost of the resource represented as the service rate. The adaptive algorithm would possibly be a reasonable choice considering both delay sensitivity and the resource utilization issue. The adaptive algorithm shows superior delay performance to the non-adaptive algorithm and has a lower service rate than that of AEIND or AEIAD.

2. Proportional Delay Differentiation

In observing the proportional delay, we set the same Pareto traffic generation parameters used in the absolute delay. The results are shown in Fig. 7. In this simulation, the adaptive algorithm shows overall improvement against the non-adaptive algorithm; therefore, we can conclude that our algorithm is robust to a bursty traffic environment.

Another simulation is also performed for delay ratio with various traffic loads. Most work-conserving schedulers cannot meet the proportional delay differentiation when the traffic load is low [15]. In [15], the authors suggested that it is because their algorithms assume that the queue buffer is always saturated. However, there is a greater possibility of queue buffers being empty when the traffic load is low.

Our algorithm solves this problem by dynamically allocating the service rate and adapting the prediction error in order to maintain a certain amount of buffer traffic, though it is based on the work-conserving scheduler. As shown in Fig. 8, the delay ratio does not depend on the traffic load. Accuracy of the delay ratio is a little worse than that of other algorithms [15]. The attempt by the adaptive algorithm to reduce the relative delay

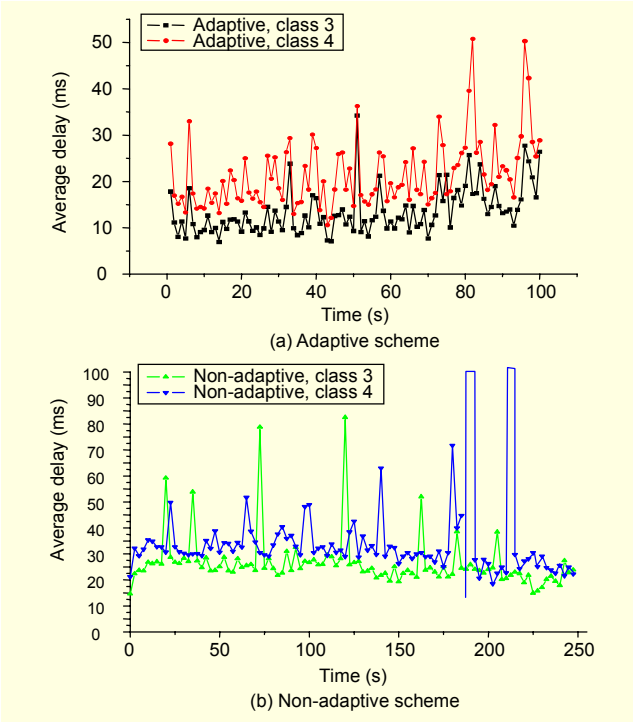


Fig. 7. Proportional delay to bursty traffic.

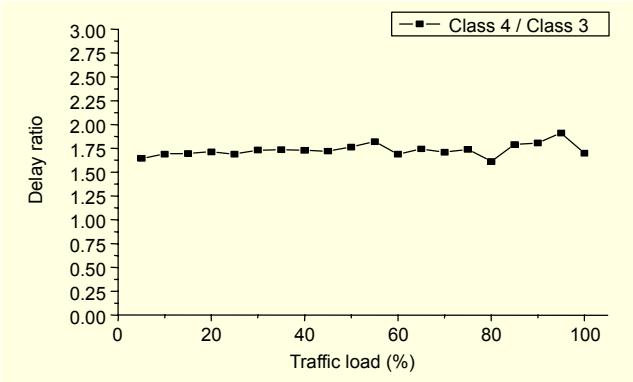


Fig. 8. Dependency of proportional delay on input load.

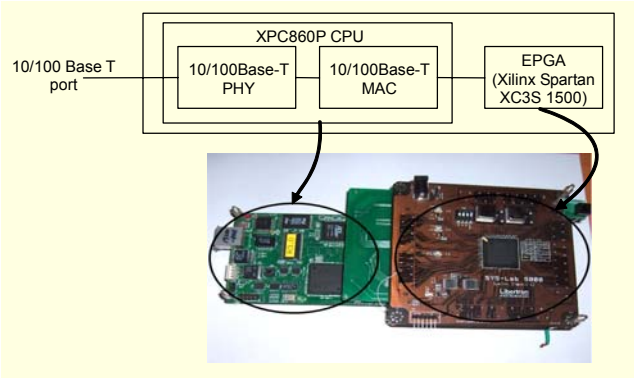


Fig. 9. Test board.

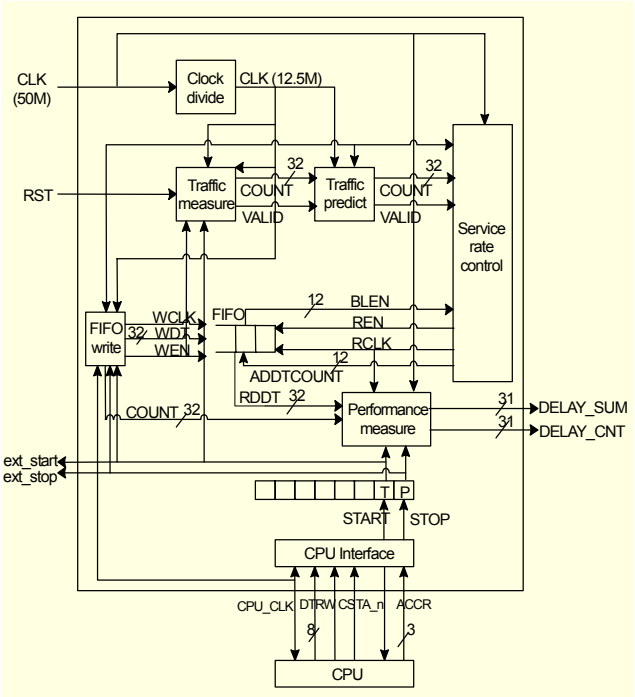


Fig. 10. Internal blocks for the absolute delay differentiation algorithm.

seems to cause the accuracy of the delay ratio. However, the tendency of the delay differentiation between higher and lower class is still maintained.

IV. Implementation

The suggested algorithm is implemented with VHDL on Xilinx Spartan XC3S1500 FPGA on the test board shown in Fig. 9. It is mainly composed of 10/100Base-T PHY/MAC and XPC860P CPU.

1. Absolute Delay Differentiation

Figure 10 shows the internal blocks for the absolute delay differentiation algorithm.

The clock divide block divides the 50 MHz clock to generate a 12.5 MHz clock to accommodate the 100 Mbps Ethernet signal with 8-bit operations. The FIFO write block accounts for receiving the Pareto distributed traffic, which is generated from the CPU board, and inserting it to the FIFO. The traffic measure block constantly measures the input traffic rate and sends the measured traffic value, that is, $\lambda_i(n)$ in (14), to traffic predict block. The traffic predict block predicts $\tilde{\lambda}_i(n)$ with $\lambda_i(n)$ as in (3), and the predicted one is sent to the service rate control block. The service rate control block derives the service rate $\gamma_i(n)$ as in (17), (18), and (19) with the incoming $\lambda_i(n)$, $\tilde{\lambda}_i(n)$, and $B_i(t_{n-1})$ from the class buffer. Finally, the performance measure block calculates the delay

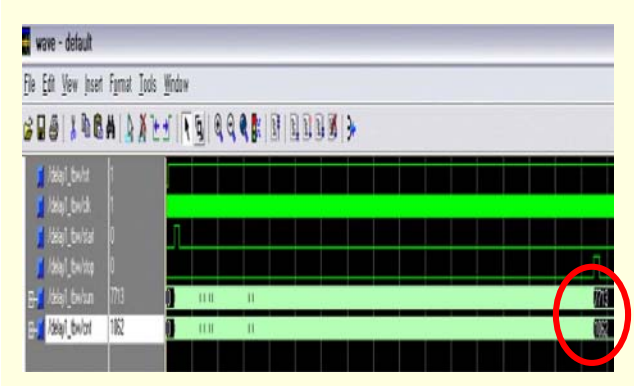


Fig. 11. Post route simulation result (target delay: 1 μs).

Table 1. Delay performance measured at the test board.

Target delay (μs)	Measured delay (μs)
1	0.5
7	5
15	11
23	17
87	66
255	176

by measuring the difference between the arrival time and the service time of the traffic.

Delay_sum and delay_cnt, which are the outputs of performance measure block, indicate the sum of delays for the measured packets and the number of packets during the measured time, respectively. As the input packets arrive at the rate of 12.5 MHz, the accurate average delay is derived from (delay_sum / delay_cnt) / 12.5 μs.

Figure 11 shows the result of the post route simulation when the target delay is set to 1 μs. The post route simulation is performed with ModelSim SE 5.7d. As delay_sum and delay_cnt are 7,713 and 1,062 respectively, the target delay is satisfied because the simulated delay is

$$\frac{7713}{1062} \times \frac{1}{12.5 \times 10^6\ [\text{Hz}]} = 0.58\ \ [\mu\text{s}].$$

The post route simulation is performed for target delays of 3 μs, 5 μs, and 15 μs, and the resulting delays are 1.84 μs, 3.26 μs, and 9.54 μs, respectively. All the delay constraints are satisfied.

Table 1 shows the delay results measured at the test board, where the implemented VHDL code is downloaded. As the results indicate, the proposed algorithm guarantees the target delay.

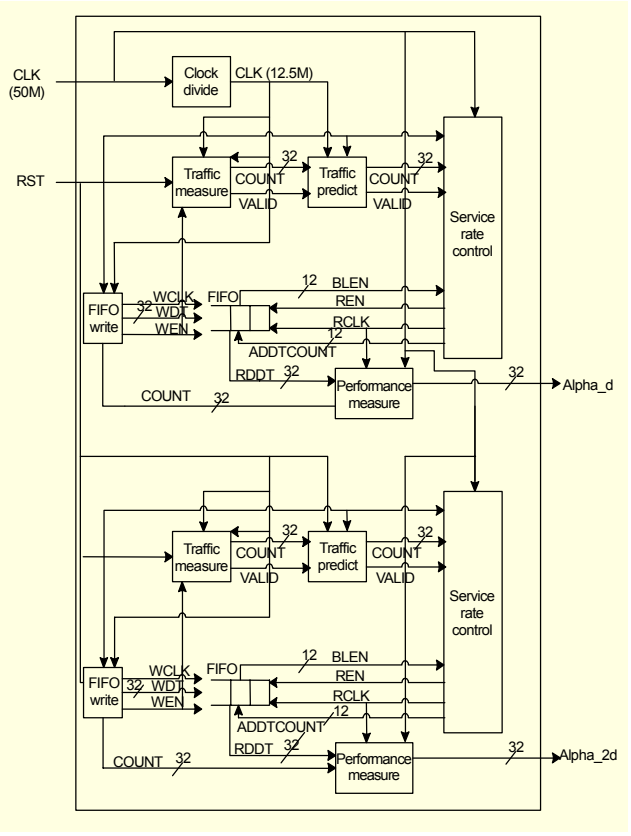


Fig. 12. Internal blocks for the proportional delay differentiation algorithm.

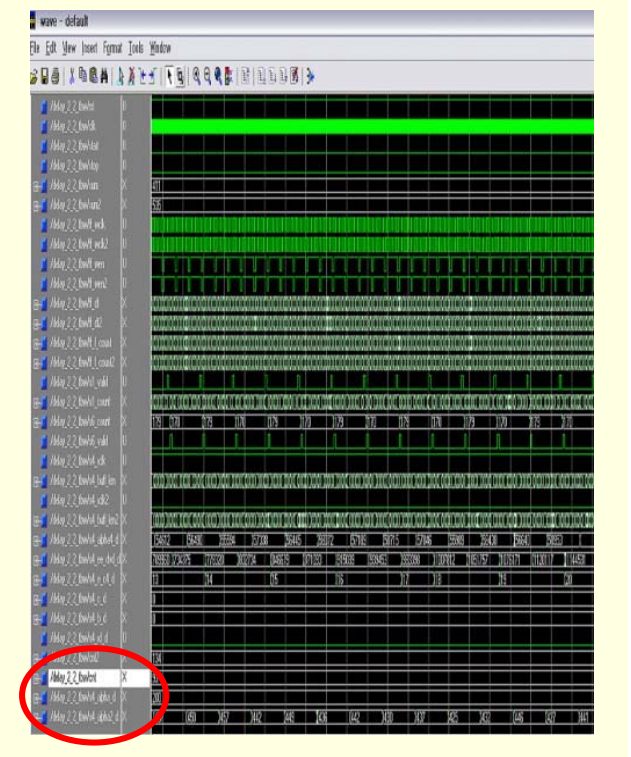


Fig. 13. Post route simulation result for the target delay ratio of 2.

2. Proportional Delay Differentiation

Figure 12 shows the block diagram for the proportional delay differentiation algorithm. As it handles only two classes, classes 1 and 2, there are two FIFOs. The functions of other blocks are identical to those of the absolute delay differentiation algorithm.

In Fig. 12, signals alpha_d and alpha_2d represent the delays of classes 1 and 2, respectively. Therefore, the relative delay ratio for the two classes is given as a ratio of alpha_d and alpha_2d.

Figure 13 shows the post route simulation results for the target delay ratio of 0.5. Referencing two signals, alpha_d and alpha_2d, the bottom of the figure shows that the delay ratio is about 0.4545. After running another post route simulation, whose result is not shown in the figure, the delay ratio of 0.3125 was obtained for the target delay ratio of 1/3. Thus, the results demonstrate that our proportional delay differentiation algorithm closely follows the target delay ratio.

V. Conclusion

In this paper, a delay differentiation algorithm that provides absolute and proportional QoS is proposed. The main feature of this algorithm is that it continually adjusts the target delay with reference to the traffic prediction deviation in the preceding time section. We have demonstrated that the proposed schemes perform well in terms of providing absolute and proportional QoS. Moreover, our algorithm shows better adaptability to traffic fluctuation than the conventional approach, and it presents a feasible approach to future Internet where QoS differentiation is essential and bursty traffic is prevalent.

We have shown that our algorithm can be realized by making a prototype system with VHDL. The prototype system is more applicable to high-speed Ethernet systems than the conventional software-based approach, which is implemented in a network processor, because it is a hardwired implementation.

In our study, only the delay factor was handled for the QoS differentiation. Loss and throughput should also be considered to increase applicability. In addition, an optimization process should be applied to the service rate derivation to increase the utilization of the output link. These issues are very important and will be further explored in the future.

References

[1] S. Shenker and J. Wroclawski, *General Characterization Parameters for Integrated Service Network Elements*, IETF RFC 2215, Sept. 1997.

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

[3] I. Stoica and H. Zhang, "Providing Guaranteed Services without Per Flow Management," *Proc. ACM SIGCOMM*, Cambridge, MA, Sept. 1999, pp. 81-94.

[4] C. Dovrolis, D. Stiliadis, and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling," *Proc. ACM SIGCOMM*, Cambridge, MA, Sept. 1999, pp. 109-120.

[5] A.K. Parekh and R.G. Gallager, "A Generalized Processor Sharing Approach to Flow Control: The Single Node Case," *Proc. INFOCOM*, vol. 2, May 1992, pp. 915-924.

[6] A. Demers, S. Keshav, and S. Shenker, "Analysis and Simulation of a Fair Queueing Algorithm," *Internetworking: Research and Experience*, vol. 1, no. 1, 1990, pp. 3-26.

[7] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," *ACM Trans. Computer Systems*, vol. 9, May 1991, pp. 101-124.

[8] M. Katevenis, S. Sidiropoulos, and C. Courcoubetis, "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip," *IEEE J. Selected Areas in Communications*, vol. 9, Oct. 1991, pp. 1265-1279.

[9] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," *IEEE/ACM Trans. Networking*, vol. 4, no. 3, June 1996, pp. 375-385.

[10] T. Nandagopal, N. Venkitataman, R. Sivakumar, and V. Barghavan, "Delay Differentiation and Adaptation in Core Stateless Networks," *Proc. IEEE INFOCOM*, Tel-Aviv, Israel, Apr. 2000, pp. 421-430.

[11] L. Liebeherr and N. Christin, "JoBS: Joint Buffer Management and Scheduling for Differentiated Services," *Proc. IWQoS 2001*, Karlsruhe, Germany, June 2001, pp. 404-418.

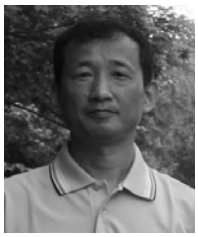
[12] H. Qi, C. Dovrolis, and M. Ammar, "On the Predictability of Large Transfer TCP Throughput," *Int'l Computer and Telecommunications Netoworking*, vol. 51, no. 14, Oct. 2007, pp. 3959-3977.

[13] C. Hu, R. Zheng, J.C. Hou, and L. Sha, "A Microscopic Study of Power Management in IEEE 802.11 Wireless Networks," *Int'l J. Wireless and Mobile Computing*, vol. 1, no. 3/4, 2006, pp. 165-178.

[14] J. Qiu and E. Knightly, "Measurement-Based Admission Control with Aggregate Traffic Envelopes," *Proc. 10th IEEE ITWDC*, vol. 9, no. 2, Sept. 1998, Ischia, Italy, pp. 199-210,

[15] Y. Lai and A. Chaing "A Non-work-conserving Scheduler to Provide Proportional Delay Differentiated Services," *GLOBECOM 2004*, vol. 3, pp. 1723-1727.

Junghoon Paik received the BS, MS, and PhD degrees in electronic engineering form Hanyang University, Rep. of Korea, in 1986, 1988, and 2005, respectively. He worked for ETRI from 1988 to 1996, where he was involved in developing transmission and switching systems. Since joining Dong-Ah Institute of Media and Arts in 2002, he has been a professor. His research interests include transmission and switching system design, analysis, and implementation.



Jeong-dong Ryoo is a principal member of research staff in ETRI, Rep. of Korea. He holds Masters and PhD degrees in EE from Polytechnic University, Brooklyn, NY, USA, and a Bachelor's degree in EE from Kyungpook National University, Rep. of Korea. After completing his PhD study in the area of telecommunication networks and optimization, he started working for Bell Labs, Lucent Technologies, NJ, in 1999. While he was with Bell Labs, he was mainly involved in performance analysis, evaluation, and enhancement study for various wireless and wired network systems. Since he left Bell Labs and joined ETRI in 2004, his work has been focused on next generation network and carrier class Ethernet technology research. He has especially participated in OAM and protection standardization activities in ITU-T. He co-authored *TCP/IP Essentials: A Lab-Based Approach* (Cambridge University Press, 2004). He is a member of the Eta Kappa Nu association.

Bheom-Soon Joo received the BS degree from Seoul National University, Rep. of Korea and the MS degree from KAIST, Rep. of Korea in electrical and electronic engineering. Since December 1983, he has been working with ETRI and has participated in system development projects in areas including, TDX10, ATM exchange, a 10-gigabit Ethernet switch system, and a carrier-class Ethernet system. Currently, he is a leader of a carrier Ethernet research team. His research interests include Ethernet technology, broadband convergence networks, high-speed networks, and network synchronization.