A Robust Packet Scheduling Algorithm for Proportional Delay Differentiation Services[†]

Jianbin Wei and Cheng-Zhong Xu
Department of Electrical & Computer Engineering
Wayne State University, Detroit, MI 48202
Email: {jbwei, czxu}@wayne.edu

Xiaobo Zhou
Department of Computer Science
University of Colorado at Colorado Springs, CO 80933
Email: zbo@cs.uccs.edu

Abstract— **Proportional delay differentiation (PDD) model is an important approach for relative differentiated services provisioning on the Internet. It aims to maintain pre-specified packet queueing-delay ratios between different classes of traffic at each hop. Existing PDD packet scheduling algorithms are able to achieve the goal in long time-scales when the system is highly utilized. This paper presents a new PDD scheduling algorithm, called *Little's average delay* (LAD), based on a *proof* of Little's Law. It monitors the arrival rate and the cumulative delays of the packets from each traffic class, and schedules the packets according to their transient queueing properties so as to achieve the desired class delay ratios in both short and long time-scales. Simulation results show that, in comparison with other PDD scheduling algorithms, LAD can provide no worse level of service quality in long time-scales and more accurate and robust control over the delay ratio in short time-scales. In particular, LAD outperforms its main competitors significantly when the desired delay ratio is large.**

I. Introduction

The past decade has seen an increasing demand for provisioning of different levels of quality of service (QoS) on the Internet to support different types of network applications and different user requirements. Differentiated Services (DiffServ) is one of the major effort to meet this demand [1]. It aims to provide differentiated services among classes of aggregated traffic flows within a router. Two different schemes exist for DiffServ: Absolute DiffServ and relative DiffServ. Absolute DiffServ aims to guarantee a class's received resource, such as bandwidth. Relative DiffServ is to quantify the quality spacing between different classes.

Recently, a proportional delay differentiation (PDD) model was proposed in support of relative DiffServ [2], [3]. It ensures the quality spacing between classes of traffic to be proportional to certain pre-specified class differentiation parameters. Since then, many packet scheduling algorithms have been developed to implement the PDD model. Representatives of the PDD algorithms include backlog-proportional rate (BPR) [2], joint buffer management and scheduling (JoBS) [5], proportional average delay (PAD) [3], waiting-time priority (WTP) [3], adaptive WTP [4], hybrid proportional delay (HPD) [3], mean-delay proportional (MDP) [6], and VirtualLength [8]. They demonstrated various characteristics in support of the PDD model in different class load conditions and different time-scales. Most of them are capable of achieving desired delay ratios, if the ratios are feasible, under heavy load conditions and in long time-scales. However, for light load conditions and in short time-scales, they exhibit various limitations. We shall compare them with our algorithm in Section IV.

In this paper, we present a new PDD algorithm, called Little's average delay (LAD), based on a *proof* of Little's Law. Little's Law regarding a queueing system states the *stationary* relationship between queue length, arrival rate, and queueing delay on average in the long run. The proof reveals a *transient* property regarding the queueing length [7]; that is, the queueing length of a class at any time is equal to the product of the traffic arrival rate and the waiting time of backlogged packets, plus the experienced delay of departed packets. Accordingly, LAD monitors the average arrival rate of every traffic class and the queueing delay of arrived packets, including both the waiting packets in the queue and departed packets, for the purpose of controlling the delay ratio in both long and short time-scales.

Simulation results show that LAD overcomes the limitations of its main competitors: AWTP, HDP, and MDP. Specifically, whenever the PDD model of a desired class delay ratio is feasible, LAD is capable of providing more accurate and robust control over the delay ratio than its competitors in short time-scales. The improvement is significant when the desired delay ratio is large. In long time-scales, LAD performs no worse than its competitors under any load conditions. Moreover, the performance of LAD is independent of the distributions of packet inter-arrivals and packet sizes because of the generality of Little's Law.

The remainder of the paper is organized as follows. Section II overviews the PDD model and the existing PDD algorithms. Section III presents the LAD algorithm and discusses its design and implementation issues. Section IV compares it with other PDD algorithms via extensive simulations. Section V concludes this paper.

II. Background and Related Work

We consider packet scheduling of a lossless, work-conserving, and non-preemptive link that services M ($M \geq 2$) first-come-first-served (FCFS) queues, one for each traffic class. The objective of the PDD model is to control the quality spacing between different classes so that their average delay ratios be proportional to certain class differentiation parameters pre-specified by network operators. Let W_i denote

[†]This research was supported in part by U.S. NSF grant ACI-0203592.

the average delay of class i, and δ_i the pre-defined delay differentiation parameter. The PDD model requires to ensure that for any two classes i and j, $1 \leq i, j \leq M$,

$$\frac{W_i}{W_j} = \frac{\delta_i}{\delta_j}. \tag{1}$$

Many packet scheduling algorithms have been proposed for PDD service model. Rate-allocation algorithms, as exemplified by BPR [2] and JoBS [5], adjust service rate allocations of classes dynamically to meet the proportional delay differentiation constraints. However, for accurate rate allocation, the system should operate under high load conditions, this limits the applicability of the rate-based PDD algorithms. In contrast, our algorithm has good performance in various load conditions.

Dynamic-priority algorithms adjust the priority of a backlogged class according to its currently measured states. In WTP, the priority of a backlogged class is adjusted to be proportional to its head-of-line packet's delay normalized with respect to its delay differentiation parameter. Albeit simple, WTP implements the PDD model only when the system utilization approaches unity [2]. To overcome such limitation, adaptive WTP adjusts the priority of a class not only according to its experienced delay, but also based on the current class load condition. We find out that such adjustment is valid for certain network traffic with small degree of self-similarity. In contrast, the performance of our algorithm is independent of network traffic characteristics and load conditions.

Some algorithms determine the next packet according to the average queueing delay of backlogged classes. It is known that at time t, arrived packets of a class in a time window $[t-\tau, t]$, can be in one of the two states: departed or waiting in the queue. PAD considers the average delay of departed packets in the time window only. It is capable of achieving the PDD model constraints in various load conditions. However, PAD exhibits a pathological behavior in short time-scales; that is, occasionally higher classes to experience larger delays than lower classes, which is caused by its ignorance of those backlogged packets. To address this issue, HPD was proposed to take into account the average delay of departed packets and the delay of the head-of-line packet simultaneously. HPD enhances the average control quality of PAD, and meanwhile avoids its pathological behavior problem. It, however, achieves the class delay ratio with large statistical variations in short time-scales, which will be shown in Section IV. MDP considers the delay of all arrived packets of each class in the time window. In addition, it also takes into account the estimated delay of backlogged packets. In Section IV we shall show that MDP delivers performance comparable to HPD. However, its performance deteriorates as the target quality spacing between the classes is enlarged. Note that PAD, HPD and MDP schedule backlogged packets of different classes based on heuristic delay information of arrived packets. In [8], we proposed VirtualLength based on Little's Law. It measures the average arrival rates and queue lengths of different classes over a time window and calculates their forwarding priorities. LAD presented in this paper is based on a proof of Little's Law [7]. It considers the delay of departed packets as well as the delay of the packets in the backlogged queue in the time window.

III. Little's Average Delay Algorithm

A. Little's Law

In G/G/1 queueing system, Little's Law states that the average number of packets in the system is equal to the product of average arrival rate of packets and the average waiting time of the packets in the system. Define $L(T)$ as the average number of the packets in the system during the time interval $[0, T]$, $W(T)$ as the waiting time per packet averaged over all packets, $\lambda(T)$ as the average arrival rate. Suppose $W(T)$ and $\lambda(T)$ have limits as $T \to \infty$, that is

$$W = \lim_{T\to\infty} W(T), \text{ and } \lambda = \lim_{T\to\infty} \lambda(T).$$

Then, the limit of $L(T)$, denoted by L, exists and

$$L = \lambda W. \tag{2}$$

The beauty of Little's Law (2) is that it does not depend upon any particular queueing discipline (packet scheduling algorithms); nor does it depend upon any specific assumptions regarding the packet arrival distribution or the packet size distribution. It is applicable to the queueing system of each traffic class in the PDD model.

LAD algorithm controls the delay ratios between different classes based on the Little's Law. Substituting L/λ for W, the objective of PDD model in (1) leads to a new constraint:

$$\frac{L_i}{\lambda_i \delta_i} = \frac{L_j}{\lambda_j \delta_j}, \tag{3}$$

for any two classes i and j. To ensure proportional delay differentiation between two classes, their normalized queue length with respect to their respective arrival rates and delay differentiation parameters should be kept equal. The LAD algorithm is to control the delay ratio by adjusting their average queueing lengths according to their arrival rates.

Notice that (2) reveals an asymptotic (or stationary) relationship between the queue length, packet arrival rate, and packet waiting time in the system. It is not enough to guide PDD scheduling because the objective of proportional delay needs to meet in small time windows. Because most of Web requests are small in size, provisioning of relative delay differentiation service in short time-scales is as important as in long time-scales. LAD algorithm is based on a transient property of the queueing system, as revealed by a proof of the Little's Law [7]. Following is a sketch of the proof.

Suppose that packets $p_1, p_2, \ldots$ arrive at time $t_1, t_2, \ldots$ $(0 \leq t_i < t_{i+1})$, and depart at $t_1^d, t_2^d, \ldots$. The packets are not necessarily forwarded in FCFS discipline. Denote $N(T)$ the total number of arrived packets in the time interval $[0, T]$; $N^d(T)$ and $N^c(T)$ the number of departed packets and the number of waiting packets in queue, respectively. It follows that at time T,

$$N(T) = N^d(T) + N^c(T). \tag{4}$$

Define $I_i(t)$ as the presentation function of packet p_i at time t, that is

$$I_i(t) = \begin{cases} 1, & \text{if packet } p_i \text{ is present at time t;} \\ 0, & \text{otherwise.} \end{cases}$$

Then, we have

$$N^c(T) = \sum_{i=1}^{N(T)} I_i(t). \tag{5}$$

Since packet p_i stays in queue during the interval $[t_i, t_i^d]$ and its queueing delay $w_i = t_i^d - t_i$, we have

$$\int_0^T I_i(t)dt = \begin{cases} w_i, & t_i^d \le T; \\ T - t_i, & t_i^d > T. \end{cases} \tag{6}$$

Therefore, the cumulative queue length in the interval $[0, T]$ is

$$\begin{aligned} \int_0^T N^c(t)dt &= \sum_{i=1}^{N^d(T)+N^c(T)} \int_0^T I_i(t)dt \\ &= \sum_{\{i:t_i^d \le T\}} w_i + \sum_{\{i:t_i \le T, t_i^d > T\}} \big(T - t_i\big), \end{aligned} \tag{7}$$

and the average queue length in interval $[0, T]$ is

$$L(T) = \frac{1}{T}\int_0^T N^c(t)dt = \lambda(T)W(T), \tag{8}$$

where

$$\lambda(T) = \frac{N(T)}{T}, \tag{9}$$

$$W(T) = \frac{\sum_{\{i:t_i^d \le T\}} w_i}{N(T)} + \frac{\sum_{\{i:t_i \le T, t_i^d > T\}} \big(T - t_i\big)}{N(T)}. \tag{10}$$

Assuming that $\lambda(T)$ and $W(T)$ exist as $T \to \infty$, (8) leads to that

$$L = \lim_{T \to \infty} \lambda(T)W(T) = \lambda W. \tag{11}$$

This completes the proof.

B. The LAD Algorithm

The basic idea of LAD algorithm is to control the delay ratio of classes by monitoring their arrival rates and queueing delays of their arrived packets based on transient relationship between the queue length, arrival rate and waiting time, as revealed by (8). In particular, (10) defines the average waiting time per packet in a window of size T. The numerator of the first term actually represents the accumulated delays of all departed packets and the numerator of the second term represents the accumulated waiting time of the packets in the backlogged queue so far at time T. Accordingly, we define the LAD algorithm as follows.

For class i, the LAD scheduler maintains three control variables to monitor its traffic flow over finite time window T: the cumulative delays of departed packets W_i^d; the number of arrived packets N_i; and current queue length N_i^c. At the beginning of each time window, these variables are (re)initialized. Note that the size of T is in terms of number of successively departed packets from the system. These control variables are updated according to the following rules:

1) At the beginning of each time window, $N_i \leftarrow N_i^c$ and $W_i \leftarrow 0$.
2) Upon the receipt of a packet of class i, the packet is time-stamped and $N_i \leftarrow N_i + 1$, and $N_i^c \leftarrow N_i^c + 1$.
3) After transmitting a packet of class i, $N_i^c \leftarrow N_i^c - 1$ and $W_i^d \leftarrow W_i^d + w$, where w is the measured delay of the packet.

Let W_i^c denote the current cumulative delay of backlogged packets in the queue i. According to (10), we set the priority of class i as

$$\mathcal{P}_i = \frac{W_i^d + W_i^c}{N_i \delta_i}. \tag{12}$$

Whenever the queueing system is available for packet transmission, a backlogged packet of class j^* with the highest priority is selected. That is,

$$j^* = \arg \max_{1 \le i \le M} \mathcal{P}_i. \tag{13}$$

Ties for the highest priority are broken by serving the packet that has entered the queueing system earliest. Note that the validity of Little's Law does not depend upon any particular queueing discipline. Therefore, the next packet can be any backlogged packet if a more complicated scheduling algorithm is needed.

The determination of the time window size T is one important implementation issue of the LAD algorithm. It is known that Little's Law is valid when the time window is sufficiently large. However, provisioning PDD services in short time-scales is as important as in long time-scales. A good choice of T should strike a balance between system stability and responsiveness. On one side, a large T would avoid abrupt changes of average queueing delay due to bursty traffic. Particularly, when T is sufficiently large, the average delay of the packets in the time window would hide the effect of the distributions of packet arrivals and packet sizes. On the other side, a small T would lead to an agile scheduler that responds to the change of traffic conditions quickly. Although we leave T to be an adjustable parameter by network operators, in our simulations, we show that LAD is able to provide PDD services in both long and short time-scales.

In addition, for each packet transmission, LAD needs to calculate and compare the priorities of all backlogged classes, which requires at most N calculations and $N-1$ comparisons. The calculation overhead is mainly due to the update of control variables and time-stamping operations. The cost for update is small because it involves only a few number of add operations; the timestamping operation is assumed in the implementation of WTP and MDP as well.

IV. Simulation Results

In this section, we compare LAD with other PDD algorithms, including WTP, AWTP, PAD, HPD, and MDP. A primary performance metric is error between desired class delay ratio and achieved ratio. The results are an average of 1000 runs.

The experiments assumed the distributions of packet inter-arrivals and sizes are similar to those in [3], [4]. That is, the inter-arrivals between packets of a class follow a Pareto or Poisson distribution. The packets size are variable with a small number of choices. The transmission time of a packet is proportional to its size.

We compared LAD with other PDD algorithms, including WTP, AWTP, PAD, HPD, and MDP. In the experiments, we assumed two classes of traffic with the equal class loads. The packet arrivals of each class followed a Pareto distribution ($\alpha = 1.5$) and all the packets had equal size. The same stream of packets was used for all the experiments with different PDD algorithms. Our implementation of AWTP used the jumping window method, as suggested in [4], to estimate the arrival rate of traffic. HPD is a hybrid of WTP and PAD with a weighting parameter g. We set the parameter g to 0.875 as recommended in [3]. MDP takes into account the delay of departed packets and the estimated delay of all other waiting packets in the determination of class priorities. Although the MDP authors suggested a simplified method to approximate the average delay for all arrived packets to make a tradeoff between quality and run-time overhead [6], we implemented its original version in this experiment.

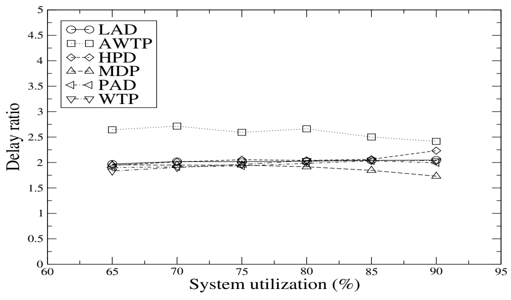
We also carried out experiments under various system conditions. The simulation results show that the algorithm is able to provide predictable and controllable differentiated services. The performance of the algorithm is independent of traffic characteristics. Due to the limitation of space, readers are referred to [9] for details.

A. Comparison in short time-scales

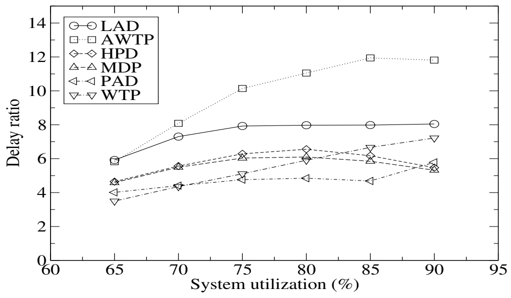
We first compared the short time-scale performance of the algorithms under different system utilization rates. The time window was set to $T = 100$ packets. The simulation results for the cases of $\delta_1/\delta_2 = 2$ and 8 are plotted in Fig. 1(a) and Fig. 1(b), respectively.

Fig. 1(a) shows that all the PDD algorithms, except AWTP, can meet the PDD constraints to an acceptable extent for a small delay ratio under moderate and high system load conditions. In particular, LAD achieves the desired delay ratio with minimum errors consistently. In contrast, HPD and MDP demonstrate good performance under moderate load conditions, but yield relatively large errors when the system utilization rate goes up to as high as 90%. Recall that HPD is a hybrid of WTP and PAD. Both WTP and PAD gain performance as the utilization rate increases, but their improvement rates are different. Hence, a linear combination of the WTP and PAD with a constant weighting parameter g in HPD is expected to generate a convex performance plot with respect to the utilization rate. This impact of linear combination can be seen more clearly in Fig. 1(b) for the case of a large desired delay ratio.

The reason for the inaccuracy of MDP in highly utilized systems is the estimation error of the delays of backlogged



(a) $\delta_1/\delta_2 = 2$.



(b) $\delta_1/\delta_2 = 8$.

Fig. 1. Delay ratios of class 1 to class 2 using different PDD algorithms in different system utilizations.

packets in a time window of $[t-\tau, \infty)$ at any time t. Although MDP can measure the delay of packets in the time window $[t - \tau, t]$, MDP uses a lower bound to estimate the delay of the packets in future $[t, \infty)$. With the increase of the system utilization, there are more packets in a backlogged queue during the interval τ, and consequently the estimation error increases. When the system utilization rate goes beyond certain point, the impact of estimation accuracy becomes significant and the overall performance of MDP starts to deteriorate. Fig. 1(b) shows that the estimation error is exaggerated in the case of a large desired delay ratio and the gap between LAD and MDP is enlarged.

Fig. 1 shows that WTP yields relatively large errors when the system utilization rate is moderate. This is consistent with the findings of other researchers [3], [4]. AWTP was proposed as a remedy of this problem [4]. It relies on a policy iteration algorithm to adjust the feasible set of control parameters according to the delay of the head-of-line packet in each class and the class load distributions. The algorithm is based on an assumption that the arrival process of each traffic class is a Poisson distributions. The authors showed that AWTP be applicable to the traffic of a *Pareto* distribution with the shape parameter $\alpha = 1.9$. The impact of α on the performance of AWTP is discussed in [9].

B. Comparison in long time-scales

We compared LAD with other PDD algorithms, focusing on their robustness in different time-scales. The experiment settings remain the same as in the last one, except that the system utilization rate is fixed at 90%. Fig. 2(a) and Fig. 2(b) show three percentiles (the 5th, 50th, and 95th) of achieved delay ratios for the target ratio of 2 and 8, respectively. We give the numbers in the figures directly for large percentiles.

Fig. 2 shows that LAD achieves the target ratios accurately in all of the time-scales that we tested and outperforms its competitors consistently in terms of the errors in various percentiles. This implies that LAD is more robust to keep the class delay ratio under control and deliver the desired ratio with small statistical variations. Although all the algorithms are able to meet the PDD constraints in terms of their medians with small deviations in long time-scales, LAD is outstanding to provide tight and robust control in a statistical sense over the class delay ratio in short time-scales.

Fig. 2(a) shows that all the PDD algorithms, except MDP, are able to achieve the delay ratio of 2 with a high probability in the short time-scale of 100 packets. LAD demonstrates an excellent robustness because more than 90 percentage of the total runs would produce ratios between 1.6 and 2.4. MDP is robust, as well, but its achieved ratios center around 1.6. In contrast, AWTP, HPD, and PAD exhibit a "heavy tail" property in that majority of the runs, under the control of the algorithms, would lead to delay ratios that are close to the target ratio of 2, but the algorithms could lose the control in a few occasions.

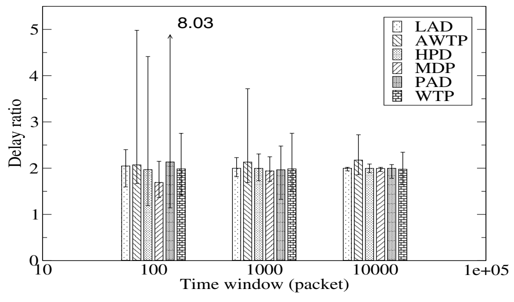
Fig. 2(a) also shows that the success probability of the algorithms increases with the time scale. In the long time-scale of 10000 packets, all the algorithms are able to achieve the target delay ratio robustly.

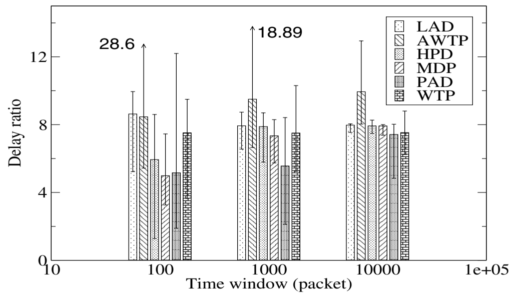
In comparison with Fig. 2(b), we observe that all the PDD algorithms lose certain degrees of robustness when the desired delay ratio δ_1/δ_2 is large. In the short time-scale of 100 packets, LAD performs slightly better than WTP and AWTP, but outperforms PAD, HPD and MDP significantly in terms of their medians. The goodness of WTP and AWTP are mainly due to the high utilization ratio (90%) that we assumed in this experiment. WTP and AWTP provide consistent levels of QoS, independent of the desired delay ratio. This is because they use extra control parameters to adjust the impact of the pre-defined delay ratio. But they are lack of robustness because of their medians with large statistical variations. PAD, HPD and MDP perform in a similar way to LAD. They differ in the way of delay estimation of arrived packets. Fig. 2(b) shows that their performance gap in short time-scales gets larger as the delay ratio increases. As the time-scale increases, all the PDD algorithms gain more control over the delay ratio. In the long time-scale of 10000 packets, LAD provides similar levels of QoS to HPD and MDP.

We conclude that in short time-scales, LAD consistently outperforms its competitors for large target delay ratios. For small target ratios, most of the algorithms can provide an acceptable level of quality of service. Under heavy load conditions and in long time-scales, LAD performs similarly to HPD and MDP. WTP, AWTP, and PAD are not as robust as the others due to their large statistical variations.

V. Conclusions

We have proposed a new proportional delay differentiation algorithm, called LAD, to implement the PDD model. The algorithm is derived from a proof of Little's Law. It monitors the arrival rate of the packets in each traffic class and their cumulative delays and achieves the desired class delay ratios in both short and long time-scales. Simulation results have shown, in comparison with other PDD algorithms, LAD provides no worse level of service quality in long time-scales and more accurate and robust control over the delay ratio in short time-scales. Our future work will address issues on Internet servers so as to provide end-to-end differentiated services to network applications and users.





(b) $\delta_1/\delta_2 = 8$.

Fig. 2. Percentiles of achieved delay ratios using different PDD algorithms in different time-scales.

References

[1] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. *An Architecture for Differentiated Services*. RFC 2475, December 1998.

[2] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. In *Proceedings of SIGCOMM*, pages 109–120, 1999.

[3] C. Dovrolis, D. Stiliadis, and P. Ramanathan. Proportional differentiated services: Delay differentiation and packet scheduling. *IEEE/ACM Transactions on Networking*, 10(1):12–26, 2002.

[4] M. K. Leung, J. C. Lui, and D. K. Yau. Adaptive proportional delay differentiated services: Characterization and performance evaluation. *IEEE/ACM Transactions on Networking*, 9(6):801–817, December 2001.

[5] J. Liebeherr and N. Christin. JoBS: Joint buffer management and scheduling for differentiated services. In *Proceedings of IWQoS 2001*, pages 404–418, June 2001.

[6] T. Nandagopal, N. Venkitaraman, R. Sivakumar, and V. Bharghavan. Delay differentiation and adaptation in core stateless networks. In *Proceedings of IEEE Infocom*, pages 421–430, April 2000.

[7] S. J. Stidham. A last word on $L = \lambda W$. *Operations Research*, 22(2):417–421, 1974.

[8] J. Wei, Q. Li, and C. Xu. VirtualLength: A new packet scheduling algorithm for proportional delay differentiation. In *Proceedings of ICCCN*, pages 331–336, October 2003.

[9] J. Wei, C. Xu, and X. Zhou. LAD: Little's average delay scheduling for proportional delay differentiation. Technical report, Department of Electrical and Computer Engineering, Wayne State University, June 2003.