A New Class of QoS Routing Strategies Based on Network Graph Reduction

C. Casetti, R. Lo Cigno, M. Mellia, M. Munafò
Politecnico di Torino – Dipartimento di Elettronica, Italy
Corso Duca degli Abruzzi, 24 – I-10129 Torino, Italy
e-mail: {casetti,locigno,mellia,munafo}@polito.it
Zsóka Zoltán
Technical University of Budapest – Hungary
e-mail: zsoka@hit.bme.hu

***Abstract*—This paper discusses a new approach to QoS routing, introducing the notion of algorithm resilience (i.e., its capability to adapt to network and load modifications) as performance index of the algorithm itself, for a given network topology, load and traffic pattern.**

The new approach can be summarized as Network Graph Reduction, i.e., a modification of the graph describing the network before the routing path is computed, in order to exclude from the path selection over-congested portions of the network. This solution leads to a class of two-step routing algorithms, where both steps are simple, hence allowing efficient implementation.

Simulation experiments, run on randomly-generated topologies and traffic patterns, show that these routing algorithms outperform both the standard Minimum Hop algorithm and those QoS-based algorithms based on the same metrics but not using the notion of Network Graph Reduction.

I. Introduction

Dynamic, QoS-based routing received considerable attention in recent years [1], [2], [3], [4], [5], [6], [7], especially considering the difficulty in predicting Internet traffic patterns and the consequent impossibility to properly plan and dimension the network.

The core of any QoS-based routing algorithm is a network-status-dependent *cost* function that is used to find the optimal (or at least a suitable) route across the network by solving an optimization problem. In particular, given the best-effort nature of the current Internet where elastic data flows are in the majority, the most commonly used metric aims at the maximization of either network utilization or user throughput. Several proposals introduced cost functions, algorithms and protocols that give them advantages over traditional, topology-based algorithms such as the *Minimum Hop* (*MH*)[1] presently used in TCP/IP networks [8], [9]. For example in [1], the authors introduce the bottleneck bandwidth as a metric and then define the maximization of user throughput as optimization target. Similarly, in [2], [3] the authors study how to improve the throughput of high-bandwidth traffic, such as large file transfers, in a network where resources are fairly shared among connections. Their findings, obtained by simulation, show that, at high loads, a Minimum-Hop routing algorithm maximizes network and user performance; for low-congested networks, instead, they propose an algorithm, named *Minimum Distance* (*MD*) routing, offering better performance. However they are unable to provide an algorithm that merges the two behaviors. Researchers focused their attention upon other aspects of QoS routing, namely protocol overhead [5], [7], implementation issues [4], [6], impact of update policies [5].

A major drawback, however, affects all QoS-based routing algorithms. The cost function at the core of the algorithms tries to find portions of the network where resources are underutilized and exploits them to the benefit of connections that would otherwise cross a congested portion of the network. In doing so, as shown in [11] for the case of simple alternate routing, the algorithm ends up consuming more resources than Minimum Hop routing does, hence, in case of heavy congestion, QoS-based routing wastes resources and performs poorly compared with *MH*. A formal proof of this observation can be found in [10]. The extention of this property to TCP/IP networks is not straightforward, since flows often exhibit a greedy, elastic behavior, using up the available bandwidth. Minimum Hop routing has been already conjectured to be asymptotically optimal as the load ρ offered to the network tends to infinity, and many simulation results confirm this intuition [2], [3], [12], [15]. We stress at this point that the poor performance of QoS-based routing at high loads is not due to the sub-optimality of the used algorithms, but is rooted in the locality of the routing decision. Whenever a call is routed, the given cost function is minimized/maximized for the current state of the network, necessarily disregarding the future network evolution. Under heavy load, the overall network benefit does not coincide with the cost function of a single call, hence the minimization of the one does not lead to the maximization of the other.

In the light of the above discussion, the drawback of QoS routing in the Internet is clear: whatever is gained at low or medium network loads, it is paid for at high network loads. What is needed to solve this problem is a resilient algorithm that allows the migration of a QoS-based routing algorithm to

This work was supported in part by the Italian Ministry for University and Scientific Research through the PLANET-IP Project.

[1] In this paper we refer to the Minimum Hop path as a special case of the Shortest Path, in which all links have unitary costs.

MH routing as ρ grows large. The problem is that ρ is typically not known to the routing algorithm, not even in the case of centralized routing algorithms.

The contribution of this paper lies in the identification and definition of a class of routing algorithms, named *NGR* (Network Graph Reduction), whose performance encompasses that of QoS-based routing algorithms at light and medium loads, and the optimality of *MH* routing at high loads. The goal is obtained by reducing the graph that describes the network topology, and applying a suitable metric to the reduced graph. Results are reported showing both the elementary properties of the algorithm, and its behavior in randomly-generated topologies, both with uniform and non-uniform, time-varying traffic.

The rest of the paper is organized as follows. Section II provides a general description of the *NGR* algorithm, along with implementation issues. Section III presents the network and traffic models used in the simulation presented in Section IV, in which the performance of the *NGR* algorithm is compared to the one of classic QoS routing algorithms, both in simple network scenarios, and in more complex ones. Finally, conclusions and future work are discussed in Section V.

II. *NGR* ROUTING ALGORITHMS

Routing can be formalized as the problem of finding a suitable set of edges connecting two nodes in a directed graph. QoS routing relies on the definition of a suitable edge metric w, which depends on the networks status, and of a cost function $c(\cdot)$, which should be minimized/maximized, yielding the optimal route selection under specific network conditions. As discussed in the Introduction, however, all metrics and cost functions that try to maximize network utilization or/and user throughput suffer from the same drawback: as the network load increases ($\rho \to \infty$) their performance drops below that of a simple Minimum Hop algorithm. Finding metrics that let the path selection process converge to Minimum Hop routing as ρ increases seems to be a hard problem. As shown in [15] the elastic nature of the Internet traffic just worsens the problem. The difficulty of finding a suitable metric lies in the fact that the goal of such a metric should change with the network load, making the problem not easily tackled by a standard minimization approach.

Indeed, we can look at the routing problem from a slightly modified perspective. Instead of trying to find adaptive metrics, the same metric that works well for light loads can be applied at high loads to a *reduced graph* that only contains uncongested links, i.e., links whose load is under a given threshold. Since the reduced graph may not be connected, i.e., no path may exist from a source to a destination, the minimum-hop path should always be included as possible solution of the routing problem. Thus, as the offered load ρ grows, all links become congested and the algorithm necessarily chooses the minimum-hop path. While the direct measure (or evaluation) of ρ is a complex task, the identification of single congested links is extremely easy, so that, overall, the methodology is simple and its implementation straightforward.

This key idea leads to the definition of the Network Graph Reduction (*NGR*) strategy which can be applied to any existing QoS routing algorithms.

A. *Formal Description*

For the purpose of providing a formal description of the class of *NGR* algorithms in their most general form, we use a standard graph theory formalism. Thus, we refer to a generic network as a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices (nodes, in our case), and $\mathcal{E}$ is the set of edges (links). A path $\pi(v_j, v_k)$ of length n is defined as a sequence of n distinct edges e_i joining v_j and v_k, where $v_j, v_k \in \mathcal{V}$, $e_i \in \mathcal{E}$, $\pi(v_j, v_k) = \{e_1, e_2, ..., e_n\}$.

A set $\mathcal{P_G}$ is defined as the set of all paths existing between any two distinct vertices of $\mathcal{G}$, i.e., $\mathcal{P_G} = \{\pi(v_j, v_k) | v_j, v_k \in \mathcal{V}, v_j \neq v_k\}$. Each link is assigned a weight w using any metric that combines topological, physical or traffic-related characteristics of that link. To each path π a cost $c(\pi)$ is assigned using a combination of the weights of its links. Following this, an order relation $\prec$ in the set $\mathcal{P_G}$ is established among paths, and we will refer to a path π_i as "lighter than" π_j if $\pi_i \prec \pi_j$ holds.

The *NGR* algorithm operates two subsequent alterations upon the path set $\mathcal{P_G}$, and applies selection criteria on the resulting subset:

- STEP 1: the first alteration transforms the directed graph $\mathcal{G}$ into $\mathcal{G}' = (\mathcal{V}, \mathcal{E}')$, $\mathcal{E}' \subseteq \mathcal{E}$, selecting only those links whose weight satisfies a *Cut-off criterion* $\mathcal{C}$ (e.g., if w represented the link capacity, "slower" links could be discarded from path selection if $w < w_M$, where w_M is the smallest acceptable capacity); as a result, the path set $\mathcal{P_G}$ is reduced to $\mathcal{P_{G'}}$ after removing those paths containing cut-off links;
- STEP 2: let $\mathcal{P}_{mh}$ be the set including one minimum-hop path from each source to each destination. In case more than one minimum-hop path exist from the same source to the same destination, one at random is selected[2]. The second alteration transforms the path set $\mathcal{P_{G'}}$ into $\mathcal{P_{G''}} = \mathcal{P_{G'}} \bigcup \mathcal{P}_{mh}$; the new subset is built from i) the paths belonging to $\mathcal{P_{G'}}$ and ii) the minimum-hop paths between any two nodes in $\mathcal{V}$, *even if their links were discarded in the first step*.

Finally, the decision on how to route packets between a source and a destination is made by applying the order relation $\prec$ to the path set $\mathcal{P_{G''}}$ and picking the path, joining source and destination, that turns out to be the *lightest* according to the cost metric defined by the QoS routing algorithm.

B. *NGR-MD and NGR-WS algorithms*

In a broad sense the *NGR* methodology can be coupled with any metric and cost function. In order to exemplify this property, we considered the application of this methodology to two algorithms already proposed in the literature, which were shown to offer good performance [1], [2]:

- *Widest-Shortest* (*WS*): for each source-destination pair, the algorithm determines all the paths with the minimum-hop count; if more than one such path exist, it breaks the tie by choosing the one with the largest available bandwidth [1];

[2]It is possible to optimize the $\mathcal{P}_{mh}$ set, for example using load-balancing criteria.

- *Minimum-Distance* (*MD*): for each source-destination pair, the path π is chosen which minimizes the sum on each link of the inverse of the equal bandwidth share [2].

To help the reader, we report a simple description of the above algorithms, using the notation introduced in this paper.

- *WS path weight computation*: a weight w_l associated to a generic link l is computed as

$$w_l = b_l \tag{1}$$

where b_l is an estimate of the currently available bandwidth currently on link l;
The set of paths among which the path $\mathcal{P}_\mathcal{G}$ is chosen is restricted to the minimum-hop paths. Then, the cost $c(\pi_i)$ of path π_i is defined as follows:

$$c(\pi_i) = \min_{l \in \pi_i}\{w_l\}; \tag{2}$$

and the ordering relation is such that $\pi_i \prec \pi_j$ if $c(\pi_i) > c(\pi_j)$ (i.e., π_i is lighter than π_j if its available bandwidth is larger than π_j's).
- *MD path weight computation*: a weight w_l associated to a generic link l is computed as

$$w_l = \frac{n_l + 1}{C_l} \tag{3}$$

where C_l is the link capacity and n_l is an estimate of the number of active flows currently routed over link l.
The cost $c(\pi_i)$ of path π_i is then defined as follows:

$$c(\pi_i) = \sum_{l \in \pi_i} w_l; \tag{4}$$

and the ordering relation is such that $\pi_i \prec \pi_j$ if $c(\pi_i) < c(\pi_j)$.

To implement the *NGR* algorithms, we define the following

Cut-off criterion $\mathcal{C}$: a link is discarded if its available bandwidth is found to be below a given threshold W_t.

We further combine this with either the *MD* or *WS* QoS selection defined, thus obtaining the *NGR-MD* and *NGR-WS* algorithms.

In the rest of the paper, we assume $W_t = 0$, i.e., if a link shows a 100% utilization, it is discarded by the Cut-off criterion.

C. Implementation issues

After defining the algorithms, we are interested in the viability of a hop-by-hop implementation. Unfortunately, the algorithms described above cannot guarantee consistent routing in case of straight hop-by-hop implementation, i.e., routing decisions taken at an upstream node are liable to be superseded by downstream nodes finding a locally-optimal alternative to the path chosen earlier by upstream nodes. The proof in case of the *NGR-MD* metric is by counter-example, and the one relative to the *NGR-WS* metric can be obtained similarly. Consider the simple directed graph in Figure 1, consisting of four

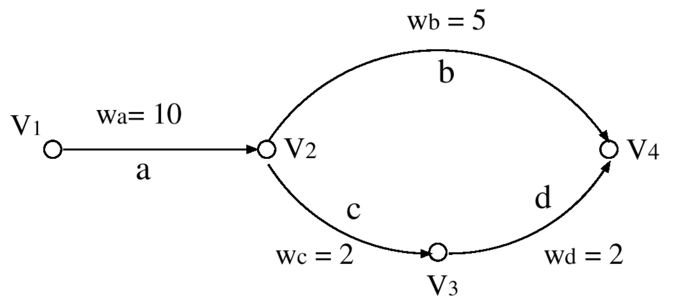


Fig. 1. Simple topology where of hop-by-hop implementation of the *NGR* algorithm fails

links, a, b, c, d, each labeled with its own weight, i.e., $w_a = 10$, $w_b = 5$, $w_c = 2$ and $w_d = 2$. Our aim is to route a packet from V_1 to V_4 according to the *NGR-MD* criterion, locally implemented at each node. For the sake of simplicity, we consider V_4 as the only destination, such that only path ending in V_4 are shown in $\mathcal{P}_\mathcal{G}$.

Let us suppose that, of all links, only link a, joining V_1 and V_2, does not make the cut at STEP 1, being overloaded. As a consequence, the path set is $\mathcal{P}_{\mathcal{G}'} = (\{b\}, \{c, d\}, \{d\})$. STEP 2, however, reintroduces path $\{a, b\}$ because it is the minimum-hop path from V_1 to V_4. Therefore, we have $\mathcal{P}_{\mathcal{G}''} = (\{b\}, \{c, d\}, \{d\}, \{a, b\})$. Since only one path joining V_1 and V_4 is available, such path is the natural choice to go from V_1 to V_4.

However, from V_2's standpoint, the ordering is such that path $\{c, d\}$ is lighter than path $\{b\}$. It follows that a packet reaching node V_2 will travel over links c and d, and not over link b, as was originally meant by V_1.

This could happen every time a node along the path *follows* the congested links, so that, from that node on, the destination can be reached on the path set $\mathcal{P}_{\mathcal{G}'}$ and not necessarily on the minimum-hop path only, i.e., $\pi(v_1, v_4)$ is congested, while $\pi(v_2, v_4)$ is not.

For consistent routing to be preserved, the forwarding procedure must be integrated so as to signal the routing decision taken at upstream nodes. This solution can be implemented using the route-pinning property of the MPLS technology [13], or a slight modification of the IP forwarding procedure.

If MPLS is used, each node can open an LSP (Label Switched Path) on the minimum-hop path toward each destination. Then, for each packet, the edge router selects whether to route it using an MPLS LSP, or using the classic hop-by-hop IP routing over the best path selected using the QoS routing algorithm. Potentially, a total number of $N(N-1)$ LSPs can be set up, where N is the number of routers in the domain. Another possibility is to always have an LSP from each source to each destination, that can either be routed through the QoS path, or through the minimum-hop path in case no available QoS path exists. This simplifies the forwarding procedure at each node, but requires a larger amount of MPLS signalling, because the QoS paths are more frequently updated than the minimum-hop paths.

If, on the other hand, standard IP forwarding is used, it is possible to have the edge router tag packets (i.e., by setting a bit in the TOS field of an IPv4 packet header) when they are supposed to travel along a minimum-hop path. This solution prevents downstream nodes from choosing locally-optimal paths.

By adding the above mentioned tag, the algorithm is implementable using OSPF. Each node has to compute and maintain two routing tables, the first for the minimum-hop paths, the second for the QoS-based routing information. In more detail:

- routers should advertise QoS metrics for their outgoing links according to the selected algorithm, i.e., as computed by (2) or by (3);
- upon advertising them, routers should mark links that fail the cut-off criterion;
- when compiling their routing table, routers should run the *MH* algorithm with the usual 30-minute periodicity, using unitary metrics for each link in the *complete* network graph; in this way, the minimum-hop paths towards any destination are identified;
- with smaller periodicity, they should run the QoS algorithm, thus identifying the lightest path toward any destination;
- for each destination, a router should store next-hop entries associated to minimum-hop and to lightest paths.

Upon reception of a packet, a router should inspect its destination address and TOS field; if a match is found, the router will forward the packet toward the next hop on the lightest path, unless the packet is tagged, in which case the next hop in the minimum-hop path is chosen.

As a final remark, it should be noted that the *asymptotical* computational complexity of the NGR algorithms is the same as the complexity of the QoS algorithms without graph reduction.

III. Evaluating Routing Algorithms in the Internet

The performance evaluation of routing algorithms is typically based on simulations. In this paper, we adopt a traffic model and performance evaluation tool, recently introduced in [14], [15], that matches the characteristics and the dynamics of the Internet better than traditional connection-based simulation tools. Additional information about the tool itself, based on the ANCLES simulator, can be found in [14], [15], [16].

TCP/IP networks do not provide for connection admission control (CAC), hence there is no limitation on the number of concurrent connections on a given path. The dominant Internet applications (namely, all data-based applications) are *elastic*, adapting the transmission speed to the available resources. A connection is opened with a given amount of data to transfer and it is closed either when the transfer is complete, or when long, repeated timeouts cause the connection to reset, or when the user tires of waiting and drops the session. In other words, connection durations are not known in advance, but are determined by data file sizes and network conditions.

The evaluation tool we use takes the elastic model into account. Specifically:

- flows are characterized by a given amount of data to transfer;
- the amount of resources the flow can exploit at any given time is the minimum between a target bit rate for the flow, e.g., the access line speed, and what the flow would be assigned under perfect max-min fair resource sharing;
- the flow holding time depends on the resources it receives throughout its "life", which determine the amount of data already transmitted at any given time;
- if the resources usable by a flow fall below a given threshold, the connection may be shut down prematurely, mimicking the behavior of impatient users.

The performance metrics we take into account are:

- the rate at which connections are shut down due to lack of resources;
- the average throughput of connections that complete the data transfer;
- the overall throughput of the network.

Further details about the traffic model and the evaluation methodology, as well as evidence of the difference between this approach and more traditional ones can be found in [14], [15].

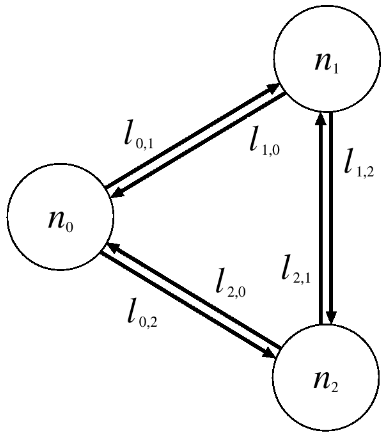


Fig. 2. Tri-node topology used to illustrate the elementary behavior of the routing algorithms

IV. Algorithm Performance and Resilience

The algorithm described in Section II necessarily behaves as a *MD* or *WS* algorithm for uncongested networks and as a *MH* algorithm as the offered load increases causing congestion; however, the behavior at intermediate loads can only be investigated by simulation. For the sake of comparison, result reports include the classic *MD*, *WS* algorithms and the *MH* algorithms.

We separately consider two cases, analyzed in the following two Sections. The first one presents elementary results on a trivial topology, which can aid the understanding of the algorithms and verify that the algorithms behave as expected. The second one analyzes results in complex, randomly-generated topologies and different traffic patterns, ensuring that the properties of the algorithm are not bound by the choice of the topology or of the traffic pattern.

A. Elementary Behavior

We consider the transient behavior on a trivial tri-node topology as described in Figure 2. Node n_0 generates all the traffic which is directed to nodes n_1 and n_2. All links $l_{i,j}$ are unidirectional with capacity 1. n_0 starts sending all connection requests

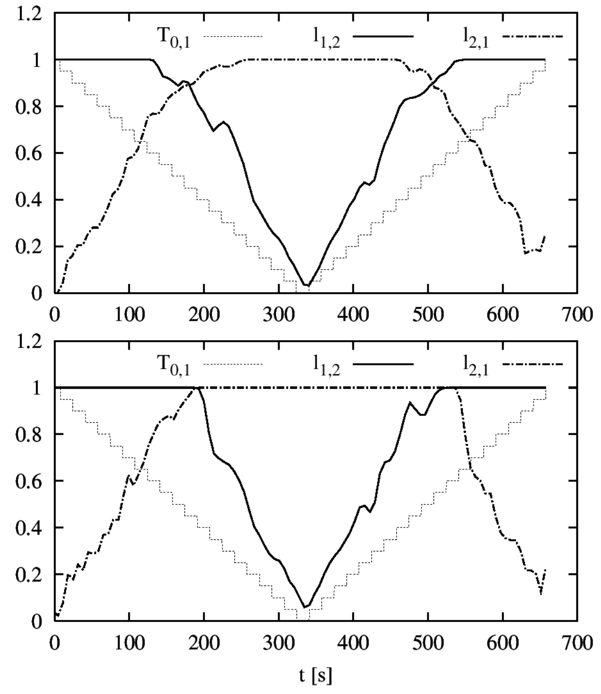


Fig. 3. Free bandwidth on links $l_{1,2}$ and $l_{2,1}$ as the load shifts from traffic relation $T_{0,1}$ to $T_{0,2}$ and back: *NGR-MD* algorithm in the lower plot, *MD* algorithm in the upper plot

to n_1, then progressively (and linearly) decreases the amount of traffic directed to n_1, increasing the fraction directed to n_2. Let $0 \leq T_{0,1} \leq 1$ and $0 \leq T_{0,2} \leq 1$ represent the fraction of the overall traffic routed from node n_0 to node n_1 and to node n_2 respectively. $T_{0,1} + T_{0,2} = 1$ at any time. The resulting overall network offered traffic is constant and always set equal to 2.

The behavior of *MH* in this simple case is obvious: the traffic fraction $T_{0,1}$ is always routed on link $l_{0,1}$ and the traffic fraction $T_{0,2}$ is always routed on link $l_{0,2}$. Notice that only when $T_{0,1} = T_{0,2} = 0.5$ (in the example, when node n_0 is equally sending its traffic to node n_1 and n_2) does the *MH* algorithm manage to transport all the offered traffic. Otherwise, the "widest" connection is throttled on each corresponding link. Also note that, in this simple scenario, both the *WS* and the *NGR-WS* algorithms behave like *MH*, as only one minimum-hop path exists between any two nodes.

If either the *MD* or the *NGR-MD* algorithm is used, the behavior is not so straightforward. Indeed, we expect that, when one of the traffic relations is idle, all available resources are devoted to the other, thus allowing to completely exploit the network capacity. In particular, links $l_{1,2}$ or$l_{2,1}$ should be used by part of the traffic from the active (non-idle) traffic relation. Figure 3 reports the free bandwidth on those links for the *NGR-MD* algorithm (lower plot) and classic *MD* algorithm (upper plot). Link loads are plotted versus the simulation time. The dotted, stepwise line is the fraction of traffic $T_{0,1}$; the fraction $T_{0,2}$ is not shown but has a complementary profile. Ideally, traffic should never be routed on *both* links $l_{1,2}$ and $l_{2,1}$ at the same time, since this would entail that there is traffic flowing from n_0 to n_1 through n_2 and from n_0 to n_2 through n_1, at the same time: it would be desirable to simply "swap" the traffic on the alternate paths so that links $l_{1,2}$ and $l_{2,1}$ are completely free, and they can be used to accommodate traffic among nodes n_1 and n_2.

Analyzing Figure 3 it is clear that the *NGR* algorithm achieves this result almost perfectly, while the *MD* algorithm displays transitions (around times 150 and 500), where both links are used at the same time, thus wasting resources. This phenomenon is at the root of the poor performance of *MD* (in fact, of any QoS-based algorithm) when the network load increases, due to frequent transients similar to the simple ones purposedly generated in our example.

B. Performance on Random Topologies

In this Section, we present results obtained on randomly-generated network topologies using the GT-ITM software [17]. All topologies were generated using the "flat random graph" model, and include 32 nodes, with an average connectivity degree of 4. Every link has the same capacity (10 Mb/s), and there is a best-effort traffic source generator connected to each node, opening connections with a maximum bandwidth of 1 Mb/s; each connection attempts to perform a bulk data transfer whose size is randomly chosen from an exponential distribution with average 2.5 MBytes. Each node adopts a fixed update period for the QoS metric, set to 30 seconds. The simulator does not model the overhead resulting from updates flooding the network.

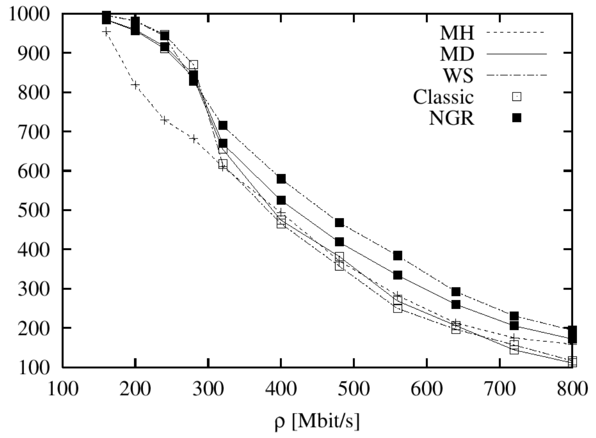
To model the starvation effect [15], a starvation threshold B_t is set to 50 kb/s and used to identify starved connections: if the current per-connection bit rate estimate on a bottleneck link drops below B_t, then the most-backlogged connection on that bottleneck is picked and shut down. This is repeated until the sending rate of starved connections raises above the threshold. This allows us to define the *starvation probability* P_s as the ratio between connections that are prematurely aborted and the total number of connections that entered the network.

To get accurate results, each simulation was ended when the performance indices were such that the 95% confidence interval was within 0.1% of the point estimate.

We report results for two networks, named (A) and (B) for short. More simulations were ran on similar, random topologies, although they are not reported here for lack of space; they show results comparable to the ones reported in the paper, and thus similar conclusions can be drawn.

Results for network (A) reflect a simple stationary traffic scenario. A uniform traffic pattern is simulated, i.e., when a new connection request is generated, the source and the destination are randomly chosen with the same probability. All the sources have the same probability of generating a new connection. These probability are stationary, i.e., they do not change during the simulation.

On the contrary, results reported for network (B) refer to a more complex traffic pattern, mimicking a "client-server" scenario. Besides, the traffic matrix changes over time. In more detail, among the 32 nodes, 5 nodes, picked among those with the highest number of outgoing links, are defined as "servers", while all the remaining one are defined as "clients". Each server node generates 10 times the traffic generated from a client, with a non-stationary behavior: it randomly cycles over time among



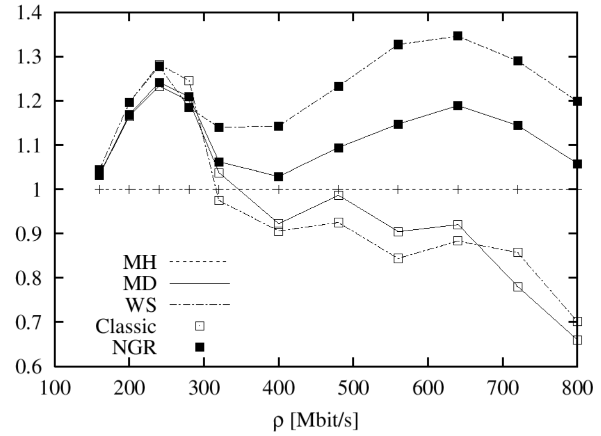


Fig. 4. Average per-connection throughput on network (A) with uniform and stationary traffic pattern; absolute value [kb/s] (upper plot) and value normalized w.r.t *MH* (lower plot)

two states, high traffic and low traffic. When it is in the low-traffic state, it reduces its offered load by a factor of 10, i.e., it behaves as a client node. The average time spent in either the high- or low-traffic state is 3 hours, following an exponential distribution.

1) Network (A): Figure 4 reports the average throughput obtained by single connections that have completed their information transfer. The upper plot reports the absolute value in kb/s versus the aggregate load offered to the network, expressed in Mb/s. The throughput is obviously decreasing with the network load for all routing algorithms, since the number of competing connections increases, and elastic connections adapt their sending rate according to a max-min fair-share algorithm.

The advantage of QoS routing algorithms over *MH* at low loads is clear, but while the classic *MD* and *WS* implementations fall below the *MH* at high loads, *NGR* algorithms always remains above *MH*. The behavior is all the more evident if we take the throughput normalized with respect to *MH* (lower plot). This can also be seen as the gain of the algorithms as compared to the well-established, simple *MH*. It is clear that the *MD* and *WS* algorithms are extremely sensitive to the network load, while the *NGR* versions perform better than *MH*, converging to its performance as ρ grows larger. In particular, the *NGR-WS* algorithm performs better than the *NGR-MD*, since the path set it chooses from only includes minimum-hop ones. Indeed, the achieved gain is always larger than 15%. The non-monotonic relative behavior of the algorithms is due to the "randomness" of the topology that is neither regular nor a "well defined" hierarchical topology, which causes different portion of the network to become congested depending on the values of offered load, and on the specific QoS routing algorithm implemented.

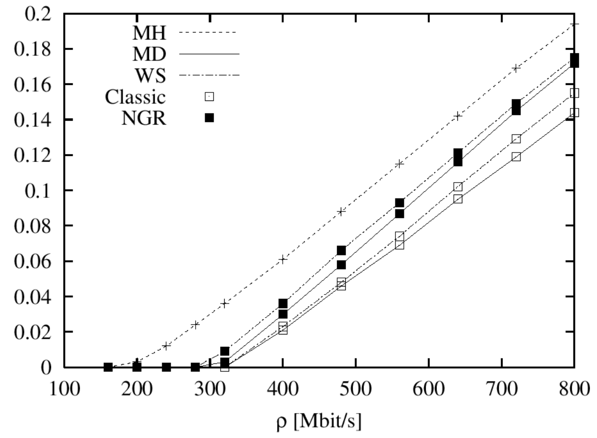


Fig. 5. Average starvation probability on network (A); uniform and stationary traffic pattern

One might wonder whether connections completing their information transfer obtain a larger throughput simply thanks to the larger number of starved connections shutting down, thus leaving more resources up for grabs. Figures 5 and 6 shed some light on this issue. Figure 5 reports the probability that connections close up because they are starved. *NGR* algorithms show a starvation probability that is between *MH* (the highest) and the classic *MD* and *WS* algorithms. In more detail, the starvation probability measured is negligible for all the QoS algorithms for values of the offered traffic smaller than about 300 Mb/s, while the *MH* algorithm shows an earlier increase in the starvation probability, at about 200 Mb/s of offered traffic. Then, a linear increase is shown by all the proposed algorithms.

Figure 6, finally, reports the total network throughput, i.e., the total amount of *useful* information carried by the network,

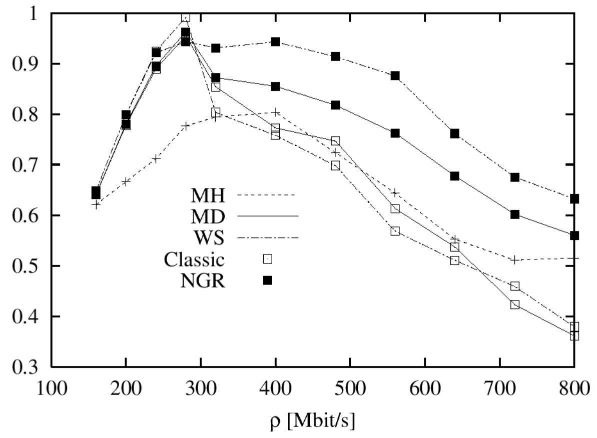


Fig. 6. Total network throughput on network (A); uniform and stationary traffic pattern

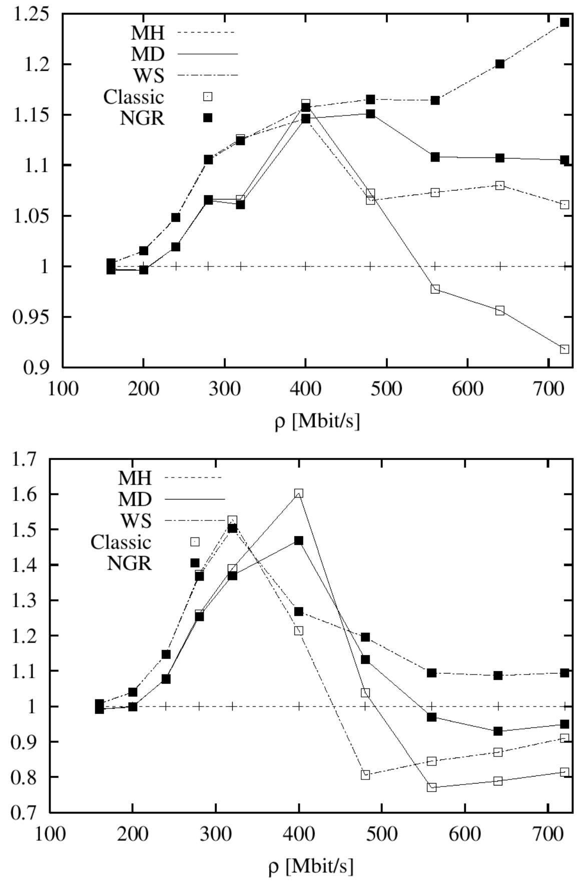


Fig. 7. Per-connection average throughput (normalized w.r.t. M.H.) of clients (upper plot) and servers (lower plot) on network (B) with client-server, non-stationary traffic pattern

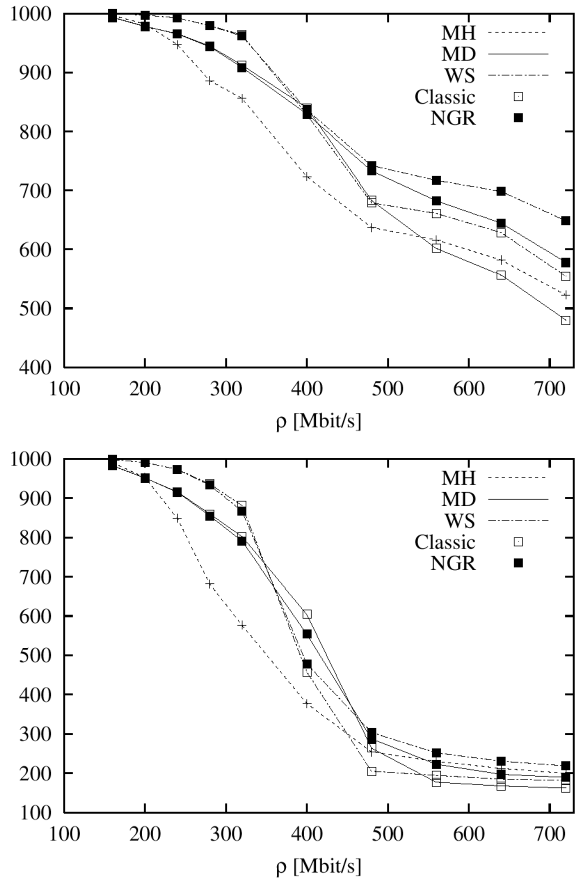


Fig. 8. Per-connection average throughput of clients (upper plot) and servers (lower plot) on network (B) with client-server, non-stationary traffic pattern

normalized with respect to the total installed bandwidth[3].

While all the QoS routing algorithms sport a better utilization of the network capacity at low values of offered load, the classic *MD* and *WS* algorithms show steep decreases in the throughput, mitigated by the *NGR* implementations, which shows the highest throughput, up to 30% larger than either *MH*, *MD* or *WS* algorithms.

2) Network (B): In the case of the client-server model with non-stationary traffic, it is more interesting to separately look at client's and server's behavior, especially in view of the high level of traffic generated by the servers during their active cycles. Figure 7 shows the average throughput per connection normalized to the one obtained by *MH*, while Figure 8 reports the absolute values. The upper plot refers to clients while the lower plots refer to servers. Also in this case the *NGR* version of the algorithms outperforms the classic QoS routing. The more complex nature of the traffic pattern which keeps changing during time, reduces the gain observed for the client connections to less than 20%. On the contrary, the performances obtained by server nodes are greatly improved by any QoS routing for relatively low load, with the classic *MD* showing a clear edge. But as soon as the network load increases over 500Mbps, their performance worsen, while the *NGR* versions still exhibit a better utilization of the network capacity.

[3]The total installed bandwidth is simply defined as the sum of the capacity of all the links in the network

Figures 9 and 10 report the probability of starvation and the total network throughput (again, only the useful portion of it) respectively; in both cases, results are averaged including both servers and clients. The starvation probability confirms results discussed about network (A). It shows that the use of QoS routing greatly reduces the starvation probability, reflecting a better distribution of the traffic on the network. Also in this case, the *NGR-MD* and *NGR-WS* algorithms exhibit a slightly higher starvation probability.

Considering the network throughput reported in Figure 10, we can observe that the network capacity is not yet completely saturated by the offered traffic, at least during the period of time when the servers are in the low-traffic regime. Again, given the complex network topology and traffic pattern, it is very difficult to predict the performance, especially for high values of the offered load. However, also in this case the best performance is obtained by the *NGR* algorithms.

V. DISCUSSION AND WORK EVOLUTION

The search for efficient and robust routing algorithms that can replace the traditional Minimum Hop routing used in TCP/IP networks has received, and is still receiving, considerable attention; however, so far, none of the proposals seems to offer enough benefits to counter-balance the necessarily increased complexity. In particular, robustness to changes in traf-

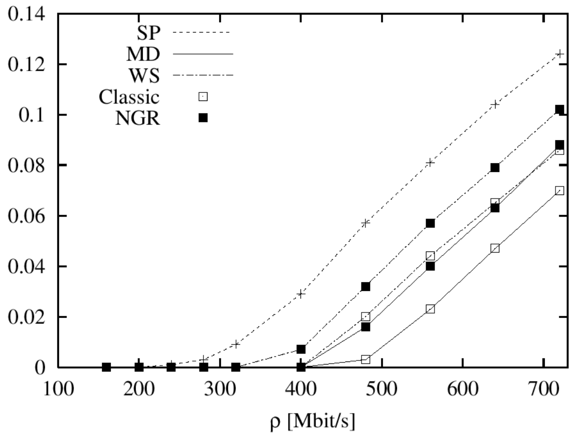


Fig. 9. Average starvation probability on network (B); client-server and non-stationary traffic pattern

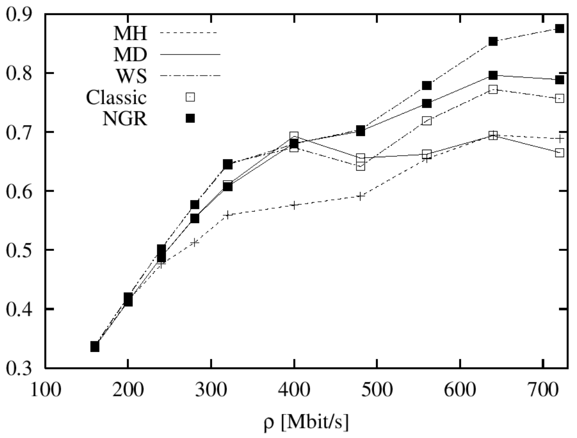


Fig. 10. Total network throughput on network (B); client-server and non-stationary traffic pattern

fic pattern and, most of all, to sudden load increases is a major concern, since all QoS-based algorithms seem to perform worse than the simple Minimum Hop algorithm as the network load increases.

Since the optimality of Minimum Hop routing can be demonstrated under some simplifying assumptions, and can be inferred under more realistic scenarios, a robust and resilient routing algorithm should converge to the Minimum Hop routing as the traffic load grows larger and larger.

Pursuing this idea, in this paper we have proposed and discussed a new class of routing algorithms that exhibit this property and that are shown to outperform both the Minimum Hop and traditional QoS-based algorithms in simulation. These routing algorithms are based on a technique which reduces the graph describing the network to a graph containing only uncongested links. The section of the path is then performed on a set containing paths evaluated on the reduced graph, plus the Minimum Hop itself. In their current form, these routing algorithms cannot be implemented with a standard hop-by-hop procedure, but require either the use of MPLS techniques, or a (simple) modification of the IP forwarding.

Further research in this area should address the possibility of finding a similar class of routing algorithms that are, however, implementable in a hop-by-hop fashion without requiring any modification to the standard IP forwarding.

REFERENCES

[1] Z. Wang, J. Crowcroft, “QoS Routing for Supporting multimedia applications”, *IEEE JSAC* 14(7):1228–1234, Sept. 1996.

[2] Q. Ma, P. Steenkiste, and H. Zhang. “Routing High-Bandwidth Traffic in Max-Min Fair Share Networks”, in *Proceedings of the ACM SIGCOMM’96*, pages 206–217, Stanford, CA, USA, Aug. 1996.

[3] Q. Ma, P. Steenkiste, “Routing Traffic with Quality-of-Service Guarantees in Integrated Services Networks”, In *8th IEEE/ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV’98)*, England, July 1998.

[4] D. Cavendish, M. Gerla, “Internet QoS Routing Using the Bellman-Ford Algorithm”, *High Performance Networking (HPN98)*, IFIP, Vienna, Austria, 1998.

[5] G. Apostolopoulos, R. Guérin, S. Kamat, S. K. Tripathi, “Quality of Service Based Routing: A Performance Perspective”, *ACM SIGCOMM’98*, Vancouver, Canada, Sept. 1998.

[6] G. Apostolopoulos, D. Williams, S. Kamat, R. Guerin, A. Orda, T. Przygienda, “QoS Routing Mechanisms and OSPF Extensions”, *RFC 2676*, IETF, Aug. 1999.

[7] A. Shaikh, J. Rexford, K. Shin, “Load-sensitive routing of long-lived IP flows”, *ACM SIGCOMM ’99*, Cambridge, Massachusetts, USA, pp. 215–226, august 1999.

[8] J. Moy, “Open Shortest Path First version 2”, *RFC 2328*, April 1998.

[9] Ross Callon, “Use of OSI IS-IS for Routing in TCP/IP and Dual Environments”, *RFC 1195*, December 1990

[10] D. Bertsekas, R. Gallager, “Data Networks”, *PrenticeHall*, 2nd ed., 1992.

[11] S. Sibal, A. De Simone, “Controlling Alternate Routing in General-Mesh Packet Flow Networks,” *ACM SIGCOMM ’94*, London, U.K., August 1994.

[12] C. Casetti, G. Favalessa, M. Mellia, M. Munafò, “An Adaptive Routing Algorithm for Best-effort Traffic in Integrated-Services Networks”, *16th International Teletraffic Congress (ITC-16)*, Edinburgh, UK, June 1999

[13] Rosen, E., Viswanathan, A., R. Callon, “Multi-Protocol Label Switching Architectur”, *RFC 3031*, January 2001.

[14] C. Casetti, R. Lo Cigno, M. Mellia, M. Munafò, Z. Zsoka, “Routing Algorithms Evaluation for Elastic Traffic”, *IEEE Workshop on High Performance Switching and Routing (HPSR 2001)*, Dallas, Texas USA, May 29-31, 2001.

[15] C. Casetti, R. Lo Cigno, M. Mellia, M. Munafò, Z. Zsoka, “A Realistic Model to Evaluate Routing Algorithms in the Internet”, *IEEE Globecom 2001*, San Antonio, Texas USA, Nov. 25–29, 2001.

[16] ANCLES - A Network Call-Level Simulator.
URL: `http://www.telematics.polito.it/ancles`

[17] GT-ITM *Georgia Teach-Internetwork Topology Models*
URL: `http://www.cc.gatech.edu/projects/gtitm`