*Regular Paper*

# Load Early Detection (LED): A Congestion Control Algorithm Based on Routers' Traffic Load

Arjan Durresi,[†1] Leonard Barolli,[†2] Mukundan Sridharan,[†3]
Sriram Chellappan[†3] and Raj Jain[†4]

Efficient bandwidth allocation and low delays remain important goals, especially in high-speed networks. Existing end-to-end congestion control schemes (such as TCP+AQM/RED) have significant limitations to achieve these goals. In this paper, we present a new and simple congestion control scheme, called Load Early Detection (LED), that achieves high link efficiency and low persistent queue length. The major novelty of LED is that it gauges the congestion level by measuring the traffic load instead of the queue length. We show analytically that this feature of LED enables it to detect congestion earlier and react faster than other schemes. To gain insight into the behavior of LED and compare it with RED, we analyze a simple fluid model, and study the relation between stability and throughput, especially for low delays. We evaluate the performance of LED using extensive *ns2* simulations over a wide range of network scenarios. Our simulation results show that LED achieves significantly higher link efficiency than RED (up to 83%), REM (up to 141%), and PI controller (up to 88%), especially in the presence of low delays.

## 1. Introduction

End-to-end traffic management and congestion control continues to be one of the major pillars for the robustness of the Internet [1]. Because of the randomness of Internet traffic, at least within short periods, over-provisioning cannot fully avoid congestion. Moreover, insufficient congestion control could lead to congestion collapse [1]. Minimizing queuing delays is especially important in high bandwidth networks. The end-to-end delay that affects users' traffic is the sum of three components: (a) propagation delay, (b) transmission delay, and (c) queuing delay. Propagation delay depends on the distance and speed of a signal, therefore for a given distance it cannot be minimized. Transmission delay is the ratio of the size of the data to be transmitted to the bandwidth. Therefore, it can be reduced by increasing the bandwidth. It is for this precise reason that users are willing to pay for more bandwidth. But, the advantage of using high bandwidth to reduce transmission delay can be lost in the presence of queuing delay, which increases the total end-to-end delay. Therefore, congestion control, with the active participation of routers, will be an important part of high-bandwidth Internet solutions. Since TCP is still the dominant protocol for the Internet, and in views of TCP's known behavior of congesting the network in order to probe its capacity, we need schemes that have a very fast response to an increase in the traffic level.

Although a number of schemes have been proposed for congestion control, the search for new schemes continues [1)~11)]. As a result of the increase in web traffic on the Internet, there is a need for algorithms to react faster to congestion. Surveys of various congestion control algorithms proposed for use in routers can be found in Low, et al. [12] and Medina, et al. [13].

The proposed solutions cover a wide spectrum of improvements. At one end of this spectrum there are simpler, more incremental, and more easily employable changes to the current TCP. Examples of such proposed solutions are Random Early Detection (RED) [14] and Explicit Congestion Notification (ECN) [15]. At the other end of the spectrum are solutions incorporating more powerful changes that result in new transport protocols with higher performance but with less chance to be deployed on a large scale on the Internet, at least in the immediate future. An example of such a solution is XCP [11]. Other proposals, such as Random Exponential Marking (REM) [2], Proportional In-

†1 Department of Computer Science, Louisiana State University, Baton Rouge, Louisiana, USA
†2 Department of Information and Communication Engineering, Fukuoka Institute of Technology (FIT), Japan
†3 The Ohio State University, Columbus, Ohio, USA
†4 Washington University in St. Louis, St. Louis, Missouri, USA

tegral (PI) controller [7], HighSpeed TCP [6], and Quick Start TCP [8] reside along the simplicity-deployability spectrum. In the end, the choice among all these solutions depends on making a tradeoff between performance and practical use that is best suited to the Internet. Because of the size and multidimensional complexity of the Internet, robustness in heterogeneity is valued over efficiency of performance, which causes evolutionary change to be favored over revolutionary change. For this reason, in our solution we propose minimal changes to RED/ECN and try to derive the maximum performance improvements out of them.

Today's routers use some form of Active Queue Management (AQM) scheme based on the queue length. RED [14] and ECN [15),16] are well-known algorithms, and are considered the current standards. RED randomly drops packets even before the queue is full, with a probability of packet drop calculated from the queue size. ECN is a modification to RED where a bit in the IP header is set and used when the routers decide to drop a packet. The probability-calculating function used in ECN is the same as that in RED (or with some minor modification, but still based on the size of the queues built in routers). In our scheme, we still use the framework of the ECN to convey the congestion information to the source. That is, we still mark the same bits used in ECN to indicate congestion, but we intend to change the probability-calculating algorithm of ECN. REM [2] and PI controller [7] are two of the other well-known algorithms to which we compare our scheme. Both REM and PI controller attempt to tune the average queue length to a target queue size. REM embeds the difference in target and average queue size along with the difference in input rate and the capacity of the link into a parameter price, which serves as a measure of congestion. Packets at the link are then marked or dropped exponentially in relation to the price. PI also employs the difference in average queue size and target queue size, as well as the difference between queue size in the previous time period and target queue. These are then scaled by factors $a$ and $b$ [7] to obtain the loss probability. In LED, presented here, the probability of packet mark/drop is calculated on the basis of the traffic load at routers. We also show that our LED performs better than RED, REM, and PI Controller. In particular, we show by theoretical and experimental analysis that LED reacts faster to congestion than other solutions. The predominance of short-lived traffic on the Internet requires routers to employ congestion control algorithms that detect congestion early and react fast.

The remainder of the paper is organized as follows. In Section 2, we present the LED algorithm and describe it in detail. In Sections 3 and 4, we analyze our scheme, using control theoretic tools. We use Delay Margin (DM) principles to check the stability of the system and analyze its effect on the throughput of LED in comparison with RED. The *ns2* simulations have been used to compare the performance of LED to that of the other schemes (RED, REM, and PI Controller). In Section 5, we present our simulation results using *ns2* [17] simulator. In Section 6, we discuss parameter settings and implementation issues. In Section 7, we present the conclusions of our research.

## 2. Algorithm for Calculating the Probability of Packet Marking Using LED

It is a known fact that in any AQM-based scheme the delay in the system could be traded for better throughput. The significance of our scheme is that it achieves a better tradeoff than the current AQMs. The principle behind the scheme is to use the traffic load at the router to determine the packet mark strategy. By traffic load, we imply the ratio of the number of bytes received to the capacity of the router in a particular interval of time. If an algorithm can keep the value of the traffic load always one, that is, if it can make the in-rate equal to the out-rate, then such a system will be able to guarantee a 100 percent throughput with zero delay.

The ultimate aim of any traffic management scheme is to keep the traffic load at the router equal to one. A traffic load of less than one leads to under-utilization of bandwidth, while a traffic load more than one results in congestion and queue build up. Rather than trying to control the queue at an optimum value and thereby indirectly controlling the traffic load, a scheme that directly controls the traffic load will perform better by itself adapting faster to changes in traffic load. Traffic-load-based schemes such as LED can sense an increase in traffic prior to queue build-up, which is a consequence of congestion.

A scheme based on queues can never sense an increase in traffic, unless there is an actual

queue build-up. For example, in RED, if the $min_{th}$ is set to, say, 20 packets, then the router would not start marking the packets unless the queue reaches at least 20 packets, but in absolute terms, even a queue of one would mean an imbalance between the in-rate and the out-rate. On the other hand, if we set the thresholds very low, the RED queue oscillates more and reaches the value zero often, thus, leading to a significantly lower throughput in the low-delay region. We also show in Sections 3 and 4, using control theoretical tools, that the queue in a scheme based on traffic load, such as LED, oscillates less and consequently leads to higher throughput in the low-delay region, than the queue in RED. In the high-delay region, which is not desirable for users, all schemes produce good throughput. Our proposed algorithm used to calculate the probability of packet marking or dropping is described in following section.

### 2.1 Algorithm

LED computes an average traffic load running average similar to the one used to calculate the queue length in TCP-RED. That is, the average load factor is computed as a weighted average of the previous average and the new (instantaneous) load estimate. The average traffic load is calculated at discrete intervals of time $t = n\tau$, where $n = 0, 1, 2, 3, \ldots,$ $n$ and $\tau$ is the time interval. The traffic load is given by Eq. (1):

$$L_{avg}(n\tau) = (1-\alpha)L_{avg}((n-1)\tau)+\alpha L(n\tau), \quad (1)$$

where

$$L_{avg}(n\tau) = Average\ traffic\ load\ at\ t = n\tau, \quad (2)$$

$$L(n\tau) = Traffic\ load\ at\ t = n\tau = \frac{Bytes\ arrived\ in\ (n\tau,(n+1)\tau)}{C\tau}, \quad (3)$$

$$\alpha = Averaging\ weight, 0 < \alpha < 1. \quad (4)$$

The reason for using an average load factor rather than an instantaneous one is that it captures more accurately the notion of congestion. Because of the bursty nature of the Internet traffic, the load factor can oscillate rapidly. Therefore, the weighted average calculation tries to balance the reaction to long-lived and instantaneous congestion, by filtering short-term changes in the load factor. The weighting parameter $\alpha$ is selected to smooth the instantaneous traffic load. A large $\alpha$ tracks changes in traffic load but might be heavily influenced by temporary fluctuations. On the other hand, a small $\alpha$ might not be quick enough to adapt to real changes in traffic. More details of how to select the weighting parameter $\alpha$ are given in Section 6.2. The average traffic load calculated at time $t = n\tau$ is used in the interval $[n\tau, (n+1)\tau)]$ to mark or drop packets.

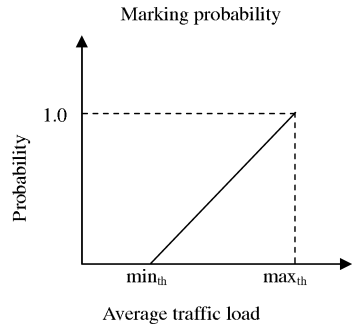


**Fig. 1** Probability of a packet being marked.

Two thresholds, namely, minimum threshold ($min_{th}$) and maximum threshold ($max_{th}$), are maintained for the traffic load. At the time of each packet arrival, the calculated average traffic load is compared to thresholds. If the average traffic load is below ($min_{th}$), no action is taken. If the traffic load is found to be between ($min_{th}$) and ($max_{th}$), the packet is marked if it is ECN-capable and dropped if it is not with some probability, whose calculation is described later in this section. If the traffic load is above ($max_{th}$), the packet is marked if it is ECN-capable and dropped if it is not.

$$If\ L_{avg}(n\tau) > max_{th}\ Mark/Drop\ packet;$$
$$else\ if(min_{th} < L_{avg} < max_{th})\ Mark/Drop\ packet\ with\ Probability\ p;$$

The calculation of the probability of packet marking or dropping follows a linear pattern as shown in **Fig. 1**. As the traffic load varies from minimum to maximum thresholds, the probability of marking varies from 0 to 1 linearly. The maximum probability of dropping in this scheme is 1. This is then adjusted against the *count*, which is the number of packets since the last drop or mark. The probability is also modified to take care of the packet length. Thus, the probability is given by

$$p = \frac{p' \times PacketSize}{Maximum\ Packet\ Size}, \quad (5)$$

where

$$p' = \frac{p''}{1 - count \times p''}$$

and

$$p'' = \frac{L_{avg} - min_{th}}{max_{th} - min_{th}}.$$

As shown in this section, the implementation of LED is simple. Only the probability-calculating algorithm need to be changed in routers that implement RED/ECN. No change is required in TCP end users.

### 2.2 Initialization and Idle Period Settings

The average traffic load is set to the minimum threshold during the initialization of the queue. When the link is idle for a particular period of time, which is denoted by the absence of any packet arrivals, the average traffic load is calculated during that period by setting the instantaneous traffic load to zero. Hence, if the link is idle for a period of $n$ time intervals, at the end of the idle period the average traffic load is scaled to

$$L_{avg}((n+k)\tau) = (1-\alpha)^n L_{avg}(k\tau). \quad (6)$$

At the end of the idle period, the average traffic load is set to the minimum of the value calculated above and the minimum threshold set for the average traffic load. Thus, the average traffic load is gradually reduced during the idle period, but never allowed to go below the minimum threshold. This is done to ensure a faster response of the router to the congestion information when it comes back from the idle period.

## 3. Stability Analysis

To study the stability of the LED system, we have developed a control theoretical model, which is presented in the Appendix. We use DM as the major metric [18] to estimate the stability of LED as a closed feedback system. If DM is negative, the system will oscillate. A positive DM represents how much the Round Trip Time (RTT) can be increased without violating the stability of the feedback system. Our goal is to keep the system stable, while the delay is low. If the system oscillates when the delay is low (i.e., when the queue is small), the system will have often an empty queue and therefore a low link utilization.

Assuming that a steady-state queue length exists at equilibrium, from Eq. (A.18), Eq. (A.19) and Eq. (A.21) the open loop transfer function of the LED scheme is

$$G_{LED} = \frac{\frac{C}{2N}s}{\left(s + \frac{2N}{R_0^2 C}\right)\left(s + \frac{1}{R_0}\right)} e^{-R_0 s} \times \frac{L_{LED}}{1 + \frac{s}{k_{LED}}}, \quad (7)$$

where

$$L_{LED} = \frac{1}{max_{th} - min_{th}}.$$

We further assume that the low-pass filter pole $k_{LED}$ is less than the corner frequency of TCP's dynamics and that it dominates the closed-loop system behavior. The unity-gain crossover frequency $\omega_{LED}$ (i.e., $|g_{LED}(j\omega_{LED}| = 1)$) thus satisfies

$$\omega_{LED} \ll min\left\{\frac{2N}{R_0^2 C}, \frac{1}{R_0}\right\}. \quad (8)$$

Then at low frequency we have

$$G_{LED}(s) \approx \frac{K_{LED}s}{1 + \frac{s}{k_{LED}}} e^{-R_0 s}, \quad (9)$$

where

$$K_{LED} = \frac{R_0^3 C^2}{(2N)^2} L_{LED}. \quad (10)$$

The phase margin of the LED system Eq. (7) without delay is:

$$PM_{LED}(\omega_{LED}) = \pi + \frac{\pi}{2} - \arctan\left(\frac{\omega_{LED}}{k_{LED}}\right) = \frac{3\pi}{2} - \arctan\left(\frac{\omega_{LED}}{k_{LED}}\right), \quad (11)$$

and the DM, which represents how much the RTT can be increased without violating the stability of the feedback system, is then

$$DM_{LED}(\omega_{LED}) = \frac{PM_{LED}(\omega_{LED})}{\omega_{LED}} - R_0 = \frac{\frac{3\pi}{2} - \arctan\left(\frac{\omega_{LED}}{k_{LED}}\right)}{\omega_{LED}} - R_0, \quad (12)$$

where $\omega_{LED}$ is the smallest solution of $|G_{LED}(j\omega_{LED})| = 1$, i.e.,

$$\omega_{LED} = \frac{1}{\sqrt{K_{LED}^2 - \frac{1}{k_{LED}^2}}}. \quad (13)$$

On the other hand, as shown in Hollot, et al. [20], the open-loop transfer function of the classical TCP-RED scheme is

$$G_{RED}(s) \approx \frac{K_{RED}s}{1 + \frac{s}{k_{RED}}} e^{-R_0 s}, \quad (14)$$

where,

$$K_{RED} = \frac{R_0^3 C^2}{(2N)^2} L_{RED}, \tag{15}$$

$$L_{RED} = \frac{P_{max}}{max_{th} - min_{th}}, \tag{16}$$

$$k_{RED} = -C \ln(1 - \alpha_{RED}), \tag{17}$$

where $\alpha_{RED}$ is RED's queue-averaging weight. The DM for RED is thus

$$DM_{RED}(\omega_{RED}) = \frac{\pi - \arctan\left(\frac{\omega_{RED}}{k_{RED}}\right)}{\omega_{RED}} - R_0, \tag{18}$$

where

$$\omega_{RED} = k_{RED}\sqrt{K_{RED}^2 - 1}.$$

Note that the derivative term appearing in the open-loop transfer function from Eq. (A.18) adds a 90° phase and thus increases the DM of the system. Therefore, the oscillations in the queue are reduced, leading to an improvement in the throughput, with correspondingly low delays.

To achieve high stability, we need to keep (by changing the scheme parameters) a large positive value of DM in the presence of low delays. The delay level is set by $min_{th}$ and $(max_{th} - min_{th})$. A larger DM would lead to fewer oscillations in the system. In **Fig. 2**, **Fig. 3**, and **Fig. 4**, we plot the DM as a function of the network parameters $N$, $(max_{th} - min_{th})$ and $\alpha$. The DM increases as $N$, $(max_{th} - min_{th})$, increases. It decreases as $\alpha$ increases.

Figure 2 shows the variation of the DM with $N$, for the link capacity $C = 200$ packets/sec, $R_0 = 0.06$ s, $\alpha = 0.05$, $min_{th} = 0.8$, and $max_{th} = 1.2$. Figure 3 shows the variation of DM with $(max_{th} - min_{th})$ for $C = 200$ packets/sec, $R_0 = 0.06$ s and $N = 5$. Figure 4 shows the variation of DM with $\alpha$ for $C = 200$ packets/sec, $R_0 = 0.06$ s, $min_{th} = 0.8$, $max_{th} = 1.2$, and $N = 5$.

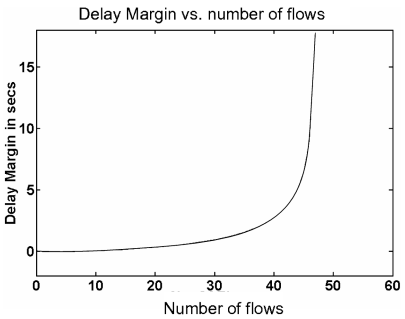


**Fig. 2** Variation of DM with $N$.

The estimator of stability for low delay is the DM, which is shown in Fig. 2, Fig. 3, and Fig. 4. Because the fluid model used for the system is an approximation of TCP, we validate the model with *ns2* simulations. In the following, we analyze the performance of the LED scheme. Our comparisons are based on *ns2* plots of average traffic load and queue length versus time. **Figure 5** shows the performance of LED scheme with corresponding parameters specified in **Table 1**.

We now tune the system for a more stable response. From Eq. (12), we can do this by decreasing $k_{LED}$ (decreasing $\alpha$), as this will give a higher delay margin, making the system less oscillatory. We thus decrease $\alpha$ to 0.001. The response of the system is shown in **Fig. 6**, where

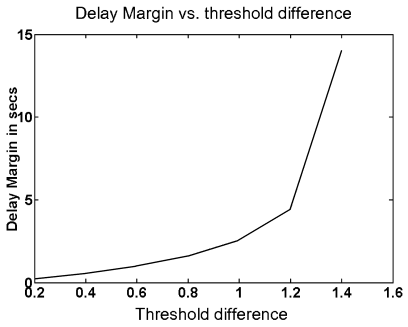


**Fig. 3** Variation of DM with $min_{th} - max_{th}$.

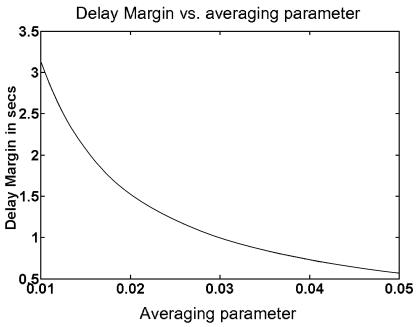


**Fig. 4** Variation of DM with $\alpha$.

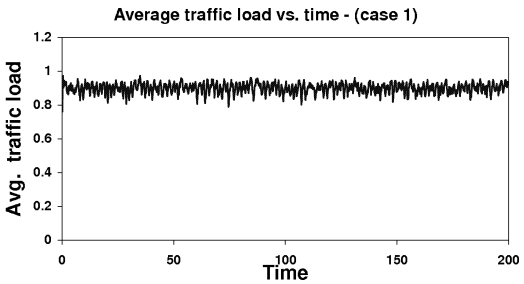


**Fig. 5** Traffic load vs. time in case 1.

**Table 1** Simulation parameter choices.

| Case | Weight | $max_{th}$ | $min_{th}$ | C | N |
|---|---|---|---|---|---|
| 1 | 0.05 | 1.2 | 0.8 | 200 | 10 |
| 2 | 0.01 | 1.2 | 0.8 | 200 | 10 |
| 3 | 0.005 | 1.2 | 0.8 | 200 | 10 |
| 4 | 0.05 | 1.2 | 0.8 | 200 | 20 |

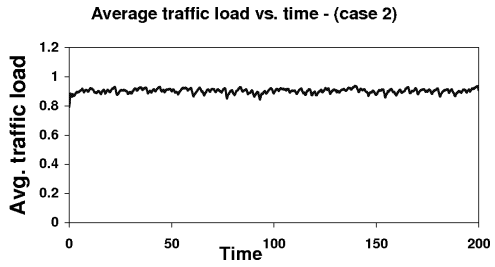


**Fig. 6** Traffic load vs. time in case 2.

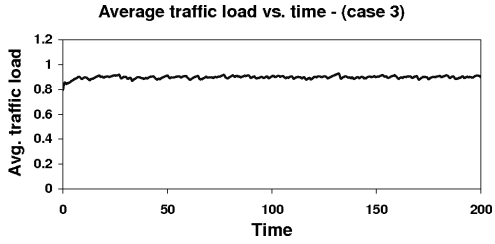


**Fig. 7** Traffic load vs. time in case 3.

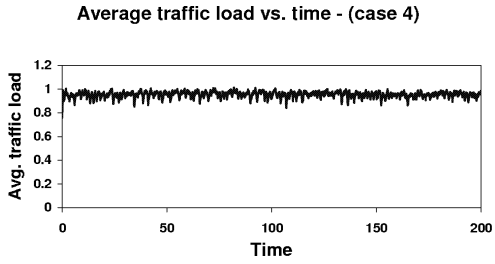


**Fig. 8** Traffic load vs. time in case 4.

the system is less oscillatory. However, an increase in the averaging parameter causes a more sluggish response, and there are still certain oscillations in the queue.

We can further stabilize the system by decreasing the averaging parameter as shown in **Fig. 7**, where $\alpha = 0.005$.

The number of flows in the router also plays an important role in the queue oscillations. A larger $N$ will lead to fewer oscillations, as shown in Eq. (12). As in case 1, we keep $\alpha = 0.05$ and increase $N$ to 20. The response shown in **Fig. 8** is less oscillatory than that in Fig. 5.

As shown by the above experiments, the predictions of our model are validated by *ns* simulations. Our model can thus be used to tune the parameters in the LED scheme.

## 4. Performance Improvement

In this section, we use the DM to analyze why the LED scheme performs much better than the RED scheme in the low-delay region, as in Quet, et al. [19].

The DM measures the stability of the system. In the low-delay region, we will show that RED has more oscillations in the queue than the LED scheme. Oscillations in the queue in the low-delay region cause the queue to go to zero often. In this case, the link is greatly underutilized, which reduces the throughput. In contrast, LED enables higher throughput in the low-delay region because there are fewer oscillations. We proceed to explain this result by comparing the DM for both cases.

Equation (12) gives us the DM for the LED scheme. From Hollot, et al. [20], we derived DM for RED in Eq. (18).

We can see that DM is a decreasing function of $\omega_g$ and $k$ (for both the RED and LED schemes). Then, from Eq. (12) and Eq. (18), if $K_{LED}$ is less than $K_{RED}$, and $k_{LED}$ is less than $k_{RED}$, we have a higher DM for the LED scheme than for RED. Our comparison of the RED and LED schemes uses the same value of $R_0$. From Eq. (10) and Eq. (15), we see that $K_{LED}$ is less than $K_{RED}$ in the low-delay range, where $min_{th}$ and $(max_{th} - min_{th})$ are low for the RED scheme. If $min_{th}$ is set high, then we operate in the high-delay range, where oscillations in the queue have no effect on the throughput of the system. From Eq. (A.9) and Eq. (17), we see that $k_{RED}$ is higher than $k_{LED}$ in most cases. This implies that, in the low-delay region, oscillations are higher in the RED scheme than in the LED scheme.

The above DM analysis shows that LED enables fewer oscillations than RED in the low-delay regions. This means that the traffic load in the RED scheme oscillates more than the traffic load in the LED scheme. From Eq. (A.7), we see that oscillations in the queue in the LED scheme are fewer than in the RED scheme. Thus, in the low-delay region, in the LED scheme the queue does not go to zero as often as it does in similar regions for RED. We can see the confirmation of this phenomenon in Fig. 11, Fig. 12, Fig. 13, and Fig. 14, which were obtained by *ns2* simulations.

Consider the cases shown in Fig. 12 and

Fig. 14. The capacity of the router's output link is $C = 187.5$ packets/sec, number of flows $N = 30$, $\alpha = 0.05$, and $T_p = 52$ ms. For the case of the LED scheme, the thresholds were set to 0.8 and 1.5. We observe an $R_0$ value of 0.2 s in Fig. 12. Furthermore, in the cases we have analyzed, we assume that $\tau = R_0$ to compute $k_{LED}$ from Eq. (A.9). The calculated DM is higher in the LED scheme than in RED. Therefore, the throughput is expected to be higher in LED than in RED.

The above theoretical analysis is confirmed by the simulation results presented in Section 5. We would like to stress that, for high-bandwidth networks, it is important to obtain high throughput while keeping the queuing low. The main reason for users to use and pay for higher-bandwidth networks is to reduce the whole communication time by reducing transmission delay. Therefore, it is very important to minimize the queuing delay. Otherwise, what would be gained in terms of delay from using high bandwidth would be lost through queuing delays. In queuing systems, including RED and LED, it is generally possible to trade throughput for delay, which means that such systems could provide high throughput in the presence of high delays. But, as we explained, especially in high-bandwidth networks long queuing delays cannot be tolerated. As is shown by the simulation results, LED guarantees higher throughput than RED in the low-delay range. Therefore, the ability of the LED scheme to provide higher throughput in the low-delay region is its principal advantage, which finally justifies its implementation.

## 5. Simulation Results

### 5.1 FTP Traffic Model

Three different simulation configurations are used to compare the LED scheme to RED [14], REM [2], and PI controller [7]. Our goal is to minimize the changes required in routers, in order to have a chance for the protocol to be used in practical conditions. For this reason, we do not compare LED to solutions that require the implementation of new protocols in routers and end users such as XCP [11].

The first configuration simulates FTP traffic. This model is used to study the effects of the scheme on the TCP dynamics. A number of sources, $S_1$, $S_2$, $S_3$, ..., $S_n$, are connected to a router $R_1$ through 10 Mb/s, 2 ms delay links. Router $R_1$ is connected to $R_2$ through a 5 Mb/s, 20 ms delay link, and is the congested link in this configuration. A number of destination nodes $D_1$, $D_2$, $D_3$, ..., $D_n$ are connected to the router $R_2$ via a 10 Mb/s, 4 ms delay link. The packet size is 1,000 bytes. **Figure 9** shows the simulation configuration.

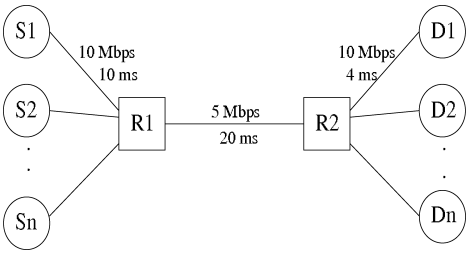


**Fig. 9** Network configuration.

**Table 2** Parameter values for RED, REM, and LED.

| Parameter | RED | LED | REM |
|---|---|---|---|
| Queue weight | 0.002 | | |
| Maximum probability | 0.2 | | |
| Minimum threshold | 1–40 | 0.8 | |
| Maximum threshold | 3–120 | 0.95–1.75 | |
| LED weight | | 0.05 | |
| Update interval (sec) | | 0.05 | 0.002 |
| $\gamma$ | | | 0.01 |
| $\phi$ | | | 1.001 |
| Target queue | | | 1-80 |

The parameter settings for RED, REM, and LED are shown in **Table 2**. Another simulation is performed using the above topology, but with variation in the RTT of the sources. The link delay for source and destination links varies from 1–10 ms, resulting in different RTT for different flows. In addition reverse path traffic is induced by adding some sources starting at nodes $D_1$, $D_2$, and so on.

### 5.2 Mixed Traffic Model

The mixed traffic model is configured to simulate web traffic. It consists of

- Several exponential traffic sources running over the UDP, which create a self-similar traffic, used to model the Internet background traffic.
- Several web sources and clients, to model the web traffic to be analyzed.
- Between five and ten FTP sources to model the bulk data traffic.

The network configuration used was an unbalanced dumbbell topology, very similar to the configuration for FTP traffic. The sources connected to node $R_1$ were replaced by a combination of web sources, FTP sources, and exponential sources, as mentioned above. The destination nodes were collection web clients, TCP

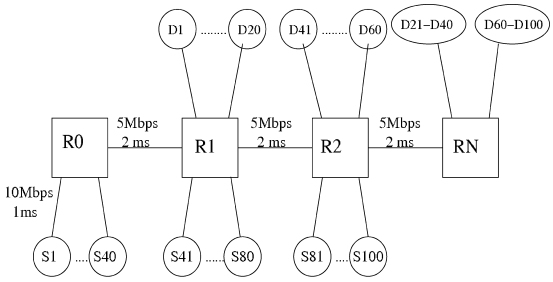


**Fig. 10** Simulation configuration for multiple congested gateways.

and UDP sinks.

In this model, we wanted to simulate typical high-bandwidth Internet conditions, which are shown to be characterized by self-similar traffic. We measure the "self-similarity" of our model traffic by using the Hurst parameter. This parameter indicates the degree of traffic self-similarity. Usually, the Hurst parameter of web traffic is between 0.75 and 0.85. For our mixed model, the Hurst parameter was measured by using a variance-time plot [21], and was found to be 0.807.

### 5.3 Multiple Congested Gateways Model

The multiple congested gateways model is used to observe the response of the scheme when multiple congested links all following the same queue management scheme are used. It is a typical parking lot configuration. Different flows in the network travel for different distances. There are $N+1$ routers in the network, $R_0$ to $R_N$. At each router, 20 flows enter the network. In our experiment, we used a configuration with four routers. In addition, 20 flows exist between routers $R_0$–$R_1$ and $R_1$–$R_2$. The simulation configuration is shown in **Fig. 10**.

### 5.4 Simulation Results

In all configurations, the throughput versus the queuing delay at the congested link is plotted. The parameters used for the schemes are shown in Table 2 and **Table 3**. RED, REM, and LED have the same parameters for all the configurations. For PI, the two critical parameters $a$ and $b$ are calculated by following the procedure outlined in Hollot, et al. [7]. For REM, the guidelines specified in Athuraliya [22] have been followed to obtain high responsiveness. For the multiple congested gateways, configuration the total system throughput versus the end-to-end queuing delay is compared. For flows using TCP as transport protocol, the throughput is calculated for the total number of acknowledged packets. For UDP packets, only the packets successfully reaching the destination node are included in the calculation. It should be noted here that this throughput is different from the actual number of packets that has actually left the congested link, since duplicate acknowledgments and packets that may have been dropped at destination nodes are not included in throughput measurement.

**Table 3** Parameter values for PI.

| Parameter | FTP | Mixed | Multiple gateways |
|---|---|---|---|
| Target queue | 1–80 | 1–80 | 1–80 |
| A | 0.1965 | 0.09 | 0.0592 |
| B | 0.024 | 0.013 | 0.0102 |

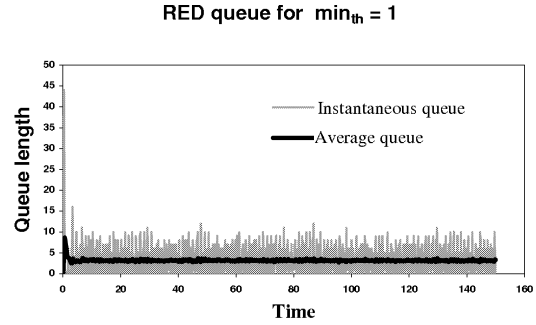


**Fig. 11** RED queue for $min_{th} = 1$.

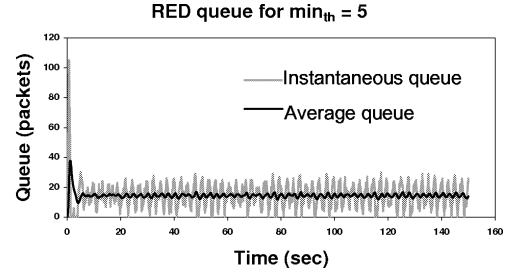


**Fig. 12** RED queue for $min_{th} = 5$.

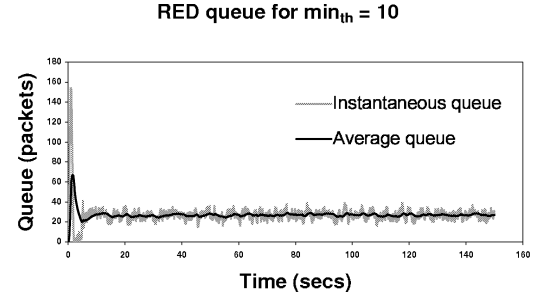


**Fig. 13** RED queue for $min_{th} = 10$.

**Figure 11**, **Fig. 12** and **Fig. 13** plot the average queue length at the output link for RED, while **Fig. 14** does the same for LED. The

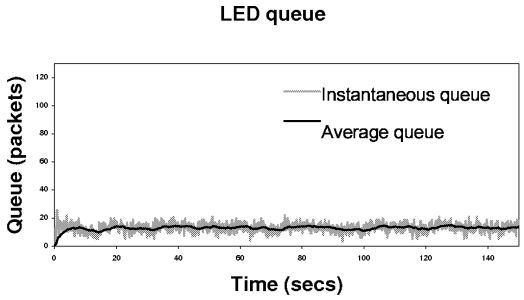


**Fig. 14** Queue for the LED scheme.

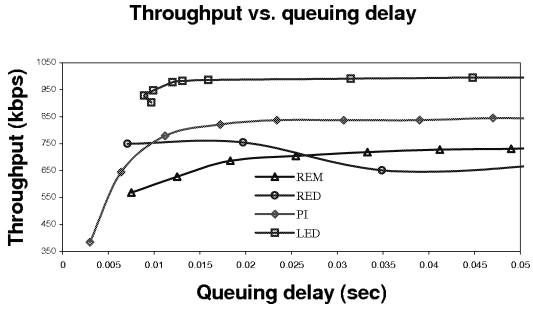


**Fig. 15** Throughput vs. queuing delay (FTP traffic).

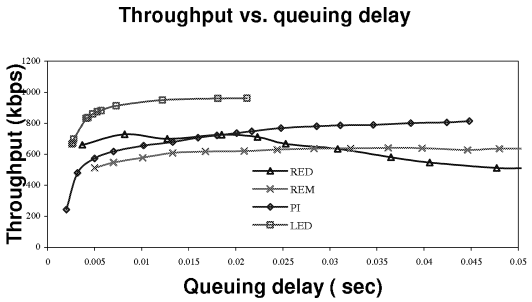


**Fig. 16** Throughput vs. queuing delay (with reverse traffic, FTP traffic).

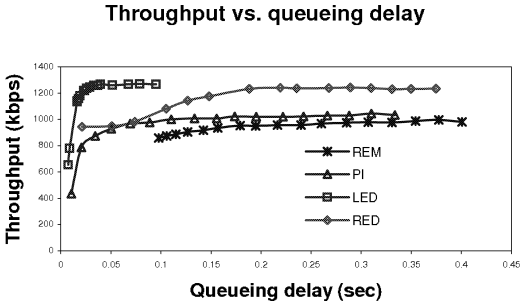


**Fig. 17** Throughput vs. queuing delay (multiple congested gateways).

oscillations in the queue decrease for RED as the difference between the minimum and maximum thresholds is increased from 2 to 20 packets (when the minimum threshold is increased from 1 to 10, the maximum threshold becomes three times the minimum threshold), with a corresponding increase in the throughput, albeit at a higher delay. The average queue for the LED scheme is shown in Fig. 14, for a minimum threshold of 0.8 and a maximum threshold of 1.2.

Comparing the queues in Fig. 12 and Fig. 14, it can be observed that LED leads to fewer oscillations in the queue than RED, for similar queuing delay. Therefore, LED enables a higher utilization of the link than RED. These results can be explained by the faster response to congestion by LED than that of RED. As shown in Fig. 13, to achieve a similar throughput as LED, RED needs to operate with an average delay (20 packets) higher than LED. Therefore, these experimental results validate our theoretical conclusion that LED with correspondingly low delays can achieve higher throughput than RED.

We plot four cases of throughput versus queuing delay graphs that illustrate superiority of LED performance to those of RED, REM, and PI controller. The first two cases, the **Fig. 15** and **Fig. 16**, show the throughput for FTP sources. The simulation configuration is shown in Section 5.1. In Fig. 15, we set the minimum threshold to 0.8 and vary the maximum threshold from 1.0 to 1.75 in such a way that the average delay increases with throughput. Figure 15 shows the FTP throughput with forward path traffic only. As can be seen, LED performs better in terms of link efficiency than RED (up to 115%), REM (up to 141%), and PI controller (up to 44%), in the presence of low delays. Figure 16 shows the throughput when reverse traffic is also included. Again, in the low-delay region, LED performs better in terms of link efficiency than RED (up to 42%), REM (up to 65%), and PI controller (up to 188%).

**Figure 17**, shows the result for the multiple congested gateways configuration. As can be seen, LED performs better in terms of link efficiency than RED (up to 34%), REM (up to 46%), and PI controller (up to 43%), especially in the presence of low delays. It can also be observed that the effect is multiplied in comparison with the simple FTP configuration in Fig. 15. For the results shown in **Fig. 18**, we used the simulation configuration described in Section 5.3. This configuration is used to test our scheme on a more realistic traffic model, which includes simultaneous web, FTP, and

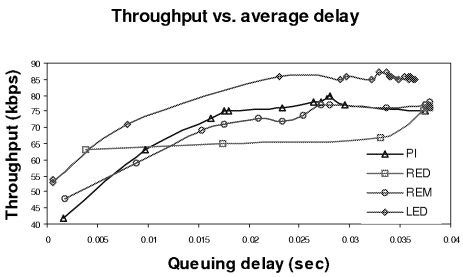


**Fig. 18** Throughput vs. queuing delay (mixed traffic model).

UDP traffic. As shown in Fig. 18, in the low-delay region LED performs better in terms of link efficiency than RED (up to 83%), REM (up to 75%), and PI controller (up to 75%).

## 6. Discussion of Parameters and Implementation

### 6.1 Time Interval

The first parameter to be considered is the interval at which the traffic load is estimated. The effect of the actions taken by the traffic management scheme affects the router one round trip time later. Hence the time constant of the dynamics of the traffic load cannot be faster than the lowest RTT among all of the flows. While the choice of estimated interval in the presence of different RTT-s is beyond the scope of this paper, a practical solution would be to set $\tau$ equal to the average RTT. For all our simulations, we chose an estimation interval comparable to RTT.

### 6.2 Traffic Load Weight $\alpha$

Following the same principles of the RED time constant as in Floyd and Jacobson [14] and Floyd, et al. [23], we assume that it takes $-1/ln(1-\alpha)$ time intervals for the traffic load estimator to reach 63% of a new value, or that the time constant of our system is $-\tau/ln(1-\alpha)$ seconds. Letting this time constant in our system to be 1 second, we get

$$\alpha = 1 - e^{-\tau}. \quad (19)$$

### 6.3 Thresholds ($min_{th}$ and $max_{th}$)

Minimum threshold setting for the algorithm determines how soon the packet marking is commenced. A threshold of 0.8 indicates that the algorithm starts marking or dropping as soon as the number of bytes that have arrived exceeds 80% of the capacity of the link. The maximum threshold determines the size of burst the link can tolerate. The higher the maximum threshold, the higher the link delay and vice versa. The setting for the thresholds depends on the amount of acceptable delay for the system and the input traffic pattern. One broad guideline would be to set $max_{th}$ and $min_{th}$ on either side of 1.0, both for stability and performance reasons. We are working on tuning the minimum and maximum thresholds adaptively, according to the traffic load.

### 6.4 Implementation Issues

The implementation of the LED scheme is going to be similar to that of RED, since they have a similar architecture. The overhead involved in calculating the probability is going to be less than in RED, since it is done once every second and not at the time of each packet arrival. The overhead is comparable with that of REM and PI controller, since both employ a similar time period. The complexity of setting the parameters is also similar to that of REM, PI, or RED.

## 7. Conclusions

In this paper, we have presented LED, a new and simple congestion control scheme. LED has a faster response to congestion than other solutions, because it measures the congestion level by using the traffic load at routers.

To better understand the behavior of LED and compare it with RED, we developed a fluid model, and used this model to analyze the stability and efficiency of LED, especially for low delays.

Using extensive *ns2* simulations, we show that LED achieves high utilization for low delays. Our simulation results show that LED performs better in terms of link efficiency than RED (up to 83%), REM (up to 141%), and PI controller (up to 88%), especially in the presence of low delays.

## Appendix

### A.1 Appendix: Mathematical Model

In the following, we develop a mathematical model that will allow us to justify our design by carrying out a stability analysis of the implicit feedback control system the scheme forms. Moreover, this model will help explain the performance improvement we observe while comparing the LED to the RED-ECN scheme, and will provide guidelines for tuning the design parameters.

We consider a network configuration consisting of a single router supporting $N$ homogeneous TCP flows. As shown in Hollot, et al. [24] and Misra, et al. [20], the congestion avoidance

mode of TCP can be modeled by using a fluid flow approximation:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t) \times W(t - R(t))}{2 \times R(t - R(t))} \times p(t - R(t)), \quad \text{(A.1)}$$

where $W(t)$ is TCP's congestion window size, $R(t)$ is the RTT delay and $p(t)$ is the probability of packet mark due to the AQM mechanism at the router. The RTT $R(t)$ is composed of the propagation delay $T_p$ and the queuing delay $\frac{q(t)}{C}$, where $q(t)$ is the queue length of the buffer and $C$ is the outgoing link capacity, and thus

$$R(t) = T_P + \frac{q(t)}{C}. \quad \text{(A.2)}$$

The traffic load $L(t)$ is defined as the ratio of the aggregate incoming rate to the router and the outgoing link capacity. Hence,

$$L(t) = \frac{BytesArrived(n\tau, (n+1)\tau)}{C\tau}. \quad \text{(A.3)}$$

Now if $W_x(t)$ is the window of packets that could be sent in $\tau$ seconds, then

$$L(t) = \frac{NW_x(t)}{C\tau}. \quad \text{(A.4)}$$

$W_x(t)$ is given by

$$W_x(t) = \frac{W(t)\tau}{R(t)}. \quad \text{(A.5)}$$

$L(t)$ becomes

$$L(t) = \frac{NW(t)}{CR(t)}. \quad \text{(A.6)}$$

We then have

$$\frac{\dot{q}(t)}{C} = L(t) - 1. \quad \text{(A.7)}$$

The above equation is true for both the RED and LED schemes. Using similar techniques to those used in Hollot, et al. [20], Eq. (1) is equivalent to the frequency-domain relationship:

$$L_{avg}(s) = \frac{L(s)}{1 + \frac{s}{k_{LED}}} \quad \text{(A.8)}$$

where

$$k_{LED} = \frac{-\ln(1-\alpha)}{\tau}, \quad \text{(A.9)}$$

and the packet marking probability obeys the following equation:

$$p(t) = \frac{L_{avg}(t) - min_{th}}{max_{th} - min_{th}}. \quad \text{(A.10)}$$

From the above equations, the equilibrium of the dynamical system satisfies the following equations:

$$W_0^2 p_0 = 2, \quad \text{(A.11)}$$

$$p(t) = \frac{L_0(t) - min_{th}}{max_{th} - min_{th}}, \quad \text{(A.12)}$$

$$L_0 = \frac{NW_0}{CR_0}, \quad \text{(A.13)}$$

$$R_0 = T_P + \frac{q_0}{C}. \quad \text{(A.14)}$$

Thus, since we would like to operate with a unitary traffic load ($L_0 = 1$) and an equilibrium queue $q_0 = 0$, the traffic management scheme parameters $min_{th}$ and $max_{th}$ should satisfy:

$$\frac{Tp^2C^2}{N^2} \frac{1 - min_{th}}{max_{th} - min_{th}} = 2. \quad \text{(A.15)}$$

Following similar techniques to the ones used in Hollot, et al. [20], we then carry out a linearization of Eq. (7) around the operating point. To do so, we make the following assumptions: we neglect the effect of an eventual queue length on the RTT dynamic (which is reasonable if we keep a small queue size), we assume that $L_0 = 1$, and we neglect some high-frequency dynamics (similarly to Hollot, et al. [20]). The perturbed variables about the operating point then satisfy

$$\frac{\delta W(s)}{\delta p(s)} = \frac{\frac{R_0C^2}{2N^2}}{s + \frac{2N}{R_0^2C}} e^{-sR_0}. \quad \text{(A.16)}$$

From Eq. (10), we also have

$$\frac{\delta L(s)}{\delta W(s)} \approx \frac{N}{R_0C}. \quad \text{(A.17)}$$

Thus, the plant to be controlled by the traffic management scheme (transfer function form $\delta p$ to $\delta TL$) is

$$P(s) = \frac{\frac{C}{2N}}{s + \frac{2N}{R_0^2C}} e^{-sR_0}, \quad \text{(A.18)}$$

while our scheme (transfer function form $\delta TL$ to $\delta p$) has the following linearization:

$$C(s) = \frac{L_{LED}}{1 + \frac{s}{k_{LED}}}, \quad \text{(A.19)}$$

where

$$L_{LED} = \frac{1}{max_{th} - min_{th}}. \quad \text{(A.20)}$$

A block diagram of our scheme is shown in **Fig. 19**. We further assume that the low-pass filter pole $k_{LED}$ is less than the corner frequency of TCP's congestion window dynamic and that it dominates the closed-loop system behavior. The unity-gain crossover frequency $\overline{w}g$ (i.e., $|G_{LED}(jw_g)| = 1$), where

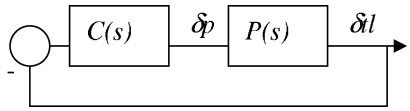


**Fig. 19** Block diagram of LED model.

$$G_{LED}(s) = P(s)C(s) \tag{A.21}$$

is the open loop transfer function thus satisfies

$$w_g \ll \frac{2N}{R_0^2 C}. \tag{A.22}$$

Then, at low frequency, we have

$$G_{LED} \approx \frac{K_{LED}}{1 + \frac{s}{k_{LED}}} e^{-R_0 s}, \tag{A.23}$$

where

$$K_{LED} = \frac{(R_0 C)^2 L_{LED}}{(2N)^2}. \tag{A.24}$$

## References


1) Floyd, S. and Fall, K.: Promoting the Use of End-to-End Congestion Control in the Internet, *IEEE/ACM Transactions on Networking*, Vol.7, Issue 4, pp.458–472 (Aug. 1999).
2) Athuraliya, S., Low, S., Li, V. and Yin, Q.: REM Active Queue Management, *IEEE Network Magazine*, Vol.15, No.3, pp.48–53 (Aug. 2001).
3) Clark, D.D. and Fang, W.: Explicit Allocation of Best-effort Packet Delivery Service, *IEEE/ACM Transactions on Networking*, Vol.6, No.4, pp.362–373 (Aug. 1998).
4) Durresi, A., Sridharan, M., Liu, C., Goyal, M. and Jain, R.: Congestion Control Using Multilevel Explicit Congestion Notification in Satellite Networks, *Proc. 10th IEEE International Conference on Computer Communications and Networks* (*ICCCN*), pp.483–488, Phoenix, AZ (2001).
5) Feng, W., Kandlur, D., Saha, D. and Shin, K.: Blue: A New Class of Active Queue Management Algorithms, Technical Report UM-CSE-TR-387-99, University of Michigan (1999).
6) Floyd, S.: HighSpeed TCP for Large Congestion Windows, IETF, RFC 3649 (Dec. 2003).
7) Hollot, C., Misra, V., Towsley, D. and Gong, W.B.: On Designing Improved Controllers for AQM Routers Supporting TCP Flows, *Proc. INFOCOM-2001*, San Francisco, CA, Vol.3, pp.1726–1734 (2001).
8) Jain, A., Floyd, S., Allman, M. and Sarolahti, P.: Quick-Start for TCP and IP, IETF Internet-draft, draft-amit-quick-start-04.txt (Feb.2000).
9) Jain, R., Kalyanaraman, S. and Viswanathan, R.: Ordered BECN: Why We Need a Timestamp or Sequence Number in the RM Cell, *ATM Forum/94-0987* (Oct. 1994).
10) Kalyanaraman, S., Jain, R., Fahmy, S., Goyal, R. and Vandalore, B.: The Erica Switch Algorithm for ABR Traffic Management in ATM Networks, *IEEE/ACM Transactions on Networking*, Vol.8, No.1, pp.87–98 (Feb. 2000).
11) Katabi, D., Handley, M. and Rohrs, C.: Internet Congestion Control for Future High Bandwidth-delay Product Environments, *Proc. ACM SIGCOMM-2002*, pp.69–102 (Aug.2002).
12) Low, S.H., Paganini, F. and Doyle, J.C.: Internet Congestion Control, *IEEE Control Systems Magazine*, Vol.22, pp.28–43 (Oct. 2002).
13) Medina, A., Allman, M. and Floyd, S.: Measuring the Evolution of Transport Protocols in the Internet, *ACM CCR*, Vol.35, No.2, pp.35–51 (Apr. 2005).
14) Floyd, S. and Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, Vol.1, Issue 4, pp.397–413 (Aug. 1993).
15) Ramakrishnan, K. and Floyd, S.: Proposal to Add Explicit Congestion Notification (ECN) to IP, IETF, RFC 2481 (Jan. 1999).
16) Floyd, S.: TCP and Explicit Congestion Notification, *ACM SIGCOMM Computer Communications Review*, Vol.24, No.5, pp.8–23 (Oct. 1994).
17) NS Project Homepage: Available at http://www.isi.edu/nsnam/ns/
18) Özbay, H.: *Introduction to Feedback Control Theory*, CRC Press LCC, Boca Raton, FL (1999).
19) Quet, P.F., Chellappan, S., Durresi, A., Sridharan, M., Özbay, H. and Jain, R.: Guidelines for Optimizing Multi-level ECN Using Fluid Flow Based TCP Model, *Proc. ITCOM-2002*, Quality of Service over Next Generation Internet, Boston, MA, pp.106–116 (2002).
20) Hollot, C., Misra, V., Towsley, D. and Gong, W.B.: A Control Theoretic Analysis of RED, *Proc. INFOCOM-2001*, San Francisco, CA, pp.1510–1519 (2001).
21) Leland, W., Taqqu, M., Willinger, W. and Wilson, D.: On the Self-similar Nature of Ethernet Traffic, *IEEE/ACM Transactions on Networking*, Vol.2, Issue 1, pp.1–15 (Aug.1999).
22) Athuraliya, S.: Available at http://netlab.caltech.edu/pub/papers/REMparameter.pdf (Mar. 2002).
23) Floyd, S., Gummadi, R. and Shenker, S.: Adaptive RED, An Algorithm for Increasing the Robustness of RED, Technical Report, ATT Center for Internet Research at ICIR (2001).
24) Misra, V., Gong, W.B. and Towley, D.: Fluid-

based Analysis of a Network of AQM Router Supporting TCP Flows with an Application to RED, *Proc. ACM/SIGCOMM-2000*, pp.151–160 (2000).

(Received May 18, 2005)
(Accepted November 1, 2005)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.94–107.)

**Arjan Durresi** received the BSEE, M.S. and Ph.D. (all summa cum laude) all in Electronic and Telecommunications, in 1986, 1991 and 1993, respectively and a Diploma of Superior Specialization in Telecommunications from La Sapienza University in Rome, Italy and Italian Telecommunications Institute in 1991. He is currently an Assistant Professor in the Department of Computer Science, Louisiana State University. His current research interests include network architectures, heterogeneous wireless networks, security, QoS routing protocols, traffic management, optical and satellite networks, and bioinformatics. Dr. Durresi has authored more than 100 papers in refereed Journals and International Conference proceedings. He is an area editor for the Ad Hoc Networks Journal. He is Program Chair of IEEE AINA-2006. Dr. Durresi is the recipient of the Lumley Research Award from The Ohio State University in 2002. He received the appreciation certificate from IEEE Computer Society in 2005. He is a senior member of the IEEE.

**Leonard Barolli** received B.E. and Ph.D. degrees from Tirana University and Yamagata University in 1989 and 1997, respectively. Presently, he is a Professor at the Department of Information and Communication Engineering, Fukuoka Institute of Technology (FIT). Dr. Barolli has published more than 100 papers in Journals and International Conference proceedings. He was an Editor of the IPSJ Journal and has served as an Guest Editor for many Special Issues of International Journals. Dr. Barolli has been a PC Member of many International Conferences. He was a PC Co-Chair of AINA-2003, PC Chair of AINA-2004 and PC Chair of ICPADS-2005. He is a General Co-Chair of AINA-2006. His research interests include network traffic control, fuzzy control, genetic algorithms, agent-based systems, ad-hoc networks, sensor networks and P2P systems. He is a member of SOFT, IPSJ, IEEE Computer Society and IEEE.

**Mukundan Sridharan** is currently a research assistant for the Dependable Distributed and Networked Systems Lab and is pursuing his Ph.D. degree in The Ohio State University. He received his B.E. and M.S. degrees from University of Madras and The Ohio State University in 2000 and 2003, respectively. His research interest includes wireless and sensor networks, Internet measurements and congestion control.

**Sriram Chellappan** is a graduate student in the Department of Computer Science and Engineering at The Ohio-State University. His current research interests are in network security, congestion control and wireless networks. He received his B.E. from University of Madras and M.S. from The Ohio State University in 2003.

**Raj Jain** is a Professor of Computer Science and Engineering at Washington University in St. Louis. He is also a Co-founder and Chief Technology Officer of Nayna Networks, Inc. He was a Professor of Computer and Information Sciences at The Ohio State University in Columbus until August 2002 and then an Adjunct Professor until August 2004. He is a Fellow of IEEE, a Fellow of ACM and is on the Editorial Boards of Computer Communications, Journal of High Speed Networks, Mobile Networks and Nomadic Applications, International Journal of Virtual Technology and Multimedia, and International Journal of Wireless and Optical Communications. He has 14 patents, more than 40 journal and magazine papers, and more than 60 conference papers. His papers have been widely referenced and he is known for his research on congestion control and avoidance, traffic modeling, performance analysis, and error analysis.