A Method with Node Autonomy to Preserve QoS in Networks with Per-Route Admission

A. Mateos-Papis

División de Ciencias de la Comunicación y Diseño.
Universidad Autónoma Metropolitana. Unidad Cuajimalpa.
México, DF., México

ABSTRACT
This paper presents a method called the Short Term Protection (STP) which is applied to fixed networks where flows[1] are constrained to follow specific routes, where the admission-process is distributed, per-route, and where the traffic is sensitive to delay (like MPEG4 traffic). The share of bandwidth of any route is protected, in the short-term, against the traffic-increment of its intersecting-routes. In the long term, in every congested link, the share of bandwidth between the routes that intersect at that link is proportional to the average relative measured demands of those routes. The nodes act autonomously, without central administration. This method is expected to: 1) help network administrators to have confidence, within a time-frame , about the amount of bandwidth they count on for every route; 2) allow a simple bandwidth management in the network, with prospect to be scalable to at least tens of nodes. This paper presents the general and detailed operation of the method, the evaluation of the method, by simulations, including a comparison with other method, a discussion of a possible scenario of application, conclusions and possible paths for further research.

Keywords: quality of service (QoS); bandwidth sharing; distributed traffic control.

RESUMEN
Este artículo presenta un método llamado short term protection -STP- (Protección a Corto Plazo) que se aplica a redes fijas donde los flujos[1] están restringidos a seguir rutas específicas, donde el proceso de admisión es distribuido, por ruta, y donde el tráfico es sensible a retrasos (como el tráfico MPEG4). La porción de ancho de banda de cada ruta está protegida, en el corto plazo, contra incremento de tráfico en rutas de intersección. En el largo plazo, en cada enlace congestionado, la proporción de ancho de banda entre las rutas que se interceptan en ese enlace es proporcional a las demandas relativas promedio medidas de esas rutas. Los nodos actúan de manera autónoma, sin administración central. Se espera que este método: 1) Ayude a los administradores de la red a tener confianza, por un periodo de tiempo, acerca de la cantidad de ancho de banda con la que cuentan para cada ruta; 2) Permita una administración simple en la red, con prospecto de ser escalable hacia al menos decenas de nodos. Este artículo presenta la operación general y detallada del método, la evaluación del método, por simulaciones, incluyendo una comparación con otro método, un diálogo sobre un posible escenario de aplicación, conclusiones y posibles rutas para investigación futura.

1. Introduction

In contrast with other works which, for optimization purposes, propose schemes to share bandwidth between the diverse classes in networks which implement Quality of Service (QoS), this paper proposes a method which manages the available bandwidth inside a single class of a QoS-implementing network, to help simplify the admission-process to this class. So, in order to facilitate the explanations in this paper, the concept of class is not used; the concept of available bandwidth of a class is expressed, simply, as the available bandwidth of a network, and the concept of flow-admission, to a class, is expressed as the flow admission to a network.

Networks with limited resources, which offer satisfactory QoS to their admitted flows, may

[1] A flow is a sequence of related packets that enter a network through the same source-node and leave the network through the same egress-node.

suffer from deterioration of the QoS offered, after increasing their traffic (maybe as a consequence of admitting new flows). This deterioration may be small, and consequently permissible, while there are still plenty of available resources in the network; but along with the increase of more and more traffic in the network this deterioration may grow to the point where it becomes intolerable.

The admission-process for a flow, to enter a network, implies the operation of a predictive evaluation to avoid granting admissions which could turn the offered QoS, in the network, unacceptable. A network where every flow is admitted, with the restriction of traversing the network through a specific route, may be called: a network with per-route admission. The QoS offered to flows, in this kind of networks, is expected to be easier to maintain, compared to the QoS offered in networks where flows may take different paths at different moments.

The admission-process to a network with per-route admission may be called a per-route admission-process. In this case, every admission decision is made for a specific route. A network with per-route admission, which does not implement bandwidth reservation in any one of its routes, allows for more flexibility in the use of its bandwidth, with the drawback that these routes may intersect one another.

For example, a portion of a network is represented in Figure 1 with three routes, $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$, that converge at node *x*, arriving, each one, to the node's input interface *A*, *B* and *C*, respectively, and sharing the bandwidth of the node's output-interface *D*. In this paper, this node and the routes are referred to as *intersection node x* and *intersecting-routes*, respectively. These routes converge, specifically, at the output-interface *D* of node *x* (it can be said that the routes converge at the link coming out of node *x* through the output-interface *D*). For the sake of clarity, the output-interface *D*, of node *x*, is called the *intersection output-interface D*.

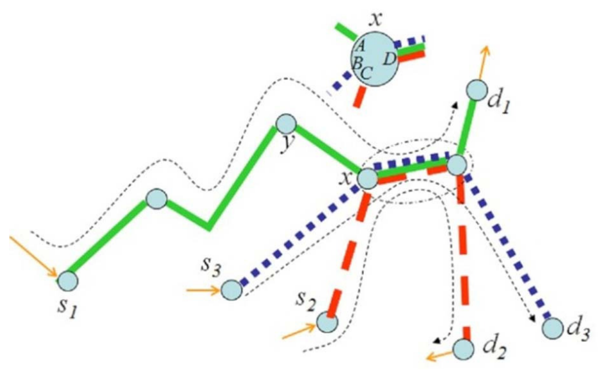

Figure 1. Routes $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$ intersect at the output-interface of node *x*, the intersection-node. This node has three input interfaces: *A*, *B* and *C*, as well as one output-interface *D* (which is an intersection output-interface) at which the traffic of the three intersecting-routes converge.

Whenever one route increases its traffic it may cause a decrease in the available bandwidth of its intersectingroutes, with a possible consequential deterioration of the QoS offered in those routes.

A per-route admission-process can be centralized or distributed. It is centralized if it is aware, at all times, of the traffic conditions in all the routes of the network, so that, before granting an admission to a route, it considers the possible effects of the admission in all the routes of the network.

A per-route admission-process is distributed if there is a separate admission-process for every route on the network, meaning that every admission-process is aware, only of the traffic conditions in its route, without considering the possible effects in the intersecting-routes when granting an admission. When evaluating if an admission to a route is granted or not, it is easier to consider the traffic conditions in just that route. This simplicity makes the distributed admission-processes to be more scalable, compared with its centralized-admission counterpart, but also, this simplicity is concomitant with the possible, inadvertent, deterioration of the QoS in intersecting-routes.

Within the type of networks which implement QoS without bandwidth reservation, those which implement per-route, distributed, admissions represent an important subset of networks, where the difficulty of the admission-process is alleviated.

This paper introduces a method called short-term protection (STP) method, which is intended to protect the routes of a network with per-route, distributed, admission, without bandwidth reservation, against the bandwidth decrease resulting from sudden increments in traffic in intersecting-routes. This method is easy to manage and adjusts in accordance to the bandwidth necessities of the routes. This method is prospected to be scalable to, at least, tens of nodes, and it is projected to be helpful for the network's route-administrators to have confidence, at least within a time- frame, about the amount of bandwidth they can count on.

The STP method is implemented at every output-interface of a core-node, on a network. Each node working with the method acts autonomously, without needing to be aware of the existence of routes, or of the existence of flows on the network, and without marking any packet. This method does not require any central admission authority on the network; still, this method is expected to attain a single extensive behavior on the networks where it is implemented.

The STP method is aimed to protect the bandwidth of routes, so the QoS parameter to evaluate this method is associated with the routes. To understand the operation of the STP method, think that a copy of the STP method operates in every output-interface of every node. At an output-interface, this method protects every queue at that interface, against the increase traffic in the other queues. Each queue represents a route intersecting at that node. So, the protection of the queue has an extending effect to protect the associated route (the input interfaces of nodes are considered to have no queuing delay [1]). At the ending, the protection offered by this method is reflected in the protection of the delay in the routes.

The QoS parameter of interest, in this paper, is the end-to-end delay in every route, so, the terms “route-delay” and “delay in route” refer to this end-to-end delay. Thus, the protection of the method is evaluated with regard to this delay. The delay in a given route depends on the available bandwidth at every node of the route, including those nodes where the route intersects with other routes.

To the best of the author's knowledge, there is no method which is oriented to work inside a class, which assigns, in the output-interfaces of every core-node, a different queue for every input interface of the node, as the STP method does (as it is explained in section 3).

The rest of the paper is organized as follows: Section 2 presents the motivation to do this work, and presents related work. Section 3 presents the conceptual framework of the STP method. Section 4 presents experimental results. Section 5 explores the scaling possibilities of the STP method. Section 6 presents a discussion of the scenarios where the STP method could be used; some application-considerations of the method; and a discussion of the STP method with regard to tendencies of QoS. Section 7 presents the conclusions derived from this paper, as well as some possible paths for further research. Finally, the demonstration of a theorem, related with the STP method, is given in Appendix A.

2. Background and related work

The STP method is motivated by some characteristics of the types of nodes used by the DiffServ model [2, 3] and by the network conditions proposed in [4, 5], where the networks considered have predefined routes, the admission-processes is per-route, distributed, and where the admission decisions are taken at the edges of the routes. These last two works highlight the problem of the traffic interactions between intersecting-routes, and they make reference to a method in which the routes can get information about the traffic of the other routes, to take admission decisions which do not unacceptably impact the intersecting-routes, with the cost of affecting the simplicity of their proposed per-route admission-process. However, the main research goal of these work is the admission-process itself, rather than the route intersection problem.

There are methods which obtain the most effective share of bandwidth between classes in a network, in terms of a global indicator to be optimized; however, these methods do not address the route-intersection problem inside a single class. A method like this is presented in [6], where the authors describe a method referred to as a Reinforcement-Learning algorithm, to attain the most effective share of bandwidth between different classes in a network, through the maximization of a global gain-figure (resulting from the addition of the gains of all the classes in that network).

This method uses a central bandwidth manager which uses a feedback scheme to periodically propose a new bandwidth allocation, which is expected to be better than the one being used. The new setting is applied and tested and the algorithm learns. The prospective characteristic of this algorithm is the reason why the authors identify it as “proactive”, in contrast with other reactive algorithms which are based on control-feedback principles. This method considers that there is contention between classes for bandwidth. The allocation of bandwidth for the diverse classes is done through the adjustment of the weight of each class, used by the scheduler located at each output-interface, in every node. There is a central bandwidth-manager, in charge of running the algorithm and evaluating the global gain.

Comparatively, the STP method considers the contention for bandwidth, not between classes but between intersecting-routes within a class. This method allocates bandwidth through dynamic queue-weight adjustments, at every output-interface of every node where two or more routes intersect; however, there is no central authority which allocates bandwidth or which evaluates a global gain: the nodes operate autonomously when allocating bandwidth.

A Coordinated-Schedulers method is presented in [7], which provides guaranteed service, in terms of end-to-end delay, without per-flow state in the network core. In this method, the nodes modify the value of each arriving-packet's priority-index (stored in its header), which represents the eligible time for the packet to be serviced. Any packet that was serviced late, at an upstream node, as a consequence of the excess of traffic in other classes contending for bandwidth in that node, arrives to the downstream node with a favorable priority-index value, in comparison with packets of other flows contending for service, and which might have been serviced early at their respective upstream nodes. This method provides natural coordination between the nodes to attain a global behavior and with the nodes operating autonomously, with the complexity that it requires the marking of every packet. This method is of special interest in the present paper (see subsection 4.5), since it can be used, for comparison purposes, applied to protect a route against intersecting-routes within a single class.

In [8], a scheduling algorithm is proposed, which intends to deliver fair bandwidth between classes and enhance the bandwidth utilization on a network offering QoS. In the scheduler, this method adjusts the weight of the queue of every class, based on the queue-length. A class gets more bandwidth if the size of its queue is bigger than the size of the other queues. Each node operates autonomously.

In [9], a method is proposed, which adaptively adjusts the weights of a weighted packet-scheduler, such as a weighted-round-robin scheduler, in every core-node of a network, to protect the Premium Service against the transient burstiness of the Expedited-Forwarding per-hop behavior (PHB), in the differentiated service architecture. The weights adjust according to the average queue-sizes. The nodes operate autonomously and cooperate to obtain QoS indicators, achieving low loss rate, low delay and delay jitter for the premium service.

In [10], a scheme with two parts is presented. The first part is an algorithm which operates autonomously at every core-node. The algorithm changes the weights of the per-class work-conserving WFQ (weighted fair queuing) scheduler, at the time of every packet-arrival. With a PGPS-like (packet-by-packet generalized processor sharing) method, the algorithm calculates the packet attention-time (*Next*(*t*)), and if the waiting-time is longer than that calculated for the last packet arrived to the queue, then the queue-weight is increased. The purpose of the algorithm is to maintain dynamic fairness according to the delay status of every queue. The second part of the scheme is a flow shaping algorithm operating at every class-queue of the edge-nodes, to maintain fairness between different flows. This algorithm has an excessive work-load calculating weights at the time of every packet arrival.

Finally, in [11], a sophisticated scheme for operation in a DiffServ (differentiated services) network is presented. This scheme includes a self-adaptive algorithm operating autonomously at every core-node, which controls local delay and loss values, by adjusting buffer sizes, queue-dropping thresholds and weights of the per-class weighted schedulers of the node, with the objective of maintaining delay bounds, differential loss bounds, and bandwidth priority assurances, across the service classes throughout the core network.

In order to attain global QoS objectives, all these methods may allow sudden impacts on flows "already admitted" which have not increased their traffic. Besides, all these methods focus on the share of bandwidth between classes.

3. The STP method

This section presents the operation of the STP method. The general scenario for this method is presented in Figure 1. Let us consider that node *x* works with the STP method. A node like this has, at every one of its output-interfaces, one queue for every one of its input interfaces (considering that the queues in a node are formed only at its output-interfaces [1]). Specifically, node *x* would have three queues at its output-interface *D*, one for every one of its three input interfaces. Each queue of the intersection output-interface has a weight which may change, slowly, according to an updating algorithm which periodically compares the traffic coming from each input interface, through the evaluation of the average length of each queue. With this algorithm, the queues at the output-interface compete for bandwidth.

3.1 Initial assumptions, general procedure and nomenclature of the method

This subsection explains the operation of the STP method, implemented in one intersection output-interface of a node, where it is supposed that there are *N* queues, where $N \geq 2$. At the beginning of the operation of the method the queues are empty and all the weights have the same value.

The method calculates the average length of every queue each time a packet arrives or departs from that queue. This average length (*avg*) is computed with the use of Equation 1 (as it is done in [12]), where w_q is the averaging parameter, and *q* is the instantaneous queue-length.

$$avg = (1 - w_q)\, avg + w_q\, q \qquad (1)$$

This equation acts as a low-pass filter. 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 burst behavior of the instantaneous queue-length.

Besides the calculations of the average-lengths of the queues, the method makes its main bulk of calculations at the ending part of every time-interval of size τ (for simplicity it is called "a time interval τ") the first time-interval beginning at time t_0. The time interval τ should be sufficiently big as to be able to observe many arrivals and departures of packets. In this paper $\tau = 1(s)$.

These calculations have the objective to obtain a new set of weights for the queues. The beginning of the ending part of each time interval τ is represented with $(t_0 + (r + 1)\tau)^-$ (meaning: just before time $(t_0 + (r + 1)\tau)$), where *r* is an integer such that $r \geq 0$.

The time-interval required to complete this bulk of calculations is very small, compared with τ, so, in the algorithm of the method it is considered that these calculations take no time[2] and begin and finish at time $(t_0 + (r + 1)\tau)^-$.

Only at time $(t_0 + (r + 1)\tau)$, the new set of values, recently computed for the queue-weights, replace the set of current queue-weights (those that were in use in the time interval τ that just ended). There may be a small time to wait before doing this updating in the scheduler, as required by the normal operation of the scheduler. So, the weights used by the scheduler are fixed from time $(t_0 + r\tau)$ to time $(t_0 + (r + 1)\tau)^-$, and the new weights take effect at time $(t_0 + (r + 1)\tau)$. The weight of queue *i*, at time *t*, is written as $\phi_i(t)$.

[2] The STP method should take less than 200 floating-point operations, which are calculated in several nanoseconds by modern processors.

In this paper the STP method uses a WFQ scheduler [13] (also called packet-by-packet generalized processor sharing), but the method is not intended to be restricted to this kind of work-conserving scheduler.

At the beginning of every bulk of calculations, the method checks if all the queues are empty. If so, the time resets to t_0, leaving the queue-weights as they were. If at least one of the queues is not empty, then the calculation of a new set of values, for the queue-weights, takes place.

In the calculations for every queue *i*, the method computes an indicator (represented with I_i –see Equation 9), to compare the current weight of the queue with its current relative average-length. The term "relative" means that the average length of the queue is divided by the sum of the average lengths of all the queues of the intersection output-interface. The weight of every queue is decreased if its relative average length results smaller than its current weight; otherwise, that weight is increased. In this method, every weight-change is always very small, and always all the queue-weights add up to 1. The weight-increment of queue *i*, from time $(t_0 + r\tau)$ to time $(t_0 + (r + 1)\tau)$, is written as $\Delta_i(t_0 + r\tau)$, such that:

$$\Delta_i(t_0 + r\tau) = \phi_i(t_0 + (r+1)\tau) - \phi_i(t_0 + r\tau) \quad (2)$$

The average length of queue *i*, at time *t*, is written as $Q_i(t)$.

Whenever, after a calculation for updating the weights, the indicator I_i indicates that queue *i* has to decrease its weight, the value for its decrement is calculated as:

$$\Delta_i(t_0 + r\tau) = f\frac{\tau}{T}\phi_i(t_0 + r\tau) \quad (3)$$

In Equation 3. *T* is a period of time such that T/τ is an integer, T/τ >> 1, and *f* is a negative factor which causes the value of $\Delta_i(t_0 + r\tau)$ to be negative too. If from time $(t_0 + r\tau)$, queue *i* obtained T/τ consecutive results indicating that it has to decrease its weight, then it would hold that:

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) - \phi_i(t_0 + r\tau) \approx \phi_i(t_0 + r\tau)\left(e^f - 1\right) \quad (4)$$

The proof of Equation 4 is given in Theorem 1, in Appendix A, which makes use of Equations 2 and 3.

In Equation 4, it is useful to make *f* to be equal to $f(P) = ln(1 - P)$, where *P* is a small positive constant, such that $P > 0$ and $P << 1$. So, given that $1 - P < 1$, then $f(P) = ln(1 - P) < 0$, such that the value of $\Delta_i(t_0 + r\tau)$ is negative. Then Equation 4 turns into Equation 5.

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) - \phi_i(t_0 + r\tau) \approx \phi_i(t_0 + r\tau)\left(e^{ln(1-P)} - 1\right) = -P\phi_i(t_0 + r\tau) \quad (5)$$

Equation 5 can be rewritten as:

$$\frac{\phi_i(t_0 + r\tau + T)}{\phi_i(t_0 + r\tau)} \approx (1 - P) \quad (6)$$

This ratio means that the weight of queue *i* has lost (*P* x 100)% of its value, in the big interval of length *T*. For example if *P* = 0.2, this ratio is equal to 0.8 and the weight loss is 20% of the weight value.

That is why the argument *P* is called "the loss factor".

With the above results, it is observed that the STP method enforces the weight-loss of the queue to be limited in relation with the factor *P*, in a big interval of length *T* (in the evaluation part –see Section 4– of this paper the value of *T* is 960(*s*)). Equation 3, then turns into Equation 7.

$$\Delta_i(t_0 + r\tau) = ln(1-P)\frac{\tau}{T}\phi_i(t_0 + r\tau) \quad (7)$$

Also, it is important to notice that, as τ/T << 1, then $|\Delta_i(t_0 + r\tau)|$ << 1, and $\sum_{\forall i}|\Delta_i(t_0 + r\tau)|$ << 1.

Note. For the calculation of Equation 7, when the value of P is very small, then $ln(1 - P) \approx -P$, and the calculation of the logarithm could be considered to be useless, but when the value of P is not too small (0.2 for example), then the calculation of the logarithm is necessary.

STP method steps

BEGINNING.

Require: At $t = t_0 + r\tau$

The values of the queue-weights were left intact, or they have just been calculated. In the last case, with the calculated weights, make sure[3] that:

$$\sum_{i=1}^{N} \phi_i(t_0 + r\tau) = 1 \qquad (8)$$
$$0 \le \phi_i(t_0 + r\tau) \le 1$$

To update the weights of the queues in use, with the new calculated weights, follow the next steps.

Require: At $t = (t_0 + (r+1)\tau)^-$

if all the queues are empty **then**

Make $t \leftarrow t_0$

The values of the weights are kept intact.

$r \leftarrow r + 1$

Go to the BEGINNING.

end if

At this stage at least one of the queues is not empty, so a calculation to obtain a new set of values, for the queue-weights, is about to commence.

In this recently finished interval of length τ, obtain the average lengths of the queues, and the values of the weights in use.

for $\{ i = 1 \rightarrow N \}$ **do**

[3] Each one of the calculated-weighs is divided by the sum of these weights. This corrects any possible, very small, deviation from 1, which the sum of the weights could have.

Calculate the indicator $I_i(t_0 + (r+1)\tau)^-$, as:

$$I_i(t_0 + (r+1)\tau)^- \leftarrow \frac{Q_i(t_0 + (r+1)\tau)^-}{\sum_{j=1}^{N} Q_j(t_0 + (r+1)\tau)^-} - \phi_i(t_0 + r\tau) \qquad (9)$$

Note. The second member of the assignment-expression 9 has two terms. The first term is the relative average length of queue i. The second term, $\phi_i(t_0 + r\tau)$, is the weight in use for queue i. So this indicator is a comparison of these two values.

if $I_i(t_0 + (r+1)\tau)^- < 0$ **then**

Queue i is marked to lose weight and it is considered to be in the set L of queues (where L stands for *Loss*).

$\Delta_i(t_0 + r\tau)$ is calculated with Equation 7, repeated, as an assignment expression, in expression 10.

$$\Delta_i(t_0 + r\tau) \leftarrow ln(1 - P)\frac{\tau}{T}\phi_i(t_0 + r\tau) \qquad (10)$$

end if $I_i(t_0 + (r+1)\tau)^- < 0$

if $I_i(t_0 + (r+1)\tau)^- = 0$ **then**

For this uncommon situation, queue i is considered to be in the set L of queues, and its weight-increment is assigned the zero value.

$$\Delta_i(t_0 + r\tau) \leftarrow 0 \qquad (11)$$

end if $I_i(t_0 + (r+1)\tau)^- = 0$

if $I_i(t_0 + (r+1)\tau)^- > 0$ **then**

Queue i is marked to gain weight, and it is considered to be in the set G of queues (where G stands for *Gain*).

No calculation of $\Delta_i(t_0 + r\tau)$ is made, but after the ending of this loop.

```
end if $I_i(t_0+(r+1)\tau)^- > 0$
end for $i = 1 \rightarrow N$
for $\forall i \in G$ do
    $\Delta_i(t_0+r\tau) \leftarrow \dfrac{I_i(t_0+(r+1)\tau)^-}{\sum_{j\in G} I_j(t_0+(r+1)\tau)^-}\left[-\sum_{j\in L}\Delta_j(t_0+r\tau)\right]$    (12)
end for $\forall i \in G$
$r \leftarrow r+1$
Go to the BEGINNING
END
```

Remarks of the STP method.

When adding both members of expression 12, $\forall i \in G$, expression 13 is obtained:

$$\sum_{i\in G}\Delta_i(t_0+r\tau) \leftarrow \frac{\sum_{i\in G} I_i(t_0+(r+1)\tau)^-}{\sum_{j\in G} I_j(t_0+(r+1)\tau)^-}\left[-\sum_{j\in L}\Delta_j(t_0+r\tau)\right] \tag{13}$$

Such that:

$$\sum_{i\in G}\Delta_i(t_0+r\tau) = -\sum_{j\in L}\Delta_j(t_0+r\tau) \tag{14}$$

So the sum of the weight-increments is equal to the negative of the sum of the weight-decrements.

4. Evaluation

This section presents the evaluation of the STP method with the use of the ns-2 simulator [14, 15]. To evaluate the STP method, the DS tools for ns-2 [16] were modified to add the WFQ packet-scheduling proposed in [17], with some amendments and also several additions, made in order to dynamically change the weights of the scheduler according to the proposed STP method.

Note that on the network used for the experiments, every input interface of the intersection nodes of the network carries the traffic of only one route which intersects at that node. That is why each queue of each one of the intersection output-interfaces is referred to as "the queue of" a specific route. It must be clear that these naming simplifications may not be valid in other general cases, for the STP-method application.

4.1 Construction of the network topology

The network topology used in the experiments is shown in Figure 2. This topology is a bounded network, similar to that found in the Figure 4 in [7].

The network has three core-nodes (c_0, c_1 and c_2), and eight edge-nodes (e_1, ..., e_8). In this network, the core-nodes are the only nodes where the routes intersect, so the core-nodes are the intersection-nodes. More specifically, the intersection output-interfaces of the core-nodes are those at the links: $c_0 - c_1$, $c_1 - c_2$ and $c_2 - e_8$. All the other interfaces of the core-nodes and all the output-interfaces of the edge-nodes use a single queue.

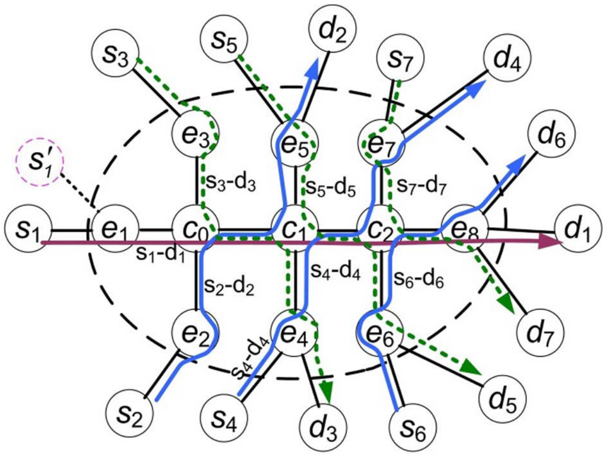

Figure 2. Topology of the bounded network used in all the experiments. The network has three core-nodes c_i, and eight edge-nodes e_i. Outside the network there are source nodes s_i, and destination nodes d_i. The network is traversed by seven routes, which intersect in groups of three routes. Even though the STP method is presented with its benefits, a route $s_1' - d_1$ is used to observe a weakness of the STP method.

Outside the network boundaries there are seven source nodes, s_1, ..., s_7, and seven destination nodes, d_1, ..., d_7. All the output-interfaces of these nodes have one queue.

In this network topology, there are seven routes (s_1-d_1, s_2-d_2, ... s_7-d_7). Route s_1-d_1, which is the longest one, traverses 7 nodes: s_1, e_1, c_0, c_1, c_2, e_8 and d_1. The other routes are smaller, in terms of traversed nodes. For example, Route s_2-d_2 traverses 6 nodes: s_2, e_2, c_0, c_1, e_5 and d_2, and Route s_3-d_3 traverses 6 nodes: s_3, e_3, c_0, c_1, e_4 and d_3. These three routes intersect at the intersection output-interface of core-node c_0 (going to core-node c_1).

Routes s_1-d_1, s_4-d_4 and s_5-d_5 intersect at the intersection output-interface of core-node c_1 (going to core-node c_2). Finally, routes s_1-d_1, s_6-d_6 and s_7-d_7 intersect at the intersection output-interface of core-node c_2 (going to core-node e_8). There are no other intersections on this network.

Only route s_1-d_1 intersects with two routes three times, that is, at the intersection output-interface of each one of the three core-nodes. All the other routes intersect with two routes at the intersection output-interface of just one core-node.

The links inside the network have a bandwidth of 30(*Mb*/*s*). The links connecting the edge-nodes with the outside nodes have a bandwidth of 100(*Mb*/*s*). Every link has 0.05(*ms*) of propagation delay.

Figure 2, also shows route $s_1'-d_1$. If this route were in use, it could not be distinguished apart from route s_1-d_1 by the STP method, at nodes c_0, c_1 and c_2, as both routes enter the network through node e_1. If the STP method were also used at the intersection output-interface of node e_1, then these two routes could be distinguished by the method at that node.

The conditions for the experiments are:

- As it has been highlighted, in every experiment of this paper, throughout all the experiment-time, each flow follows the same route. That is, there is just one flow considered for every route, therefore the packets of a flow are referred to as the packets of the corresponding route.
- The result of each experiment is the calculated-delay of the packets to pass through their respective routes (so this is an end-to-end packet-delay). The delay obtained for every route is the 98-percentile-delay, that is, this delay is the delay of the packet in the limit of the 2% most delayed packets to go across the route. This delay is calculated for every interval of 120(*s*), inside the experiment time-duration.
- The delay-calculations use the trace-files obtained from the tests done.
- The reported result of each experiment is the average of the results of 10 simulations, which is referred to as the result of the experiment[4].
- In the experiments of this paper, a flow is created from the traffic generated by the “traffic sources” (or simply the “sources”). A source is a place where traffic is generated in ns-2. A source is connected to a “source node” (like any one of the source-nodes, s_i, of Figure 2).
- This network was tested with UDP-based MPEG4 traffic (see subsection 6.2 for a discussion about source traffic). For this purpose, the algorithm to generate this kind of traffic [19] was added to ns-2. The average rate, with the selected settings for this algorithm, turned out to be 0.621(*Mb*/*s*) for each source. The lengths of the generated packets

[4] The result of every experiment is the mean value of the results of 10 simulations. This is an estimated mean. In order to locate the confidence interval to indicate the reliability of the estimate, the Bootstrap Percentile Confidence Interval method was used, using 1000 different re-samples of the 10 simulation-results. This method has been successful with a broad range of probability distributions [18]. The method indicated that, with a 90% probability, the real mean value lies within a window located from 6.67% below to 6.99% above the estimated mean value.

varied, but generally that length was of 1000(*bytes*).

- In comparison to the newer traffic-generation methods, which use test beds [20], the traffic generation method in this paper could be considered to be restrictive in terms of its lack of precision in its inter-packet generation times, and also in terms of its lack of responsiveness with regard to changes on the network conditions, including the network traffic; nevertheless, the following points may be noted:
    - As the buffers, of the nodes, used in the experiments of this paper are large, to avoid the loss of packets, as a result of queue saturation, the results in these experiments should not be sensitive to slight deviations in packet-generation patterns. The time-scales used in the experiments of this paper are, rather, capacity-planning scales.
    - The limitation of the lack of responsiveness indicated is handled, in this paper, by doing diverse experiments, all of them beginning with the same traffic conditions. Each experiment has 1560(*s*) of experiment time-duration, and at time 600(*s*), there is a change in the traffic conditions, to test the reaction of the STP method, and the reaction of other methods also tested in this paper, for comparison purposes. This change in the traffic conditions is caused when route $s_2 - d_2$ increases its traffic at time 600(*s*).
- In the experiments, the initial traffic in every route is big enough to make the waiting times of the packets, in the queues of the intersection output-interfaces, considerable. In this way, the delay-increment in the routes, caused by the traffic-increment in route $s_2 - d_2$, can be observed in the results of the experiments.
    - In every experiment, from 0(*s*) to 600(*s*) of the experiment-time, the flow of every route has 12 sources. Then, at time 600(*s*), there is a change in the traffic conditions, as the flow of route $s_2 - d_2$ increases its number of sources at that time. As indicated, the purpose of this increment is to observe the delay-impact on the other routes of the network.
- In this paper, diverse solutions to handle traffic are tested, besides the STP method which is presented in subsection 4.4. All these solutions are presented to compare their results.
- The experimental settings for the tested-solutions, that is, the topology and traffic situations of the experiments are the same for all the solutions, where four routes are selected to show their results. These routes are: $s_1 - d_1$, $s_2 - d_2$, $s_3 - d_3$ and $s_4 - d_4$.
- The experimental settings of the experiment of subsection 4.6 are an exception. In this subsection the experimental settings are a different from those of the other experiments.

4.2 Solution with one queue at the intersecting output-interfaces of the core nodes

In this solution all the intersection output-interfaces of the core-nodes have just one output-queue, a situation which could be referred to as “the traditional solution” to handle the traffic.

Figure 3, shows the results of the experiments for this solution. Subfigures 3-A through 3-D show the results of four different experiments. The difference, from one experiment to the following one, is the number of sources increased in route $s_2 - d_2$ at time 600(*s*). The increments are: 2, 4, 6 and 10 sources, for each experiment, at time 600(*s*). Label “**s1 02**” stands for “Route-delay in route $s_1 - d_1$, where, at time 600(*s*), route $s_2 - d_2$ increases its traffic in 2 sources”. The other labels have similar meaning.

In all subfigures it is observed that, at time 600(*s*), there is an increase of delay in every one of the routes $s_1 - d_1$, $s_2 - d_2$ and $s_3 - d_3$. These increments depend on the traffic-increase in route $s_2 - d_2$. In every subfigure the delay in route $s_1 - d_1$ is bigger than the delay in each one of the other routes, as route $s_1 - d_1$ is the longest one. Every route-delay results, mainly, from the delay at the intersection output-interface of core-node c_0

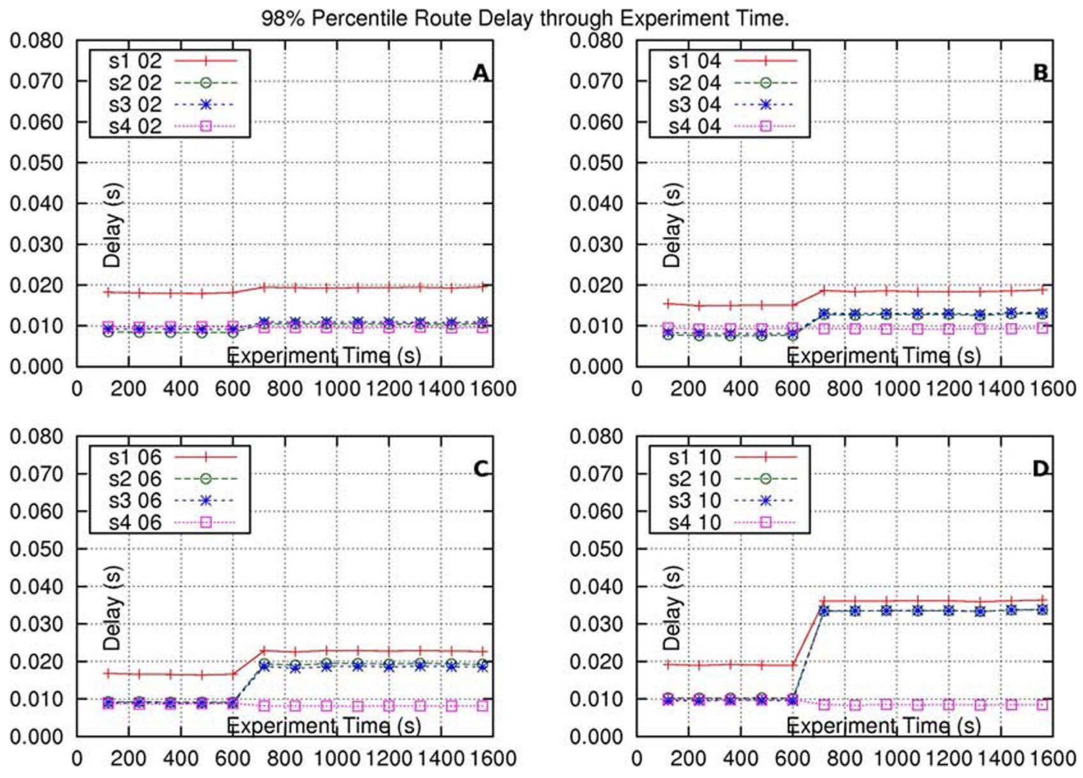

Figure 3. Delay-results of the experiments where each core-node has just one queue at its intersection output-interface. Label "**s1 02**" stands for "Route-delay in route $s_1 - d_1$" where at time 600(*s*) the route $s_2 - d_2$ increases its traffic in 2 sources". The other labels have similar meaning.

(going to core-node c_1). All these three routes suffer the same delay at this output-interface because the packets of these routes are placed in the same single queue of the interface. Note that the two lines corresponding to the delay in route $s_2 - d_2$ and the delay in route $s_3 - d_3$ are, practically, overlapped.

The delay in route $s_4 - d_4$ is not affected by the increment of traffic in route $s_2 - d_2$ because these two routes do not intersect.

In this, and in all the tested solutions, it is the route $s_2 - d_2$ (its administrators) itself which would have to evaluate, within its admission procedure, the amount of delay it can tolerate before admitting new traffic.

4.3. Solution with three queues, with fixed weights, at the intersecting output-interfaces of the core nodes

In this solution, at the intersection output-interface of every core-node, c_0, c_1 and c_2, there are three queues, one for each one of the three input interfaces of the node. Each queue has a fixed weight, equal to the weight of each one of the other two queues. This is another solution which can be referred to as "traditional".

The subfigures of Figure 4, show the results of the experiments for this solution. The meaning of the labels of these subfigures is similar to those of Figure 3. In all subfigures it is observed that, as a result of the increase of traffic in route $s_2 - d_2$, at time 600(*s*), there is a delay-increment in this route; the delay in route $s_1 - d_1$ and the delay in

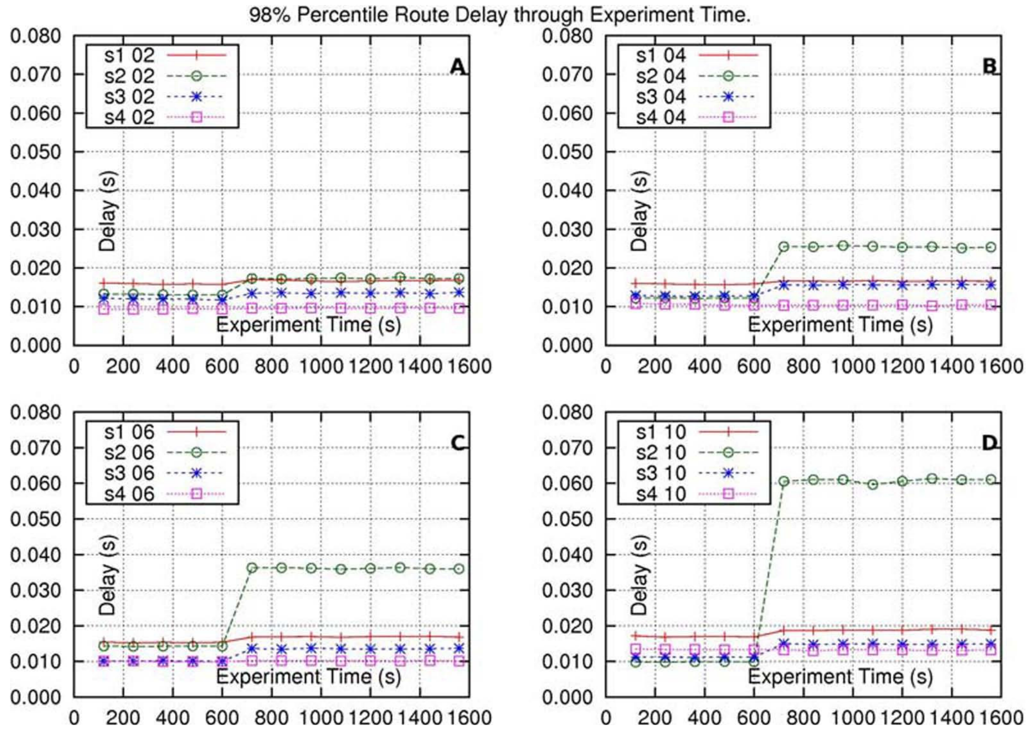

Figure 4. Delay-results of the experiments where each core-node has three queues at its intersection output-interface. The three queues have equal, fixed, weights. The meaning of the labels of these subfigures is similar to those of Figure 3.

route $s_3 - d_3$ are both barely affected; and the delay in route $s_4 - d_4$ is not affected.

In this case, the fixed weights of the queues of routes $s_1 - d_1$ and $s_3 - d_3$, at the intersection output-interface of core-node c_0, have protected these routes from the traffic-increase in route $s_2 - d_2$.

4.4 Solution with the STP method

In this solution, at the intersection output-interface of every core-node c_0, c_1, and c_2, the STP method operates with parameters T = 960(s) and P = 0.25.

For this solution, at the intersection output-interface of every core-node, there are three queues, each queue corresponding to each one of the three input interfaces of the node.

The subfigures of Figure 5 show the results of experiments for this solution. The meaning of the labels of these subfigures is similar to those of Figures 3 and 4.

It has been observed that, at time 600(s), the delay in route $s_2 - d_2$ increases as a consequence of the increase of traffic in this route, also at time 600(s). The delay-increment in route $s_2 - d_2$ is more pronounced depending on the amount of traffic increased in this route.

After time 600(s), the delay in route $s_2 - d_2$ decreases as a result of the gradual weight-increase of the queue of this route (remember that this queue is located at the intersection output-interface of core-node c_0). This weight increases as a result of the bigger average length of the queue.

The delays in routes $s_1 - d_1$ and $s_3 - d_3$ have a slight increase, at time 600(s), as these routes

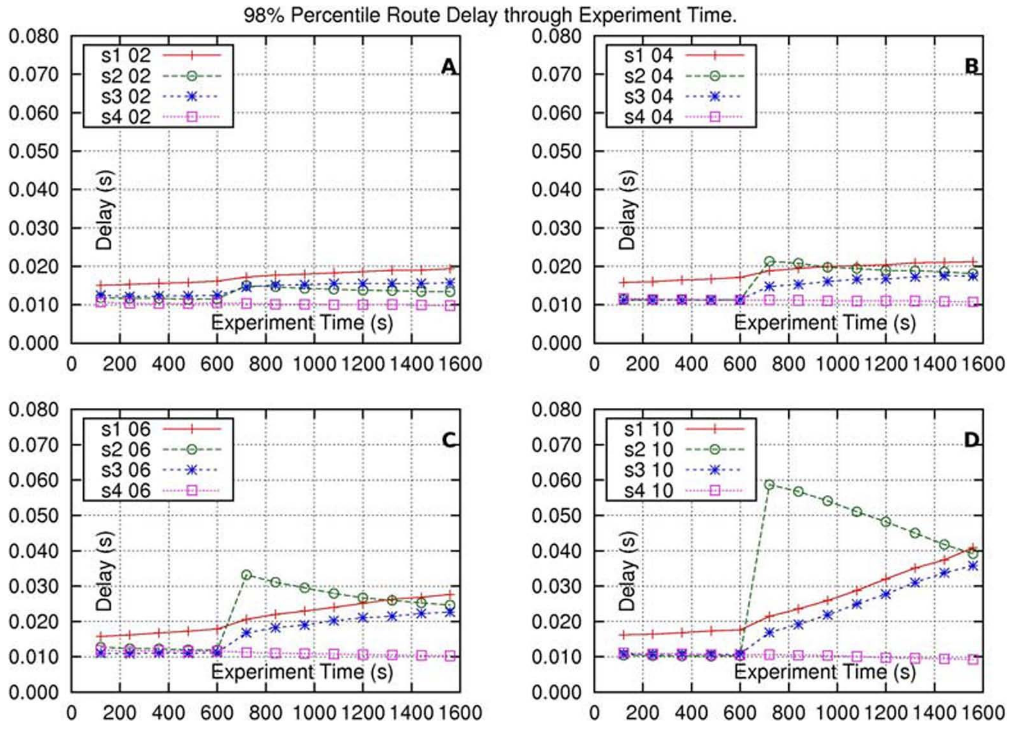

Figure 5. Delay-results of the experiments using the STP method at the intersection output-interfaces of the core-nodes. The meaning of the labels of this figure is similar to those in Figures 3 and 4.

are protected, to a void having a sudden loss of bandwidth and, consequently, a sudden increase in delay. From this time on, within the experiment-time, these delays increase slowly and steadily. The delay in route $s_4 - d_4$ is not affected, as this route does not intersect with route $s_2 - d_2$.

It is important to note that, although the STP method is acting at the intersection output-interface of each one of the three core-nodes, the method is effectively giving protection to route $s_1 - d_1$, only at the intersection output-interface of core-node c_0, because that is the only output-interface having a change in its average traffic (Section 5 shows results of other evaluations of the STP method where route $s_1 - d_1$ has more intersecting-routes increasing its traffic).

Figure 6, shows the weights of the three queues at the intersection output-interface of core-node c_0. Subfigures 6-A through 6-D show these weights for the cases corresponding to traffic-increments, in route $s_2 - d_2$, of 2, 4, 6 and 8 sources, respectively, at time 600(*s*). Note that the addition of the weights of the three queues is equal to 1, for every abscissa point of any one of the graphs. Notice also that in no case, a weight (every weight has a beginning value of 0.333) decreases its value in more than 25% at the ending of the experiment; in other words, no weight becomes as small as 0.333 x 0.75 = 0.25, at the ending of the experiment.

Before time 600(*s*), the tendency of the weight may vary, even though these results come from the average of several simulations; but from time 600(*s*) on, the tendencies are clear: the weight, corresponding to the queue of route $s_2 - d_2$, increases, and the weights, corresponding to the queues of the other two routes, decrease.

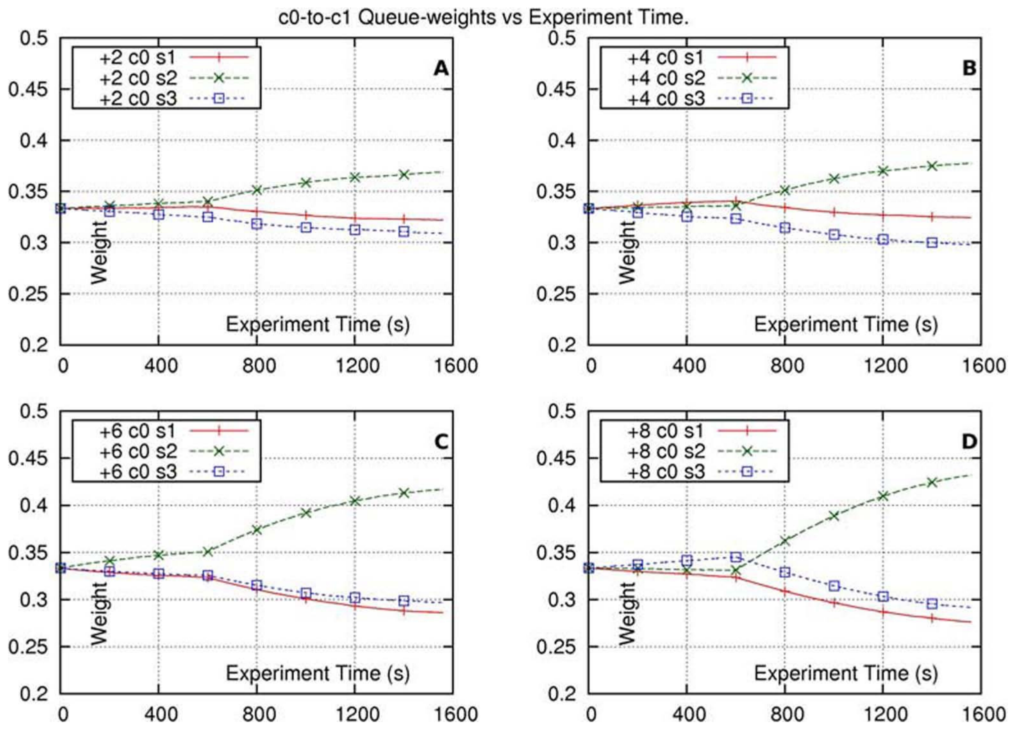

Figure 6. Weights of the queues, at the intersection output-interface of core-node c_0, in the experiments using the STP method at the intersection output-interfaces of the core-nodes. The label "**+2 c0 s1**" stands for "Weight of queue of route $s_1 - d_1$, at the intersection output-interface of core-node c_0, when route $s_2 - d_2$ increases its traffic in 2 sources, at time 600(s)". The other labels have similar meaning.

4.5 Solution with the method of coordinated multi-hop scheduling service

In the explanation of this solution the term "delay of flow" is preferred over the term "delay in route" (even though there is still one flow per route) due to the fact that this method regards very specifically to the flows on the network.

The method presented in this section is a special case of the Coordinated Multi-hop Scheduling Service (CMS): a work-conserving variant of the core-stateless jitter virtual clock (CJVC). This method was presented in [7] to protect flows of one class against the traffic increase of flows of other classes in a network. In the case where a flow is delayed as a consequence of the increase of traffic of another flow, which shares the same output-interface of a core-node, then the method helps the affected flow giving it a higher priority in the next core-node, where the flow goes through, allowing it to "catch up" (as [7] indicates). The purpose of this method is to try to maintain a low value in the overall delay of the flow, across its whole route.

In this paper, this method is applied on a network to protect flows in the same class, against the increase of traffic of flows in intersecting-routes. A general explanation of the form of operation implemented is as follows. At the ingress node, the method stores, in the header of each packet, a priority-index which is used by the output scheduler of the node, to serve the packet (the smaller the priority-index value the higher the priority). The value of this priority-index depends on the time the packet arrives at the ingress node (the smaller the time the smaller the index value). Then, every time the packet arrives at a downstream core-node, that node uses the priority-index value of the packet to recalculate a new priority-index value for that packet (basically adding the maximum time that may be needed to retransmit the packet), which is again stored in the packet. Then, the packet waits to be served at a specific queue corresponding to the packet's flow, located at the output-interface of the node. The scheduler of the output-interface serves that packet which has the smallest priority-index value (highest priority), among those packets which wait in front of each queue of the interface.

Specifically, Equation 15 shows the calculation of the priority-index of packet *k*, of flow *i*, at node *j*, where l_i^k is the packet size, r_i is the reserved bandwidth for flow *I* (l_i^k / r_i is the maximum time to retransmit the packet), t_i^k is the time of arrival of the packet at the first hop, and ξ_i^k is a slack-variable assigned to the packet. The unit of the priority-index is given in seconds.

$$d_{i,j}^k = \begin{cases} max\{t_i^k, d_{i,1}^{k-1}\} + \dfrac{l_i^k}{r_i} & j = 1 \\ d_{i,j-1}^k + \dfrac{l_i^k}{r_i} + \xi_i^k & j > 1 \end{cases} \qquad (15)$$

For this method, in this subsection, a set of experiments is presented where the flow which increases traffic is not able to harm the flows of intersecting-routes, whereas in subsection 4.6 another single experiment is presented where the flow which increases traffic is able to harm other flows.

For the first set of experiments r_i is equal to 2.4(*Mb*/*s*), meaning that the flow of every one of the three intersecting routes, at every core-node, has a reserved bandwidth of 8% of the capacity of the output link of the core node (of 30(*Mb*/*s*)); ξ_i^k is equal to 0.0003(*s*) for the flow which follows the long-sized route, $s_1 - d_1$; ξ_i^k is equal to 0.000375(*s*) for the flow which follows any one of the middle-sized routes, $s_2 - d_2$, $s_3 - d_3$, $s_4 - d_4$, $s_5 - d_5$; and ξ_i^k is equal to 0.0005(*s*) for the flow which follows any one of the small-sized routes, $s_6 - d_6$ and $s_7 - d_7$.

Subfigures 7-A through 7-D show the results of four different experiments, where the flow of route $s_2 - d_2$ increases its number of sources in 2, 4, 6 and 10, respectively, at time 600(*s*). As a result of these increments, it can be observed that, in terms of delay, the flow of route $s_2 - d_2$ has a sudden impact at 600(*s*); the flows of routes $s_1 - d_1$ and $s_3 - d_3$ are both barely affected; and the delay of the flow of route $s_4 - d_4$ is not affected. The reason for this "self-inflicted" delay-impact in the flow of route $s_2 - d_2$ is that the packet arrival-rate of this flow is bigger than the packet-arrival rate of the other flows. As each packet of a flow has a bigger priority-index value (representing less priority) compared to that of the previous packet, of the same flow, then, the priority-index values of the packets of the flow of route $s_2 - d_2$, queued at the intersection output-interface of core-node c_0, are bigger (representing less priority) than the priority-index values of the packets of the flows of routes $s_1 - d_1$ and $s_3 - d_3$, queued at the same interface. So the packets of these last two flows have advantage, when evaluated for service, with respect to the packets of the flow of route $s_2 - d_2$.

In this case, the flow packets of route $s_1 - d_1$ arrive at core-node c_1 having priority-index values not necessarily smaller or bigger than those of the packets of the flows of routes $s_4 - d_4$ and $s_5 - d_5$, so the method does not favor any one of these flows, at the output-interface of node c_1.

4.6 CMS-method solution where flow $s_2 - d_2$ has advantage over the other flows

This is the only solution which has an experiment with traffic conditions different from those of the experiments of the other solutions . The reason for this is to give the flow of route $s_2 - d_2$ the biggest advantage in this experiment. In this experiment, the results of a single simulation are observed.

The number of initial sources are 13, 13, 8, 14, 14, 0, 0, for the flows of routes $s_1 - d_1$ through $s_7 - d_7$, respectively. The flow of route s_2-d_2 increases its traffic, at time 600(*s*), in 6 flows. The reserved bandwidth for all flows is 6(*Mb*/*s*), except for the route flow of s_2-d_2, which has 9(*Mb*/*s*).

In this experiment, the values for ξ_i^k are smaller to obtain smaller priority-index values. These index values are equal to 0.00012(*s*), for the flow of the long-sized route s_1-d_1; and equal to 0.00015(*s*), for the flow of each one of the middle-sized routes s_2-d_2, s_3-d_3, s_4-d_4 and s_5-d_5; and equal to 0.0002(*s*), for the flow of each one of the small-sized routes s_6-d_6 and s_7-d_7.

CMS is very sensible to the amount of bandwidth reservation for flows. In this experiment, the flow of route s_2-d_2 has more reserved-bandwidth than the bandwidth reserved for each one of the other flows (the flows of routes s_1-d_1 and s_3-d_3), at the output-interface of node c_0. Looking at Equation 15, the priority-index values of the packets, of the flow of route s_2-d_2, have a smaller growing rate, from one packet to the next one, meaning that the tendency is to give smaller priority-index values (higher priority) to these packets, and thus, contributing to cause an important impact to the other two flows (the flows of routes s_1-d_1 and s_3-d_3), at the output interface of node c_0 when the flow of route s_2-d_2 increases its traffic.

In this case, it can be seen how CMS protects the flow of route s_1-d_1 against the increase of traffic of the flow of route s_2-d_2, by giving the flow of route s_1-d_1 a higher priority at the output-interface of node c_1, but, affecting, as a consequence, other flow, in an immediate form (the flow of route s_5-d_5 is the one which is affected -as it is shown subsequently).

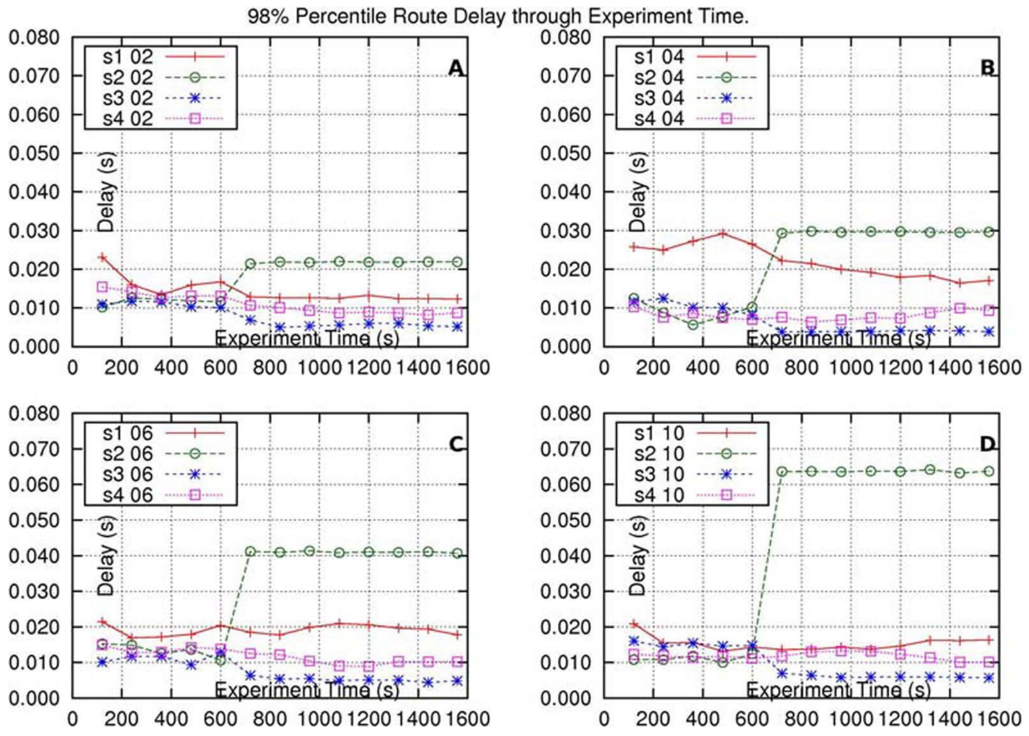

Figure 7. Delay-results of the CMS-method experiments. CMS being in its special case, the work-conserving variant of the CJVC method. This method is applied to a network with intersecting-routes, working with a single class. Label "**s1 02**" stands for "Delay of the flow of route s_1-d_1 where, at time 600(*s*), the flow of route s_2-d_2 increases its traffic in 2 sources". The other labels have similar meaning.

Subfigure 8-A shows the delay of the flow of each one of the routes s_1-d_1, s_2-d_2, s_3-d_3 and s_4-d_4. It can be seen that the flow of route s_1-d_1, at time 600(s), suffer of an increase of delay, and the flow of the route s_2-d_2 barely increases its delay as a consequence of its increase of traffic. The flow of route s_3-d_3 is almost not affected, in terms of its delay, because it has less traffic, and so the priority-index values of its packets are smaller (higher priority). The flow of route s_4-d_4 decreases its delay, and the reason is explored in the explanation of subfigure 8-C.

Subfigure 8-B shows the results for the delay of the flow of each one of the routes: s_1-d_1, s_2-d_2 and s_3-d_3, at the intersection output-interface of core-node c_0.

Subfigure 8-C shows results for the delay of the flow of each route: s_1-d_1, s_4-d_4 and s_5-d_5, at the intersection output-interface of core-node c_1. It can be seen that the flow of route s_1-d_1 has almost no delay throughout all the time of the experiment. The delay of the flow of route s_4-d_4 is roughly 43(ms), before time 600(ms), and it decreases to roughly 9.3(ms) after 600(ms). The delay of the flow of route c_5-d_5 is the one affected as it increases its delay from around 5(ms) to around 43(ms), at time 600(ms).

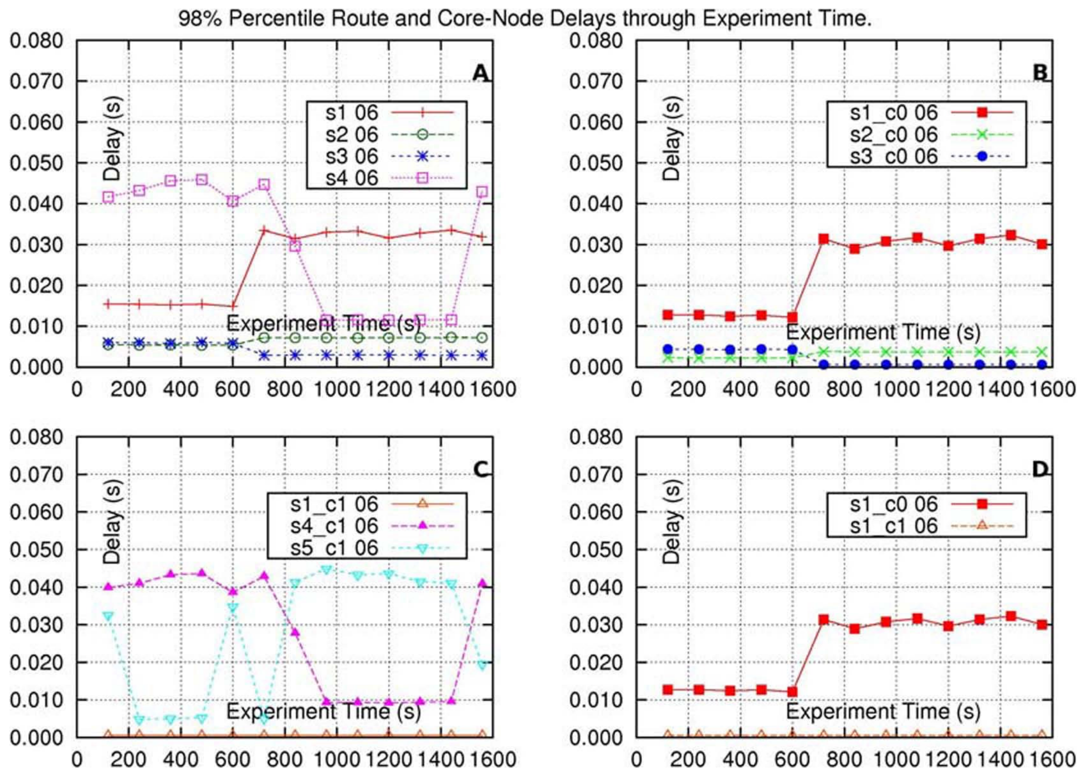

Figure 8. Delay-results of the CMS-method experiments with the special case of the work-conserving variant of the CJVC method. This method applied to a network with intersecting-routes, working within a single class. The flow of route s_2-d_2 has advantage over the other flows. Label "**s1 06**" stands for "End-to-end delay (route-delay) of the flow of route s_1-d_1", and Label "**s1_c0 06**" stands for "Delay, at the intersection output-interface of core-node c_0, of the flow of route s_1-d_1". Both labels also indicate that at time 600(s), the flow of route s_2-d_2 increases its traffic in 6 sources. The other labels have similar meaning.

This affectation is caused by: 1- the increase of traffic of the flow or route $s_2 - d_2$, 2- the protection of the flow of route $s_1 - d_1$[5], and 3- because even though the flows of routes $s_4 - d_4$ and $s_5 - d_5$ have the same number of sources, in this experiment the flow of route $s_4 - d_4$ has less traffic than that of the flow of route $s_5 - d_5$.

Subfigure *8-D* shows the delay-results for the flow of route $s_1 - d_1$, at the intersection output-interface of core-node c_0, and at the intersection output-interface of core-node c_1. The flow of route $s_1 - d_1$ has approximately 12.5(*ms*) of delay at the output-interface of core-node c_0 (to go to core-node c_1) before 600(*ms*), and after that time, the delay at that output-interface increases to approximately 30(*ms*). At the intersection output-interface of core-node c_1 (to go to core-node c_2) the delay of the flow of route $s_1 - d_1$ is at all times very close to 0.57(*ms*). So, the delay of the flow of route $s_1 - d_1$ is compensated by this method.

5. Scaling possibilities of the STP method

In order to explore the scaling possibilities of the STP method, six additional experiments were done, with mainly the same settings used for the evaluation of the solution with the STP method (subsection 4.4), but with a change in the value of parameter *P*.

The settings of these experiments are shown in Table 1.

[5] From subfigures 8-A and 8-D, it is important to notice that the result of adding the delay observed at the output-interface of node c_0, plus the delay observed at the output-interface of node c_1, for the packets of the flow of route $s_1 - d_1$, is not necessarily smaller than the delay observed for the packets of this flow to go end-to-end through this route. The reason for this is that the 2% most delayed packets to go across the intersection output-interface of core-node c_0 are not necessarily the same 2% most delayed packets to go across the intersection output-interface of core-node c_1.

In the experiments route $s_1 - d_1$ is subject to the increment of traffic, in 10 sources, at time 600(*s*), in one, two or three of its intersecting routes: $s_2 - d_2$, $s_4 - d_4$ and $s_6 - d_6$, as Table 1 indicates. In this Table, for every experiment, a row with an "x" indicates that there is an increment of traffic in the corresponding route, for example, in experiment #2, routes $s_2 - d_2$ and $s_4 - d_4$ increment their traffic, each one, in 10 sources, at time 600(*s*).

The first group of three experiments corresponds to parameter *P* = 0.25, and the results are given in Figures 9-A and 9-B. The second group of three experiments corresponds to parameter *P* = 0.10, and the results are given in Figures 9-C and 9-D.

Subfigure 9-A presents the delay in route $s_1 - d_1$ for experiments 1, 2 and 3 (corresponding to *P* = 0.25). It can be observed that, from time 600(*s*), the delay in this route grows, as more intersecting-routes increase their traffic.

Exper.	1	2	3	4	5	6
P	0.25			0.1		
$s_2 - d_2$	x	x	x	x	x	x
$s_4 - d_4$		x	x		x	x
$s_6 - d_6$			x			x
	Figs. 9-A and 9-B			Figs. 9-C and 9-D		

Table 1. Settings of the six experiments, to explore the scaling possibilities of the STP method.

Subfigure 9-B presents the delay in route $s_2 - d_2$ for experiments 1, 2 and 3 (corresponding to *P* = 0.25). It can be observed that, as routes $s_2 - d_2$, $s_4 - d_4$ and $s_6 - d_6$ do not intersect each other, the delay of route $s_2 - d_2$ is not affected if routes $s_4 - d_4$ and $s_6 - d_6$ increase their traffic.

Subfigures 9-C and 9-D are similar, respectively, to subfigures 9-A and 9-B, but the value of parameter *P* is 0.1 (10%). It can be observed that the delay-changes in routes $s_1 - d_1$ and $s_2 - d_2$ are smaller than those observed in subfigures 9-A and 9-B.

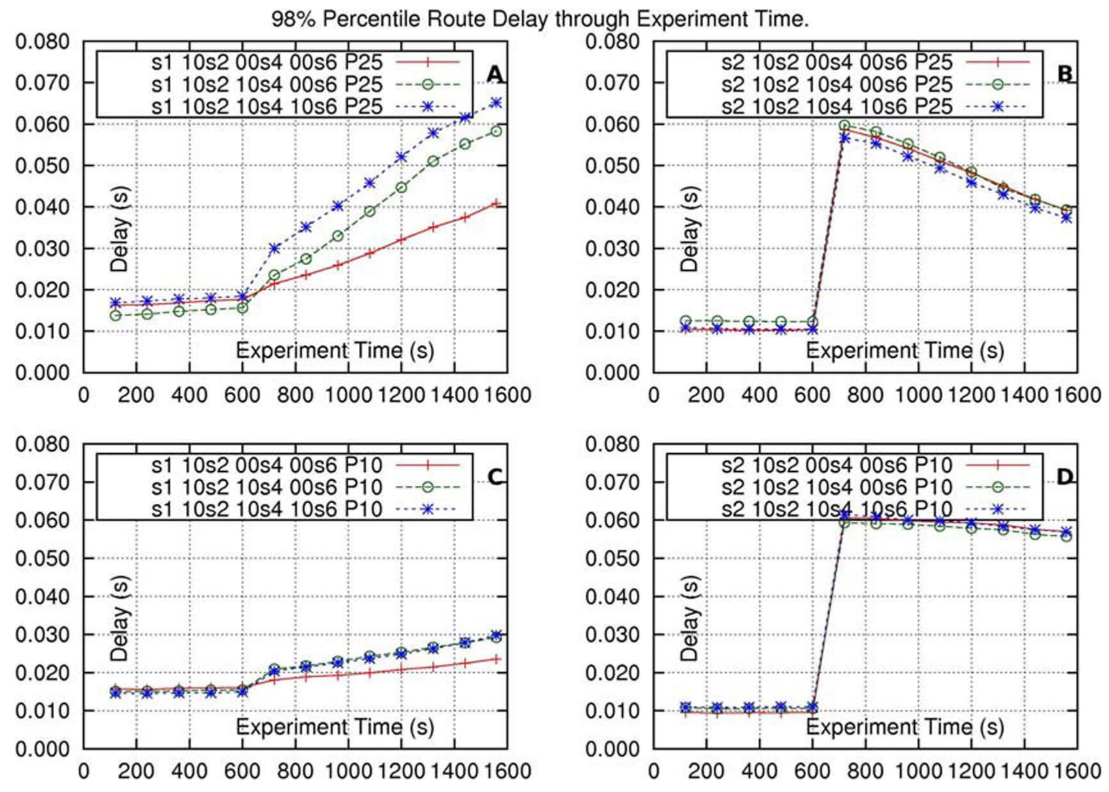

Figure 9. Delay-results of the experiments using the STP method at the intersection output-interfaces of the core-nodes. Label "**s1 10s2 10s4 00s6 P25"** stands for "Route-delay in route $s_1 - d_1$, where at time 600(s), each one of route $s_2 - d_2$, and route, $s_4 - d_4$ increase their traffic in 10 sources, and where P = 0.25".
The other labels have similar meaning.

Wrapping up, when a route has tenths of intersecting-routes, and a worst-case occurs in which many of those intersecting-routes synchronize increasing their traffic within a small window of time then, the impact on the intersected route could be severe. To limit this possible impact, the selection of the values for parameters *P* and *T* should be either: 1- A smaller value for parameter *P*, 2- A bigger value for parameter *T*, 3- A combination of both possibilities.

The relation about the time increment, of the delay in a route, as a function of the values of parameters *P* and *T*, and as a function of the number of the intersecting-routes, is still a matter of study .

6. Discussion

6.1. A scenario where the STP method could be used

The Internet has a basic design as a Best-Effort network, composed of thousands of different networks [21]. In the "middle-mile" Internet, internetwork data communication is affected by peering-point congestion between, possibly competing, networks; performance limitations of routing protocols (BGP); and unreliable networks.

One approach to use the Internet, without changes, to address quality issues, is the Delivery

Network. A delivery network is a virtual network over the Internet, which needs no changes to the underlying Internet, offering enhanced reliability and performance of delivery. One main objective of a delivery network [22] can be to minimize the long-haul communications in the Internet, where middle-mile bottlenecks are found (like peering points between networks). This is done through the use of many distribution servers, the use of application-layer multicast services, and the use of multiple alternate-paths across the middle mile of the Internet.

A scenario for the operation of the STP method can take ideas of operation of the delivery network. One scenario could be a delivery network devoted to the delivery of time-sensitive traffic, like that of video-conference. This network could span several underlying specific networks of Internet Service Providers (ISPs), with interconnection points which did not represent bottlenecks. The ISPs would jointly cover big business areas.

The network would have fixed-routes to connect to the networks of the clients. The nodes of the ISPs would run the STP method to give the required service to the delivery network. The admission-processes would be per-route, which would be in charge of verifying the fulfillment of the delay-limit expected in the route. There would be no bandwidth reservation in the routes and the output-interfaces of the nodes should have big-enough buffers to limit the amount of lost packets because of possible overload-situations.

Regarding the transport protocol suggested, there would be two variants of the scenario. For the first variant, it is interesting to observe the tendencies of use of UDP and TCP in the Internet. By the year 2009, TCP sessions were still responsible for most packets and bytes in the Internet, but in terms of flows, UDP was the dominant transport protocol [23] (mainly from P2P applications using UDP for their overlay signaling traffic). There is a report about UDP traffic, in a campus connection to the Internet, which states that UDP traffic has grown steadily, in the years 2008 to 2011 [24], to reach a share of 22% of the total traffic.

The use of UDP as the transport protocol is suggested for the first variant of the scenario. The reason for this is that the network of the scenario is devoted to delay-sensitive traffic, and it operates with admission-control. The main use of the network would be for the strict delay-sensitive traffic (like that of video-conference), although less strict delay-sensitive traffic could also be admitted (like that of streaming traffic originated from stored or live content distribution). The nodes of the underlying ISPs might still use the spare bandwidth to send low-priority traffic, like TCP traffic, in other class, using the best effort service.

For the second variant of the scenario, the transport protocol selected would be a TCP-Friendly protocol, like a form of DCCP [25] (suitable for flows with timing constraints such as streaming media or flows from or multi-player on-line games or telephony and video-conference). In this variant, TCP traffic could also use the delivery network, in the same class, but subject to the admission-process.

Regarding transmission capacity, as the considered schedulers are work-conserving, the STP method should not impose a decrease of the transmission capacity in the network.

6.2 STP-method application considerations

The first consideration, to be in mind, is that, under situations of little traffic on a network, the STP method is not better than the case of using a single queue at every intersection output-interface of the network.

Another consideration is that a possible problem of the method can be the situation where, in an output-interface, the queue of a given route "A" has the biggest weight of them all. This route may decrease its traffic, allowing the other intersecting-routes, at that output-interface, to increase their traffic. As the weight decrement of a queue is slow, route "A" could suddenly increase its traffic, while its queue still has the biggest weight. This could cause important problems to the other routes.

6.3. The STP method and the tendencies of QoS

Some modern traffic engineering approaches try to achieve near-optimal traffic distribution in the networks, without the use of overlay networks (networks like those with fixed routes), which are considered to be administratively costly as they may require the nodes to establish many logical connections. The optimal distribution functions used in these approaches do not observe the concept of QoS so, for these approaches, the adoption of a method like the STP method does not seem to be suitable. It seems that the issues of quality in these approaches should be addressed by application-layered solutions.

On the other side, other approach to QoS, called Quality-of-Service Routing [26, 27, 28], tries to identify, on a fixed topology, a feasible source-to-destination path that satisfies a set of QoS constraints, i.e., a path on which all links have sufficient resources to satisfy the diverse constraints, like delay, delay-jitter, cost, reliability, throughput. This approach does not include the concept of admission, but it rather tries to find a route for every "routing request". The routes on these networks have intersections, so, the STP method, applied to these networks, seems to be suitable.

7. Conclusions and future work

This paper addressed the problem of the intersection between routes, on networks admitting delay-sensitive flows, where each one of these flows is constrained to follow a specific route. The main contribution of this paper was to present an innovative method, called short term protection method (STP method), operating independently in every node of the network, oriented to be easily managed, with a prospect to be scalable to at least tenths of nodes, and oriented to attain a single extensive behavior in the network. The goal of the method is to help maintain the QoS in the routes of these networks, while these routes are subject to the increasing traffic in intersecting-routes.

In the results of the tests done, this method proved to be effective and with promising possibilities with regard to its scaling orientation. The STP method is compared to other method, oriented to protect flows against the increment of traffic in other flows, on a network. The benefits of the STP method are observed.

Future lines of research are: 1- Make the STP method become proactive (instead of reactive); 2- Study the way in which the STP-method's parameters affect its performance; 3 Integrate an additive increase/multiplicative decrease (AIMD) mechanism to the STP method, observing if the weight-loss of a route is due to the own decrease of traffic in the route, or to the increment of traffic in the intersecting-routes; 4- Study the STP method with more types of traffic.

Acknowledgements

The author appreciates the helpful discussions with Francisco García-Ugalde, José Incera-Diéguez and Fabián García-Nocetti. Also, the author appreciates the valuable comments of Francisco López-Fuentes and of the blind reviewers, with regard to the previous versions of this paper.

References

[1] V. Kumar et al, "Router Architectures for the Differentiated Services of Tomorrows Internet", IEEE Communications Magazine, Vol. 36, No. 5, pp. 152-164, 1998.

[2] S. Blake et al, "An Architecture for Differentiated Services", RFC 2475, IETF, 1998.

[3] K. Nichols et al, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, IETF, 1998.

[4] C. Cetinkaya and E. Knightly, "Egress Admission Control", in: 19th IEEE International Conference on Computer Communications, INFOCOM 2000, The Conference on Computer Communications, Tel Aviv, Israel, 2000, pp. 1471-1480.

[5] C. Cetinkaya et al. "Scalable Services via Egress Admission Control", IEEE Transactions on Multimedia: Special Issue on Multimedia over IP, Vol. 3, No. 1, pp. 69-81, 2001.

[6] T. Hui and C. Tham, "Adaptive Provisioning of Differentiated Services Networks Based on Reinforcement Learning", IEEE Transactions on Systems, Man, and Cybernetics, Vol. 33, No. 4, pp. 492-501, 2003.

[7] C. Li and E. Knightly, "Schedulability Criterion and Performance Analysis of Coordinated Schedulers", IEEE/ACM Transactions on Networking, Vol. 13, No. 2, pp. 276-287, 2005.

[8] M. F. Horng et al, "An Adaptive Approach to Weighted Fair Queue with QoS Enhanced on IP Network", in: IEEE Region 10 International Conference on Electrical and Electronic Technology, (TENCON 2001), Singapore, Singapore, 2001, pp. 181-186.

[9] H. Wang et al, "Adaptive-Weighted Packet Scheduling for Premium Service", in: IEEE International Conference on Communications, ICC 2001, Helsinki, Finland, 2001, pp. 1846-1850.

[10] D. Hang et al, "TD2FQ: An Integrated Traffic Scheduling and Shaping Scheme for DiffServ Networks", in: IEEE Workshop on High Performance Switching and Routing", Dallas, Texas, USA, 2001, pp. 78-82.

[11] C. C. Li et al, "Proportional Delay Differentiation Service Based on Weighted Fair Queuing", in: IEEE Ninth International Conference on Computer Communications and Networks, ICCCN 2000, Las Vegas, Nevada, USA, 2000, pp. 418-423.

[12] S. Floyd and V. Jacobson V., "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, 1993.

[13] A. Parekh and R. Gallager, "A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case", IEEE/ACM Transactions on Networking, Vol. 1, No. 3, pp. 346-357, 1993.

[14] ns-2 Network Simulator, Information Science Institute, 2011, Marina del Rey, California, USA, Available from: http://www.isi.edu/nsnam/ns/

[15] E. Altman and T. Jimenez, "NS Simulator for Beginners, Lecture Notes, 2003-2004", Univ. de Los Andes, Merida, Venezuela and ESSI, Sophia-Antipolis, France. Available from: http://www-sop.inria.fr/maestro/personnel/Eitan.Altman/COURS-NS/n3.pdf.

[16] P. Pieda et al, "A Network Simulator Differentiated Services Implementation", Open IP, Nortel Networks. 2000. Available from: http://www-sop.inria.fr/members/Eitan.Altman/COURS-NS/DOC/DSnortel.pdf.

[17] A. Mrkaic, "Porting a WFQ Scheduler into ns-2's Diffserv Environment", Student Thesis, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, 2001.

[18] J. L. Devore, "Probability and Statistics for Engineering and Sciences". The 6th Edition. Brooks Cole, ISBN 0534-39933-9. ISBN 13: 978-0534-39933-7. 2003.

[19] A. Matrawy et al, "MPEG4 Traffic Modeling Using the Transform Expand Sample Methodology", in: IEEE 4th International Workshop on Networked Appliances, Gaithersburg, Maryland, USA, 2002, pp. 249-256. Also available, 2012, from: "ns-2 contributed-code", http://nsnam.isi.edu/nsnam/index.php/Contributed_Code #Topology_and_Traffic_Generation

[20] G. Salmon et al, "NetFPGA-Based Precise Traffic Generation", in: NetFPGA Developers Workshop'09, Standford University, California, USA, 2009.

[21] CIDR REPORT, 2012. Available from: http://www.cidr-report.org/as2.0/

[22] E. Nygren et al, "The Akamai Network: A Platform for High-Performance Internet Applications", ACM SIGOPS Operating Systems Review, Vol. 44(3), pp. 2-19, 2010.

[23] Analyzing UDP usage in Internet traffic. The Cooperative Association for Internet Data Analysis. 2012. Available from: http://www.caida.org/research/traffic-analysis/tcpudpratio/

[24] C. Lee et al, "Unmasking the growing UDP traffic in a Campus Network", in: Passive and Active Measurement (PAM) Conference, Vienna, Austria, 2012.

[25] E. Kohler and S. Floyd, "Datagram Congestion Control Protocol (DCCP) Overview", ICSI Center for Internet Research, Berkeley, CA, USA, 2003.

[26] J. Huang et al, "An Effective Approximation Scheme for Multiconstrained Qualiy-of-Service Routing", in: IEEE Global Telecommunications Conference (GLOBECOM 2010), pp. 1-6, 6-10, 2010.

[27] G. Xue et al, "Finding a Path Subject to Many Additive QoS Constraints", IEEE/ACM Transactions on Networking, Vol. 15, No. 1, pp. 201-211, 2007.

[28] Y. Xiao et al. "Computing a Most Probable Delay Constrained Path: NP-Hardness and Approximation Schemes", IEEE Transactions on Computers, Vol. 61, No. 5, pp. 738-744, 2012.

Appendix A

Theorem 1. Proof of Equation 4.

Consider that queue i obtains T/τ consecutive results indicating that it has to lose weight, where, as already stated, τ and T are a small and a big intervals of time, respectively, such that T/τ is a big integer.

Equations 2 and 3 are repeated here, for convenience, as Equations. A-1 and A-2.

$$\Delta_i(t_0 + r\tau) = \phi_i(t_0 + (r+1)\tau) - \phi_i(t_0 + r\tau) \qquad \text{(A-1)}$$

$$\Delta_i(t_0 + r\tau) \leftarrow f\frac{\tau}{T}\phi_i(t_0 + r\tau) \qquad \text{(A-2)}$$

Mixing these equations the next one is obtained:

$$\phi_i(t_0 + (r+1)\tau) = \phi_i(t_0 + r\tau)\left(1 + \frac{f}{T/\tau}\right) \tag{A-3}$$

This was the result of the weight of queue *i* after the first interval, as a function of its initial weight. Similarly, the weight for the next interval would be:

$$\phi_i(t_0 + (r+2)\tau) = \phi_i(t_0 + (r+1)\tau)\left(1 + \frac{f}{T/\tau}\right) \tag{A-4}$$

Mixing these last two equations it is obtained that:

$$\phi_i(t_0 + (r+2)\tau) = \phi_i(t_0 + r\tau)\left(1 + \frac{f}{T/\tau}\right)^2 \tag{A-5}$$

So, for the T/τ consecutive increments, the following equation is obtained:

$$\phi_i(t_0 + (r + T/\tau)\tau) = \phi_i(t_0 + r\tau)\left(1 + \frac{f}{T/\tau}\right)^{T/\tau} \tag{A-6}$$

If T/τ is big, then the factor with the exponent can be approximated to:

$$\left(1 + \frac{f}{T/\tau}\right)^{T/\tau} \approx e^f \tag{A-7}$$

And with this, it is obtained that:

$$\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) \approx \phi_i(t_0 + r\tau)\,e^f \tag{A-8}$$

And,

$$\begin{aligned}&\phi_i\left(t_0 + \left(r + \frac{T}{\tau}\right)\tau\right) - \phi_i(t_0 + r\tau) \approx \\ &\phi_i(t_0 + r\tau)\left(e^f - 1\right)\end{aligned} \tag{A-9}$$

And with this the theorem is demonstrated.