Active Queue Management Exploiting the Rate Information in TCP-IP Networks

Abderrahmane Boudi, *Member, IEEE,* and Malik Loudini

***Abstract*—In this paper, we propose a new mechanism called explicit rate notification (ERN) to be used in end-to-end communications. The ERN scheme encodes in the header of transmission control protocol (TCP) packets information about the sending rate and the round trip time (RTT) of the flows. This new available information to the intermediate nodes (routers) is used to improve fairness, increase utilization, decrease the number of drops, and minimize queueing delays. Thus, it induces a better management of the queue. A comparison of our scheme with preexistent schemes, like the explicit congestion notification scheme, shows the effectiveness of the proposed mechanism.**

***Index Terms*—Active queue management, congestion control, fairness, transmission control protocol (TCP).**

I. Introduction

In a network, routers transmit incoming packets over links with finite bandwidth. Links become congested when the amount of incoming packets exceeds the capacity of outgoing links. Congestion results typically in longer delays for data delivery, and in extreme cases, it can lead to a congestion collapse, where the load exerted on the network is extremely high and the useful throughput is low. Thus, congestion control is essential to the functioning of the Internet. That is why it gathered a great amount of attention in the past three decades.

Congestion control can be seen as a distributed algorithm that shares the resources of a network among competing users. An important aspect of this system is that the network condition changes rapidly and unpredictably over time. For instance, the number of users changes greatly during the day or from one day to the next. Controlling the congestion can be done by the end hosts and/or by the routers (links). While the former type of control is achieved by mechanisms like transmission control protocol (TCP) [1], the latter type of control is achieved by active queue management (AQM) mechanisms. The philosophy behind AQM is to start dropping/marking packets at the early stages of the congestion. Doing this would convey the congestion information to the end hosts, which in turn adjust their congestion windows (CWNDs) according to the congestion signal. AQM helps to achieve smaller queueing delays, higher throughput, and avoid flow synchronization by purposely dropping packets before congestion becomes severe, which differs from previous passive queueing management mechanisms like drop-tail.

The most well-known and prominent AQM scheme is random early detection (RED) [2]. Until recently, the internet engineering task force (IETF) recommended in [3] the use of RED in internet routers. However, even if the newer request for comments (RFC) [4] still recommends the use of AQM schemes in routers, it obsoleted the recommendation that states that RED should be used by default in routers. The RED scheme was developed and introduced into Internet routers mainly to avoid the flow synchronization problem. It also helps to distinguish between a state of congestion and the bursty nature of the network. The RED algorithm manages the queue by maintaining a drop probability that is calculated according to the queue length. Nevertheless, the complexity and the difficulty in tuning its parameters have been the subject of several studies. To address the limitations and improve the performance of RED, several variants of RED have been proposed such as [5]−[10]. While RED uses the queue length as the congestion metric, other works argued that this information alone is not sufficient to achieve a good management of the queue. By categorizing the mechanisms with respect to their congestion metrics, it would result into four main categories: 1) The mechanisms belonging to the first category use the queue length as their main congestion metric, whether it is the instantaneous queue length or the filtered one as was used in RED and its variants. 2) The second category of mechanisms uses the queue length along with its variation. The mechanisms that used this metric are mainly the ones that were developed using a control theoretic approach [11]−[14]. 3) The third set of mechanisms uses the difference between the aggregate of incoming rate and the aggregate of the outgoing rate [15], [16]. 4) Finally, the mechanisms of the last category use the sojourn time [17]. Actually, using the sojourn time instead of the queue length would ensure a robust control even when the link's capacity varies.

In the internet, the number of active flows that traverse a single router is great. It could reach thousands and even more in core-routers. All these flows do not have any information about the actual state of the network, and this is more true in TCP-IP networks. Therefore, when a source wants to send data, it cannot know which sending rate will ensure a high utilization without overloading the network. That is why it gradually increases its rate by increasing its congestion window. And when it receives a congestion notification, it reduces its congestion window, and thus its rate. According to [18], [19] the TCP's self-clocking mechanism generates a traffic pattern where the packets are sent in bursts, which is

Manuscript received January 9, 2016; accepted January 16, 2017. Recommended by Associate Editor Xiaoou Li. *(Corresponding author: Abderrahmane Boudi.)*

Citation: A. Boudi and M. Loudini, "Active queue management exploiting the rate information in TCP-IP networks," *IEEE/CAA J. of Autom. Sinica*, vol. 5, no. 1, pp. 223−231, Jan. 2018.

A. Boudi and M. Loudini are with the Laboratoire de la Communication dans les Systèmes Informatiques, École nationale Supérieure d'Informatique, ESI, Algiers 16309, Algeria (e-mail: a_boudi@esi.dz; m_loudini@esi.dz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JAS.2017.7510763

the main cause of burstiness in the networks. Another source of burstiness is long term TCP connections that send data in short bursts, one example of such applications is the one that requires user interaction. The bursty behavior of the network is the reason why the queueing policy of the router must allow the bursts and notify the flows only if the load persists. Doing so, would help to achieve a high utilization by not discarding packets unnecessarily when the excess of load is due to bursts. It would also prevent high delay by dropping packets before the queue overflows. And above all, the policy should maintain a high level of fairness among flows. Ideally, each flow, in a relatively large time scale, would use its fair share of the bandwidth. One way to do this is by doing some sort of fair queueing where each flow traversing the link has its own queue, and where the access to the link is ensured by using some sort of a scheduling algorithm, like round robin for instance. Nevertheless, this kind of mechanisms, where the router stores the list of all the flows traversing its link, can hardly be used in core-routers where the traffic is huge. This is the reason why the majority of the proposed solutions do not maintain any per flow state.

Given the distributed nature and the size of the Internet, the scale-up property of any proposed solution is considered as one of the most important properties. That is why the majority of the proposed AQM schemes rely only on stateless information that can be gathered in the router. This means that, unlike fair random early drop (FRED) [20], a router should not store the list of the flows traversing its link; on the other hand, relying only on queue length and its variation in time is not sufficient to ensure a good control of the queue length [21], and even less to achieve a high fairness among flows [22]. For instance, it is well known that the network could be considered as a delayed control system, with the round trip time (RTT) being the delay. The network model proposed by Hollot in [23], [24] was modeled using the RTT, the number of flows, and the link capacity. However, as mentioned previously, the condition of the network is not stable and it changes over time. Thus, mechanisms that were built from these models were tuned to operate only under a predefined range of operating points. To the best knowledge of the authors, only a few studies [25], [19] took RTTs of the flows into account when implementing their AQM schemes. In [19], the authors used a passive method to calculate the RTT, but they calculated the RTT of a single flow only. They assumed, then, that all the other flows have the same RTT which is rarely, if ever, the case.

To address the lack of information in routers, numerous works proposed to involve end-to-end congestion control mechanisms to work along with intermediate nodes. The most popular work is the explicit control protocol (XCP) [26]. The authors of XCP proposed a new transport protocol that appends to each packet the rate of its flow. As these packets travel within the network, the routers update the value of the rate according to the level of congestion perceived by each one of them. In [27] Xia *et al.* argued that XCP requires a non-trivial and time-consuming standardization process, because of the lack of space in the IP header. Instead, they proposed to encode the congestion-related information in the existing two explicit congestion notification (ECN) bits. However, the authors admitted that while their technique approaches the performance of XCP in terms of queue management, it converges slowly to a fair allocation of the bandwidth among the flows.

In this work, we propose a new method for managing the queue during times of congestion. We aim to ensure fairness among TCP flows without storing any kind of per flow state. We propose to involve endpoints in the control by sending information about the characteristics of the flows. Therefore, the router will know the number and the rates of the flows traversing its link. This proposition differs from other works by two main points. Firstly, it is not a new protocol of the transport layer, such as XCP [26] and Variable-structure congestion control protocol (VCP) [27]. Instead, it can be seen as a new component that sends information about the rate of the flow. This is encouraged by the fact that this mechanism can be used with existing and already proven transport layer protocols. The second point where this solution differs from the existing ones is that instead of appending the rate information to each packet, as was done in [28], the information is sent only once per RTT. This is encouraged by the fact that a source needs not less than one RTT to adjust its sending rate. Thus, the behavior of the flow is predictable within one RTT, so there is no need to send the flow's characteristics with each packet.

The remainder of this paper is organized as follows. In Section II, we discuss how the new available information to the routers would allow a better management of the queues. In Section III, we present the explicit rate notification (ERN) scheme and how it enables the flow's source to share its state with routers, then we give the design guidelines of a new AQM scheme that takes advantage of the information proposed by the ERN scheme. Section IV discusses the simulation results, and provides a comparison between the RED scheme when enabling or disabling ECN and the proposed ERN scheme. In Section V, we will present some deployment and implementation considerations. Finally, we draw our conclusions and future works in Section VI.

II. The Need for More Information Within the Network

In this paper, we aim to improve fairness among flows by making the endpoints actively contribute in the management of the queue. Thus far, the endpoints were passively contributing by responding to the congestion notifications. One way to actively contribute is to share some information about the flows' states with the network (routers). Therefore, each flow should send a packet once per RTT that contains its rate. The rate $rate_i$ of any given flow i can easily be calculated using

$$rate_i = \frac{cwnd_i}{rtt_i} \tag{1}$$

where $cwnd_i$ and rtt_i represent the congestion window and the round trip time of the flow i, respectively.

Sending the rate of the flow once per RTT would have three advantages.

1) The first advantage is to enforce fairness by using the RTT when choosing which flows should be notified. For instance, if two TCP flows, with different RTTs, compete against each other over the bandwidth, the flow with the lower RTT would always prevent the other flow from getting its fair share. The reader can find a detailed discussion about the impact of the RTT in the distribution of the bandwidth in [11]. Furthermore, using the inequality

$$T < \frac{1.5 \times \sqrt{\frac{2}{3}} \times MSS}{RTT\sqrt{p}} \tag{2}$$

that was introduced in [29], makes it possible to calculate the fair share T that a flow should expect. A flow is characterized by the maximum segment size MSS, the round trip time RTT and the drop probability p. Thus, using the RTT to differentiate between flows could lead to a better fairness among flows.

2) The second advantage is a better control of the queue length. This is done by taking the number of the flows traversing the link into account when calculating the drop probability. As mentioned in [30], the weakness of some proposed AQM schemes is that the number of flows passing through a router does not affect the rate at which the congestion notifications are sent. In [30], Feng *et al.* showed that in a link filled with N connections, sending a congestion notification to one connection would reduce the load by a factor of $1-1/(2N)$. The number of flows can be calculated by counting the number of packets that contain the rate information.

3) The third and last advantage is to reduce the number of unnecessarily dropped packets by the routers. Already existing schemes maintain a drop probability that is applicable to all packets passing through the router. In this work, the flows are robustly notified by marking, when necessary, only the packets carrying the RTT information.

Given the fact that not all flows are congestion control aware, responsive flows should be distinguished from unresponsive ones. While the packets of responsive flows should be marked, the packets of unresponsive ones should be dropped. A responsive flow is a flow that significantly reduces its rate upon the reception of a congestion notification. For instance, a long lived file transfer protocol connection that continuously sends data is a good example of such type of flows. Whereas, unresponsive flows are flows that do not implement any congestion control algorithm in the transport layer, like User Datagram Protocol (UDP) flows. Another type of unresponsive flows is when there is no decrease of the load exerted on the congested link, when the flows get notified about the inherent congestion. Hypertext Transfer Protocol (HTTP) and Telnet are good examples of such type of flows. The reason why these flows should be treated as unresponsive, lies in the fact that by the time that the congestion notification would reach the sender, the HTTP or the Telnet sender would have already sent all the data he had to send. One way to differentiate between TCP responsive flows from TCP unresponsive ones is to treat all the flows that are in the slow start phase as unresponsive. Finally, it is clear that even under a heavily-congested network, any flow with enough data to send would eventually end up by reaching the congestion avoidance phase.

III. Implementation

In order to implement the proposed AQM scheme, there should be some alteration in the structure of the TCP header and even in the behavior of the TCP protocol. In what follows, we shall present the changes made to the TCP protocol, then the implementation of the AQM scheme.

A. Changes to the TCP

To implement the ERN scheme, two bits are added to the TCP header (Fig. 1). The first bit is called ERN-capable (ERC) and the second bit is called ERN-rate (ENR). With these two bits there are four possible combinations that are set by the sender. The first combination 10 indicates that the packet belongs to an ERN-capable flow. The second one 11 indicates that the options field of the current packet contains the rate information. The packet carrying the rate information is called the ERN-packet. Finally, the last two combinations 00 and 01 are used interchangeably, and they only indicate that the flow at which the current packet belongs to is not ERN-capable, thus routers should manage these flows along with the unresponsive ones. It should be noted that it is possible for a flow to be ERN-capable in one way but not the other.

Data offset	Reserved 0	ERC	ENR	NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN

Fig. 1. TCP flags.

Each ERN-capable flow should send its rate once per RTT. When the flow sends the ERN-packet, it should put the CWND size and the value of the RTT in the TCP options field. In this work, the level of granularity required for the RTT value is in milliseconds. Therefore, since a useful RTT does not go beyond a few seconds, using 12 bits to encode the RTT value should be enough (from 1 ms up to 4 s). The remaining 4 bits are used to indicate if the CWND is extended or not. To be more precise, the 4 bits specify by how much the CWND should be shifted. Finally, the CWND size can be encoded in 16 bits (2 octets). Thus, using 6 octets in the TCP options field is enough to convey all the required information (Fig. 2). The TCP options kind namespace is under the responsibility of internet assigned numbers authority (IANA). Thus, IANA should assign a value to the Kind field to be used for the ERN-TCP option.

Kind=ERN-knd	Length=4	RTT	shift.cnt	CWND
8	8	12	4	16

Fig. 2. TCP options.

B. The AQM Implementation

In order to evaluate the effectiveness of this proposition, we took a well known AQM scheme, which is RED, and modified it to take advantage of the new possibilities offered by the proposed ERN scheme. The first step in such endeavor was to separate flows by their RTTs. Having different drop probabilities for flows with different RTTs would help flows

with large RTTs to have enough time to respond to the congestion notifications before the AQM algorithm becomes more aggressive. Each category would have its own drop probability that is updated once per RTT. The drop probability is calculated, as in RED, by

$$p_b = max_p \times \frac{\overline{avg} - min_{\text{th}}}{max_{\text{th}} - min_{\text{th}}} \tag{3}$$

where p_b is the drop probability; avg is the average queue length; min_{th} and max_{th} are the minimum and the maximum thresholds, respectively; and finally, max_p is the maximum drop probability, it is set to 0.1 in the original RED.

The flows were divided into five different categories. The first category contains all the flows with an RTT less than 40 ms. The maximum RTT value is, then, doubled each time for the next three categories, which means that the categories 2, 3, and 4 contain flows with RTTs less than 80 ms, 160 ms, and 320 ms, respectively. Finally, the last category contains all the flows with an RTT above 320 ms. Choosing five categories is the authors' choice to cover all the universe of discourse of the values of RTTs. Having more than five categories would not result in a significant gain when controlling the queue length, while having less would result in a poor segmentation of the universe of discourse. It should be noted that when doubling the RTT, the rate of the source would also double. That is why the length of the categories is doubled between every two adjacent categories.

There are multiple ways to calculate the fair rate. The most direct way would be to divide the capacity of the link by the number of flows traversing it. But given the fact that the link's capacity varies over time and ERN-capable flows coexist on the same link with non-ERN flows, calculating the fair share by dividing the capacity by the number of ERN-capable flows would result in a discrimination toward non-ERN flows. So, a better way to calculate the fair rate would be done by calculating the average rate of all ERN-capable flows traversing the link. To calculate this average, we used an exponentially weighted moving average (EWMA) filter with a decaying factor of 0.98. Thus, when a router receives an ERN-packet, it increments the number of active connections of the category at which the flow belongs to, then it updates the fair rate using

$$\overline{rate} = \overline{rate} \times 0.98 + \frac{cwnd_i}{rtt_i} \times 0.02. \tag{4}$$

The classic RED mechanism has a module used to space between two consecutive drops characterized by

$$p_a = \frac{p_b}{1 - count \times p_b} \tag{5}$$

where p_a is the drop probability applied to each packet and $count$ is the number of packets that traversed the router since the last drop. Spacing between drops is done by increasing the drop probability according to the number of packets that traversed the router since the last drop. This module was introduced to improve the fairness among flows by addressing the problem of the bursty nature of TCP flows. In this work, the module described in (5) was replaced by a new one characterized by

$$p_a = p_b \times \frac{rate_i}{\overline{rate}}. \tag{6}$$

This new module uses the difference between the actual rate ($rate_i$) of the flow i, and the fair rate ($\overline{rate}$) of all the flows calculated previously in (4). Equation (6) shows clearly that the drop probability is greater for the flows with sending rates above the fair share compared to the flows with sending rates below the fair share.

Finally, all the flows that are not ERN-capable are notified using the global drop probability. This global drop probability is updated and maintained as in RED. In other words, when the proposed mechanism deals with non-ERN flows it reverts back to the original functioning of the RED algorithm. It should be noted that all the flows traversing the link share the same queue, only the drop probability that allows access to the queue differs.

IV. Evaluation

In this section, we demonstrate, through simulation, the effectiveness of the proposed solution (RED-ERN). The objective in the simulation process is to evaluate the impact of using the ERN scheme instead of the ECN or the DROP schemes. To reach our objective, a comparison is made between RED-ERN, RED-ECN and RED-DROP, where RED-ERN is the proposed mechanism, RED-ECN and RED-DROP represent the RED algorithm coupled with either the ECN or the DROP schemes, respectively. The performance of these AQM schemes is evaluated using the OMNeT++ simulator. In all simulations, a simple dumbbell topology is used with a single link as the bottleneck (Fig. 3). The number of TCP flows varies from 50 to 800 flows. These TCP flows consist of long lived FTP connections with an infinite amount of data to send. The TCP flavor used is NewReno [31], and the maximum transmission unit (MTU) is set to 576 Bytes (MSS of 536 Bytes) The bottleneck bandwidth varies from 5 Mbps to 50 Mbps, and its delay is set to 8 ms. The delays of the other links are set, by following a truncated normal distribution, so the RTTs of the flows would be evenly divided between the five categories. For instance, the delays of the links belonging to the first category were calculated with a mean of 4 ms and a standard deviation of 2 ms. It means that the delay of these links vary between 2 ms and 6 ms; by adding the bottleneck delay of 8 ms, the one way delay (propagation delay) would range between 12 ms and 20 ms. Therefore, the RTTs of the flows belonging to the first category would range between

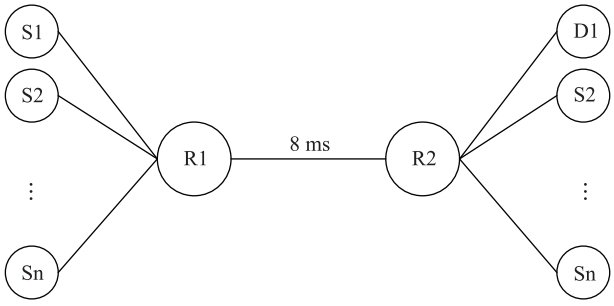


Fig. 3. Simulation topology.

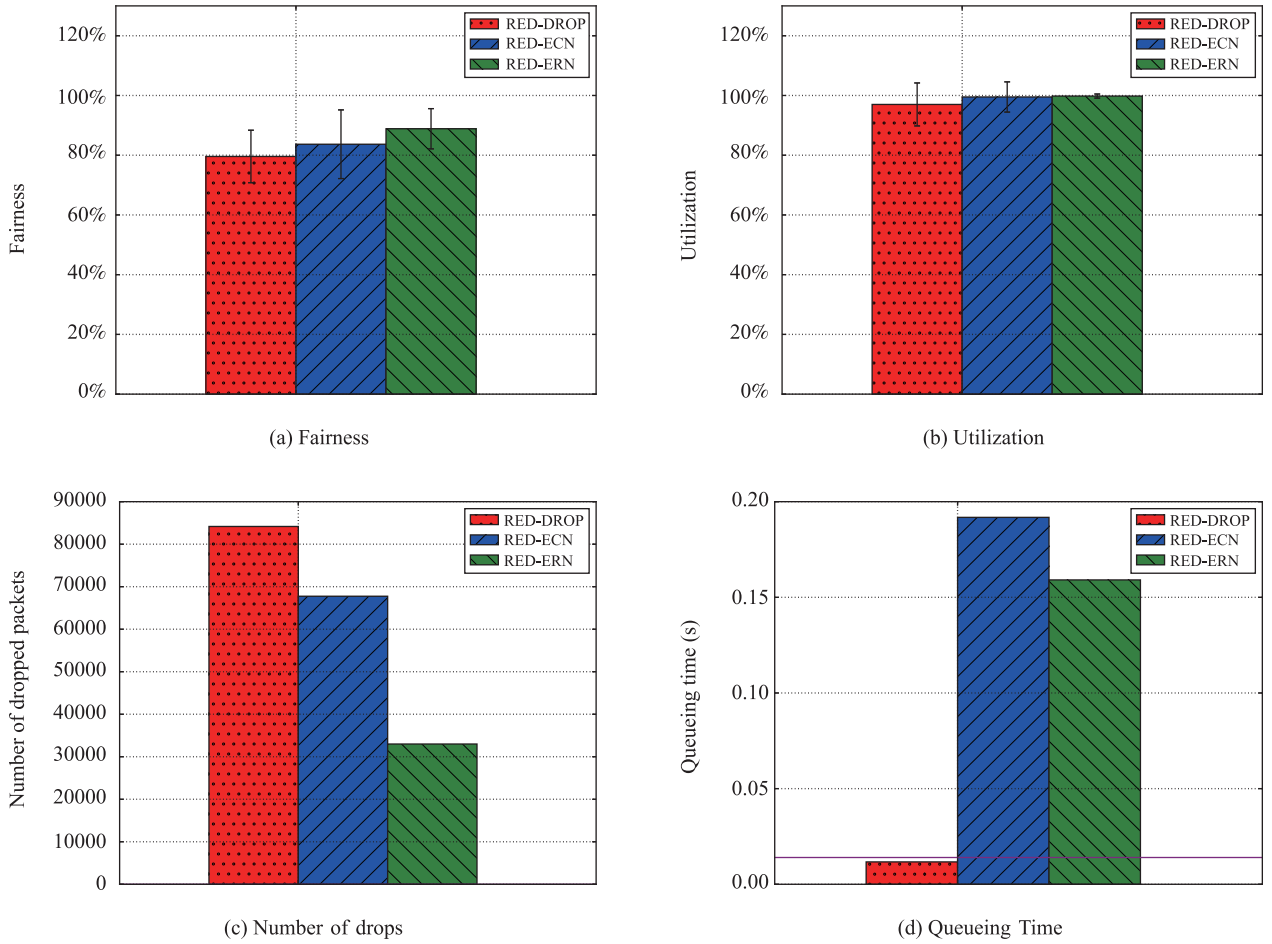


Fig. 4. General results.

24 ms and 40 ms. For the four remaining categories the mean and the standard deviation are (11 ms, 4 ms), (26 ms, 8 ms), (56 ms, 18 ms), and (101 ms, 22 ms). The buffer size is set to 800 packets, and the queue target is set so the queueing delay would be 14 ms. For instance, with a bottleneck of 30 Mbps, the queue target would be 91 packets. In order to evaluate the fairness of each scheme, the Jain's fairness index is used. It is calculated with respect to the throughput experienced by each flow. This index was introduced in [32] and it is characterized by

$$J = \frac{(\sum_{i=1}^{n} x_i)^2}{n \sum_{i=1}^{n} x_i^2} \tag{7}$$

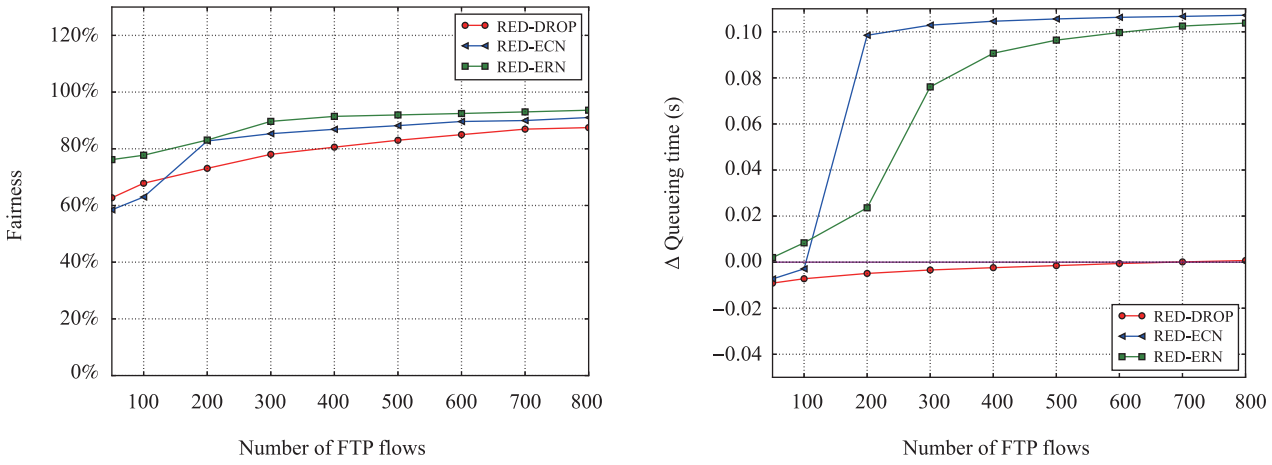
where J is the fairness index that falls in the unit interval [0,1], n is the number of flows sharing the link, and x_i is the throughput of flow i.

A. General Results

Fig. 4 shows the average and the standard deviation for all of the fairness, the utilization, the number of drops, and the queueing time when the bottleneck bandwidth and the number of FTP flows vary. The average is calculated from the results of all the simulations when varying the bottleneck bandwidth from 5 Mbps to 50 Mbps and the number of FTP flows from 50 to 800. As shown in Fig. 4(a), in average, the fairness of the RED-ERN mechanism is above the two other mechanisms. It also shows a gain of about 6 % over RED-ECN and 10 % over RED-DROP. The utilization of all of the three mechanisms is nearly optimal as shown in Fig. 4(b). As Fig. 4(c) shows, the ERN mechanism reduces considerably the number of dropped packets. It reduced that number by a factor of 2.5 compared to RED-DROP, and by a factor of 2 compared to RED-ECN. Concerning the queueing time, as depicted in Fig. 4(d), there is a great gap between RED-DROP and the two other mechanisms. This is due to the fact that in times of congestion, the DROP scheme drops the packets while the two other schemes can only mark them. Thus, the queue control of the ECN and ERN schemes is greatly impacted when the multiplexity over the link is high. Henceforth, we define the multiplexity of a link as being the number of flows traversing the link; a high multiplexity means that the number of flows is great, while a low multiplexity means the opposite.

B. Varying the Number of FTP Connections

In this set of simulations, the bottleneck bandwidth is set to 30 Mbps and the number of FTP flows varies from 50 to 800. As shown in Fig. 5(a), the fairness of the RED-ERN mechanism is located in the vicinity of 85 %. As a matter of fact, the gap in fairness between RED-ERN and the other two mechanisms can reach 20 % when the multiplexity over the link is low.



(a) Fairness

(b) Δ Queueing time

Fig. 5. Fairness and queueing time with respect to number of FTP sessions.

But as the multiplexity increases, the gap between the schemes decreases to become between 4 % and 8 %. This is due to the fact that the DROP and the ECN schemes are enhancing fairness by spacing between drops. Therefore, in a low multiplexity environment, the probability of dropping/marking packets belonging to the same flow is high. While for the ERN scheme, irrespective of the number of flows sharing the link, all the flows that are sending above the fair rate will be notified more frequently than the flows sending below the fair rate.

Fig. 5(b) shows the subtraction of the target queueing time 14 ms from the queueing times of the three schemes. It is clear from Fig. 5(b) that the queueing time increases with the increase of the number of FTP sources. Only RED-DROP is able to maintain the queueing time to the desired target in very high multiplexed configurations, while the ECN and ERN schemes fail to drive the queue to the desired target, so the queue overflows. As stated previously, this is due to the way the different schemes react to congestion; where the DROP scheme drops packets when the congestion occurs, the other two schemes mark them, which is not enough when the number of FTP flows is great. But when the number of FTP flows is high and the bottleneck bandwidth is low, the fairness of the DROP scheme falls, and up to half of the flows are excluded from the link. For instance, running a simulation with 800 FTP sessions sharing a 1 Mbps link would result in a very low fairness for the DROP scheme, which is in the vicinity of 26 %, while the fairness of the ECN and the ERN schemes exceeds 75 %.

In configurations where the multiplexity over the link is low to moderate, the ERN scheme shows very promising results. And by comparing RED-ERN with RED-ECN, it is clear that the former is more robust, and it could control the queueing time even when the multiplexity level is moderate; whereas the latter lost the control of the queue even in low multiplexity levels.

C. Varying the Bottleneck Bandwidth

In what follows, the performance of the different schemes is studied when the bandwidth of the bottleneck varies. The number of the flows sending data simultaneously is fixed to 200 FTP sessions. The bottleneck bandwidth varies from 5 Mbps to 50 Mbps. In Fig. 6(a), it can be seen that in low bandwidth configurations all of the three schemes have a high fairness. But as the bandwidth increases, the fairness of RED-ECN and RED-DROP drops to become in the vicinity of 60 %, while the RED-ERN mechanism keeps the fairness above 80 %. Fig. 6(b) depicts, as previously, the queueing time of each scheme minus the target queueing time 14 ms. It could be easily noticed that the queuing time of the ERN and the ECN schemes decreases with the increase of the bandwidth, while the DROP scheme manages to control the queueing time irrespective of the bandwidth of the bottleneck. As it can be seen in Fig. 6(b), in high bandwidth configurations the queueing times are small, which means that the queue length is also small. Having an average queue length in the vicinity of min_{th} means that the RED algorithm is triggered repeatedly but only for a short period of time. Therefore, whenever the average queue length becomes below min_{th}, the module responsible for spacing between drops is reinitialized. Thus, given the fact that RED-ECN and RED-DROP rely on spacing between drops to improve the fairness, then it becomes clear why these two mechanisms have not been able to keep a high fairness in high bandwidth configurations. Whereas the RED-ERN mechanism has kept the fairness above 80 % irrespective of the bottleneck bandwidth.

D. UDP Flows

In order to investigate the behavior of the ERN scheme in the presence of UDP flows, we ran the following simulations. The bottleneck bandwidth is fixed to 25 Mbps. There are two types of flows, FTP flows with an infinite amount of data to send representing the TCP flows, and UDP flows with a constant bit rate (CBR) application that represent the unresponsive flows. The number of TCP flows is fixed to 50, and the number of UDP flows varies from 200 to 800 with a throughput of 64 Kbps per flow. Fig. 7(a) shows that the ERN scheme manages to keep a high, fairness among FTP flows

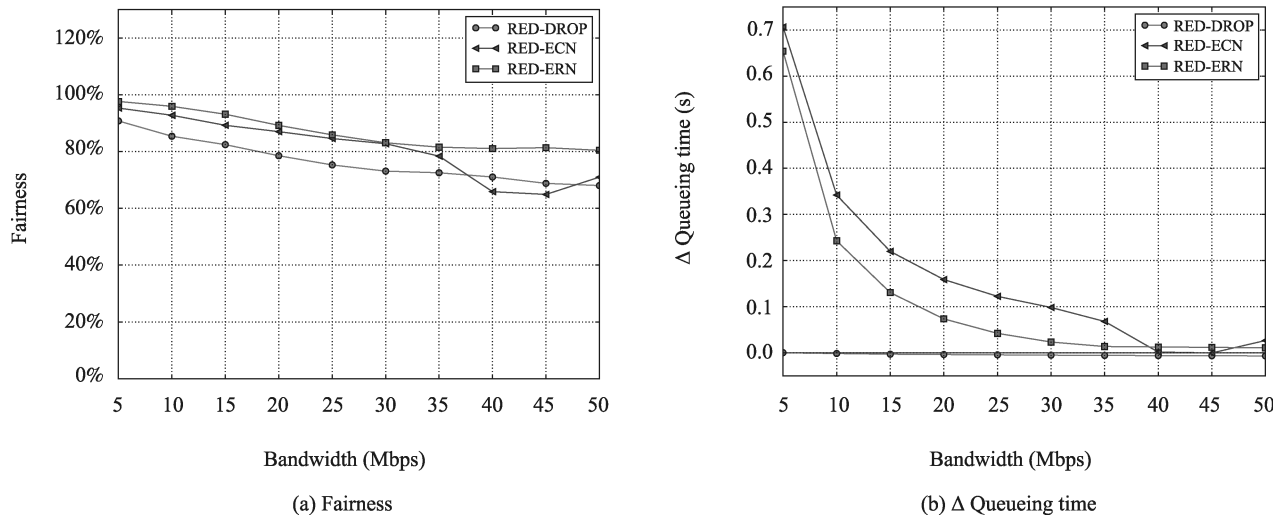


Fig. 6. Fairness and queueing time with respect to link bandwidth.

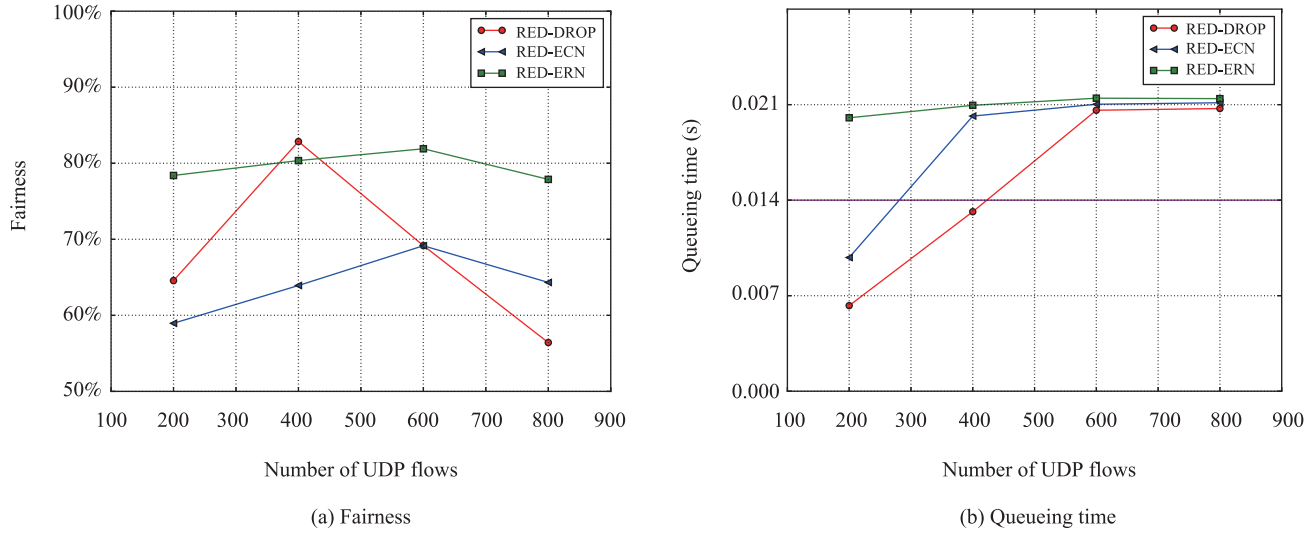
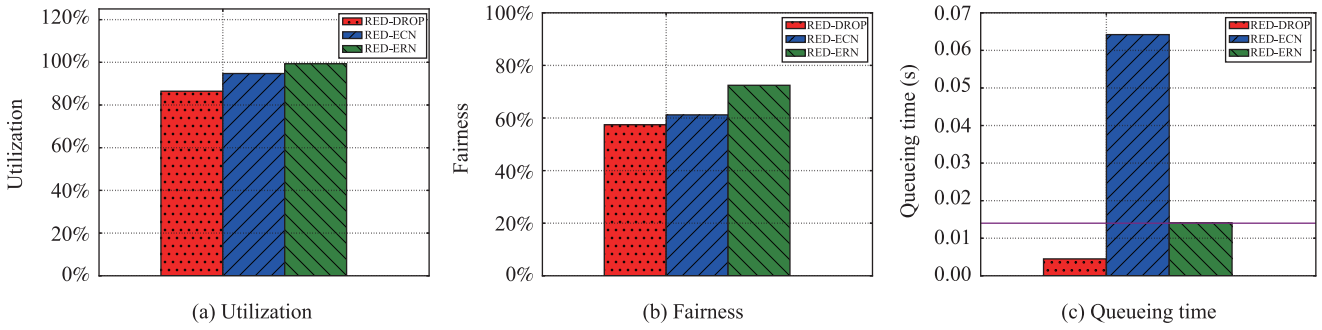


Fig. 7. Effect of UDP flows on fairness and queueing time.

irrespective of the number of UDP flows. On a side note, the simulations results showed that the fairness among UDP flows is ranging between 96 % and 99 % for all of the three schemes. Regarding the queueing time, Fig. 7(b) shows that all of the three mechanisms keep low queueing times, and they are slightly impacted by the number of the UDP flows. The reason why the queueing times of the ERN and ECN schemes are not impacted is due to fact that these two schemes drop UDP packets instead of marking them. Which means that regardless of the number of UDP flows, all of the three schemes behave in the same way when dealing with UDP flows.

E. HTTP Flows

In what follows, we study the behavior of the three mechanisms in the presence of HTTP flows. The bottleneck bandwidth is set to 25 Mbps. There are 20 long lived TCP flows continuously sending data. We ran this simulation four times where the number of HTTP flows was set to 200, 400, 600, and 800 HTTP sessions (from 10 to 40 times the number of FTP flows). The HTTP flows have a size of approximately 30 packets per connection and an idle interval of 5 s between two consecutive connections. The results in Fig. 8 show that the proposed scheme ensures both high utilization and fairness compared to RED-DROP and RED-ECN. The gap in utilization reaches 13 % compared to RED-DROP and 5 % compared to RED-ECN Fig. 8(a). While the gain in fairness of the proposed mechanism is about 10 % over the two other mechanisms Fig. 8(b). As depicted in Fig. 8(c), RED-DROP and RED-ERN manage to keep low queueing delays while the RED-ECN does not. The reason of this behavior lies in the fact that the ERN scheme, unlike ECN, can distinguish between long FTP flows and short HTTP flows. Thus, while RED-ECN marks HTTP packets, RED-ERN drops them.



(a) Utilization (b) Fairness (c) Queueing time

Fig. 8. Simulation results when HTTP flows coexist with FTP flows.

V. Deployment Considerations

Even if throughout this paper the mean to communicate with intermediate nodes was done by altering the TCP header, it is not an imperative. There are other ways to achieve the same goal with less intrusive techniques than altering the structure of the TCP header.

One way to do this requires the use of the two bits already existing in the IP header that are used for congestion notifications (ECT and CE). The ECN algorithm uses two code points to indicate that a flow is ECN capable; the first code point is 01 called ECT(1), and the second one is 10 called ECT(0). Sources use these two ECT code points alternatively to know if the destination node is responding to congestion notifications or not. Our idea is to use the ECT(1) code point to differentiate between ERN capable flows and other flows. Therefore, the ERN mechanism should set ECT(1) on all the packets belonging to an ERN capable flow. However, the only way to differentiate ERN-packets (packets containing the rate) from the other packets would be to check the options in the packet's header searching for the rate. It should be noted that only an ECT(1) packet or a CE packet can become an ERN-packet. Thus, the router should only check these packets when searching for the rate in the packet's header. The downside of using ECT(1) to mark ERN flows lies in the fact that there could be ECN capable flows that use only ECT(1). Therefore, these flows will be mistakenly viewed as ERN capable and their packets will not be marked nor dropped. A solution to this problem would be the use of some monitoring mechanism to monitor the link, but this will have a negative impact on the scale-up property.

Another way to make available the same set of information to the routers, and this time without altering the endpoints, would be the use of passive estimation techniques [33], [18], [34], [35]. But solutions like these would be in contradiction with the stateless claim of the original ERN, and thus the scale-up property.

VI. Conclusion

Active Queue Management mechanisms are important to solve the congestion in the Internet. Existing techniques focus only on controlling the queue length without taking into account the flows composing the traffic. In this paper, we proposed a new AQM mechanism that uses new information that generally does not exist in routers. We enabled the endpoints to share their flow information with routers. The endpoints send the size of their congestion windows along with their RTTs. These two pieces of information should help the routers to enhance the fairness among flows. In order to demonstrate the effectiveness of our proposition, a simulation study was carried out. As a matter of fact, the simulation results did show that the ERN scheme had a better fairness among flows, a better queue management, and a better utilization of the link, especially compared to the ECN scheme. In future works, we aim to propose a new AQM mechanism that is purposely built to support the use of the new information available in the ERN scheme. As we think that the RED-ERN mechanism does not take full advantage of the new features and possibilities induced by the ERN scheme.

References

[1] V. Jacobson, "Congestion avoidance and control," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314−329, Aug. 1988.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," *IEEE/ACM Trans. Network.*, vol. 1, no. 4, pp. 397−413, Aug. 1993.

[3] B. Braden, D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, "Recommendations on Queue Management and Congestion Avoidance in the Internet," RFC 2309, 1998.

[4] F. Baker and G. Fairhurst, "IETF Recommendations Regarding Active Queue Management," RFC 7567, 2015.

[5] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Comput. Netw.*, vol. 36, no. 2−3, pp. 203−235, Jul. 2001.

[6] J. S. Sun, K. T. Ko, G. R. Chen, S. Chan, and M. Zukerman, "PD-RED: to improve the performance of RED," *IEEE Commun. Lett.*, vol. 7, no. 8, pp. 406−408, Sep. 203.

[7] N. X. Xiong, A. V. Vasilakos, L. T. Yang, C. X. Wang, R. Kannan, C. C. Chang, and Y. Pan, "A novel self-tuning feedback controller for active queue management supporting TCP flows," *Inf. Sci.*, vol. 180, no. 11, pp. 2249−2263, Jun. 2010.

[8] X. L. Jiang, J. G. Yang, G. Jin, and W. Wei, "RED-FT: a scalable random early detection scheme with flow trust against dos attacks," *IEEE Commun. Lett.*, vol. 17, no. 5, pp. 1032−1035, May 2013.

[9] A. H. Ismail, A. Elsayed, Z. Elsaghir, and I. Z. Morsi, "Enhanced random early detection (ENRED)," *Int. J. Comput. Appl.*, vol. 92, no. 9, pp. 25−28, Apr. 2014.

[10] J. Domanźka, A. Domański, D. R. Augustyn, and J. Klamka, "A RED modified weighted moving average for soft real-time application," *Int. J. Appl. Math. Comput. Sci.*, vol. 24, no. 3, pp. 697−707, Aug. 2014.

[11] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "On designing improved controllers for AQM routers supporting TCP flows," in *Proc. IEEE INFOCOM 2001. Twentieth Ann. Joint Conf. IEEE Computer and Communications Society*, Anchorage, AK, USA, USA, 2001, pp. 1726−1734.

[12] F. Y. Ren, Y. Ren, X. M. Shan, "Design of a fuzzy controller for active queue management," *Comput. Commun.*, vol. 25, no. 9, pp. 874−883, Jun. 2002.

[13] Y. Fan, F. Y. Ren, and C. Lin, "Design a PID controller for active queue management," in *Proc. Eighth IEEE Symp. Computers and Communications (ISCC 2003)*, Kemer-Antalya, Turkey, Turkey, 2003, pp. 3−8.

[14] C. Chrysostomou, A. Pitsillides, and Y. A. Sekercioglu, "Fuzzy explicit marking: a unified congestion controller for Best-Effort and Diff-Serv networks," *Comput. Netw.*, vol. 53, no. 5, pp. 650−667, Apr. 2009.

[15] W. C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "BLUE: A new class of active queue management algorithms," Univ. Michigan, Tech. Rep. UM CSE-TR-387−99, 1999.

[16] S. Athuraliya, S. H. Low, V. H. Li, and Q. H. Yin, "REM: active queue management," *IEEE Netw.*, vol. 15, no. 3, pp. 48−53, May 2001.

[17] K. Nichols and V. Jacobson, "Controlling queue delay," *Commun. ACM*, vol. 55, no. 7, pp. 42−50, Jul. 2012.

[18] D. Carra, K. Avrachenkov, S. Alouf, A. Blanc, P. Nain, G. Post, "Passive online RTT estimation for flow-aware routers using one-way traffic," in *Proc. 9th IFIP TC 6 Int. Conf. Networking*, Chennai, India, 2010, pp. 109−121.

[19] H. Hoshihara, H. Koga, and T. Watanabe, "A new stable AQM algorithm exploiting RTT estimation," in *Proc. 2006 31st IEEE Conf. Local Computer Networks*, Tampa, FL, USA, 2006, pp. 143−150.

[20] D. Lin and R. Morris, "Dynamics of random early detection," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 127−137, Oct. 1997.

[21] R. Adams, "Active queue management: a survey," *IEEE Commun. Surveys Tutor.*, vol. 15, no. 3, pp. 1425−1476, 2013.

[22] G. Abbas, Z. Halim, and Z. H. Abbas, "Fairness-driven queue management: a survey and taxonomy," *IEEE Commun. Surveys Tutor.*, vol. 18, no. 1, pp. 324−367, 2016.

[23] V. Misra, W. -B. Gong, and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 30, no. 4, pp. 151−160, Oct. 2000.

[24] C. V. Hollot, V. Misra, D. Towsley, and W. B. Gong, "A control theoretic analysis of RED," in *Proc. IEEE INFOCOM 2001. Twentieth Ann. Joint Conf. IEEE Computer and Communications Society*, Anchorage, AK, USA, 2001, pp. 1510−1519.

[25] M.-L. Shyu, S.-C. Chen, and C. Ranasingha, "Router active queue management for both multimedia and best-effort traffic flows," in *Proc. 2004 IEEE Int. Conf. Multimedia and Expo*, Taipei, Taiwan, China, 2004, pp. 451−454.

[26] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 32, no. 4, pp. 89−102, Oct. 2002.

[27] Y. Xia, L. Subramanian, I. Stoica, and S. Kalyanaraman, "One more bit is enough," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 35, no. 4, pp. 37−48, Oct. 2005.

[28] A. Almeida and C. Belo, "Explicit congestion control based on 1-bit probabilistic marking, *Comput. Commun.*, vol. 33, pp. S30−S40, Nov. 2010.

[29] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the Internet," *IEEE/ACM Trans. Network.*, vol. 7, no. 4, pp. 458−472, Aug. 1999.

[30] W. C. Feng, D. D. Kandlur, D. Saha, and K. G. Shin, "Techniques for Eliminating Packet Loss in Congested TCP/IP Networks," Univ. Michigan, Tech. Rep. UM CSE-TR-349−97, 1997.

[31] T. Henderson, S. Floyd, A. Gurtov, and Y. Nishida, "The NewReno Modification to TCPs Fast Recovery Algorithm," RFC 6582, 2012.

[32] R. Jain, D.-M. Chiu, and W. R. Hawe, "A Quantitative Measure of Fairness and Discrimination for Resource Allocation in Shared Computer System," Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, DEC-TR-301, 1984.

[33] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Inferring TCP connection characteristics through passive measurements," in *Proc. IEEE INFOCOM. Twenty-third Ann. Joint Conf. IEEE Computer and Communications Societies*, Hong Kong, China, 2004, pp. 1582−1592.

[34] A. Moosbrugger and P. Dorfinger, "Passive RTT measurement during connection close," in *Proc. 2010 Int. Conf. Software, Telecommunications and Computer Networks (SoftCOM)*, Split, Dubrovnik, Croatia, 2010, pp. 392−396.

[35] H. Ding and M. Rabinovich, "TCP stretch acknowledgements and timestamps: findings and implications for passive RTT measurement," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 3, pp. 20−27, Jul. 2015.

Abderrahmane Boudi (M'16) was born in Algiers, Algeria, in 1990. He received the engineer degree in computer science from the École nationale Supérieure d'Informatique (ESI), Algeria, in 2013. Since 2014, he is a Ph.D. student in the Laboratoire de Communication dans les Systèmes Informatiques (LCSI) of ESI. His work is devoted to the study of active queue management, congestion control, and intelligent feedback control systems.

Malik Loudini was born in Bouzina, Batna, Algeria in 1966. He is currently a Professor of automatic control at the Computer Systems Engineering Department and a Director of Research at the Laboratoire de Communication dans les Systèmes Informatiques (LCSI) of the École nationale Supérieure d'Informatique (ESI), Algiers, Algeria. He received the Magister (M.Sc.) and the Doctorat d'état (Ph.D.) degrees in automatic control both from the École Nationale Polytechnique of Algiers. His research activities include techniques, methodologies and tools for modelling and intelligent control of engineering and computing systems. In the context of such activities, he took part and conduct research teams in numerous research projects (PNR, CNEPRU), funded by the Algerian Ministry of Higher Education and Scientific Research (MESRS).