

A Combined Delay and Throughput Proportional Scheduling Scheme for Differentiated Services

Samyukta Sankaran and Ahmed E.Kamal
Department of Electrical and Computer Engineering
Iowa State University
Ames, IA 50011
{samyukta,kamal}@iastate.edu

Abstract—The proportional differentiation model is a newly introduced approach for differentiated services networks. This paper proposes and evaluates a scheduling mechanism for the combined control of delay and throughput metrics, according to the proportional differentiation model. The scheme is based on the well known Little’s Law. A moving window averaging mechanism and an active queue management scheme are simultaneously, and respectively used to achieve control over the relative throughputs as well as the relative delays between classes. The scheme does away with measurement of the actual packet delays, and state information is minimized. Some feasibility bounds are presented, and a simulation study shows the effectiveness of this scheme.

I. INTRODUCTION

The Internet Engineering Task Force (IETF) introduced the *Differentiated Services (DiffServ)* architecture [1] as a lightweight, scalable approach to high-performance networking. In this approach, flows with similar requirements are aggregated into a limited number of service classes, and are marked with a suitable DiffServ codepoint (DSCP) [2]. Routers inside the DiffServ domain have class sensitive packet forwarding mechanisms that provide service differentiation. Since state information is maintained for a limited number of classes, DiffServ scales well. Further, policing mechanisms are required only at the edge of domain.

Research within DiffServ has been proceeding along three broad directions: *Absolute*, *Relative*, and *Proportional DiffServ* [3]. Proportional Differentiation is a refinement and generalization of Relative DiffServ, and aims to achieve the two goals of *Predictability*, or consistent differentiation independent of class loads, and *Controllability*, or the ability to externally adjust the quality spacing between classes based on dynamically changing criteria. The model aims to control some chosen class performance metric, proportional to the differentiation parameters chosen by the network operator. If m_i is the chosen metric for class i , and c_i is the corresponding operator-chosen quality differentiation parameter, then the proportional differentiation model attempts to achieve $m_i/m_j = c_i/c_j$ over all classes. So, even though actual service levels vary with class loads, service ratios are always maintained.

Several protocols have been proposed within the framework of Proportional Differentiation. The MulTCP approach [5] introduced *bandwidth proportional differentiation* using weighted proportional fairness, where the weight of each flow is related to the price paid by the user. A connection with a weight of N behaves like an aggregate of N TCP connections.

Reference [4] proposes two schedulers to achieve *proportional delay differentiation*. Given that \bar{w}_i is the mean queuing delay of a class i and W_i is the *Delay Differentiation Parameter* for class i such that ($W_1 > W_2 > \dots W_N > 0$), proportional delay differentiation was defined as satisfying $\frac{w_i}{w_j} = \frac{W_i}{W_j}$ over all pairs of classes. Two schedulers were proposed: the *Backlog Proportional Rate (BPR) Scheduler*, in which class service rates are dynamically adjusted so that they are weighted proportional to the class backlog; and the *Waiting Time Priority Scheduler*, in which the priority of a packet increases with the waiting time of the packet. The same authors extend the proportional differentiation model to *loss differentiation* in [6], and propose a Proportional Loss Rate (PLR) dropper to achieve this.

This paper proposes a scheduler based on the Proportional DiffServ model. During periods of congestion or saturation, the throughput is limited by the throughput weights. Under conditions of light load, when the buffer sizes are never exceeded, throughput is equal to the offered load. Bursty traffic, where there are alternating periods of heavy and light loads, show a combination of these two behaviors. Note that, in all cases, we achieve delay differentiation. In addition, this is achieved using simple mechanisms which do not require book-keeping of packet delays, or even measurement of packet delays.

Our paper is organized as follows. In Section II, we introduce our strategy. Numerical examples based on a simulation model are presented in Section III. Finally, Section IV concludes the paper with a few remarks.

II. STRATEGIES FOR COMBINED DIFFERENTIATION

This section introduces our strategy to achieve combined proportional differentiation. We lay out the theoretical foundations of our model, and present the limitations to achieving combined proportional differentiation. Theoretical limits to the achievable ratios are presented, along with the formal algorithm.

A. Little’s Result and Proportional Differentiation

This work has its basis in the fundamental result, first proven by J. D. C. Little, and known as Little’s Result, e.g., see [7]. Consider any general queuing system, with notations as defined in Table I. Little’s result states that $\bar{q}_i = s_i \cdot \bar{w}_i$. This result makes no assumptions about the nature of arrival or departure processes in the queuing system. Taking multiple classes into account, a straightforward application of the above gives:

$$\frac{\bar{q}_i}{\bar{q}_j} = \frac{s_i}{s_j} \cdot \frac{\bar{w}_i}{\bar{w}_j} \quad (1)$$

TABLE I
LIST OF SYMBOLS USED IN THE PAPER

Symbol	Definition
i, j	class number indices, with $0 \leq i, j < N$
\bar{b}_i	mean service time for pkt from class i
\bar{b}_i^2	2^{nd} moment of service time for pkt from class i
λ_i	arrival rate from class i in pkts/sec
ρ_i	offered load (Erlangs) from class $i = \lambda_i \bar{b}_i$
s_i	throughput of class i in pkts/sec
σ_i	carried load from class i in Erlangs $= s_i \bar{b}_i$
\bar{w}_i	mean packet delay for class i packets
\bar{q}_i	mean number of packets from class i in buffer
S_i	throughput weight for class i
W_i	mean delay weight for class i
B	buffer size

Differentiation Parameters can be associated with the performance metrics of each class, so that the problem of achieving delay and throughput proportional differentiation between classes then reduces to a problem of enforcing (1).

B. Limitations and Choice of metrics

Combined proportional differentiation has its limitations with regard to the metrics that can be combined. We propose the following, which is proven in *Appendix A*:

Proposition 1: *It is not possible to achieve combined proportional differentiation in the delay and loss metrics, independent of actual values of packet loss ratios.*

It is possible, however, to achieve combined delay and loss differentiation if the actual values of packet loss ratios are taken into account. Given this restriction, we have chosen to implement combined delay and throughput differentiation. That is, for every pair of classes i and j , (1) must hold. Control of any two of the three ratios in this equation will result in a proportional control of the third. Controlling the mean delay of a class involves greater complexity, since router bookkeeping and delay measurement has to be done for each packet passing through the router. Also, accuracy of measurement can vary widely between systems, as it depends on clock granularities. Overruling active delay control leads to the choice of controlling \bar{q}_i and s_i as the optimal approach.

C. Strategies for Combined Differentiation

This section discusses our strategies for controlling the queue size and throughput proportions, hence achieving combined proportional differentiation. It also establishes bounds on the achievable proportions. In Section II-C.1 we present algorithms for separately controlling the throughput and queue ratios during departure, while in Section II-C.2 we show how to integrate them. Section II-C.3 shows how to handle packet arrivals to a full buffer. The symbols defined in Table I will be used in the discussion.

1) *Mechanisms for departure-The Packet Scheduler:* Serving a packet from a class changes the throughput ratios and also the queue length ratios (hence, delay ratios) of associated classes. So, the packet scheduler must be designed to control these ratios appropriately. In this section, we define scheduling mechanisms to control each of these ratios separately.

a. Controlling s_i/s_j

To control the throughput of all classes proportionally, a moving window averaging mechanism is used. Throughput data for departing packets of each class is collected over a moving window. A moving window of size M will hold throughput information about the M most recently served packets. The moving window size has been chosen to be of the same order as the buffer size. When the throughput mechanism chooses a class to serve a packet from, it will base its choice of the class on minimizing the difference between the throughput ratios of all flows in the system and the ideal ratio, using min-max optimality. That is, packet departures are scheduled from that class for which the maximum deviation of any of its throughput ratios from the ideal, after service, is minimal. This strategy, when used exclusively, will fulfill the requested throughput weights of all flows, provided the input traffic satisfies certain conditions. The satisfaction of the throughput ratios is governed by the following proposal:

Proposition 2: *The necessary and sufficient conditions for the throughput ratios to be satisfied depend on the offered load, and are as follows:*
Case 1: When $\rho_1 + \rho_2 \leq 1$, then λ_1/λ_2 must equal S_1/S_2
Case 2: When $\rho_1 + \rho_2 \geq 1$:
 2a: *if $\rho_1/\rho_2 > S_1/S_2$, then $\lambda_2 \geq S_2/(S_1\bar{b}_1 + S_2\bar{b}_2)$*
 2b: *if $\rho_1/\rho_2 < S_1/S_2$, then $\lambda_1 \geq S_1/(S_1\bar{b}_1 + S_2\bar{b}_2)$*

The proof of Propositions 2,3 and 4 are omitted in this paper due to lack of space, and are included in our longer work [9]. When the offered traffic is less than the system capacity, it may not be possible to satisfy the throughput ratios. All incoming packets are served, so throughput differentiation is not possible unless the offered traffic satisfies the desired throughput ratios as in Proposition 2 (we assume a work conserving system). However, when input traffic exceeds the server capacity, the system is overloaded and packets must be dropped. Without route pinning in DiffServ domains such a case may arise, and it might be possible to guarantee throughput differentiation subject to the above conditions.

b. Controlling \bar{q}_i/\bar{q}_j

Ratios of queue sizes of different classes need to be also controlled in order to control the mean class delays proportionally. The scheduler can control the queue lengths by serving a packet from that class for which the maximum deviation of the queue lengths, after service, is minimal. This strategy, when exclusively used, will control the queue lengths (and hence delays) proportionally, with the following caveat: when one or more of the queues are empty, the above min-max approach presents anomalous behavior. This can be explained by understanding that when one or more of the queue sizes are 0, considering q_1/q_2 may give a different service decision than by considering q_2/q_1 . We have dealt with this by applying a simple, deterministic heuristic in such cases, i.e., serving the class with the smallest delay weight W_i (this is not shown in the pseudocode). Additionally, there are certain constraints to the delay control

```

System::onServerIdle {
  if SYSTEM-STATUS = LIGHT {
    For each class  $i$ 
      compute  $\Delta(q)_i |_{q_j=q_j-1}$ 
    find class  $j$  for which
       $\max_{0 \leq i < N} \{\Delta(q)_i |_{q_j=q_j-1}\} \leq$ 
       $\max_{0 \leq i < N} \{\Delta(q)_i |_{q_l=q_l-1}\}$ , for  $j \neq l$ 
    Serve a packet from class  $j$ 
  }
  else if SYSTEM-STATUS = HEAVY {
    For each class  $i$ 
      compute  $\Delta(s)_i |_{q_j=q_j-1}$ 
    find class  $j$  for which
       $\max_{0 \leq i < N} \{\Delta(s)_i |_{q_j=q_j-1}\} \leq$ 
       $\max_{0 \leq i < N} \{\Delta(s)_i |_{q_l=q_l-1}\}$ , for  $j \neq l$ 
    Serve a packet from class  $j$ 
  }
}

```

Fig. 1. Packet Scheduler

using min-max optimality, which we state below¹:

Proposition 3: *When $\sigma_1 + \sigma_2 < 1$, and under Poisson arrivals and general service times, the achievable delay ratio is such that:*

$$\begin{aligned}
(1 - \sigma_1 - \sigma_2) \cdot \frac{\bar{b}_0 + \bar{b}_1(1 - \sigma_1)}{\bar{b}_0 + \bar{b}_2(1 - \sigma_1)(1 - \sigma_1 - \sigma_2)} &\leq \frac{\bar{w}_1}{\bar{w}_2} \\
&\leq \frac{1}{1 - \sigma_1 - \sigma_2} \cdot \frac{\bar{b}_0 + \bar{b}_2(1 - \sigma_1)(1 - \sigma_1 - \sigma_2)}{\bar{b}_0 + \bar{b}_1(1 - \sigma_1)}
\end{aligned}$$

where \bar{b}_0 is the residual service time as seen by an arrival, and is given by

$$\bar{b}_0 = \sum_{i=1}^2 \sigma_i \frac{\bar{b}_i^2}{2\bar{b}_i} \quad (2)$$

Further, for a saturated system ($\sigma_1 + \sigma_2 = 1$):

Proposition 4: *Assuming Poisson arrivals and general service times, when a system is saturated,*

$$\begin{aligned}
\frac{(\bar{b}_0 - \sigma_1 \bar{b}_1 + \bar{b}_1) \sigma_2 \bar{b}_1}{[(B-1)(1 - \sigma_1) \bar{b}_1 - (\bar{b}_0 - \sigma_1 \bar{b}_1) \sigma_1] \bar{b}_2} &\leq \frac{\bar{w}_1}{\bar{w}_2} \\
&\leq \frac{[(B-1)(1 - \sigma_1) \bar{b}_1 - (\bar{b}_0 - \sigma_1 \bar{b}_1) \sigma_1] \bar{b}_2}{(\bar{b}_0 - \sigma_1 \bar{b}_1 + \bar{b}_1) \sigma_2 \bar{b}_1}
\end{aligned}$$

where \bar{b}_0 is the residual service time given by (2).

We present the pseudocode for this algorithm in Fig. 1. For all pseudocodes, we define a set of pairwise parameter ratio offsets for N classes as:

$$\Delta(x)_i = \frac{x_i}{x_{(i+1) \bmod N}} - \frac{X_i}{X_{(i+1) \bmod N}} \quad (3)$$

for $0 \leq i < N$, where the argument x can take the value q , s and w for the queue length, the throughput, or the mean delay, respectively, while X corresponds to x 's target weight.

¹Both bounds are also consistent with equation (4) in [8], with our result being the limit on the mean delay ratio when the WTP Scheduler is used, and when the dynamic priority control parameters for class 1 is much higher (respectively lower) than that for class 2.

2) *Modes of Operation for the Packet Scheduler:* The throughput control mechanism, if used exclusively (i.e., if packet scheduling is controlled by using this mechanism only), will exactly satisfy throughput ratios, subject to the input constraints described earlier (Proposition 2). Likewise, the queue control mechanism, when exclusively used, will provide the exact delay ratios required, subject to Propositions 3 and 4. However, our work aims at combined delay and throughput differentiation, when feasible. Accordingly, the system should employ both mechanisms in a complementary manner in order to satisfy both requirements. We informally define two modes of operation for the packet scheduler, namely, the *light* and *heavy* load modes:

- Lightly loaded state:* We define the system to be lightly loaded when the total input traffic is less than or equal to the server capacity. In this case, each class obviously receives all the throughput it has requested. Since the system is work-conserving, throughput control is neither needed nor possible under this condition. However, the delays of the classes may be controlled in this stage by controlling the queue sizes. Therefore, when the system is lightly loaded, the queue control mechanism is called upon to choose the class to be served, therefore satisfying the queue ratios.
- Heavily loaded state:* When the input traffic to the server is greater than the server capacity, some packets will need to be discarded. In this case, the throughputs of all classes are controlled by invoking the throughput control mechanism which will maintain the necessary throughput ratios.

When the number of arrivals within a predefined, discrete time frame is less than a predefined packet threshold, the system is considered lightly loaded; otherwise it is heavily loaded. The pseudocode is presented in the Part 1 of Fig. 2.

3) *Mechanisms for arrival-The Queue Manager:* To complete the mechanism, we must decide on how to handle packet arrivals. Notice that arrivals of packets from a class changes the queue ratios (and hence delay ratios) of associated classes. However, since the system is work-conserving, queue control (by dropping packets to adjust queue ratios) is not a feasible option when available buffer space exists. Recall that in this state, the packet scheduler works to control queue ratios proportionally. However when the buffer is full, multiple queuing decisions are possible: *an arriving packet may be dropped, or it may be accepted by discarding an already-queued packet from one of the other classes.* This decision must be made with a view to satisfying the delay ratios. The queue manager performs this function. In this case, we use an active queue management scheme to achieve combined proportional differentiation. Deviations of the queue ratios from the ideal are computed for each of the above-mentioned decisions. That decision is chosen for which deviation of the queue ratios from the ideal is optimal, using min-max optimality criterion alluded to earlier. Part 2 of Fig. 2 presents the pseudocode.

III. NUMERICAL RESULTS

In this section we provide several numerical examples based on a simulation model to show the effectiveness of our algo-

```

System::onPktArrival {
  /*Part 1: choosing mode of Packet Scheduler*/
  if number of pkt arrivals in window ≤ packet-threshold
    SYSTEM-STATUS = LIGHT
  else
    SYSTEM-STATUS = HEAVY
  /*Part 2: Queue Manager*/
  /* j = class of incoming packet */
  if buffer = NOT-FULL
    accept incoming packet
  else {
    for class i
      compute  $\Delta(q)_i |_{q_j=q_j}$ 
    for class k
      compute  $\Delta(q)_i |_{q_j=q_j+1, q_k=q_k-1}$ 
    find the queuing decision and class j
    for that decision in which:
       $\max_{0 \leq i < N} \{ \Delta(q)_i |_{q_j=q_j}, \Delta(q)_i |_{q_j=q_j+1, q_k=q_k-1}, \forall k \neq j \}$ 
       $\leq \max_{0 \leq i < N} \{ \Delta(q)_i |_{q_l=q_l}, \Delta(q)_i |_{q_l=q_l+1, q_k=q_k-1}, \forall k \neq l \}$ 
    Apply the chosen queuing decision.
  }
}

```

Fig. 2. Queue Manager

TABLE II
HEAVY TRAFFIC: THROUGHPUT AND DELAY RATIOS

Thruput Ratio	Target	Measured	Delay Ratio	Target	Measured
s_0/s_1	1.0	1.00	\bar{w}_0/\bar{w}_1	2.0	2.05
s_1/s_2	2.0	1.96	\bar{w}_1/\bar{w}_2	1.0	0.97
s_2/s_0	0.5	0.51	\bar{w}_2/\bar{w}_0	0.5	0.50

riethm. We show that we achieve delay control in all scenarios. Further, during heavy load periods in any scenario, throughputs are limited by their weights; during light load periods, throughput is equal to the offered load.

We consider a single server which is fed by three packet streams. Each stream generates packets according a Markov modulated Poisson process (MMPP). The modulating process is exponential, and its parameters are set to control the burstiness of the traffic. Packets are generated during ON and OFF periods according to a Poisson process with rates that determine the traffic intensity. As such, the Poisson process is a special case of the MMPP. Each stream is controlled independently.

Packets from all streams are assumed to require the same transmission time, which is exponentially distributed with a unity mean. The buffer at the server can hold a maximum of 100 packets. The sliding window size (M) which is used for throughput measurement is 100 time units. The thresholds used to determine light and heavy load are 60 packets arriving within 60 time units. If the number of packets arriving within a window exceeds this threshold, then the system is considered to be in the heavy load state; otherwise it is in the light load state.

The first scenario we consider is the heavy traffic case, where the system is almost always in the heavily loaded state. This is the case in which our scheme should be able to achieve full control over the throughput and mean delay ratios. The traffic from the three streams is adjusted such that their offered loads are 1, 1 and 2 Erlangs, respectively, and it is Poisson. The throughput weights are 2, 2 and 1, while the mean delay weights are 2, 1 and 1, respectively. In Table II we show both the target and achieved ratios for throughput and delay.

The achieved values are very close to the target values, with maximum errors of 2% in the throughput ratios, and 3% in the mean delay ratios. In Fig. 3(a) we plot the instantaneous

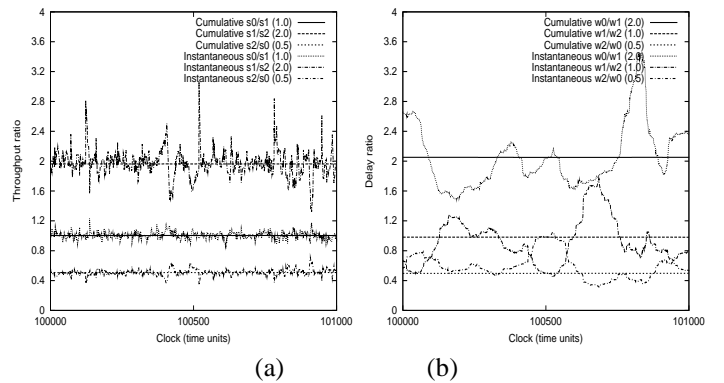


Fig. 3. Heavy traffic: (a) Throughput and (b) Delay ratios

and cumulative throughput ratios over a total duration of 1000 packet transmission times. The ratio measures are over the sliding window. The instantaneous and cumulative mean delay ratios, measured over the same interval, and using the same sliding window, are also shown in Fig. 3(b). In the figures, the ideal values are represented in braces in the legend.

Note that, for both delay and throughput, the cumulative value graph is constant and very close to the ideal, and the maximum deviations of instantaneous ratios from the ideal are reasonable². Such fluctuations occur due to the lack of packets from one or more flows at a particular instant. However, these fluctuations are corrected as soon as packets from the under-served flow arrive. Note that the slope of the delay instantaneous ratios graph is fairly low.

The second scenario is with light traffic, where the system is almost always lightly loaded. The three streams generate Poisson traffic at rates of 0.1, 0.3 and 0.2 Erlangs, for a total of 0.6 Erlangs. In this case, the queue size never exceeds 20 packets, and no packet losses are ever encountered. The individual stream throughputs are equal to the offered load, and therefore cannot be controlled. However, setting the target mean delay weights to 1, 1 and 1.25, our scheme was able to control the mean delay ratios, as shown in Table III. The error in the ratios is less than 3%.

TABLE III
LIGHT TRAFFIC: DELAY RATIOS

Delay Ratio	Target	Measured
\bar{w}_0/\bar{w}_1	1.0	0.976
\bar{w}_1/\bar{w}_2	0.8	0.812
\bar{w}_2/\bar{w}_0	1.25	1.261

The graphs of the cumulative mean delay ratios were almost flat and close to the ideal, as shown in Fig. 4(a). We note that instantaneous ratios graph (delay) has a much steeper slope than in the heavy load case (Fig. 3(b)). This is due to the fact that the idle-system case, when queues are momentarily empty, occurs more frequently when the system is lightly loaded.

The third scenario shows the effectiveness of our scheme in mixed loads, when the system alternates between periods of heavy load and light load, which are generated using the MMPP process. The three classes offer average loads of 0.4, 0.4 and

²Comparing these fluctuations to those produced by WTP and BPR in [4]

IV. CONCLUSIONS

This paper investigated the combined proportional differentiation between multiple performance metrics. It first showed that combined proportional differentiation between loss and mean delay cannot be achieved independent of the actual loss values. Therefore, we have presented an algorithm for combined proportional differentiation between mean delay and throughput ratios. The algorithm is based on an implementation that enforces the throughput ratios, and then enforces the queue ratios. Based on the satisfaction of Little's result, the mean delay ratio will then be achieved.

The paper presented several numerical examples which show the effectiveness of the algorithm. When throughput is uncontrollable, our algorithm is a new and simple implementation of proportional delay differentiation. However, under heavy load, the algorithm was able to control both throughput and mean delay ratios properly.

Appendix A

Proposition 1: *It is not possible to achieve combined proportional differentiation in the delay and loss metrics, independent of actual values of packet loss ratios.*

Proof: Let l_i be the fraction of lost packets for class i , and λ_i be the mean arrival rate for class i . Then the throughput for class i , s_i can be expressed as $\lambda_i(1 - l_i)$. Little's Result states that $\bar{q}_i = s_i \bar{w}_i$, which yields

$$\frac{\bar{q}_i}{\bar{q}_j} = \frac{s_i}{s_j} \cdot \frac{\bar{w}_i}{\bar{w}_j} = \frac{\lambda_i}{\lambda_j} \cdot \frac{(1 - l_i)}{(1 - l_j)} \cdot \frac{\bar{w}_i}{\bar{w}_j}$$

Let the target proportions be $\bar{w}_i/\bar{w}_j = W_i/W_j$ and $l_i/l_j = L_i/L_j$. Then the ratio \bar{q}_i/\bar{q}_j must be expressible in terms of these proportions only. We prove the infeasibility of this using contradiction.

Assume these proportions can be simultaneously satisfied, then

$$\frac{\bar{q}_i}{\bar{q}_j} = \frac{\lambda_i}{\lambda_j} \cdot \frac{(1 - l_j \cdot L_i/L_j)}{1 - l_j} \cdot \frac{W_i}{W_j} \quad (4)$$

However, (4) depends on l_j in addition to L_i and L_j . This contradicts the assumption, proving that combined proportional delay and loss differentiation is only possible if the actual loss ratios are taken into account. \square

REFERENCES

- [1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang and W. Weiss, "An Architecture for Differentiated Services", IETF RFC 2475, December 1998
- [2] K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 headers", IETF RFC 2474, December 1998
- [3] C. Dovrolis and P. Ramanathan, "A Case for Relative Differentiated Services and the Proportional Differentiation Model", IEEE Network, September/October 1999
- [4] C. Dovrolis, D. Stiliadis and P. Ramanathan, "Proportional Differentiated Services: Delay Differentiation and Packet Scheduling", IEEE/ACM Transactions on Networking, February 2002
- [5] J. Crowcroft and P. Oechslin, "Differentiated end-to-end Internet Services using a Weighted Proportional Fair sharing TCP", ACM Computer Communication Review, July 1998
- [6] C. Dovrolis and P. Ramanathan, "Proportional Differentiated Services, Part II: Loss Rate Differentiation and Packet Dropping", IEEE/IFIP Eighth International Workshop on Quality of Service, June 2000.
- [7] L. Kleinrock, Queueing Systems, Vol. I: Theory. John Wiley, New York, 1975.
- [8] M. Leung, J. Lui and D. Yau, "Adaptive Proportional Delay Differentiated Services: Characterization and Performance Evaluation", IEEE/ACM Transactions on Networking, Vol 9, No 6, December 2001
- [9] S.Sankaran, "A combined scheme for delay and throughput scheduling scheme for Proportional Differentiated Services", M.S.thesis, Electrical and Computer Engineering Dept., Iowa State University, 2002.

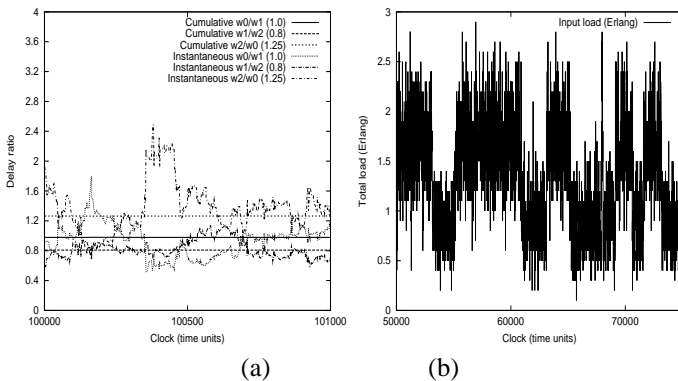


Fig. 4. (a) Light traffic: Delay ratio and (b) Mixed traffic : Input load snapshot

0.85 Erlang during the heavy load periods, and average loads of 0.2,0.2 and 0.425 Erlang during the light load, or underloaded periods. The durations of the heavy and light load periods were exponentially distributed with a mean of 2,000 time units. The length of the simulation run was 300,000 time units. Fig. 4(b) shows a snapshot of the input load in the period of 50,000 to 75,000 time units.

We set the target delay weights to 1.0,1.0 and 1.25, which are to be enforced over the entire simulation run. The target throughput weights are set to 1.2,1.0 and 1.4, and are enforced under periods of heavy load. Under the underloaded system condition, throughput is equal to the offered load. In Table IV, we show the average delay ratios for the heavy and light load periods, as well as the overall delay ratios. The overall averages are reasonably close to the target ratios, and so are the light load period ratios. The average ratios for the heavy period show somewhat larger deviations from the target.

TABLE IV
MIXED TRAFFIC: DELAY RATIOS

Delay Ratio	Target	Heavy period Avg.	Light period Avg.	Overall Avg.
\bar{w}_0/\bar{w}_1	1.0	0.78	0.93	0.84
\bar{w}_1/\bar{w}_2	0.8	0.73	0.77	0.83
\bar{w}_2/\bar{w}_0	1.25	1.74	1.38	1.42

Table V shows similar results for throughput. Note that very precise control is achieved over the ratios in the heavy load phase. During the light load duration, no throughput control is exercised, and all incoming traffic is served. Throughput ratios in this phase are the same as that for the incoming traffic.

TABLE V
MIXED TRAFFIC: THROUGHPUT RATIOS

Delay Ratio	Target (for heavy load)	Heavy period Avg.	Light period Avg.	Overall Avg.
s_0/s_1	1.2	1.19	0.99	1.11
s_1/s_2	0.71	0.71	0.47	0.59
s_2/s_0	1.16	1.17	2.1	1.50

It must be noted that, since the burst sizes are very large, we need to increase the packet threshold to achieve these results.

It is important to point out that, for the lightly loaded system, our scheme reduces to a new implementation of *Proportional Delay Differentiation*.