Quality of Service Support in Differentiated Services Packet Networks

Spiridon Bakiras and Victor O.K. Li
The University of Hong Kong
Department of Electrical & Electronic Engineering
Pokfulam Road
Hong Kong
email: {sbakiras,vli}@eee.hku.hk

Abstract—**During the past few years, new types of Internet applications which require performance beyond the best-effort service that is provided by the current Internet have emerged. These applications include the transmission of voice and video, which require a fixed end-to-end delay bound in order for the end-user to perceive an acceptable level of service quality. The Differentiated Services (Diffserv) model has been proposed recently to enhance the traditional best-effort service, and provide certain Quality of Service (QoS) guarantees to these applications. Its current definition, however, does not allow for a high level of flexibility or assurance and, therefore, it can not be widely deployed. In this paper, we introduce a new protocol for a Diffserv architecture which provides a simple and efficient solution to the above problem. It is a complete protocol, in the sense that it deals with the issues of packet scheduling, admission control, and congestion control. We will show, through experimental results, that our proposed protocol can improve the flexibility and assurance provided by current solutions, while maintaining a high level of network utilization.**

I. Introduction

The Internet was originally designed to provide best-effort services to all users. The Internet today does not provide resource reservation, and all packets are treated equally in a First-Come-First-Served (FCFS) order. In the past this approach worked very well, since the applications that made use of the Internet did not require a fixed delay bound (e.g. telnet, ftp, e-mail, etc.). However, the dramatic increase of the capacity in the Internet core, and the development of powerful compression techniques, have allowed the creation of new types of applications such as Internet telephony, video-conferencing, and streaming video. These applications are called real-time, since they require a fixed end-to-end delay bound. To address this problem, the Internet Engineering Task Force (IETF) has proposed two different service models, namely, Integrated Services (Intserv) [1] and Differentiated Services (Diffserv) [2]. The purpose of these architectures is to provide end-to-end QoS guarantees to real-time applications.

The main idea behind Intserv is resource reservation. It utilizes the RSVP [3] signaling protocol to reserve resources in each intermediate router between the source and the destination, so as to provide application-specific QoS requirements. The Intserv model, however, has some major limitations on widespread deployment. Each router has to perform management on a per flow basis. In other words, each router has to perform per flow signaling procedures (for resource reservation), perform per flow classification and scheduling, and maintain per flow forwarding and QoS state. This approach will work well when the number of flows is small, but it will be hard to implement if the number of flows is large. In other words, Intserv is not scalable.

The above limitations of Intserv led to the introduction of the Diffserv model by the IETF. The Diffserv architecture, in general, is substantially different from Intserv. First of all, Diffserv distinguishes between boundary and core routers. In a Differentiated Services (DS) capable domain, only the boundary routers process traffic on a per flow basis. The core routers forward packets based on Per Hop Behaviors (PHBs). In particular, each packet is forwarded according to the DS field (1 byte) in the IP header. There is a limited number of service classes that are defined in the DS field, and each application may select any of them based on the required type of service. Since there is no need to maintain per flow states in the core routers, the Diffserv model is more scalable. Even though Diffserv scales well with increasing number of flows, it still has some key disadvantages compared to the Intserv model. First, its service is not flexible, since the application can not specify the required end-to-end delay. In its current definition [4], Diffserv can only provide a static priority service discipline to the different classes, which can not be translated into end-to-end delay bounds. Second, the issue of admission control has not been defined yet and, therefore, QoS guarantees can not be provided. However, since the number of flows in the Internet core is expected to be very large, the more scalable Diffserv architecture seems to be more appropriate for the future Internet.

In this work, we propose a simple and efficient protocol for a Diffserv architecture. More specifically, we will present a new packet scheduling algorithm, where the priority of each packet will change continuously during its transmission from the source to the destination. In addition, we will introduce an admission control algorithm which will try to ensure that all packets will meet their end-to-end delay bounds with a very high probability. We will show, through experimental results, that our proposed protocol can improve the flexibility and assurance provided by current solutions, while maintaining a high level of network utilization. The rest of the paper is organized as follows. Our proposed packet scheduling algorithm is presented in Section II, while in Section III we describe the admission control procedure. Section IV presents the results of the simulation experiments, and Section V concludes our work.

II. Packet Scheduling

Our goal is to design a packet scheduling algorithm along with an admission control procedure, which does not require routers to maintain per flow state. The main idea behind our proposed algorithm is that most of the time the queueing delay of a random packet will be very small compared to its local (at that queue) delay bound. Traditional scheduling disciplines, such as Weighted Fair Queueing (WFQ) [5] or Earliest Deadline First (EDF) [6], do not take this fact into account in packet scheduling. They schedule each packet in exactly the same way at every node, without considering the delay already experienced at the previous nodes. Other protocols, such as the Jitter Virtual Clock [7], hold the packets that arrived earlier than expected at a rate-controller, so as to reconstruct the per flow traffic and remove the delay-jitter. This, of course, will degrade channel utilization. As part of our protocol we will introduce a new packet scheduling mechanism, called Priority-EDF (P-EDF). It is based on the EDF service discipline, but it also takes into account the delay that each packet experiences during its transmission from node to node. Packets that are ahead of their deadlines at one node will be given lower priority at the next node, while those that are behind will be given higher priority.

Without loss of generality, let us define a *time unit* to be a period of time equal to 0.1 ms. This time unit will enable us to represent the priority of each packet (which is its per node delay bound) as a small

0-7803-7097-1/01/$10.00 ©2001 IEEE

integer number. We will discuss this issue again in the next subsection, where we describe the details of the implementation. Suppose a connection i has an end-to-end delay requirement of d_i time units, and that the length of the path from the source to the destination is n_i hops. Then, we can assign a per node delay bound of $\bar{d}_i = \lfloor \frac{d_i}{n_i} \rfloor$ time units in each intermediate router, for that particular connection. Each packet k from connection i will then be assigned a priority p_i^k which will be the per node delay bound. This priority will be equal to $\bar{d}_i$ when the packet arrives at the first router of the path. It will then be updated continuously just before the transmission of the packet from each router. The priority will be updated to

$$p_i^{k'} = \left\lfloor \frac{(n_i + 1)p_i^k - w_i^k}{n_i} \right\rfloor \quad (1)$$

where n_i is the number of remaining hops, and w_i^k is the waiting time for that packet at the router (in time units). This priority will be used as an index to insert the packet in the appropriate place of the priority queue at the next router. If the priority value in equation (1) becomes negative, it means that the deadline (end-to-end) of the packet has already expired. As a result, this packet may be discarded immediately at the next or current router. Upon its arrival at a router, each packet is assigned a priority value which is equal to the arrival time plus the per node delay bound (p_i^k). Packets are transmitted in order of increasing priority value (i.e. EDF scheduling), and the number of remaining hops is used to break any tie: packets with smaller number of remaining hops are transmitted first, since they are more urgent (packets with larger number of remaining hops may be given priority at the following hops, if they miss their current per node delay bound).

Note that the per node delay bound for a random packet is not tight. Most of the packets will be transmitted before or sometimes after their deadlines. Equation (1) will rearrange the priorities so that each packet is treated according to its previously experienced delay. Any delay gain or loss at a router will be split equally among the per node delay bounds of the remaining hops.

A. Implementation issues

The proposed scheduling algorithm requires only minor changes in the current IP protocol. For its implementation we need to include two new states (integers) in the IP packet: the priority of the packet p_i^k, and the number of remaining hops n_i. These states will be updated during the transmission of the packet from a router. With 15 bits we can easily encode these two states in the IP packet; 11 bits can represent a per node delay bound (i.e. priority) of up to 200 ms, while 4 bits can represent a path length of up to 16 hops. This is mainly the reason why we chose the time unit to be equal to 0.1 ms. It is small enough to differentiate between individual priorities, and it can be easily encoded in the IP packet as an integer number, avoiding the use of floating point arithmetic. For the actual implementation of the protocol we may insert an additional header between the layer 2 and layer 3 headers, which will include these two state variables (similar to the MPLS concept).

Another implementation issue is due to the complexity of the priority queue. When the queue length is large, packet insertion may be the bottleneck for a router that has to forward packets at a speed of several Gbps. This, however, is a more general problem that is inherent to the priority queue service disciplines, and is beyond the scope of this paper. In the next section we will indicate how to perform some kind of congestion control, and prevent the individual queues from reaching large values.

B. Related work

There have been some similar packet scheduling techniques reported in the literature which aim to provide better service to real-time applications. In [8] the authors proposed a scheme called FIFO+, where each router measures constantly the average delay for each service class. If a packet is treated better than the class average, it will be given lower priority at the following hops. If, on the other hand, it experienced larger delay than the average, it will be given higher priority. The objective was to minimize the jitter across all the hops of a given path. However, the authors did not consider the problem of providing end-to-end delay guarantees.

An approach which is more similar to ours was developed independently in [9] with the Budgeted Residual-Life Dependent Discipline (BURD), and in [10] with the Hop Laxity (HL) scheduling mechanism. These schemes are practically identical, and they schedule each packet according to a dynamic priority discipline, where the priority of each packet is given by [9] $Q(t) = \frac{(Age(t) - LT)}{Hops(t)}$, where LT is the end-to-end delay bound, $Age(t)$ is the age of the packet at time t, and $Hops(t)$ is the number of remaining hops at time t (including the current node). This scheme, however, has two disadvantages. First, it has a constant $O(N)$ complexity for the dequeue operation (where N is the length of the queue), since the priority of each packet is changing according to both the age of the packet, and the number of remaining hops. As a result, all the packets in the queue have to be searched in order to find the one with the highest priority. Our scheme, though, requires only the implementation of a priority queue which has $O(\log N)$ enqueue, and $O(1)$ dequeue complexity. Second, this scheme will obviously favor packets with small end-to-end delay bounds, since the priority of each packet increases at a rate which is inversely proportional to the number of remaining hops.

III. Admission control

In order for an admission control algorithm to provide deterministic or statistical delay guarantees, per flow state has to be maintained in each router about the traffic characteristics of each flow (typical examples are the algorithms proposed in [11], [12], [13]). In our protocol, we do not maintain per flow state, and so we are not able to prove analytically that the admission control procedure can indeed provide end-to-end delay guarantees. Even if we keep the per flow state in each router, since the priority of each packet changes continuously along the path of intermediate routers, it is practically impossible to provide analytical results. We will follow, instead, an intuitive approach to admission control, and we will try to verify it with experimental results.

Each router will reserve enough resources to accommodate the average per node delay of all the active connections. In other words, it will assume that there is only one priority class with a specific per node delay bound. If, for example, there are two active connections with arrival rates λ_1 and λ_2, and per node delay requirements $\bar{d}_1$ and $\bar{d}_2$, respectively, then the router will guarantee that the probability the queue length under a FCFS service discipline exceeds the value $d_{avg} = \frac{(\lambda_1 \bar{d}_1 + \lambda_2 \bar{d}_2)}{(\lambda_1 + \lambda_2)}$ is bounded. Since the priority of each packet is updated continuously in order for every packet to meet its deadline, we can argue that the end-to-end delay requirement for every connection will be met with a very high probability.

Every router will only keep two variables for the purpose of admission control: the aggregate arrival rate $\lambda = \sum_k \lambda_k$, and the average per node delay d_{avg}. When a new request arrives, each intermediate router will check whether it can support the requested per node delay $\bar{d}_i$ of the new connection (which has an arrival rate λ_i). If the answer is affirmative in every router, the connection will be accepted and each router will update its two variables as follows

$$\lambda' = \lambda + \lambda_i$$

$$d'_{avg} = \frac{(\lambda d_{avg} + \lambda_i \bar{d}_i)}{\lambda'}$$

When a connection is terminated, these variables will be updated accordingly. The admission control procedure may be performed in either a centralized (e.g. bandwidth broker architecture) or distributed (e.g. RSVP-like signaling protocol) manner.

0-7803-7097-1/01/$10.00 ©2001 IEEE

To complete the admission control procedure, we need a formula that will actually perform the admission control, that is, a formula which will guarantee, probabilistically, that the queue length in each router will not exceed the value of d_{avg}. Recently, several studies have shown that Internet traffic exhibits self-similarity [14], [15]. Moreover, it has been shown in [16] that interarrival times produced by a Pareto distribution generate asymptotically self-similar packet counts. We will, therefore, use the G/M/1 queueing model to perform the admission control, where the interarrival times will be assumed to be Pareto distributed with a shape parameter α ($1 < \alpha < 2$). Smaller values of α indicate more bursty traffic (this is similar to the Hurst parameter concept). Given an average arrival rate λ, a service rate μ, and a delay bound d_{avg}, the admission control procedure will check whether in each router the following inequality holds [17]

$$P\{y > d_{avg}\} = \sigma e^{-\mu(1-\sigma)d_{avg}} \leq \epsilon \qquad (2)$$

where y is the waiting time at the queue under a FCFS service discipline, and ϵ is a very small number (e.g. 10^{-6}). The variable σ is the solution to the equation $\sigma = A^*(\mu - \mu\sigma)$, where $A^*(s)$ is the Laplace transform of the pdf of the interarrival time distribution (i.e. the Pareto distribution). This equation can be easily solved numerically.

As we mentioned in the previous section, implementation of the priority queue in our scheduling algorithm may become a bottleneck if the queue length of a router is allowed to grow to large values. For this reason, we will use inequality (2) to perform some kind of congestion control. A recent study [12] has shown that even for bursty sources, such as video, the utilization level at any node does not increase significantly with a queue length of more than 20-30 ms. We will, therefore, choose to limit the maximum allowable queue length at any node, to a value of 20 ms. This can be achieved by substituting d_{avg} at the LHS of inequality (2), with $\min\{20\text{ms}, d_{avg}\}$.

The main advantages of our admission control algorithm are

- There is no need for per flow state in any router.
- The admission decision is based on the average rate of the aggregated traffic, so it does not require individual traffic characteristics.
- Congestion control is performed implicitly.
- Each router need only advertise two values, λ and d_{avg}. The concept of available bandwidth is no longer required.

However, we have not shown analytically that our algorithm can indeed provide end-to-end delay guarantees. In the following section we will perform extensive simulation experiments that will investigate the potential of our protocol.

IV. Experimental Results

We simulated our protocol in the network topology of Fig. 1 where the maximum path length is 5 hops. This topology may represent an autonomous Diffserv domain. There are 6 sources generating external traffic, and 6 sinks which absorb the traffic. The arrows in Fig. 1 indicate the direction in which the traffic flows. All the links have a capacity of 45 Mbps, and to simplify the experiments, we assumed a fixed packet length of 1Kbit. We used two different traffic sources for the experiments

- An ON/OFF source with exponentially distributed ON and OFF periods. During the ON period, an exponentially distributed number of packets was transmitted, with a mean of 20 packets. The rate at which packets were transmitted was 80 packets/sec. The OFF period was exponentially distributed with mean 375 ms. This traffic source was used to model packetized voice with an average rate of 32 Kbps.
- 10 MPEG compressed video sequences [18] with an average rate of between 312 and 744 Kbps. Each trace was 40,000 frames long (approximately 30 min in time). The frame rate was 24 fps, and we assumed that the packets of each frame were generated at equally spaced intervals within the duration of the frame. These sources are very bursty, and they have been shown to exhibit self-similarity.

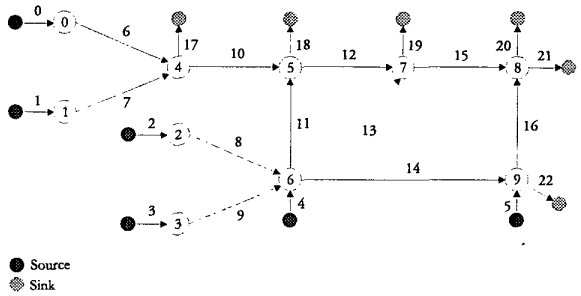


Fig. 1. The simulated network topology.

Each source node generated requests with interarrival times that were exponentially distributed with mean 500 ms. The duration of each connection was exponentially distributed with mean 5 min. These exponential interarrival and holding times are quite realistic for voice and video connections, and they do not affect the results of the experiments. Even with different distributions we would obtain similar results, since we do not make any assumptions on this matter in our protocol. A total of 27 different source-sink pairs were used, with a predefined path between them. The length of the individual paths varied between 2 and 5 hops. There are three classes of traffic with end-to-end delay bound of 10 ms, 50 ms, and 100 ms, respectively. Every new request was either for a voice or video connection with equal probability. The end-to-end delay requirement for voice calls was 10 ms (class 1), while for video calls it was 10 ms (class 1) with probability 0.2, 50 ms (class 2) with probability 0.3, and 100 ms (class 3) with probability 0.5. Every video connection started from a random point in the trace, with appropriate wrap-around at the end of the trace. The shape parameter for the Pareto distribution was set to $\alpha = 1.3$. We simulated 3000 sec of real time and collected the results after the first 1000 sec. The experiments were repeated 9 times with a different random seed, so as to obtain the confidence intervals.

Table I shows the maximum observed delays (end-to-end) and the 99% confidence intervals (CI) for the different service classes at different links. Throughout the experiments there was no delay violation, and the maximum observed delay for any packet was much smaller than its end-to-end delay requirement. It is, therefore, clear that our scheduling algorithm can potentially provide end-to-end delay guarantees, when proper admission control is performed. For a lower value of α, the end-to-end delays were very small which means that the network links were underutilized. On the other hand, with a value of $\alpha = 1.4$ we observed some delay violations which were not very frequent (in the order of 10^{-5}). In a real system, the value of α would be set after extensive experiments with real network traffic.

In Table II we have summarized the average utilization, and the maximum observed queue length for all the links of the simulated network topology. The core links of the upper part of the topology (links 10, 12, and 15) experienced lower utilization level, as they were required to carry many flows with a large path length (i.e. their d_{avg} was relatively small). The rest of the links, however, which carried more flows with smaller hop count, were able to achieve an average utilization of over 80%. The maximum queue length was also small for links 10, 12, and 15, while for the highly utilized links we did observe some very rare violations (i.e. the queue length exceeded the value of d_{avg}). The scheduling algorithm, though, was able to adjust the priorities accordingly, so that every packet was able to meet its end-to-end delay bound. For a higher value of α, these violations were more frequent, leading to occasional delay violations.

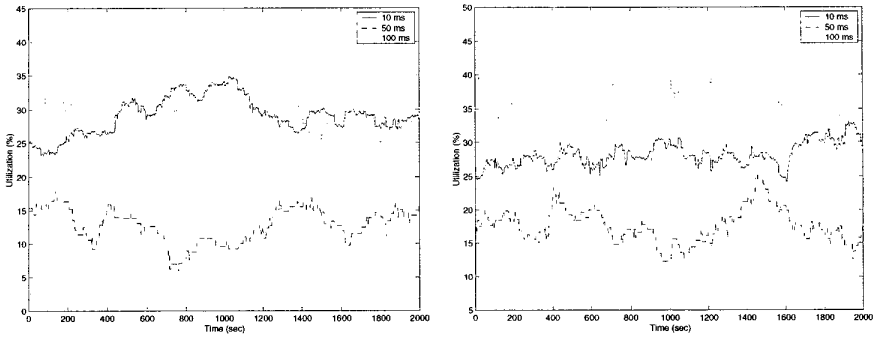
In Fig. 2 we have plotted the distribution of the link capacity among the three service classes for two different links of the simulated network topology. For both links, an average of almost 30% of the link capacity

0-7803-7097-1/01/$10.00 ©2001 IEEE

TABLE I

MAXIMUM OBSERVED DELAYS AT THE TRAFFIC SINKS.

	Service class (delay bound)					
	Class 1 (10 ms)		Class 2 (50 ms)		Class 3 (100 ms)	
Link_id	Mean (ms)	99% CI	Mean (ms)	99% CI	Mean (ms)	99% CI
17	0.31	0.28-0.34	5.7	0.1-13.4	37.1	21.2-53.0
18	0.42	0.38-0.46	1.5	0.2-2.9	31.1	22.5-39.7
19	0.38	0.34-0.42	3.2	1.0-5.4	28.3	25.8-30.9
20	0.36	0.31-0.40	4.4	0.4-8.5	25.1	18.5-31.7
21	0.32	0.28-0.36	4.5	0.4-8.5	28.7	17.0-40.3



(a) Link 12 (average utilization: 71%).

(b) Link 19 (average utilization: 81%).

Fig. 2. Distribution of the link capacity among the different service classes.

TABLE II

UTILIZATION LEVEL AND MAXIMUM QUEUE LENGTH AT DIFFERENT LINKS.

	Utilization (%)		Max queue length (ms)	
Link_id	Mean	99% CI	Mean	99% CI
6	77.5	76.3-78.7	6.1	4.7-7.4
7	77.8	76.7-79.0	8.3	5.2-11.3
8	76.5	75.4-77.5	6.4	4.5-8.2
9	75.2	74.1-76.1	5.3	3.0-7.6
10	73.9	73.3-74.5	4.0	2.7-5.3
11	80.1	79.3-80.9	7.2	5.5-9.0
12	71.3	70.8-71.9	1.4	0.9-1.8
13	82.4	82.3-82.5	10.3	9.0-11.7
14	82.5	82.4-82.6	12.8	8.2-17.5
15	72.6	72.3-72.9	3.2	2.5-3.9
16	83.0	83.0-83.0	9.9	7.5-12.3
17	81.5	81.2-81.7	15.4	5.1-25.7
18	82.7	82.6-82.7	11.4	7.4-15.4
19	81.1	80.7-81.5	8.9	6.6-11.1
20	76.8	76.1-77.4	5.5	2.7-8.2
21	78.9	78.0-79.7	8.6	2.0-15.2

(29% for link 12, and 28% for link 19) was allocated to connections with an end-to-end delay bound of 10 ms (class 1). On the other hand, connections with an end-to-end delay bound of 50 ms (class 2) were allocated an average of 13% (link 12) and 18% (link 19) of the link capacity. This was due to the lower arrival rate of class 2 customers in the system: each new request was from a class 1 customer with probability 0.6, from a class 2 customer with probability 0.15, and from a class 3 customer (100 ms) with probability 0.25. Finally, class 3 connections were allocated 29% (link 12) and 35% (link 19) of the link capacity. The above results are very promising, and they indicate that our protocol can distribute fairly the network resources among the different service classes. Even with 30% of the link capacity being allocated to connections with very small end-to-end delay bound, the admission control procedure could admit many connections from other service classes, and achieve a very high level of network utilization.

Finally, we compared our scheduling algorithm with the FCFS, BURD, static priority, and EDF service disciplines. We performed exactly the same experiment (i.e. with the same random seed) for all five algorithms, and the admission decisions were based on our admission control procedure of Section III. For the static priority discipline, the priority of each packet is equal to its per node delay bound (i.e. the value $\bar{d}_i$), and it is always the same at any router. Packets with the same priority are treated in a FCFS order. For the EDF discipline, each packet is assigned a deadline upon its arrival at the router, which is equal to the arrival time plus the per node delay bound. Packets are transmitted in order of increasing value of deadline. The results are depicted in Table III.

As expected, the FCFS service discipline had the worst performance among all the algorithms. Since each packet is treated equally independent of its service class, the maximum observed delays were practically identical for all three service classes. Even though class 2 and class 3 connections did not experience any delay violations, class 1 connections had many of their packets dropped at the destination node because of excessive delay. Moreover, the connections with larger path length experienced much larger delays (this was also demonstrated in [8], [10]).

For both static priority, and EDF scheduling we observed some rare delay violations. In static priority, the connections with small end-to-end delay bounds (class 1 and class 2) were treated much better than the ones with large delay bounds (class 3). As a result, the delay distribution among the different classes was very unbalanced. A different

0-7803-7097-1/01/$10.00 ©2001 IEEE

TABLE III

MAXIMUM OBSERVED DELAYS AT THE TRAFFIC SINKS FOR DIFFERENT SERVICE DISCIPLINES.

Service class (delay bound)	Algorithm	Link_id 17	18	19	20	21
Class 1 (10 ms)	FCFS	10.1	14.2	11.0	26.7	30.1
	BURD	0.3	0.4	0.3	0.2	0.2
	P-EDF	0.3	0.4	0.4	0.4	0.4
	EDF	0.3	0.4	0.4	3.5	13.3
	Static	0.3	0.4	0.4	0.4	0.5
Class 2 (50 ms)	FCFS	10.1	14.3	10.9	26.6	30.0
	BURD	1.4	0.9	0.9	1.0	9.3
	P-EDF	4.4	0.9	5.0	9.1	9.4
	EDF	4.5	14.7	10.2	16.7	33.3
	Static	1.4	0.9	1.1	1.5	1.5
Class 3 (100 ms)	FCFS	10.2	14.3	10.9	26.6	30.1
	BURD	43.0	35.1	37.2	27.5	59.3
	P-EDF	29.5	35.3	30.1	26.3	59.4
	EDF	29.8	46.7	50.3	41.5	62.4
	Static	48.8	138.2	61.7	43.4	184.8

unbalanced delay distribution occured also in EDF scheduling, where the connections with a large path length (i.e. terminating at links 18-21) experienced much larger delays. This can be explained by the fact that EDF scheduling does not take into account the delay that each packet experienced in the previous nodes. A packet which has to traverse many nodes is more likely to be delayed at one or more of them, but this delay will not affect its scheduling at the following nodes. However, with EDF scheduling the delay distribution among the different service classes was fair.

Finally, for P-EDF and BURD there was a fair delay distribution both among the different service classes, and among the connections with different path lengths. In addition, none of the two schemes resulted in an end-to-end delay violation. The difference between the two algorithms is that class 1 and class 2 connections receive better service with BURD, while class 3 connections receive better service with P-EDF. This is a result that we expected from the beginning, since this property of BURD was identified in Section II. We believe, however, that P-EDF generally results in a more equitable delay distribution, since with BURD scheduling class 1 and class 2 connections experience very similar delays, even though their end-to-end delay bounds are very different (i.e. five times larger for class 2). Moreover, the $O(1)$ dequeue complexity of P-EDF makes it a better candidate for an actual implementation within a Diffserv architecture.

V. CONCLUSIONS

In this paper, we presented a new lightweight protocol for a Differentiated Services architecture. Despite its simplicity, it can offer the flexibility of the Intserv architecture and, with proper admission control, it can also offer a similar assurance level. Its implementation is very easy, since it only requires minor changes to the current IP protocol. The proposed protocol is complete, in the sense that it deals with the issues of packet scheduling, admission control, and congestion control. The experimental results showed that our protocol can potentially provide end-to-end delay guarantees to real-time applications, while maintaining a high level of network utilization. For future work we plan to implement our protocol, and investigate its applicability in the Internet architecture. Moreover, we will try to incorporate QoS routing protocols in the admission control procedure.

ACKNOWLEDGEMENTS

This research is supported in part by the University Grants Committee, Hong Kong, Area of Excellence in Information Technology, Grant No. AOE 98/99.EG01, and by the State Scholarships Foundation of Greece.

REFERENCES

[1] R. Branden, D. Clark, and S. Shenker, "Integrated services in the Internet architecture: an overview," *Internet RFC 1633*, June 1994.

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

[3] R. Branden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource reservation protocol (RSVP) – version 1 functional specification," *Internet RFC 2205*, September 1997.

[4] K. Nichols, V. Jacobson, and L. Zhang, "A two-bit differentiated services architecture for the Internet," *Internet RFC 2638*, July 1999.

[5] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queueing algorithm," in *Proceedings ACM SIGCOMM*, pp. 3–12, September 1989.

[6] D. Ferrari and D. Verma, "A scheme for real-time channel establishment in wide-area networks," *IEEE Journal on Selected Areas in Communications*, vol. 8, pp. 368–379, April 1990.

[7] I. Stoica and H. Zhang, "Providing guaranteed services without per flow management," in *Proceedings ACM SIGCOMM*, pp. 81–94, September 1999.

[8] D. D. Clark, S. Shenker, and L. Zhang, "Supporting real-time applications in an integrated services packet network: architecture and mechanism," in *Proceedings ACM SIGCOMM*, pp. 14–26, August 1992.

[9] J. Kobza and S. Liu, "A head-of-line approximation to delay dependent scheduling in integrated packet-switched networks," in *Proceedings IEEE INFOCOM*, pp. 1106–1113, April 1989.

[10] H. Schulzrinne, J. Kurose, and D. Towsley, "An evaluation of scheduling mechanisms for providing best-effort, real-time communication in wide-area networks," in *Proceedings IEEE INFOCOM*, pp. 1352–1361, June 1994.

[11] E. W. Knightly, "Second moment resource allocation in multi-service networks," in *Proceedings ACM SIGMETRICS*, pp. 181–191, June 1997.

[12] E. W. Knightly and N. B. Shroff, "Admission control for statistical QoS: theory and practice," *IEEE Network*, pp. 20–29, March/April 1999.

[13] V. Sivaraman and F. Chiussi, "Providing end-to-end statistical delay guarantees with earliest deadline first scheduling and per hop traffic shaping," in *Proceedings IEEE INFOCOM*, pp. 631–640, March 2000.

[14] W. E. Leland, M. S. Taqqu, W. Willinger, and D. V. Wilson, "On the self-similar nature of ethernet traffic," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 1–15, February 1994.

[15] W. Willinger, M. S. Taqqu, R. Sherman, and D. V. Wilson, "Self-similarity through high-variability: statistical analysis of ethernet LAN traffic at the source level," in *Proceedings ACM SIGCOMM*, pp. 100–113, September 1995.

[16] J. Gordon, "Pareto process as a model of self-similar packet traffic," in *Proceedings IEEE GLOBECOM*, pp. 2232–2236, November 1995.

[17] L. Kleinrock, *Queueing systems, Vol. I: theory*. John Wiley & Sons, 1975.

[18] O. Rose, "Statistical properties of MPEG video traffic and their impact on traffic modeling in ATM systems," in *Proceedings 20th Annual Conference on Local Computer Networks*, pp. 397–406, 1995.