

# A simple framework for QoS provisioning in traffic engineered networks

S. Avallone and G. Ventre  
COMICS Lab, Dipartimento di Informatica e Sistemistica  
Università di Napoli Federico II  
Via Claudio 21, 80125 Napoli, Italy

**Abstract**—Different architectures have been proposed and standardized to support Quality of Service (QoS) in the Internet. The goal was to provide the users with a certain level of QoS guarantees. The Traffic Engineering (TE) concept was then introduced to account for resource optimization, as well as users' QoS requirements, and inspired new architectures. Routing has also been a flourishing research field. Proposed QoS routing algorithms seek a multi-constrained optimal path, while traffic engineering algorithms aim at maximizing throughput and the number of admitted requests. However, architectures and routing algorithms have been developed rather independently of each other. The aim of this work is to present a framework which provides predictable communication services to flows without requiring sophisticated queuing at individual routers. We show that the routing algorithm plays a key role to assure both QoS and resource optimization.

## I. INTRODUCTION

In the recent past years several architectures (e.g., IntServ, DiffServ, ATM) have been proposed and standardized with the primary goal of providing different levels of QoS. However, they pay little attention to service providers' viewpoint. Internet Traffic Engineering (TE) is a more general framework that addresses traffic oriented performance requirements, while utilizing network resources economically and reliably. Several works proposed architectures that employ traffic engineering techniques for network management purposes (e.g., [1][2]). Despite TE is a more general concept than QoS, many of such architectures only focused on resource optimization.

The research on QoS provisioning and traffic engineering has also produced plenty of routing algorithms. We can distinguish between QoS routing algorithms and traffic engineering algorithms. The former represent QoS requirements as constraints and address the problem of finding a multi-constrained optimal path. But, they typically ignore the perspective of service providers. The latter target resource optimization and attempt to maximize the throughput and the number of accepted requests. But, they consider the bandwidth (or the *effective* bandwidth) as the unique QoS requirement.

Typically, architectural considerations and routing algorithms are not coupled in previous work. We believe that an architecture aiming at providing QoS cannot prescind from an appropriate routing strategy, as it is difficult to offer strict QoS guarantees without having the control over the route taken by packets. The routing algorithm is the key not only to satisfy the QoS requirements of the flows but also to meet

traffic engineering objectives. Also, if properly designed, a routing scheme may avoid the employment of complicated QoS mechanisms in the core of the network, which is the main reason preventing network operators from deploying QoS infrastructures. The approach we propose in this paper succeeds in providing each individual flow with quantitative guarantees by means of as simple scheduling policies as FCFS. This result is achieved through a proper link weight assignment scheme and a constraint-based routing algorithm.

## II. THE PROBLEM OF QoS LINK WEIGHTS SETTING

We model the network topology *at time*  $t$  as a graph  $G(V, E)$ ,  $V$  being the set of nodes and  $E$  the set of links. Each link  $l$  is assigned a QoS link weight vector with as components the available bandwidth  $w_0(l; t) \geq 0$  and  $m$  additive QoS weights  $w_i(l; t) \geq 0$ ,  $i = 1, \dots, m$ . The QoS requirements of a flow are expressed as an  $(m + 1)$ -dimensional vector  $\vec{Q} = [Q_0, \dots, Q_m]$ , where  $Q_0$  is the requested bandwidth and  $Q_1, \dots, Q_m$  are constraints on the considered additive measures. The goal of a QoS routing algorithm is to find a path  $P$  at time  $t_0$  (the arrival time of the flow) such that:

$$w_0(P; t_0) \stackrel{def}{=} \min_{l \in P} w_0(l; t_0) \geq Q_0 \quad (1)$$

$$w_i(P; t_0) \stackrel{def}{=} \sum_{l \in P} w_i(l; t_0) \leq Q_i \quad \forall i = 1, \dots, m \quad (2)$$

In order for a QoS routing algorithm to be implemented in practice, it is necessary to determine how to set QoS link weights. This section addresses such a problem, which has not been adequately investigated up to now. We distinguish among bottleneck and additive QoS measures. It is safe to state that the link weight associated with the available bandwidth, the most commonly used bottleneck QoS measure, should be as close as possible to the current bandwidth availability. But, does such a principle hold for additive QoS measures too? An exact QoS routing algorithm seeks a path  $P$  such that the inequalities (2) hold. But, the QoS requirements of a flow are satisfied if the QoS *experienced* across path  $P$  is within the constraints *for the whole duration* of the flow. We denote by  $q_i(l; t)$  the average value of the  $i$ -th QoS measure experienced across link  $l$  at time  $t$ . Consequently, the QoS requirements of a flow routed along a path  $P$  are satisfied if:

$$\sum_{l \in P} q_i(l; t) \leq Q_i \quad \forall i = 1, \dots, m \quad \forall t \in T \quad (3)$$

where  $T$  is the duration of the flow. Thus, to make (2) imply (3) it is necessary to appropriately set QoS link weights. We recall that, since there is no differentiation among flows, the QoS experienced across a link  $l$  (i.e.  $q_i(l; t)$ ,  $i = 1, \dots, m$ ) is the same for all the flows traversing that link.

### III. A QoS LINK WEIGHTS SETTING BASED ON UPPER BOUNDS

The aim of this section is to show that it is possible to provide QoS guarantees by employing a proper QoS link weight setting and a routing algorithm that finds a path within additive constraints. We propose an approach based on static, load-independent additive QoS link weights, while  $w_0(l; t)$  is still dynamic and follows the fluctuations of the link bandwidth availability. The idea is to set each additive QoS link weight to a constant value which provides an upper bound to the QoS that can be experienced across link  $l$ . This means that, as long as the bandwidth allocated on link  $l$  is less than the link capacity  $C(l)$ , the QoS experienced by packets crossing that link ( $q_i(l; t)$ ) cannot exceed the QoS weight:

$$w_i(l) \geq q_i(l; t), \quad \forall t, \forall i = 1, \dots, m \quad (4)$$

The notation  $w_i(l)$  underlines that additive QoS link weights are time-independent. Using (4), we do not have to worry about future QoS deterioration when selecting a path for a flow. Routing new flows on a link  $l$  makes the experienced QoS ( $q_i(l; t)$ ) worse, but cannot make it grow beyond the link weight ( $w_i(l)$ ). Thus, if the routing algorithm seeks a path  $P$  having sufficient available bandwidth and such that  $\sum_{l \in P} w_i(l) \leq Q_i$ ,  $\forall i = 1, \dots, m$ , then we are guaranteed that the experienced QoS along the path is within the constraints for the whole duration of the flow. Indeed, from (4) it follows that  $\sum_{l \in P} q_i(l; t) \leq \sum_{l \in P} w_i(l) \leq Q_i$ ,  $\forall t$ ,  $\forall i = 1, \dots, m$ , i.e. (3) holds. In other words,  $\sum_{l \in P} w_i(l)$  represents a QoS level that cannot be exceeded along that path. So, if path  $P$  is selected for a flow, it means that even such QoS level is acceptable for that flow. During the lifetime of the flow, the experienced QoS may vary due to the beginning/ending of some other flows, but it will always be below such level and therefore within the requirements.

The architecture we propose only requires packet filters, policers and shapers to be installed on the edge nodes. Their task is to police and shape incoming traffic so as to assure that its actual rate is within the agreed profile. Core nodes do not require any QoS mechanism. Per-flow QoS is guaranteed by a proper QoS link weight setting and a QoS routing algorithm that admits a flow only if it finds a path satisfying flow's requirements.

The difficulty of the proposed approach lies in the determination of appropriate upper bounds. Indeed, if the QoS link weight is far less than the worst-case QoS value (i.e. the QoS value experienced in case all the link capacity has been allocated) then some packets may experience a QoS larger than the link weight, thus the corresponding flows might not receive the requested QoS. If the QoS link weight is much greater than the worst-case QoS value, then the link will not

be used to accommodate requests it may serve, thus leading to resource underutilization.

### IV. DISCUSSION ON QoS LINK WEIGHTS SETTING

An upper bound to the delay across a link (waiting time plus transmission time) is given by  $D_{max} = \frac{(N+1)E[L]}{C}$ , where  $N$  is the number of packets the buffer of the router can accommodate,  $E[L]$  is the average packet size and  $C$  is the link capacity. We performed ns-2 simulations to study the employment of such upper bound as the delay link weight. First, we considered an ad-hoc built scenario, such that it is as tough as possible to meet delay requirements. For such a scenario, the traffic load has been chosen and routed such a way that all the capacity of the network links is used and at every network node a "new" flow is multiplexed with the traffic outgoing the previous network node. Also, we chose on-off traffic instead of constant bit rate (CBR) traffic, in order to create packet bursts that increase the queue size and consequently the queuing delay. The results of the simulations show that the proposed framework enables to guarantee the requested QoS to all the flows. Also, we note that even though we built the simulated scenario such a way that all the network links are fully loaded, the delay experienced by packets is quite below the delay requirement. The simulation results also show that the longer the path the bigger is such a gap. Intuitively, if a flow crosses only one network link, some packets happen to be enqueued in the last available position of the queue and therefore their delay is close to  $D_{max}$ . But, if a flow crosses more links, it is unlikely that the same packet happens to enter the last position of the queue for all the nodes. These considerations suggest us that using  $D_{max}$  as the delay link weight in a more realistic scenario may lead to the overestimation of the maximum delay experienced along a path, and thus may cause the rejection of flows whose requirements may be actually satisfied.

We also performed simulation studies in a more realistic scenario, which seem to confirm that a relaxation of the upper bounds enables to increase the number of admitted flows. On the other hand, relaxing the upper bounds beyond a certain threshold leads to frequent violations of the constraints.

### V. CONCLUSIONS

In this paper we introduced a framework which enables to provide quantitative guarantees to individual flows at a low complexity. Indeed, no complex mechanisms are required in the core of the network. The basis of our novel approach is a QoS link weights assignment scheme and a constraint-based routing algorithm. The proposed link weight assignment is based on upper bounds to the experienced QoS values.

### REFERENCES

- [1] C. Scoglio, T. Anjali, J. de Oliveira, I. Akyildiz, and G. Uhl, "TEAM: A traffic engineering automated manager for DiffServ-based MPLS networks," *IEEE Communications Magazine*, vol. 42, no. 10, pp. 134-145, October 2004.
- [2] V. Tabatabaee, B. Bhattacharjee, R. La, and M. Shayman, "Differentiated Traffic Engineering for QoS Provisioning," in *Proceedings of INFOCOM 2005*. Miami: IEEE, March 2005.