A Short-Time Burst degradation Classifier for Real-Time Traffic with Application in MPLS Ingress Nodes

Masoomeh Torabzadeh and Wessam Ajib
Department of Computer Sciences,
University of Quebec at Montreal (UQAM), Montreal, Canada
masoomeh_torabzadeh@yahoo.com; ajib.wessam@uqam.ca

***Abstract*—We propose a novel classifier technique, named short-time burst degradation classifier (SBDC), with the ultimate goal of improving the short-term delay and the packet jitter for real-time traffic. The main idea of the proposed classifier is to provide the ability to manage the queuing delay in order to decrease the impact of short-time scale burstiness of the traffic in a flexible manner, since traditional mechanisms such as leaky bucket cannot have such kind of flexibility to smooth the traffic. Even though the proposed scheme is general and can be used in different points of the network, we propose to implement it in MPLS ingress nodes. To evaluate the performance of the proposed classifier, we introduce a new efficient scheduler, called short-term quality-of-service class based queuing (SQ-CBQ), to be combined with our classifier. The scheduler is based on a new scheduling algorithm named polling deficit round robin (PDRR). Also, after using the combination of the classifier and the scheduler in MPLS ingress nodes, the sort-term QoS provisioning is improved. The performance analysis shows that high quality of service provisioning for the real-time traffic can be achieved by implementing our proposed scheduler and classifier, compared to the traditionally implemented switching schemes.** [1]

***Index Terms*—traffic classification, scheduling, burstiness, Quality of Service**

I. Introduction

One of the most crucial problems in the Internet nowadays is the quality of service (QoS) provisioning. Providing QoS means the ability to provide guaranteed services to different applications. It is often measured in terms of throughput, loss rate, delay, and/or jitter. QoS provisioning is typically based on end-to-end mechanisms (e.g., connection admission control), edge mechanisms (e.g., shaping and policing), core mechanisms (e.g., buffering, queue management, and scheduling), or any combination of the three. Given that the traffic behaviour has gone through remarkable changes the last few years, there is a need for new classification and scheduling techniques that takes into account those changes and offer efficiently QoS provisioning especially for the emerging real time applications.

Packet dynamics in a network can make the prone to occasional or constant congestion, especially at routers connecting networks of widely different bandwidths. Research results have shown that traffic burstiness in short-time scales has impact on QoS mechanisms. Research in this area is usually about buffer management, queuing systems, and scheduling and it typically investigates: the influence of the stochastic features of the input traffic on the performance of some QoS mechanisms [1], the cause and the impact of Internet traffic burstiness in short-time scales [2], Internet traffic congestion control using queue thresholds [3], the possibility of using the degree of burstiness as a network QoS performance metric [4], the impossibility of guaranteeing a stable QoS in the Internet [5], the short-term QoS issue caused by the unexpected nature of traffic [6], and the queuing process in generalized processor sharing (GPS) and packet-based GPS (PGPS) with bursty traffic inputs [7].

In this paper, we propose a novel classifier algorithm and an efficient scheduling scheme. The short-time burst degradation classifier (SBDC) aims at efficiently decreasing the short-time scale bursts, the packet delay, and the packet jitter. We show that using this classifier for smoothing the traffic can notably improve the QoS provisioning for real-time traffic, compared to the widely used schemes that use leaky bucket to smooth the traffic [8-9]. We also introduce an efficient scheduler named short-term QoS class based queuing (SQ-CBQ) to be combined with the SBDC classifier. The proposed classification and scheduling algorithms are general and can be implemented in different points of the network. Anyhow, we show that the combination of the proposed classifier and scheduler leads to an efficient switching algorithm when they are implemented at MPLS ingress nodes.

The rest of this paper is organized as follows. Section 2 provides some background information about the delay and packet jitter issues, the leaky bucket, and the scheduling. We first introduce our proposed scheduler in Section 3, and then our proposed classifier in Section 4. In Section 5, we provide details of the traffic used in our

Manuscript received August 15, 2010; revised November 15, 2010; accepted January 15, 2011

doi:10.4304/jcm.6.3.215-224

performance evaluation and the results of performance evaluation for the proposed switching scheme and some discussions. Finally, in Section 6 we conclude this paper.

II. BACKGROUND

Packet delay, or latency, at each hop is in general divided into three parts: serialization, propagation and switching delays. Serialization delay, also referred to as transmission delay, is the time needed by a device to clock a packet at a given output rate. It depends on the link's bandwidth as well as on the size of the packet being clocked. On the other hand, propagation delay is the time for a bit to get transmitted from the sender to a link's receiver. Finally, switching delay is the time between the reception of the packet and its transmission.

All packets belonging to the same flow don't obviously experience the same delay in the network. The delay seen by each packet can vary based on transient network conditions. If the network is not congested, queues will not build at the routers, and the total packet delay contains only the serialization and the propagation delays. This is the minimum delay the network can offer. Also, the serialization delay may become insignificant compared to the propagation delay on fast link speeds.

If the network is congested, queuing delays will start to influence the end-to-end delays for the different packets and will contribute to the packet delays variation. The variation in packet delays is referred to as packet jitter. The Packet jitter is considered often as an important parameter because it estimates the maximum duration between the different instants of packets reception relatively to the individual packet delay. Depending on the application, a receiver can offset the jitter by adding a receiver buffer that could store packets up to the jitter bound. Playback applications that send a continuous data stream, e.g., voice calls, video conferencing, and distribution, fall into this category.

On the other hand, traffic rate management requires a traffic metering function to measure the traffic. Leaky bucket (token bucket) is a common scheme used to measure and regulate the input traffic. It is used in both policing and shaping algorithms and it reports if a packet is compliant with the configured rate parameters. Depending on whether a packet is conforming, an appropriate action (e.g., transmitting, dropping, delaying …) can be performed. A simple leaky bucket has two key parameters: (i) mean rate or committed information rate (CIR) given in bits per second, where the average traffic rate can not exceed CIR, and (ii) conformed burst size (B_C) given in bytes. The B_C is the amount of traffic allowed to exceed the token bucket on an instantaneous basis. A relevant parameter is the time interval (TI) where TI = B_C / CIR.

In addition, at times of network congestion, a router resource allocation for a specific flow is determined by the router's scheduling discipline. While the scheduler determines which packet is served next, how often the packets belonging to a specific flow are served determines that flow's bandwidth, or its allocated resources. The traditional packet scheduling mechanism on the Internet is the first-in-first-out (FIFO) scheduling, where the packets are served at the same order of their arrival. FIFO is simple and easy to be implemented but it cannot differentiate among the flows. Hence, FIFO cannot allocate specific performance bounds for a flow or prioritize one flow over the others.

Another scheduling scheme, named deficit round robin (DRR) [10], uses stochastic fair queuing to assign different flows to queues. DRR uses round-robin (RR) servicing with a quantum of service assigned to each queue. The difference with traditional RR is that if a queue is not able to send a packet in the previous round because its packet size is too large, the remainder from the previous quantum is added to the quantum for the next round. Thus, deficits are kept track off and the queues that were shortchanged in a round are compensated in the next round.

Another scheduler, called MDRR (Modified DRR) is introduced in [11] and can be implemented in the core routers. MDRR improves DRR by adding one high-priority queue. The MDRR works in two modes: strict-priority one and alternate-priority one. In strict-priority mode only when all high-priority traffic is clear, other queues will be considered. It will lead to guaranteeing a minimum latency for the packets going through the high-priority queue. On the other hand, in alternate-priority mode, a quantum of data is taken from the high-priority queue, and another quantum is taken from one of the other queues, then a quantum of the high-priority queue is taken again.

After introducing some background information in this section about the packet delay, jitter and the scheduling issues, we introduce in the next sections our proposed solutions for the traffic scheduling and classification algorithms.

III. PROPOPOSED SCHEDULER SQ-CBQ

In this section, we present the new scheduler that we call short-term quality-of-service class based queuing (SQ-CBQ). This scheduler is to be combined with our proposed classifier, presented in the next section, and hence SQ-CBQ allows us to evaluate the performance of the classifier. The proposed scheduler can be seen as an extended version of short-term QoS deficit round robin (SQ-DRR) scheduler [6]. Our SQ-CBQ uses our new version of DRR [10] scheduling named polling DRR (PDRR). The traditional DRR tracks the byte deficit d_i (the difference between the number of bytes that ought to be sent and b_i, the number of bytes that have been sent) for each queue i and uses it to regulate the long-term bandwidth assignment to the queues.

We assume that the traffic in the network can be divided into four categories: signaling traffic (ST), real-time traffic (RT) high priority best-effort traffic (HBT), and simple priority best-effort traffic (SBT) [12]. We consider that the amount of ST traffic can be neglected and we focus on three classes of traffic RT, HBT, and SBT. An advantage of our scheduling algorithm is that it also has the benefits of MDRR algorithm. In RT traffic, there are many small size

packets with relatively long inter-arrival times; whereas the HBT and SBT traffic may be too large. When the queue length of the class RT, i.e., q_{RT}, is less than its byte deficit d_{RT}, it will be polled between two classes HBT and SBT. Because we would like to give another chance to the class RT to use its entire fair share in the current round, if it has become backlogged ($q_{RT} > 0$). In this case, it can be served by b_{RT} ($b_{RT} \le d_{RT}$). We notice that when the class RT is polled, its byte deficit d_{RT} will not be increased by the quantum size Q_{RT}. The difference between our proposed PDRR scheduling and the strict or alternative MDRR is that PDRR gives another chance to RT traffic to be served and doesn't allocate to RT traffic more than its fair share. While MDRR is not a fair scheduler as DRR and PDRR, since it may serve high-priority traffic (RT) more than one quantum size per round. Fig. 1 shows a diagram of the PDRR algorithm.

It seems not possible to support the short-term QoS using the fixed quantum Q_i. Therefore we modify the quantum of delay constrained class in the case of violation of the specified delay constraint. Also, the SQ-CBQ utilizes a simple window based measurement module. The difference between our window and the deficit and surplus estimator (DSE) [6] is in the simplicity of our scheduling since we are proposing to employ packet by packet shifting instead of τ seconds shifting.

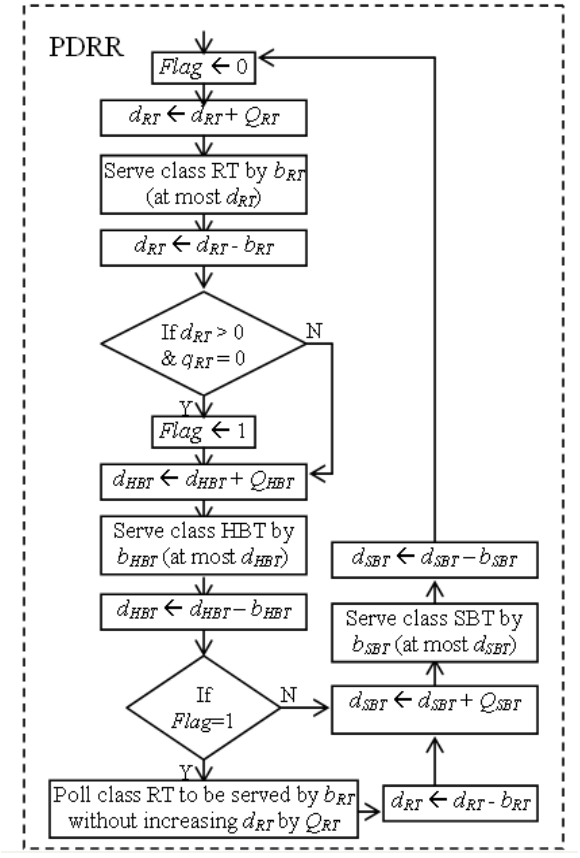

Figure 1. The algorithm of PDRR

The SQ-CBQ has two phases: examination one and quantum adjustment one. Every Δ seconds, SQ-CBQ goes to the examination phase and checks if quantum adjustment is required by calculating the expected delay. If the expected delay is larger than the delay constrained d'_{RT} ; then SQ-CBQ goes to the quantum adjustment phase. In other words, the quantum adjustment phase starts when:

$$\frac{B_{RT}(t)Q}{CQ'_{RT}} \succ d'_{RT} \tag{1}$$

where $B_{RT}(t)$ is the backlog of class RT at time t, Q is the sum of all quanta, C is the link capacity, and Q'_{RT} is quantum allocated in the previous adjustment. In the quantum adjustment phase, we compute the quantum $\hat{Q}_{RT}$ required to satisfy the delay constraint as follows:

$$\hat{Q}_{RT} = \frac{B_{RT}(t)Q}{Cd'_{RT}}. \tag{2}$$

Then SQ-CBQ admits the request if the calculated service by $\hat{Q}_{RT}$ for class RT is smaller than the measured portion of surplus $D_{t,RT}$ over the past period T. But if it would exceed $D_{t,RT}$ over the past period T as:

$$0 \prec D_{t,RT} \prec \frac{T \times C \times (\hat{Q}_{RT} - Q_{RT})}{Q} \tag{3}$$

then $\hat{Q}_{RT}$ is partially accepted so that the new quantum Q'_{RT} is updated as follows:

$$Q'_{RT} = D_{t,RT} \times Q/(T \times C) + Q_{RT}. \tag{4}$$

Else, for $D_{t,RT} \prec 0$, we set Q'_{RT} to the initially allocated quantum for class RT. Note that we consider the off times of the input traffic where "off time" means the duration when all the queues are empty and the server is idle. Therefore, we have:

$$D_{i,t} = c_i(T - T_{off}(t-T,t)) - S_i(t-T,t), \tag{5}$$

where T is the window size, $T_{off}(t_1,t_2)$ is the total time of idle for server between t_1 and t_2, and c_i is the portion of bandwidth for class i. Now we have to allocate the remainder of the quanta to the classes HBT and SBT:

$$Q_p = Q_{HBT} + Q_{SBT} \tag{6}$$

$$Q'_p = Q - Q'_{RT} \tag{7}$$

where Q_{HBT} and Q_{SBT} are the initial quantum for HBT and SBT class respectively, Q'_p is the remaining quanta

for classes HBT and SBT after quantum adjustment of class RT. Then we update Q'_{HBT} and Q'_{SBT} as newly updated quanta for classes HBT and SBT:

$$Q'_{HBT} = Q'_p \frac{Q_{HBT}}{Q_P} \quad (8)$$

$$Q'_{SBT} = Q'_p \frac{Q_{SBT}}{Q_P}. \quad (9)$$

The 'measured fairness percentage' for class i between t_1 and t_2 can be given by:

$$F_i(t_1,t_2) = \frac{S_i(t_1,t_2)}{((t_2-t_1)-T_{off}(t_1,t_2))\times C} \times \frac{Q}{Q_i} \times 100 \quad (10)$$

where $S_i(t_1,t_2)$ is the bytes served by class i between t_1 and t_2. Note that the term $((t_2-t_1)-T_{off}(t_1,t_2))\times C$ is the bytes served by all classes when the server is busy during the period (t_2-t_1). Hence, we have:

$$(t_2-t_1).C = T_{off}(t_1,t_2).C + \sum_{i=1}^{n} S_i(t_1,t_2). \quad (11)$$

Also we can see that:

$$B_{off}(t_1,t_2) = \frac{T_{off}(t_1,t_2).C}{(t_2-t_1).C} \times 100 = \frac{T_{off}(t_1,t_2)}{(t_2-t_1)} \times 100 \quad (12)$$

where $B_{off}(t_1,t_2)$ is the percentage that the server has served no traffic between t_1 and t_2.

IV. Proposed Classifier SBDC

In this section we introduce our proposed approach, the short-time burst degradation classifier (SBDC), to degrade the short-term burstiness that is our main contribution. The key point in designing the classifier is the management of the queuing delay in a way that we can overcome the short-time scale burstiness of the traffic. Using SBDC, we can notably decrease the short-term delay and the packet jitter for real-time traffic.

Fortunately, the IP network is connection less and packets are transmitted independently. We should exploit this property in congestion situations. As every burst in the network may belong to many flows, changing the order of low priority packets in short-time scales and especially in the cases of network congestion may not have a significant drawback for each end user to arrange the packets by TCP. Besides, the traditional methods such as using the leaky bucket for regulation of the traffic to overcome the burstiness is not as efficient as our scheme to guarantee the QoS since our scheme focuses on the management of the queuing delay in a flexible manner. The efficiency of SBDC compared to leaky bucket scheme will be shown in the next section. Fig. 2 presents a general view of the SBDC classifier. The classifier defines the traffic into N categories (shown as Cat_1, $Cat_2 \ldots Cat_N$ in the figure). The traffic can be divided based on the sensitivity to delay and jitter. If we assume descending sensitivity to delay and jitter as category 1, 2, 3, …, N, then we are providing different values of delay as 0, τ_1, τ_2, …, τ_N to those categories.

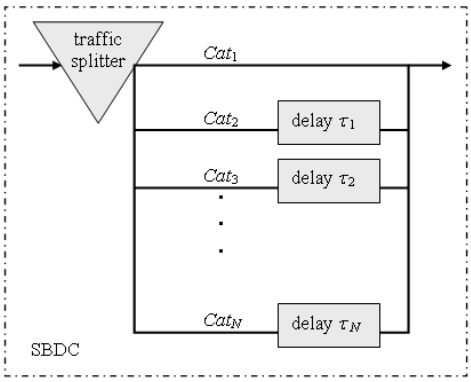

Figure 2. The short-time burst degradation classifier (SBDC)

The traffic expressing the highest delay-sensitivity, i.e., the real time traffic, can be assigned to Cat_1 and can pass through the classifier directly. On the other hand, the lower delay sensitive traffic can be split by a round robin manner into two parts (or more). The first part can be assigned to Cat_2 whereas the second part can be assigned to Cat_3 and then they can pass via the classifier with delay τ_1 and delay τ_2 respectively. Therefore, with assigning packets in the same burst to different categories with various delays taking into consideration QoS, the burst will be broken. Intuitively, it is better to divide the traffic into various categories as often as possible.

Even though, our classifier is general and can be implemented in different points of the network, we propose to use it in MPLS ingress nodes. We consider four types of traffic based on QoS guarantees for an MPLS network: real-time traffic (RT), signalling traffic (ST), high priority best-effort traffic (HBT), and simple priority best-effort traffic (SBT) [12]. The amount of traffic sent by each node on the Internet is about 10% for RT, 20% for HBT, 70% for SBT, if we ignore the traffic of class ST. Also, there are 93% TCP traffic and 7% UDP. On the other hand, the original application is 5% for SMTP, 0.5% for Telnet, 70% for Web and NNTP, 10% for FTP, and 14.5% for the others [13, 14]. In Fig. 3, we show how to bind the class of services to the various applications.

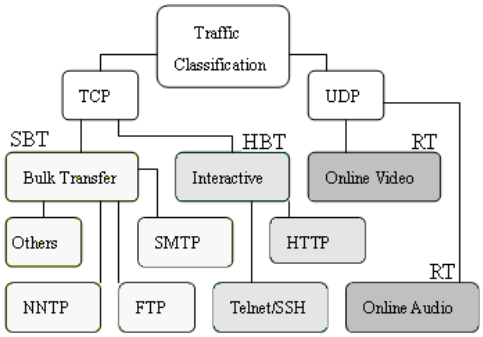

Figure 3. The class of services binding to various traffic applications

To implement SBDC, we first classify the packets to the classes of service RT, HBT, and SBT; then, using round robin we split HBT traffic equally and uniformly into two queues (one packet to the first queue and the next packet to another queue, respectively). The second queue has delay τ. Also for SBT traffic, using round robin we split the packets equally into three queues where the second queue has delay τ and the third queue has delay 2τ. After forcing these delays to some parts of the traffic, the classifier will deliver the packets to the scheduler. Fig. 4 shows the queuing part of this classifier combined with SQ-CBQ.

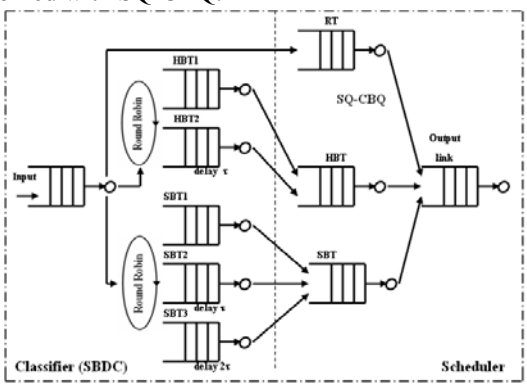

Figure 4. Switching discipline using SBDC and SQ-CBQ

Our approach can be implemented in various ways depends on the percentage of each class of the traffic and its delay sensitivity. For example, the bulk traffic can tolerate more delay and jitter; consequently we can afford forcing some delays in some parts of it. For the real time traffic, jitter is a very important parameter and we should decrease it. We present in this paper a simple version of this technique. Anyhow in the case of implementation, we can split the traffic considering that packets belonging to the same flow would go to the same queue with round robin. This can help to reduce the overhead of sorting the packets by TCP at the destinations, as each burst in backbone traffic may belong to many flows. We can also use some prediction or measurement techniques with SBDC to be activated in congestion situations. However, one of the most important advantages of this proposed mechanism is its flexibility since the implementation can be based on different policies.

To show the performance of the SBDC scheme we compare it with another scheme in which some leaky buckets are used to smooth the traffic; here we call it the leaky bucket scheme. In this case, we use two leaky buckets: one for smoothing the traffic of class HBT and a second one for smoothing the SBT traffic. We set the parameters of the leaky buckets such that we can add approximately a maximum delay of τ to class HBT and a maximum delay of 2τ to class SBT. This leads to improve the packet delay and jitter for class RT. Fig. 5 shows the switching discipline using leaky bucket and SQ-CBQ.

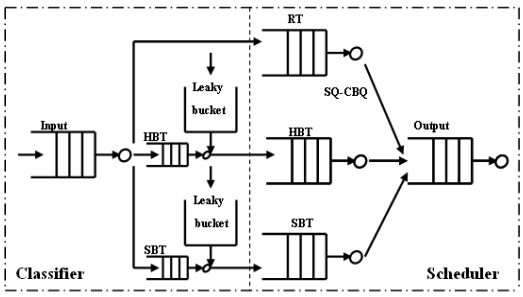

Figure 5. Switching discipline using leaky bucket scheme and SQ-CBQ

V. Simulation Results

A. Assumptions

In this paper, we make use of 40 minutes of the traffic captured from the link of Indianapolis to Cleveland in USA [15]. The average rate of the real traffic is about 23 Mbytes/s as shown in Fig. 6. We have classified the real traffic into RT, HBT, and SBT classes, ignoring the traffic of class ST, in Fig. 7.

For our simulation, we initialized quantum sizes as 1500 Bytes for class RT (maximum size of a packet is 1500 Bytes), 11000 Bytes for class HBT, and 37500 Bytes for class SBT. The permissible delay for the link capacity $C = 50$ MBytes/s is 300 us and for the congestion situation with $C = 25$ MBytes/s is about 2.5 ms.

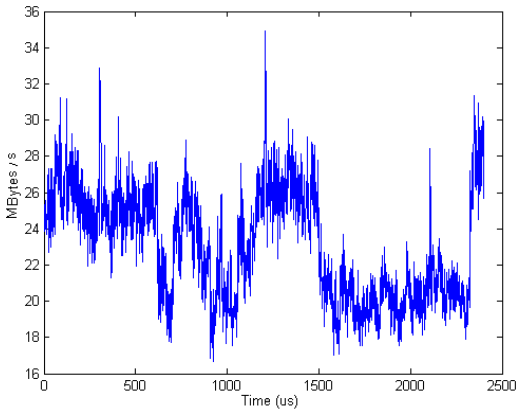

Figure 6. The rate of the real traffic

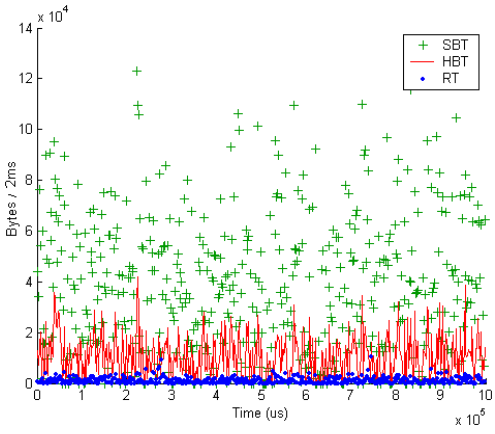

Figure 7. The classification of the real data traffic

B. System performance

Fig. 8 shows that the average packet delay for the class RT using PDRR is lower than the experienced delay using DRR. The proposed SQ-CBQ outperforms the other schedulers (DRR and SQ-DRR) as we can notice from Fig. 9 for a non-congestion situation (C = 50 MBytes/s) and from Fig. 10 for a congestion situation (i.e., C = 25 MBytes/s).

Fig. 11 shows that SQ-CBQ provides both short-term and long-term fairness among the traffic classes i, since each measured fairness F_i doesn't exceed one-hundred percent. From Fig. 12, we can notice only a one-time short-term fairness violation at around 6.8 seconds in the congestion situation. In the figures, B_{off}, the percentage that the server is idle and has no traffic to be served, also can be seen.

A variance time plot [16] can be used to represent the degree of burstiness in short-time and long-term scales. Fig. 13 shows the degradation of bursts of the real traffic in short-time scales that passes via classifier SBDC for various values of delay τ. The variance time plot in the figure shows that our technique with various values of delay τ decreases the variances for short-time scales. It is interesting to notice that for τ = 2 ms, the variance will be decreased significantly. Note that we can decrease the variances only for short-time scales and this technique can not be used for long-time scale bursts. Fig. 14 also shows how we could smooth the traffic using SBDC.

To show the gain of performance due to the utilization of SBDC scheme, we compare its performance with the performances of leaky bucket scheme (Fig. 5) that uses two leaky buckets to smooth the HBT and SBT traffics. We assume that CIR = 10.6 MBytes/s and B_C = 15 kBytes for the leaky bucket used for the HBT traffic and CIR = 32.5 MBytes/s and B_C = 27.5 kBytes for the leaky bucket used for the SBT traffic to add approximately a maximum delay around τ = 2 ms to class HBT and a maximum delay around 2τ = 4 ms to class SBT. Fig. 15 and Fig. 16 show the packet delays of HBT and SBT traffics after using the leaky bucket scheme and SQ-DRR. Fig. 17 and Fig. 18 compare the delay for class RT using SQ-DRR and SQ-CBQ with various classifiers (τ = 2 ms). We can notice that the short-term delay has been significantly decreased using SBDC and SQ-CBQ.

We also show the amounts of backlogs for class RT and its quantum sizes using various schedulers in Fig. 19 and Fig. 20. In the figures we can see how the quantum size of the class RT is being changed and its backlog is managed consequently. The amount of total served bytes by schedulers between 0 and t is shown in Fig. 21. It shows using SQ-CBQ we can achieve a better throughput for the congestion situation.

We have evaluated the performance of SBDC in terms of packet jitter combined with both SQ-CBQ and SQ-DRR. Fig. 22 shows packet jitters for RT traffic after using SBDC and SQ-DRR for various delay τ with C = 50 MBytes/s. In Fig. 23, similar results are shown using SBDC and SQ-CBQ. The figures show that we could significantly decrease the packet jitter for RT traffic with $\tau \geq$ 2 ms.

We compare the packet jitters using various schemes, taking into consideration SQ-DRR as a base when the real traffic passes via it; in this case, the packet jitter equals 86526 us^2. Using SBDC and SQ-DRR with τ = 2 ms, the packet jitter is 28301 us^2 (67.2% improvement compared with using SQ-DRR). Also, we notice that using SBDC and SQ-CBQ, we will improve the packet jitter to 20128 us^2 (76% improvement). On the other hand, if we use the leaky bucket scheme and SQ-DRR, then the packet jitter will be improved to 72101 us^2 (16.7% improvement). Finally, with the leaky bucket scheme and SQ-CBQ it will be improved to 53175 us^2 (38.5% improvement).

SBDC can also be implemented in a more simple way in MPLS ingress nodes. As we have high volume of the SBT traffic, we can send the RT and HBT traffic directly to the scheduler without delay and split the SBT traffic to two parts and send the first part without delay and another part after τ = 2 ms or τ = 4 ms to the scheduler (Fig. 24). Fig. 25 shows the comparison among various ways of the delay management in SBDC. Therefore, how the management of the delay considering classes of service is very important to have the best efficiency of SBDC.

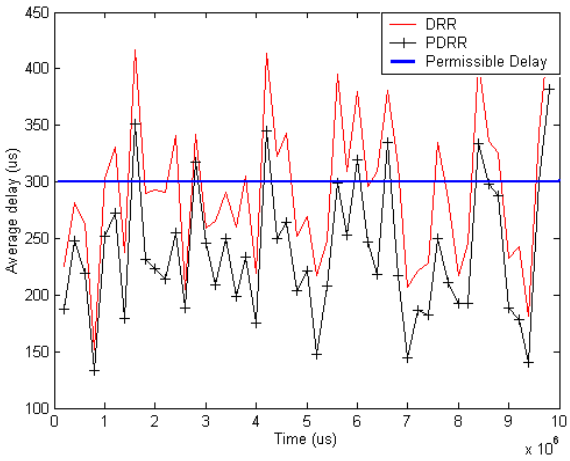

Figure 8. Average delay for class RT with C = 50 MBytes/s using PDRR and DRR

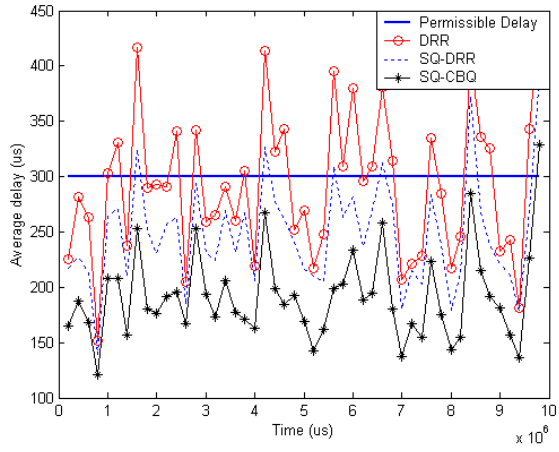

Figure 9. Average delay of class RT using different scheduling schemes with C = 50 MBytes/s

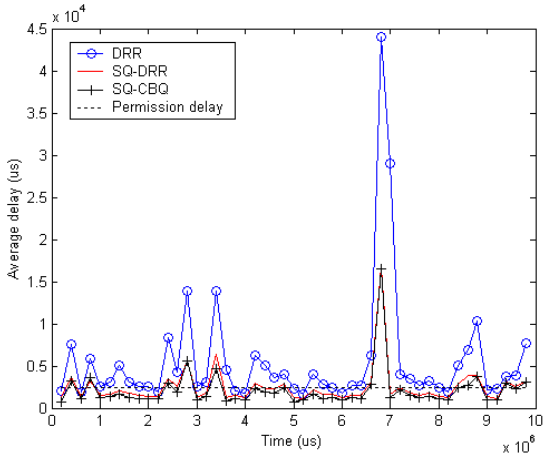

Figure 10. Average delay of class RT using different scheduling schemes in the case of congestion (C = 25 MBytes/s)

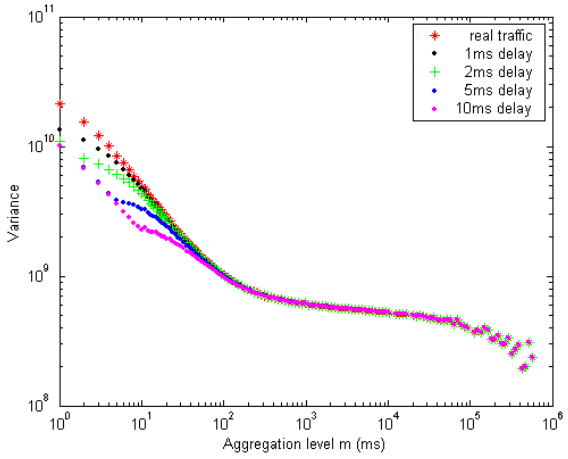

Figure 13. Variance time plot of traffic via SBDC with various delay τ

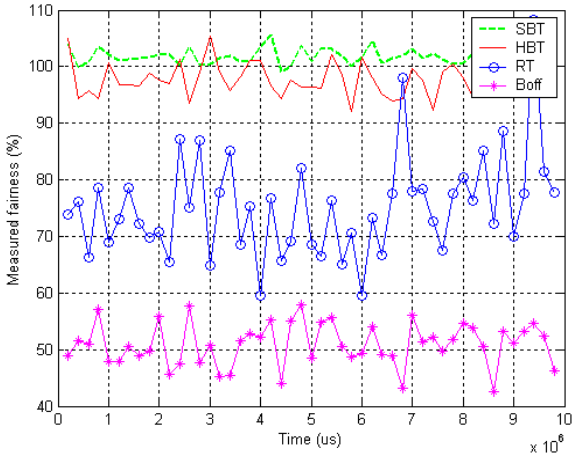

Figure 11. Fairness of all traffic classes for SQ-CBQ with C = 50 Mbytes/s

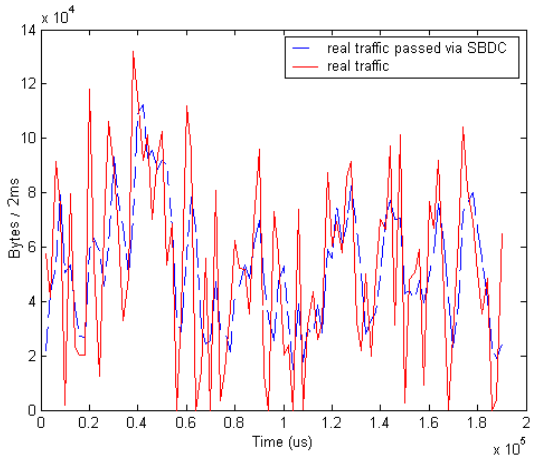

Figure 14. Rate of the real traffic passed via SBDC with τ = 2 ms

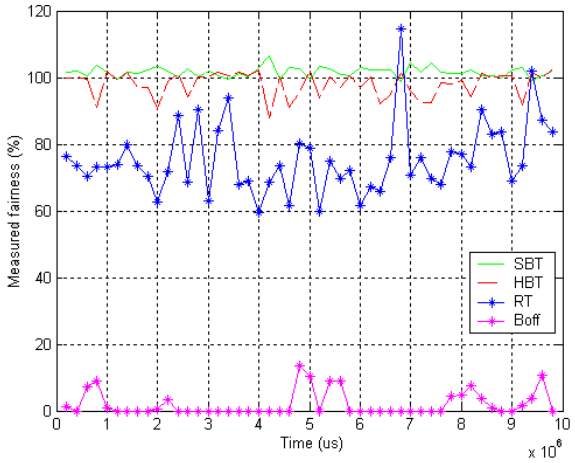

Figure 12. Fairness of all traffic classes for SQ-CBQ in the case of congestion (C = 25 MBytes/s)

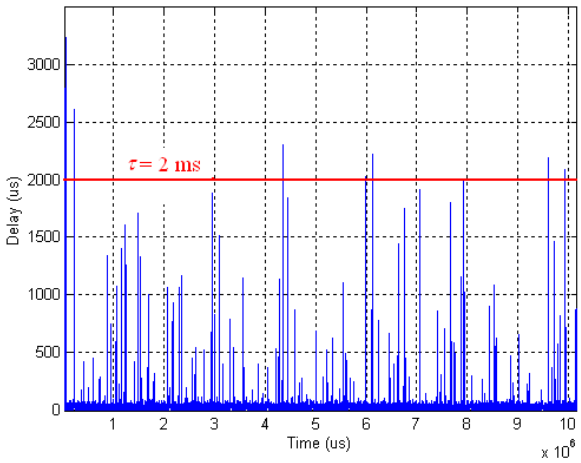

Figure 15. Delay of class HBT using the leaky bucket scheme and SQ-CBQ with C = 50 MBytes/s

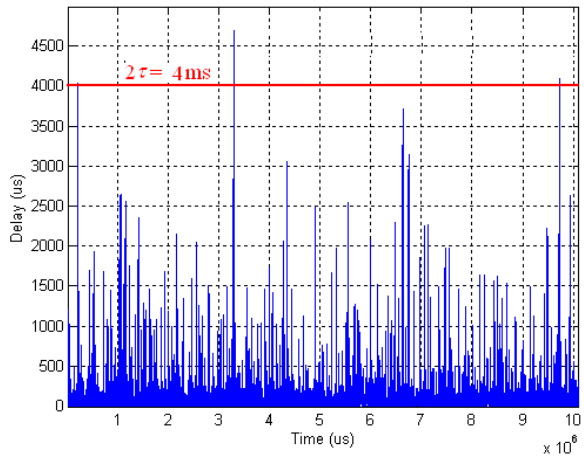

Figure 16. Delay of class SBT using the leaky bucket scheme and SQ-CBQ with C = 50 MBytes/s

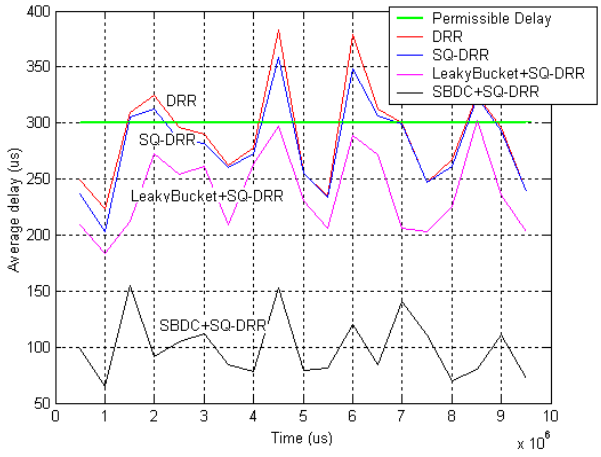

Figure 17. Delay of class RT using SQ-DRR and various classifiers (τ = 2 ms) with C = 50 MBytes/s

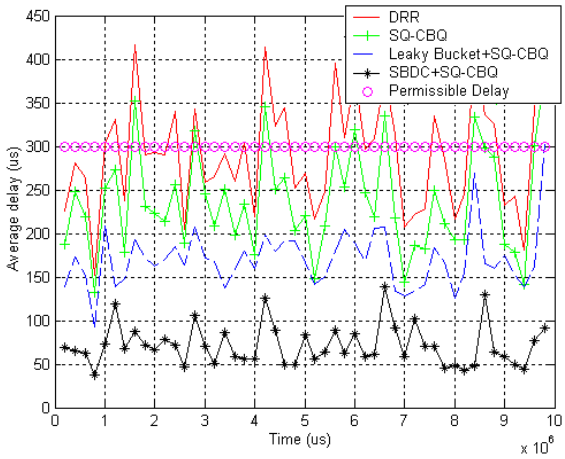

Figure 18. Delay of class RT using SQ-CBQ and various classifiers (τ = 2 ms) with C = 50 MBytes/s

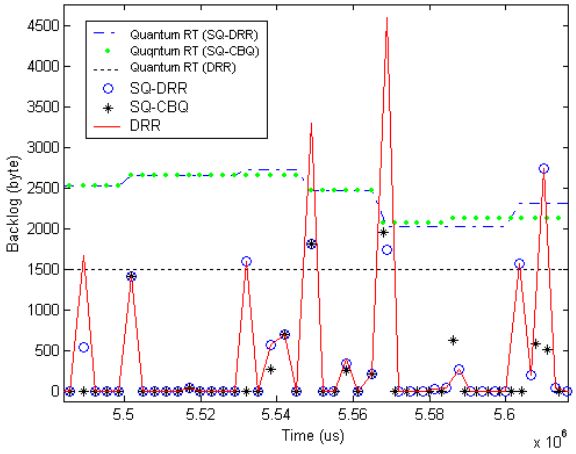

Figure 19. Backlog of class RT with C = 50 MBytes/s

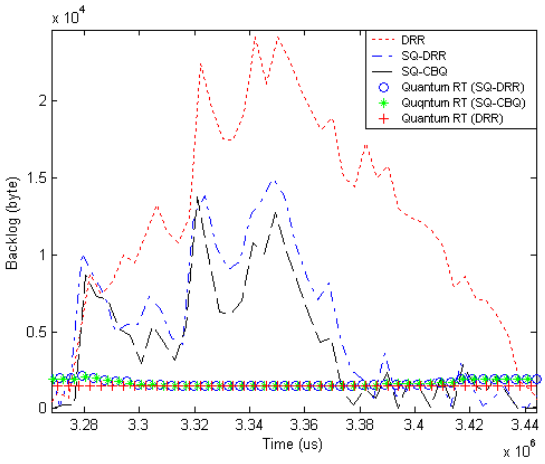

Figure 20. Backlog of class RT in case of congestion with C = 25 MBytes/s

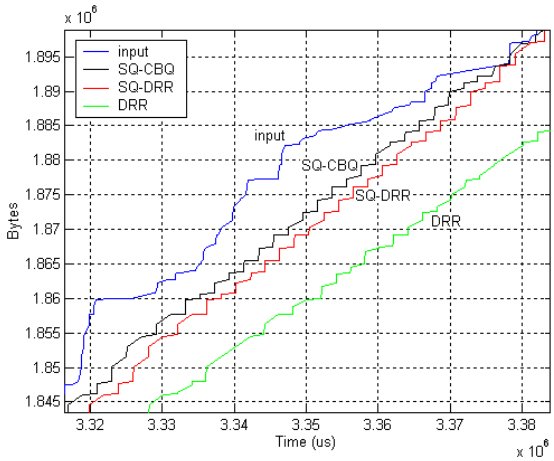

Figure 21. The amount of total served bytes by schedulers between 0 and t in case of congestion with C = 25 MBytes/s

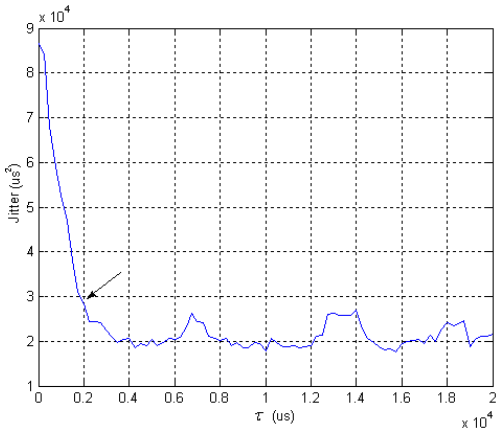

Figure 22. Jitter for class RT after using SBDC and SQ-DRR for various delay τ with C = 50 MBytes/s

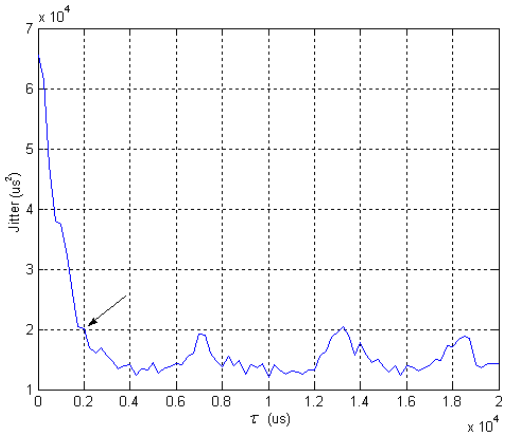

Figure 23. Jitter for RT traffic with SBDC and SQ-CBQ for various delay τ with C = 50 MBytes/s

VI. Conclusion

In this paper, we have presented a novel general classification scheme and we have proposed its implementation for MPLS ingress nodes. The purpose of the new classification algorithm is to decrease the impact of short-time scale bursts of the input traffic and to improve the packet delay and jitter for the real-time traffic class. Also we have introduced a scheduler to be combined with this classifier for providing better short-term QoS. Finally, we have provided some performance evaluation of our schemes by simulation.

The proposed scheduler, short-term quality-of-service class based queuing (SQ-CBQ), uses our new version of traditional deficit round robin scheduling named PDRR (Polling DRR). On the other hand, the design of our proposed short-time burst degradation classifier (SBDC) is based on the management of the queuing delay in a way that we can overcome the short-time scale burstiness of the traffic. Using this scheme leads to have less burstiness and decreases the size of bursts. Because with splitting the traffic (and hence the bursts) and force some delays on some parts of those bursts we will certainly have less burst sizes. The effects of using this technique in switching nodes are the decreasing of the delay and jitter for delay sensitive classes significantly.

To show the performance of the SBDC scheme we compared it with another scheme, in which some leaky buckets are used to smooth the traffic; we call it leaky bucket scheme. The SBDC outperforms the leaky bucket scheme and can notably improve the packet delay and jitter of the real-time traffic.

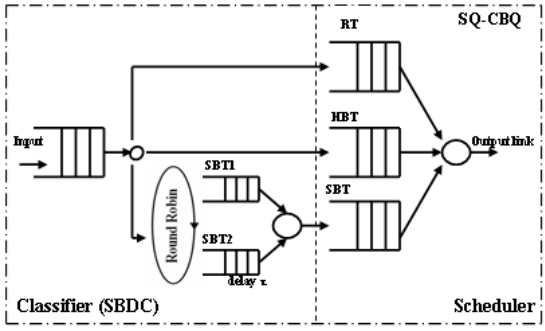

Figure 1. Second way of the delay management in SBDC

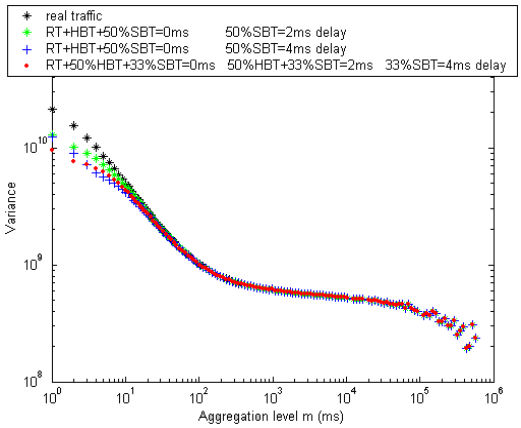

Figure 2. Comparison of various ways of the queuing delay management in SBDC

References

[1] J. Domańska, A. Domański, and T. Czachórski, The Influence of Traffic Self-Similarity on QoS Mechanisms, Proc. of Symposium on Applications and the Internet Workshops (SAINT-W) (2005) 300-303.

[2] H. Jiang, and C. Dovrolis, Why is the Internet Traffic Bursty in Short Time Scales?, Proc. of Int. Conf. on Measurement and modeling of computer systems (SIGMETRICS) (Jun. 2005) 241-252.

[3] L. Guan, I.U. Awan, M.E. Woodward, and X. Wang, Discrete-Time Performance Analysis of a Congestion Control Mechanism Based on RED Under Multi-Class Bursty and Correlated Traffic, Journal of Systems and Software, vol. 80, no. 10 (Oct. 2007) 1716-1725.

[4] H. Toral, D. Torres, C. Hernández and L. Estrada, Self-Similarity, Packet Loss, Jitter, and Packet Size: Empirical Relationships for VoIP, Proc. of Int. Conf. on Electronics, Communications and Computers (2008) 11-16.

[5] Ph. Owezarski, and N. Larrieu, Internet Traffic Characterization-An Analysis of Traffic Oscillations, High Speed Networks and Multimedia Communications (HSNMC), LNCS 3079, vol. 3079 (2004) 96-107.

[6] M. H. Kim, and H. S. Park, Scheduling Self-Similarity Traffic in Packet-Switching Systems with high Utilization, IEE Proc. Commun., vol. 151, no. 5 (Oct. 2004) 429-437.

[7] X. Yu, I. L. -J. Thng, Y. Jiang, and C. Qiao, Queueing Processes in GPS and PGPS with LRD Traffic Inputs, IEEE/ACM Trans. Networking, vol. 13, no. 3 (Jun. 2005) 676-689.
[8] Li. Qian, Ma, Zhengxin, and Du, Wen, A Low Pass Filter Traffic Shaping Mechanism for the Arrival of Traffic, International Conference on Wireless Communications, Networking and Mobile Computing (WiCom) (Sept. 2007) 1032-1035.
[9] W. JIA, H. WANG, W. TU, and W. ZHAO, A New Delay Control Method for Real-Time Flows, Journal of combinatorial optimization, Springer, vol. 12 (2006) 127-149.
[10] M. Shreedhar, and G. Varghese, Efficient fair queueing using deficit round robin, IEEE/ACM Trans. Networking, vol. 4, no. 3 (Jun. 1996) 375-385.
[11] S. Vegesna, IP quality of service, Cisco Press, 2001.
[12] G. Ahn and W. Chun, Design and Implementation of MPLS Network Simulator (MNS) supporting QoS, Proc. of Int. Conf. on Information Networking (ICOIN) (2001) 694-699.
[13] A. Arvidsson and P. Karlsson, On Traffic Models for TCP/IP, Proc. of Int. Teletraffic Congress (ITC-16) (1999).
[14] K. Thompson, G. J. Miller, and R. Wilder , Wide-Area Internet Traffic Patterns and Characteristics, IEEE Network, vol. 11, no. 6 (1997) 10-23.
[15] http://pma.nlanr.net/Traces/long/ipls1.html
[16] V. Paxon, and S. Floyd, Wide-Area Traffic: The Failure of Poisson Modeling, IEEE/ACM Trans. Networking, vol. 3, no. 3 (Jun. 1995) 226-244.

Masoomeh Torabzadeh received the M.E. degree in Computer Architecture Engineering from Amirkabir University of Technology (Tehran Polytechnic), department of Computer Engineering and Information Technology, Tehran, in 2003 and the Ph.D. degree in Informatics (Fundamental Networks) from the Graduate University for Advanced Studies, the Department of Informatics, Tokyo, in 2008. She was ranked first of M.E graduates and received the honor scholarship during the Ph.D. study. She was a research assistant at the National Institute of Informatics, Tokyo, between Aug. 2005 and Jan. 2008. She had a post-doc fellowship at the Department of Computer Sciences, Université du Québec à Montréal, QC, Canada between Feb. 2008 and Aug. 2008. Her research interests include MIMO cross-layer scheduling, wireless communication networks, traffic analysis, and the quality of service mechanisms in high-speed networks.

Wessam Ajib received an Engineer diploma from Institut National Polytechnique de Grenoble, France (INPG-ENSPG) in Physical Instruments, in 1996, a DEA (Diplôme d'Études Approfondies) from École Nationale Supérieure des Télécommunication (ENST), Paris, France in 1997 in Digital Communication Systems, and a Ph.D. degree in Computer Sciences and Computer Networks from ENST, Paris, France, in 2001. He had been an architect and radio network designer at Nortel Networks, Ottawa, ON, Canada between October 2000 and June 2004. He had conducted many projects and introduced different innovative solutions for UMTS system, the third generation of wireless cellular networks. He had a post-doc fellowship at Electrical Engineering department of École Polytechnique de Montréal, QC, Canada between June 2004 and June 2005. Since June 2005, He had been with the Department of Computer Sciences, Université du Québec à Montréal, QC, Canada, where he is an assistant professor of computer networks. His research interests include wireless communications and wireless networks, multiple access and MAC design, traffic scheduling error control coding, MIMO systems and cooperative communications. He is the author or co-author of many journal papers and conferences papers in these areas.