A Bandwidth Sharing Method under Node Autonomy and Short-Term Protection for QoS

Alfredo P. Mateos-Papis (a).

(a) Division of Sciences of Communication and Design. Universidad Autónoma Metropolitana. Cuajimalpa. Mexico DF.
amateos@correo.cua.uam.mx

Abstract. This paper proposes a method to share bandwidth between intersecting routes in networks subject to per-route distributed admission processes. With this method the current bandwidth share of a flow is guaranteed in the short-term against a possible bandwidth decrease caused by flows recently admitted in intersecting routes. However, this share-protection is decreased slowly such that, in the long term, bandwidth is distributed according to the intersecting route's observed requirements. With this method every node takes its own decisions, having an impact on the bandwidth sharing between routes and attaining, with this, global behaviors in the network. This method is expected to help preserve the Quality of Service in the routes while, because of its simplicity, allowing a simple bandwidth management in the network and possibly permitting its implementation in bigger-sized networks. The method proposed is termed: the Short Term Protection (STP) method.

Keywords. Quality of Service (QoS); Bandwidth Sharing; Distributed Traffic Control.

1. Introduction.

This article addresses in a novel way the problem bandwidth-sharing for the long-time studied QoS-aware-networks having per-route distributed admission processes, using a novel uncomplicated method where the nodes operate autonomously offering the possibility to attain global behaviors.

In bandwidth-limited data networks which offer Quality of Service (QoS), a flow-admission-process is used to foresee if the traffic increase caused by the entrance new flows to the network will originate an unacceptable deterioration of the QoS offered.

An admission process to a network can be centralized or distributed. In the centralized approach a single entity (like a bandwidth broker) makes admission decisions based on its global view of the network. This global-view requirement makes this approach less scalable compared with that of the distributed admission approach wherein there is an admission entity for each one of different parts of a network. The admission to each part is done based on a partial view of the network, with the potential risk that an admission decision could inadvertently cause an unacceptable deterioration of the QoS in other parts of the network.

The bandwidth sharing method introduced in this paper, which is called Short-Term Protection method (STP method), is aimed at networks with per-route distri-

Advances in Computer Science and Electronic Systems
Research in Computing Science 52, 2011, pp. 155-168

Received 23/01/11
Accepted 20/02/11
Final version 07/03/11

buted admission processes, as those proposed in [1] [2], where there is no bandwidth reservation considered for any route. This method is intended to protect routes against bandwidth exhaustion derived from sudden increments in traffic in intersecting routes while still allowing the network to have a simple distributed management scheme which adjusts in accordance to the bandwidth necessities of the routes. With this method each node acts autonomously and there is no central admission authority considered. The nodes are aware neither of the existence of routes nor of the existence of flows in the network. Fig. 1(a) is a schematic representation of a network with two routes that intersect at node *x* (more specifically at its output interface) and share the bandwidth of the node's output link. In this paper these routes are called *intersecting routes* and node *x* is called an *intersection node*.

An intersection node has, at every one of its output interfaces, one queue for every one of its input interfaces. As an example, Fig. 1(b) represents a node with three input interfaces: *A*, *B* and *C*, from where three routes converge at the node and leave the node through the output interface *D* which has three queues.

Inspired by the Weighted Fair Queuing[1] (WFQ) scheduling algorithm [3], the STP method gives each queue a weight, however, each weight may change slowly according to an updating algorithm that is periodically evaluated. In this paper the STP method uses a WFQ scheduler, but the scheduler could be other kind of work-conserving scheduler [4] which could adequately operate with changing weights.

In the STP method traffic coming from the different input interfaces competes for bandwidth in a special way: the weights change slowly in favor of the queues which have bigger average lengths. For example, a route could lose up 20% of its weight in a 30-minute interval, against an intersecting route, if its queue length were constantly smaller –within this interval– compared to that of the intersecting route.

The STP method offers a simple and novel option for data networks to offer scalable QoS in comparison with the DiffServ model, studied since 13 years ago, which is oriented for scalability and flexibility in bandwidth-sharing (see section 2) but which poses a native difficulty in the evaluation of the traffic conditions in the networks.

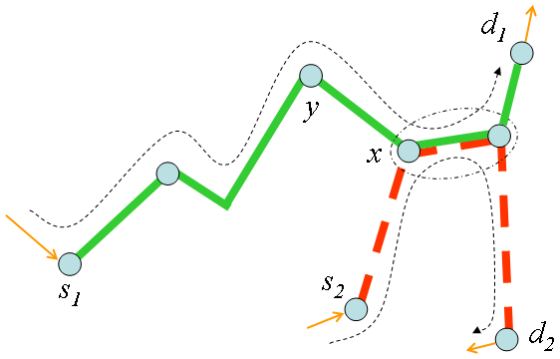
2. Background and Related Works.

The STP method has been motivated by some characteristics of the nodes used in the DiffServ[2] model [5] [6], and in the network conditions proposed in [1] which deals with networks with per-route distributed admission where the admission decisions are taken at the edge of the routes. In DiffServ the packets are classified when entering to the so-called DiffServ domain (so its admission is per-class oriented instead of per-flow oriented), and the central nodes (also called core nodes) of that domain follow a specific behavior when treating the packets of a specific class.

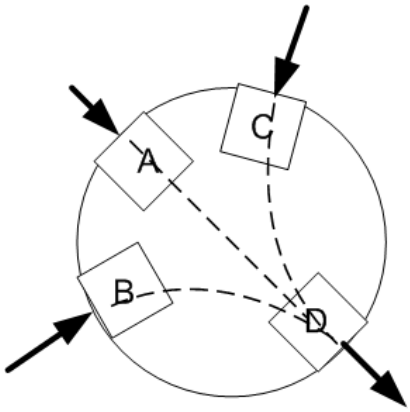
It is reasonable to consider that the behaviors provided by a node are implemented at its output interfaces [7]. Typically, an output interface of a node keeps one queue (buffer) for every one of the classes to attend, which share the total bandwidth of the output interface. Any output interface with more than one queue in operation needs a

[1] Also called PGPS (Packet-by-Packet Generalized Processor Sharing).

[2] See technical report *Rep-Tec-4-Arquitectura DS.pdf* at http://ccd.cua.uam.mx/~amateos/.



(a) Routes $s_1 \rightarrow d_1$ and $s_2 \rightarrow d_2$ in a network intersect at the output interface of node x.



(b) Representation of an intersection node with three input interfaces, *A*, *B*, and *C*, and one output interface, *D*, in which the incoming traffic converges.

Fig. 1. The concepts of two intersecting routes (subfigure a), and the concept of an intersection node (subfigure b).

scheduler to choose the next packet to be dispatched over the output interface, thus creating a bandwidth sharing process between classes.

There are methods for optimization of a global indicator in a network by obtaining the most effective bandwidth-share between classes. For example [8] presents a bandwidth-sharing method described as proactive (in contrast with *reactive* algorithms) which uses a central bandwidth manager. Another bandwidth-sharing method between classes [9] adaptively adjusts the weights of a weighted round robin scheduler in every core node and does not depend on a central bandwidth manager. It looks like the nodes can effectively operate autonomously and cooperate to obtain an expected end-to-end QoS indicator.

Centikaya et al [1] [2] propose a method of admission to a network that has predefined routes. This method is defined as *distributed* since the admissions are made per-route, not per-flow oriented, without a central management, where a route can cause a

negative impact in intersecting routes when admitting a flow. The main research goal of the paper is the admission process, not the route-intersection problem.

The STP method is oriented towards handling the negative interaction between intersecting routes with per-route admission but it is not aware of any admission process.

In [10] a *Coordinated-Schedulers* method is presented to provide delay-bounds in a network, which is not per-flow oriented. In it, the core nodes modify each arriving-packet's priority index depending on whether the packet was serviced late or early at the upstream node as a consequence of cross-traffic. The method provides natural coordination between nodes but each node operates autonomously. This method can work in the context of a single class and it seems that it could have the benefit of protecting a route against intersecting routes. The STP method is simpler although it is not in the scope of this paper to study if it can attain the benefits of the method presented by [10].

3. The STP Method.

3.1 Remarks.

The STP method was conceived to operate within one class of traffic in a network, so this paper considers traffic in a single class. The possible interaction of a class using the STP method with schemes doing bandwidth provisioning between classes, to optimize a global indicator in a network, is out of the scope of this paper.

The evaluation of the STP method is made with the results obtained from simulations with traffic generated using a mixture of constant bit rate and Pareto sources. This traffic does not represent the diversity of traffic that might exist in a network; nevertheless these results are presented as a starting point for the evaluation of this method. In the simulations a change in the amount of traffic is made through the change of the number of sources.

The evaluation of the method is limited to a single direction in the routes as it is considered that the traffic in one direction is independent of the traffic in the opposite direction.

The topology used for the simulations is rather small. It has one central node (see section 2). The STP method is used in just the central node as this paper is aimed to compare the results in a network when a single central node uses the STP method with the results when that central node is a *normal* node, that is, the node uses a single output queue for each output interface. The author has made simulations with two nodes capable of using the STP method, which will be presented in a subsequent paper.

3.2 Condensed Explanation of the STP Method.

The QoS parameter of interest in this paper is the delay. This paper evaluates the STP method based on the end-to-end delay in the routes. This delay depends on several factors, including the queuing delay in the intersection nodes which will be the

factor of interest in this paper. This paper defines the term *delay-limit* as the maximum permissible delay that a packet can experiment while traversing a route.

Consider an output interface of a node working with the STP method, where there are *N* queues being attended by the scheduler (so there are *N* routes intersecting at that interface). The weight of each queue should tend to be proportional to its relative average length compared with the sum of the lengths of the queues. The rate of weight-change should be slow, that is, the bandwidth required to satisfy a bandwidth-greedy route should be released slowly in order to protect, temporarily, the routes which would loose bandwidth.

The method starts at time t_0 where the all the weighs have the same value. At the ending of every time-interval of duration τ, the method computes an indicator for every queue of the output interface, to compare the current weight of the queue with its current relative length with regard to the lengths of the other queues. The indicator is represented with $I\Delta$ in equation (1). This time-interval should be sufficiently big as to be able to observe several arrivals and departures of packets (in this paper $\tau = 1s$). After computing this indicator the method computes the new weights of the queues and substitutes those in operation. All this computation should take place in a time much smaller than τ. In (1), the computation of this indicator is made at time $t_0 + (r+1)\tau$ for queue *i*, where *r* is an integer such that $r \geq 0$, $\phi_i^{Act}(t_0 + r\tau)$ represents the weight of queue *i*, which has been valid from $t_0 + r\tau$ to $(t_0 + (r+1)\tau)^-$), and the average lengths of the queues, obtained at time $t_0 + (r+1)\tau$, are represented with $Q_i(t_0 + (r+1)\tau)$.

$$I\Delta\left(\phi_i^{Act}(t_0 + r\tau)\right) = \frac{Q_i(t_0 + (r+1)\tau)}{\sum_{j=1}^{N} Q_j(t_0 + (r+1)\tau)} - \phi_i^{Act}(t_0 + r\tau) \quad \textbf{(1)}$$

If, from equation (1), the indicator results negative for queue *i*, then this queue should lose weight so that the result of its new weight, $\phi_i^{Act}(t_0 + (r+1)\tau)$, minus its weight in operation, $\phi_i^{Act}(t_0 + r\tau)$, should reflect a negative increment (or a decrement). Equation (2) proposes a form of calculation for this negative increment, which is represented with $\Delta\left(\phi_i^{Act}(t_0 + r\tau)\right)$.

$$\Delta\left(\phi_i^{Act}(t_0 + r\tau)\right) = \phi_i^{Act}(t_0 + (r+1)\tau) - \phi_i^{Act}(t_0 + r\tau) = \ln(1-P)\frac{\tau}{T}\phi_i^{Act}(t_0 + r\tau) < 0 \quad \textbf{(2)}$$

The two important parameters of (2) are *P* and *T*. Parameter *P* is called the *loss factor*, which is a positive constant much smaller than 1 (with values near to 0.1). When *P* is very small then $-\ln(1-P) \approx P$. The parameter *T* represents a *time span* and it might be on the order of 1200*s*. The ratio T / τ should be a big integer called *K*.

From (2) it can be seen that, as τ is much smaller than T, the decrement of the queue, from one interval to the next one, should be very small.

It can be shown that if queue i decreased its weight for K consecutive intervals of size τ within time-span T, then the ratio of its initial and ending weights, with regard to that time span, would be equal to $(1-P)$, that is: $1-P = \phi_i^{Act}\left(t_0+(l+1)T\right) / \phi_i^{Act}\left(t_0+lT\right)$. This formula can be written as: $\phi_i^{Act}\left(t_0+(l+1)T\right) - \phi_i^{Act}\left(t_0+lT\right) = -P\phi_i^{Act}\left(t_0+lT\right)$. With this it can be stated that with this method the maximum weight deterioration for a queue in a time T is P%. This result is demonstrated using the fact, which can be obtained from (1) and (2), that $\phi_i^{Act}\left(t_0+(m+K)\tau\right) = \phi_i^{Act}\left(t_0+m\tau\right)\left(1-f(P)/(T/\tau)\right)^{T/\tau}$. The variables l and m represent integers greater than, or equal to 0, and $f(P) = -\ln(1-P)$. Because of the limitation of space of this paper, all the details of this method are not developed here[3].

The STP method also proposes a calculation for the weight of every queue that should not decrease its weight, as indicated by (1), after every interval of duration τ, in such a way that the sum of the weights of all queues be always equal to 1.

Following the method proposed in [11], the average length Q_i of queue i is computed as $Q_i = (1-w_q)Q_i + w_q q_i$, where w_q is the averaging parameter and q_i is the instantaneous queue-length. This equation acts as a low-pass filter. In this equation, the smallest the value of w_q the smoother the output will be. The value for w_q is set to 0.002, to cope with the possibly burst behavior of the instantaneous queue-lengths.

4. Experimental Results

The performance evaluation of the STP method was done through a series of simulations which uses the topology shown in Fig. 2, a bounded network (or domain) with one central node, *c1*, and three edge nodes: *e1, e2* and *e3*. Outside the network boundaries there are three source nodes: *s1, s2, s3*, and three destination nodes: *d1*, *d2* and *d3*. Inside the network there are three routes: *Route1*, *Route2*, *Route3*. See that node *c1* acts as an exit node for *Route2*. Links inside the network have a bandwidth of 3*Mb/s*, for the operating Class. The links connecting the bounded network with the outside nodes have a bandwidth of 100*Mb*/*s*. The propagation delay of the links is 0.05ms. *Route1* and *Route3* intersect at the output interface of *c1* (going to *e2*). *Route2* intersects with *Route1* at the exit interface of *e1* (going to *c1*). *Route1*, then, suffers from two intersections which puts it in disadvantage against *Route3* which is not affected by *Route2*. In the experiments the situation is that *Route3* increases its traffic so that *Route1* is affected with this increase.

Node *e1* was selected to be a *normal* node in every experiment, and node *c1*, in a group of experiments uses the STP method and in other group of experiments it does not, as it is explained in the next paragraphs (remember that this paper is aimed at

[3] See technical report *Rep-Tec-2-Equations-STP-Method.pdf* at http://ccd.cua.uam.mx/~amateos/.

evaluating the effect of the STP method in just one node in the network). With the STP method *Route1*-traffic passes through one queue and *Route3*-traffic passes through the other queue of the output interface of *c1* going to *e2*. *For the sake of simplicity-of-explanation the first queue of c1 can be referred to as the queue for Route1 and the second queue as the queue for Route3*.

The results of these experiments are compared to evaluate the STP method. Because of the random nature of experiments the results are taken from the mean values of 40 to 60 experiments, for each result.

4.1. Experimental Setting

The experiments were carried out with the ns-2 simulator [12], version 2.33, also using the DS tools included in the simulator [13]. In the STP method these tools were modified to add the WFQ packet-scheduling operation, with the implementation proposed in [14], which uses the WFQ scheduler [3]. Additional modifications were made to incorporate the possibility to change the weights of the scheduler in a dynamic form, in accordance with the STP method[4].

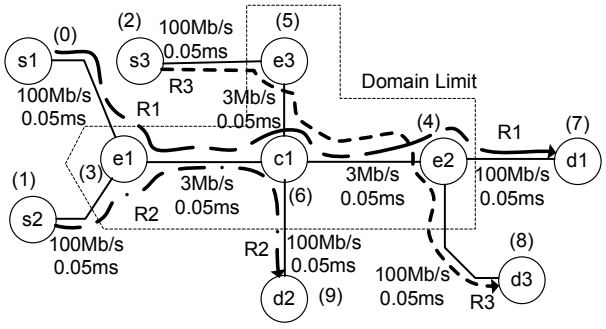


Fig. 2. The topology for the experiments has two routes: *Route1* traversing nodes *s1–e1–c1–e2–d1*, and *Route3* traversing nodes *s3–e3–c1–e2–d3*. The routes intersect at the output interface of node *c1* and separate at node *e2*. *Route1* and *Route2* intersect at the output interface of node *e1*.

The Pareto sources are *On/Off* with 250*ms* on/off duration, with 68 Kb/s during *On* periods and 95-byte packets [15][5] with Pareto shape parameter of 1.7 (infinite variance). The CBR sources have 1500-byte packets with 256Kb/s[6]. It should be clear that these rates are small compared with those actually used in the Internet, but they

[4] See technical report *Rep-Tec-3-WFQ-Changing-Weights.pdf* at http://ccd.cua.uam.mx/~amateos/.

[5] A voice packet can have 67 bytes which would make a 95-byte IP packet adding 8 bytes of UDP header and 20 bytes of IP header. A reasonable voice mean-rate of 3.0 Kbytes/s in one direction makes a traffic mean rate of 34Kb/s (3 Kbyte/s x 8 bits/byte x 95 / 67 = 34 Kb/s – considering the headers' overhead), making 68 Kb/s during the On period of On/Off sources.

[6] The rate value is taken from information from [16].

are used for the purpose of the evaluation of the STP method, within the scope of this article, as indicated in section 3.1.

It is supposed that for a long time before the beginning of each experiment the routes have had the same traffic. At the beginning of every experiment *Route3* increases its number of sources (without knowing about the possible deterioration of the QoS inflicted on *Route1*).

Three different groups of experiments were carried out, depending on the buffer used in the output interface of *c1*. In the first group, called "case *q1*", there is just one queue at the output interface of node *c1* (going to *e2*), which handles the traffic of the two routes. The second group, called "case *q2*", uses the STP method and utilizes two queues at the output interface of node *c1*, one corresponding to each one of *Route1* and *Route3*. The parameters employed for case *q2* are: $T = 1200s$, $P = 0.25$. The time τ is $1s$. The last groput, called "case *q2f*", also utilizes two queues at the output interface of node *c1*, one corresponding to each one of the routes, but the weight of each queue is fixed throughout the duration of the experiment.

For a *P* parameter value greater than 0, a *q2* case with parameter *T* being close to 0 should offer results similar to those of the *q1* case. A *q2* case with parameter *T* being close to infinite should offer results similar to those of a *q2f* case. All the other queues in all the nodes are Droptail. The queue lengths are big enough such that the total observed packet loss rate is always less than 2% in every experiment. In every experiment *Route1* and *Route2* have almost the same number of sources. *Route1* has 3 CBR sources + 17 Pareto sources, and *Route2* has 3 CBR sources + 16 Pareto sources. These numbers of sources are big enough to cause average packet-delays near to $3ms$ at the exit interface of node *e1*, without causing packet congestion losses.

Route3 initiates each experiment with 3 CBR sources + 8 Pareto sources. The initial traffic-share of *Route1* and *Route3*, at the exit of node *c1*, is: 0.5641 and 0.4359, respectively. These values are also the initial weights used for the queues of *c1* going to *e2*, for the *q2* case, which correspond respectively to *Route1* and *Rotue3*. Remember that the weights given by the STP method to intersecting routes in an output interface of a node (node *c1* in this case) tend to be the same as the bandwidth proportion that the routes take at that interface, in the long term. For the *q2f* case these values are taken as the fixed weights for the two queues of *c1* going to *e2*.

Each row of Table 1 shows the number of sources of *Route1* and *Route3*, their rates and the bandwidth percentage share at the exit interface of node *c1* going *e2* for every experiment. +PAR means the number of Pareto sources increased in *Route3* at the beginning of the experiment, for example, in the second row the total number of Pareto sources of *Route3* is 8 + 1 = 9). The heading –%R1 indicates, for a given row, the bandwidth percentage deterioration of *Route1* with regard to that which this route has in the first row. The queue-weight deterioration for a route with the STP method can not be bigger than the traffic-share deterioration of that route within a time-duration *T*). For example in the second row %R1 = 55.62 so that 1–55.62/56.41 = 0.01405 (1.41%). The maximum traffic-share deterioration for *Route1* shown in the table is 13.55%.

Table 1. Number of sources of *Route1* and *Route3*, their rates and the bandwidth-percentage share at the exit interface of node *c1*, going to *e2*, for every experiment.

CBR	CBR	PAR	PAR	+PAR	Kb/s	Kb/s	Kb/s	%	%	–%
R1	R3	R1	R3	R3	R1	R3	R3+R1	R3	R1	R1
3	3	17	8	0	1346	1040	2386	43.59	56.41	0.00
3	3	17	9	1	1346	1074	2420	44.38	55.62	1.41
3	3	17	10	2	1346	1108	2454	45.15	54.85	2.77
..	..	..	..							
3	3	17	16	8	1346	1312	2658	49.36	50.64	10.23
3	3	17	17	9	1346	1346	2692	50.00	50.00	11.37
3	3	17	18	10	1346	1380	2726	50.62	49.38	12.47
3	3	17	19	11	1346	1414	2760	51.23	48.77	13.55

4.2. Initial Results.

Fig. 3 presents the weight values for the *q2* case of the output interface in node *c1*, for four different values of *P*, through a period of time $T = 1200s$ in a special group of experiments where the traffic of *Route3* is bigger than that of *Route1*[7].

In the case of $P = 0.05$, at $t = T = 1200$ the weight of the queue attending *Route1* should be $0.564 * 0.95 = 0.5358$, and the experiments offered a weight value of 0.5291. For $P = 0.15$ this weight should be of $0.564 * 0.85 = 0.4794$, and the experiments offered a weight value of 0.4661. For $P = 0.25$ the weight obtained should be $0.564 * 0.75 = 0.4230$ and the experiments offered a weight value of 0.41. All these results are close to the projected ones.

4.3 Gain Results.

In order to assess the benefits of the STP method, this paper uses a figure of merit which consists of an overall gain for each route, for each experiment. This gain rewards with one point for every packet that traverses its route within the delay-limit of the route, and penalizes with ten points otherwise.

The first scenario considers experiments of short duration, 120*s*, for cases *q1* and *q2f*, and a delay limit is 18*ms*. The reason to make this scenario is to observe the behavior of these two cases to compare their results with those of the *q2* case, made in the next scenario. The left side of Fig. 4 shows the gains obtained for *Route1* and *Route3* (*Gain1* and *Gain3*) as a function of the number of sources added to *Route3*. Every point in the figure shows the gain obtained in each 120*s*-experiment. The upper abscissa axis shows the number of sources added to *Route3* in the experiment, and the lower abscissa axis shows the resulting rate in *Route3*. The gain in each experiment is normalized dividing it by the duration of the experiment in the simulation.

[7] *Route3* augments 30 Pareto sources at the beginning of the experiment time. Regarding Table 1, *Route1* has 40% of the traffic and *Route3* has 60% (only for this experiments the bandwidth of *c1-e2* is 4[Mb/s]).

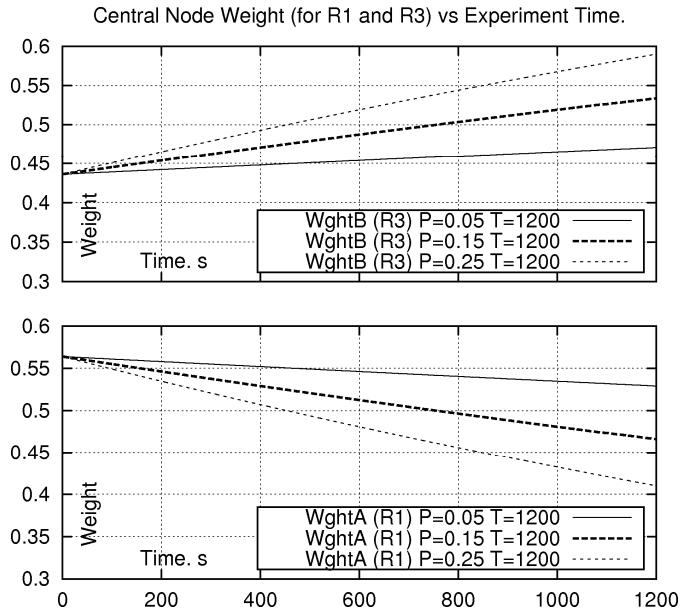


Fig. 3. Experimental results of weight-functions for the *q2* case, with different values of *P* parameter. Label "Wght A (R1) P=0.15 T=1200" stands for weight of *Route1* for the *q2* case with parameters *P*=0.15 and *T*=1200.

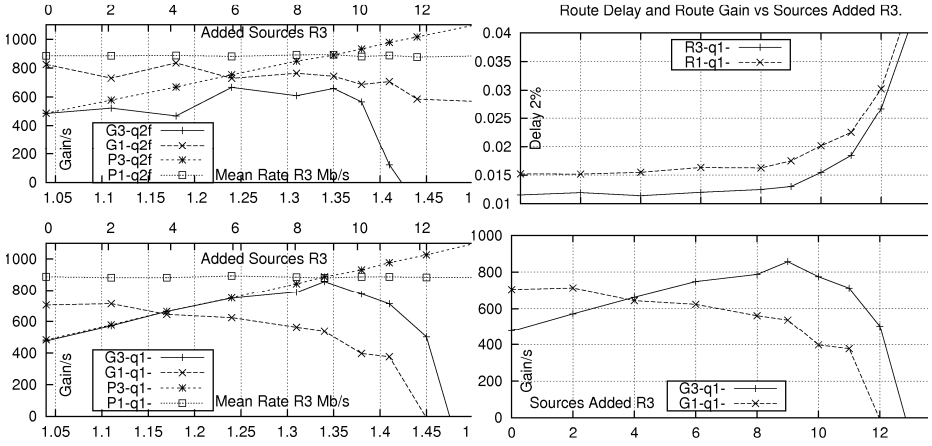


Fig. 4. The left side of the figure shows the gains of routes against number of sources added in *Route3*, for cases *q1*, and *q2f* with a delay limits of 18*ms*. Label "G3-q1" stands for *Gain3* in the *q1* case, and label "P1-q2f" stands for stands for number of packets for *Route1* for the *q2f* case. The meaning of the other labels is similar. The right side of the figure shows the gains and the delays (in *s*) of the routes for the *q1* case.

For the *q1* case *Gain3* has a maximum value at 9, decreasing afterwards. The admission process of *Route3* should avoid allowing an increase of more than 9 sources. *Gain1* deteriorates with the increase of sources in *Route3*. For the *q2f* case it is also clear that the admission process of *Route3* should neither allow more than 9 sources. From 0 to 9 sources, for the *q1* case, *Gain1* goes from 704 a 537 points, representing a loss of (704-537) / 704 = 27.7%. For the *q2f* case *Gain1* goes from 825 to 742 points, representing a loss of (825 – 742) / 825 = 10%. It is clear that the *q2f* case protects *Route1*. It is observed that the *q1* case is a good option when the available bandwidth is big compared to the traffic in the routes[8].

The right side of Fig. 4 shows the gains and the delays of the routes for the *q1* case (the delays of the *q2f* case are not shown because of space limitations in this paper). The delays are computed as the 98th percentile of the delay observed by the packets traversing their corresponding route. In other words, the delay of the route is the time exceeded by the delay of only the 2% most delayed packets. The *q2f* case is not appropriate for networks where the traffic can change in the routes and where there is no reason to give fixed protection to routes.

The second scenario is one using the *q2* case with $P = 0.25$ and $T = 1200s$, with experiments of 1200*s* duration, where it is supposed that the change in traffic at *Route3* is at $t = 0s$ of the experiment, with no further change in traffic during those 1200*s*.

It is also supposed that the traffic of the routes has been the same for a long time before time 0*s* of the experiments, so that as the STP method tends to give to the weight of each queue the same value as its relative traffic-share, then the initial traffic-share values of *Route1* and *Route3* are reflected in the initial values of the weights of the queues of *c1*, which are: 0.5641 and 0.4359, for *Route1* and *Route3* respectively. The gains are evaluated at the ending part of each subsequent interval of 120*s* duration (that is the gain at time 1200*s* is evaluated from 1080 to 1200*s*).

The left side of Fig. 5 shows how the delay in the *q2* case augments with time in every experiment for *Route1* (and decreases for *Route3*). The right side shows how the *Gain1* and *Gain3* of *q2* case tend to be equal to case *q1* at the ending of each experiment (at 1200*s*) and tend to be equal to *q2f* case at the beginning of each experiment (at 120*s*). With this, it can be seen that in the first part of each experiment the *q2* case protects *Route1* as case *q2f* does, and at the ending part of each experiment the *q2* case tend to protect *Route1* as case *q1* does.

It can also be shown, from graphics not presented in this paper, that the total gain, that is the sum of gains (*Gain1* + *Gain3*), is approximately the same for the three cases. The reason for this is that the schedulers for *q2* and *q2f* cases are work conserving, that is, the schedulers serve at full transmission rate whenever there is data to be served, ideally having a service, for the *q2* and *q2f* cases, as big as that of the *q1* case. This is clear as the STP method is not intended to maximize the total gain indicator.

The result of other experiments for *q2* scenarios with parameter $T = 120s$ (instead of 1200*s*), not shown in this paper, show that the gains and delays behave very similar to that of Fig. 5, but they change 10 times faster. This results could give an insight of how to select the parameter *T*, which could be on the order of the mean time between

[8] The standard deviation obtained for the gain was big (for most cases of increased-sources it was bigger than the mean value times 0.5. The reason of this is that the sources are of Pareto type.

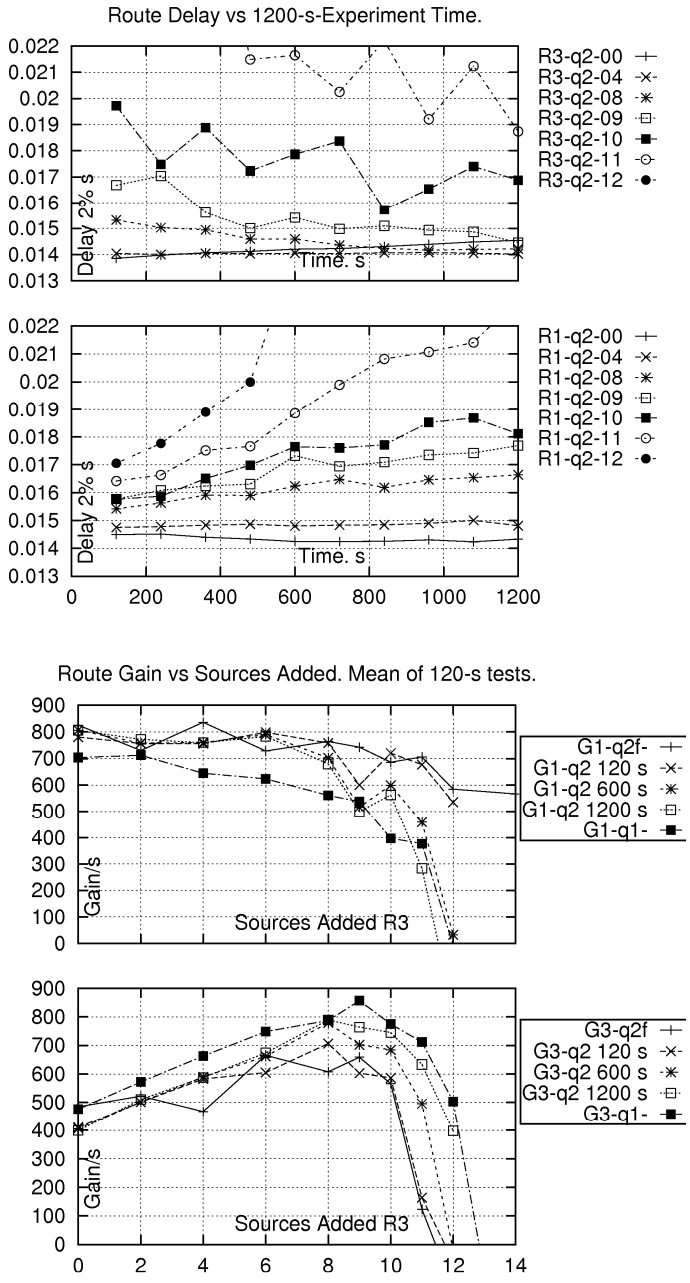


Fig. 5. The left side of the figure shows the delays of routes for *q2* case, throughout the experiment time. Label "R1-q2-04" stands for delay of *Route3* for 4 sources added in *Route3*. The right side of the figure shows the gains of routes for the *q2* case. Label "G1-q2 120 s" stands for *Gain1* of *q2* case at 120*s* of experiment. The other labels have similar meanings.

admission of flows, or the mean time of the life (duration) of the flows, in the network.

An example of a possible disadvantage of the *q2* case is, if at a certain moment *Route3* had a big weight and suddenly it decreased its traffic allowing for *Route1* to increase its traffic, and then, before *Route3* began to lose, substantially, its weight-share, it increased again its traffic, then *Route1* could suffer from this increase, even more than what it would do in a similar situation with a *q1* case.

5 Conclusions and Future Work.

This article addresses the mature problem bandwidth-sharing for QoS-aware networks with per-route distributed admission processes proposing a method which should help the network administrators to have confidence, at least within a time-window, about the amount of bandwidth they count with.

The method allows the nodes to operate autonomously, offering the possibility to attain global behaviors and being scalable. The objectives of the method are: 1- To protect, in the short term, the routes against traffic increments in intersecting routes, and 2- To dynamically assign, in the long term, a bandwidth share proportional to the average measured demands of the routes traversing congested links. As its scheduler is work-conserving this method should not impose a decrease of the transmission capacity in the network. The method's computations are separated by relatively long intervals of time (1*s*) so it should not cause noticeable performance-degradation in actual routers.

Under situation of little traffic the method is not better than the case of using single-queue configurations as there is no need to protect flows against each other.

Bandwidth reservation is still an appreciated form of service which can be a source of important economic compensations; however its overall performance depends heavily on the proper weight assignments which have to be setup by a central authority. On the contrary, the STP method is a simple method that dynamically adapts to the changing network conditions without the need of a central administration.

An analysis about how the STP parameters affect its performance is an important line of research.

References.

1 Centikaya, C., Knightly, E.: Egress Admission Control. In: IEEE INFOCOM 2000, vol. 3, pp. 1471-1480. (2000).

2 Cetinkaya, C., Kanodia, V., Knightly, E.: Scalable Services via Egress Admission Control. IEEE Transactions on Multimedia: Special Issue on Multimedia over IP, vol. 3(1), pp. 69-81. (2001).

3 Parekh, A., Gallager, R.: A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case. IEEE / ACM Transactions on Networking, vol. 1(3), pp. 346-357. (1993).

4 Neely, M.: Lecture Notes. EE 549 (from lecture 1 to 5). University of Southern California, http://www-rcf.usc.edu/~mjneely/ee549notes/EE549_Supplementary_Lecture_Notes_01.pdf

5 Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: RFC 2475. An Architecture for Differentiated Services. The Internet Society (1998).
6 Nichols, K., Blake, S., Baker, F., Black, D.: RFC 2474. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. The Internet Society (1998).
7 Kumar, V., Lakshman, T., Stiliadis, D.: Beyond Best Effort: Router Architectures for the Differentiated Services of Tomorrow's Internet. IEEE Communications Magazine, vol. 36(5), pp. 152-164. (1998).
8 Hui, T., Tham, C.: Adaptive Provisioning of Differentiated Services Networks Based on Reinforcement Learning. IEEE Transactions on Systems, Man, and Cybernetics, vol. 33(4), pp. 492-501. (2003).
9 Wang, H., Shen, C., Shin, K.: Adaptive-Weighted Packet Scheduling for Premium Service. In: IEEE International Conference on Communications'2001, pp. 1846-1850. (2001).
10 Li, C., Knightly, E.: Schedulability Criterion and Performance Analysis of Coordinated Schedulers. IEEE/ACM Transactions on Networking, vol. 13(2), pp. 276 – 287. ISSN: 1063-6692. (2005).
11 Floyd, S., Jacobson, V.: Random Early Detection Gateways for Congestions Avoidance. IEEE/ACM Transactions on Networking, vol. 1(4), pp. 397-413. (1993).
12 **ns-2** network simulator, http://nsnam.isi.edu/nsnam/index.php/Main_Page
13 Pieda, P., Ethridge, J., Baines, M., Shallwani, F.: A Network Simulator Differentiated services Implementation. Open IP, Nortel Networks, http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/DOC/DSnortel.pdf
14 Mrkaic, A. (tutor: U. Fiedler, supervisor: Prof. Dr. B. Plattner). Porting a WFQ Scheduler into ns-2's Diffserv Environment. Computer Engineering and Networks Laboratory (Insitut für Technische Informatik und Kommunikationsnetze). Swiss Federal Institute of Technology. Eidgenössische Technische Hochschule Zürich (2001).
15 Salman, S., Schulzrinne, H.: An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. Department of Computer Science. Columbia University. 2004, http://www1.cs.columbia.edu/~salman/publications/skype1_4.pdf
16 Skype, https://support.skype.com/en/faq/FA1417/How-much-bandwidth-does-Skype-need;jsessionid=5A4C03A96FCF4A2AE9CE5D72D15093CD?frompage=search&q=how+much+bandwidth+for+a+call&fromSearchFirstPage=false