Virtual Flow Deviation: Dynamic Routing of Bandwidth Guaranteed Connections

Antonio Capone, Luigi Fratta, Fabio Martignon
DEI, Politecnico di Milano
Piazza L. da Vinci 32, 20133 Milan, Italy
capone, fratta, martignon@elet.polimi.it

Abstract

Finding a path in the network for a new incoming connection able to guarantee some quality parameters such as bandwidth and delay is the task of QoS routing techniques developed for new IP networks based on label forwarding.

In this paper we focus on the routing of bandwidth guaranteed flows in a dynamic scenario where new connection requests arrive at the network edge nodes. When more than a path satisfying the bandwidth demand exists, the selection of the path is done in order to minimize the blocking probability of future requests.

We propose a new routing algorithm named Virtual Flow Deviation (VFD) which exploits the information of the ingress and egress nodes of the network and the traffic statistics. We show that this new algorithm allows to reduce remarkably the blocking probability in most scenarios with respect to previously proposed schemes.

1 Introduction

The development of the Internet has been impressive in the last years. The upcoming high-speed networks are expected to support a wide variety of communication-intensive real-time multimedia applications. However, the current Internet architecture offers mainly a best-effort service and does not meet the requirements of future integrated services networks that will be designed to carry heterogeneous data traffic [1]. In order to offer guaranteed end-to-end performance (as bounded delay, jitter or loss rate), it's necessary to introduce some sort of resource reservation mechanism in the Internet. With classical IP routing, however, when the resources are not available on the shortest path the connection request must be rejected even if they are available on alternative paths.

Many Quality of Service routing algorithms have been recently proposed [1, 2, 3, 4]. With new label based forwarding mechanisms, such as MPLS (Multi Protocol Label Switching) [5], per flow path selection is possible and Quality of Service parameters can be taken into account by the routing algorithm. The notion of Quality of Service (QoS) has been introduced to capture the qualitatively and quantitatively defined performance contract between the service provider and the user applications. The QoS requirement of a connection can be given as a set of link constraints. Such contraints can be expressed, for instance, as bandwidth constraints specifying that the path selected for the connection of the requesting user has sufficient bandwidth to meet the connection requirement.

The goal of QoS routing algorithms is twofold:

- satisfying the QoS requirements for every admitted connection;
- achieving global efficiency in resource utilization.

In this paper, we focus on the problem of QoS routing. First of all, we review some of the proposed QoS routing algorithms, such as the Min-Hop Algorithm (MH) [6], the Widest Shortest Path Algorithm (WSP) [7] and the Minimum Interference Routing Algorithm (MIRA) [8]. We'll describe in some detail MIRA, which has somewhat different features compared to other algorithms, as it takes explicitly into account the topology disposition of the ingress and egress points of the network, i.e. those routers through which the traffic enters into and exits form the network.

We analyze and compare their performance based on results obtained under a variety of simulated scenarios. We observe that these algorithms are unable to achieve good performance when the traffic statistics at each ingress point are different (for instance, when an ingress node offers to the network a traffic significantly higher than other nodes). Furthermore, all the proposed algorithms lack to consider traffic statistics which can be easily measured at each ingress node.

To overcome these limitations, we propose a new QoS routing algorithm, called Virtual Flow Deviation (VFD), which keeps into account the information of the ingress and egress nodes of the network and the traffic statistics. More precisely, VFD exploits the knowledge of the disposition of the ingress/egress nodes of the network, and uses the statistics information about the traffic offered to the network through each ingress point in order to forecast future connection arrivals. For every connection request, VFD creates a set of *virtual calls* based on the observed traffic statistics. These virtual calls represent the calls which are likely to request resources to the network in the immediate future, and will thus interfere with the current one. In order to improve the global resource utilization, VFD routes the current call together with the virtual calls using the Flow Deviation method [9].

In order to assess the effectiveness of the proposed scheme, we analyze the performance of VFD under a variety of scenarios, and we compare it with that achieved by existing routing algorithms.

The paper is structured as follows: in Section 2 we address the QoS routing problem and some existing routing algorithms. In Section 3 we introduce the Virtual Flow Deviation algorithm. In Section 4 we discuss and compare the performance of these algorithms under a variety of simulated scenarios. Finally, Section 5 concludes the paper.

2 QoS Routing

The concept of Quality of Service (QoS) indicates the performance trading between the Internet Service Provider and the hosts' applications. The QoS constraints parameters of a single connection can be specified in terms of:

- minimum guaranteed bandwidth;
- maximum tolerable delay and/or jitter;
- maximum tolerable loss rate.

The main goal of a QoS routing technique is to determine a path able to guarantee the constraints requested by the incoming connection and to reject as few connections as possible.

Let's model a network as a graph (N, A), where the nodes N represent routers and Arcs A represent communication links, as shown in Figure 1.

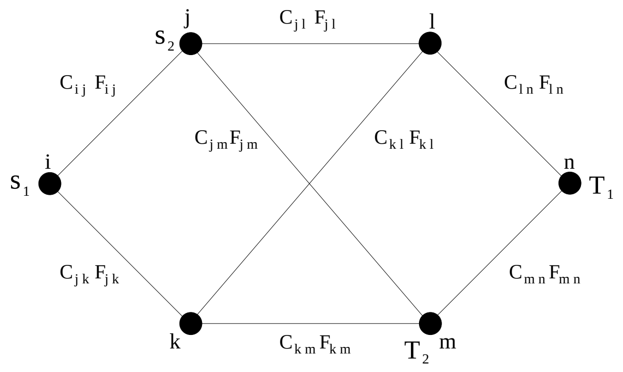

Figure 1: QoS Network State

The traffic enters the network at ingress nodes S_i and exits at egress nodes T_i. Each single connection requires a path from S_i to T_i. Each link (i, j) has associated some parameters such as the capacity C_{ij} and the actual flow F_{ij}. The residual bandwidth is defined as $R_{ij} = C_{ij} - F_{ij}$. A new connection can be routed only over links with R_{ij} greater or equal to the requested bandwidth.

Referring to a new connection with requested bandwidth d_k, a link is defined as *feasible* if $R_{ij} \geq d_k$. A connection can be accepted if at least one path between S_i and T_i exists in the feasible network. The minimum R_{ij} over a path defines the maximum residual bandwidth of that path.

In the following we review some of the algorithms available in the literature.

Min-Hop Algorithm

The Min-Hop Algorithm (MHA) [6] routes an incoming connection along the path which reaches the destination node using the minimum number of feasible links.

This scheme, based on the Dijkstra algorithm, is simple and computationally efficient. However, MHA can build up bottlenecks in the network, as it tends to overload some links leaving others underutilized. The cost given to each link, in fact, remains unvaried and independent of the current link load and therefore MHA tends to use the same paths until saturation is reached before switching to other paths with underutilized links.

Widest Shortest Path Algorithm

The Widest Shortest Path Algorithm (WSP), proposed in [7], is an improvement of the Min-Hop algorithm, as it attempts to load-balance the network traffic. In fact, WSP choses the feasible min-hop path with the maximum residual bandwidth, thus discouraging the use of already heavily loaded links.

However, WSP still has the same drawbacks as MHA since the path selection is performed among the shortest feasible paths.

Minimum Interference Routing Algorithm

The Minimum Interference Routing Algorithm (MIRA), proposed in [8], explicitly takes into account the location of the ingress and egress routers. The key idea of MIRA is to route an incoming connection over a path which least interferes with possible future requests.

Specifically, an incoming connection request between (S_i, T_i) is routed with the goal of maximizing an objective function which is either the minimum maximum-flow (maxflow) of all *other* ingress-egress pairs or a weighted sum of maxflows in the latter case, where weights α_{ST} assigned to each ST pair reflect the "importance" of the flow.

In order to achieve an on-line routing algorithm, MIRA keeps an updated list of the *critical* links, i.e. the links whose use by the incoming call diminishes the maxflow between other pairs.

When a new call has to be routed between the sorce/destination pair (S_i, T_i), MIRA determines the set L_{ST} of the critical links for all the source/destination pairs (S_j, T_j) *other* than (S_i, T_i). The weight w of each link l is then set according to the equation $w(l) = \sum_{(S,T):l \in L_{ST}} \alpha_{ST}$, and the route which causes the minimum interference to other source/destination pairs is selected.

In spite of its more sophisticated functions, MIRA still has the following limitations whose effect will be shown in the discussion of numerical results:

- MIRA discourages the use of critical links based only on the number of other S-T pairs which could use them, without verifying if these S-T pairs actually use these links. Evidently, if one of these other S-T pairs introduces a low traffic in the network, the *criticality* of the links which diminish its maxflow is far less important than that of S-T pairs which produce a large amount of traffic. As a consequence, MIRA preserves the use of certain links which remain underutilized, thus causing a suboptimal use of the network.

  To overcome this limitation, it has been proposed to maximize a weighted sum of the source/destination maxflows. However, in [8] the weights are chosen offline and do not adapt to changes in network traffic. Hence this solution does not provide the flexibility necessary to an on-line routing scheme.

- In its on-line implementation, MIRA sets the link weights almost in a static way according only to their level of criticality. In fact, the only event which can cause the redistribution of new weights is the saturation of some links, similarly to the Min-Hop algorithm.

- While chosing a path for an incoming request, MIRA does not take into account how the new call will affect the future requests of the *same* ingress/egress pair (auto-interference).

3 Virtual Flow Deviation

In the previous Sections we have summarized the features and the limitations of the existing QoS routing algorithms. In this section we propose a new technique, called Virtual Flow Deviation (VFD), which aims to overcome these limitations by exploiting all the information available which has been underevaluated in the other routing algorithms.

To better describe the current state of the network and to forecast its future one can add to the topological information on the location of ingress/egress pairs, used by MIRA, the traffic statistics obtained by measuring the load offered to the network at each source node. This information plays a key role in deciding how to route incoming requests to prevent network congestion.

Exploiting the knowledge of the offered traffic, we can forecast how many new connections will probably be generated at each S-T pair in the immediate future. These new calls, which are likely to be offered to the network, will interfere with the current call to be routed, and they should thus be considered in the routing process.

In order to take into account the future traffic offered to the network, VFD thus routes not only the real call, but also a certain number of *virtual* calls, which represent an estimate (based on measured traffic statistics) of the connection requests that will probably interfere with the current, real call. The number of these virtual calls, as well as the origin and the bandwidth requested should reflect as closely as possible the real future network conditions. Hence, we determine these parameters based on the past traffic statistics of the various ingress/egress pairs, as we'll explain in detail in the next subsection. With this mechanism, we can take into account both the interference and the auto-interference produced by the current and future calls between every S-T pair.

All the information concerning network topology and estimated offered load must be used to produce a path selection which uses at the best the network resources and minimizes the number of rejected calls. Such a path selection is performed in VFD by the Flow Deviation algorithm, which allows to determine the optimal routing of all the flows entering the network through all the different source/destination pairs.

Before describing in detail the VFD algorithm, we give an high-level scheme of its functionalities.

3.1 The Virtual Calls

Each call offered to the network, either real or virtual, can be represented with the notation (S, T, d), where S and T are the source and destination nodes of the call, respectively, and d is its bandwidth requirement. The determination of these three parameters for each virtual call is quite critical, as they must reflect as closer as possible the real evolution of the network. More precisely, we have to determine how many virtual calls should be generated, their source/destination pairs, and their bandwidth request. In this process, we can easily measure and distribute to each S-T pair the two following parameters:

- the average traffic (λ_{S_i,T_i}) offered by the i-th S-T pair, defined as the average number of connections entering the network through the node S_i in an interval Δt
- the probability density distribution of the bandwidth required at each S-T pair, which can be estimated as the ratio between the number n_b of calls which have requested b bandwidth units and the total number N of calls considered for the estimation.

Note that, for simplicity of exposition, we have considered bandwidth requests which are multiple of a given bandwidth unit. However, the algorithm works as well with bandwidth requirements which can assume any real value.

If we define the total average load offered to the network, Λ, as:

$$\Lambda = \sum_{\forall \text{ pairs } S_i - T_i} \lambda_{S_i,T_i}$$

we can evaluate the probability P_{S_i,T_i} to receive a call between the node pair (S_i, T_i) as $P_{S_i,T_i} = \frac{\lambda_{S_i,T_i}}{\Lambda}$ while the probability P_{b_i} to have a request of b bandwidth units at the i-th source node, is estimated by $P_{b_i} = \frac{n_b}{N}$.

Having derived the above information, one can easily generate the parameters (S_i, T_i, d_i) which completely determine the virtual calls. The virtual calls are routed together with the real call, represented by (S_R, T_R, d_R), using the Flow Deviation algorithm, in order to ensure an optimal flow assignment.

To determine the number N_v of virtual calls which must be generated we have considered two different approaches listed below.

The first one takes into account the variations in the total load offered to the network, and estimates the average number $\overline{N}$ of calls routed (active calls) in the network over the past T seconds. When the new call request arrives, $N_v = \lfloor(\overline{N} - N_A)\rfloor$ virtual calls are generated, where N_A is the current number of active calls. Note that if $N_A > \overline{N}$, no virtual call is generated.

The second approach is based on the maximum number of active calls routed in the network, N_{max}, and the number of virtual calls is given by $N_v = \lfloor (N_{max} - N_A) \rfloor$. The underlying assumption in this approach is to consider the network operating close to its saturation. This condition, which stresses the effectiveness of the routing algorithm, is useful when comparing performance.

3.2 The Virtual Flow Deviation Algorithm

The VFD algorithm operation is described in the flow diagram of Figure 2.

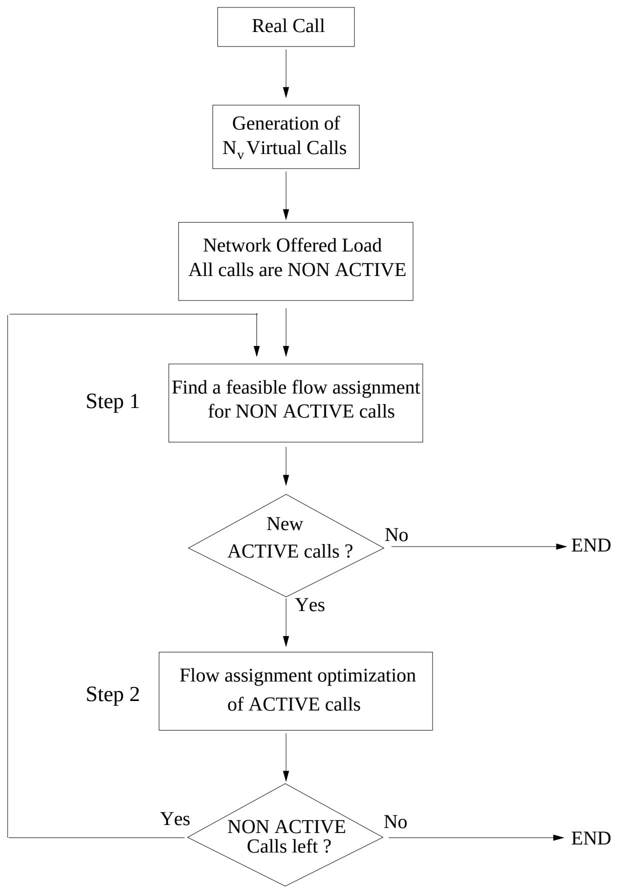
Real Call
Generation of N_v Virtual Calls
Network Offered Load
All calls are NON ACTIVE
Step 1
Find a feasible flow assignment for NON ACTIVE calls
New ACTIVE calls ?
No
END
Yes
Step 2
Flow assignment optimization of ACTIVE calls
Yes
NON ACTIVE Calls left ?
No
END

Figure 2: The Virtual Flow Deviation algorithm

Upon a new call request the process for generating N_v associated virtual calls, as described in the previous section, is activated. The real call and the virtual calls are then offered to the network. The procedure to route the new traffic operates in two steps.

In the first step an initial feasible flow assignment is obtained. Calls are routed one by one starting from the real call. A call can be either defined as ACTIVE, if a feasible path has been found, or NON ACTIVE otherwise. This step is repeated until all calls have been considered. The procedure stops if the real call cannot be routed.

In step two the routing of all ACTIVE calls is optimized using the Flow Deviation Method.

Then step one is repeated for the NON ACTIVE calls. If at least one NON ACTIVE call is declared ACTIVE the step two is repeated and the procedure is iterated until either all calls are ACTIVE or step one does not define any new call as ACTIVE.

At the end of the procedure the real call has been routed on an optimal path considering an expected future evolution of the network traffic load.

The feasible flow assignment is obtained in step one by using the Shortest Path Algorithm (Dijkstra) applied to the network whose links weights reflect the actual channel utilization. More specifically, for each link a weight $w_{ij} = \frac{1}{C_{ij}-F_{ij}}$ is assigned and updated at each iteration.

A more formal description of VFD is given by the pseudo-code in Table 1. The description of a connection has been enriched by a flag to identify ACTIVE and NON ACTIVE connections.

```
for ( $\forall$ connection $(S_k, T_k, d_k, flag_k)$)
    $flag_k =$ NON ACTIVE
end for

do
    for ($\forall$ connection $(S_k, T_k, d_k, flag_k =$ NON ACTIVE)
        for ($\forall$ link $l_{ij}$)
            weight assignment:
            $w_{ij} = \frac{1}{C_{ij}-F_{ij}}$ if $F_{ij} < C_{ij}$
            $w_{ij} = \infty$ if $F_{ij} = C_{ij}$
        end for
        execution of Dijkstra Shortest Path algorithm:
        if ($\exists$ a path between $S_k$ and $T_k$ with bandwidth $d_k$)
            update $F_{ij}$ and memorize the path
            $flag_k =$ ACTIVE
        end if
    end for

    for ( $\forall$ connection $(S_k, T_k, d_k, flag_k =$ ACTIVE))
        execution of the Flow Deviation method
    end for

while (in the last iteration at least one $flag_k$ has been set to ACTIVE)
```

Table 1: Pseudo-code specification of *Step 1* and *Step 2* introduced in Figure 2

4 Numerical Results

In this section we compare the performance of the Virtual Flow Deviation algorithm with that of the Min-Hop Algorithm and MIRA referring to three different network scenarios in order to cover a wide range of possible environments. The performance function we consider is the percentage of rejected calls versus the average total load offered to the network.

The first scenario we consider is illustrated in Figure 3. In this network the links are unidirectional with capacity equal to 120 bandwidth units. The network traffic, offered through the source nodes S_1, S_2 and S_3, is unbalanced as the traffic offered by sources S_2 and S_3 is four times that offered by S_1. Each connection requires a bandwidth uniformly distributed between 1 and 3 units.

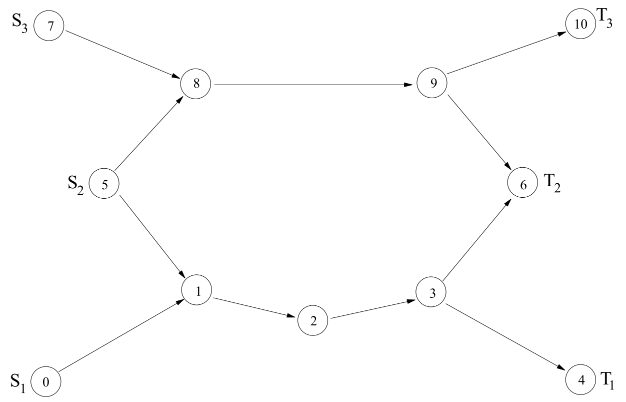

Figure 3: Network topology with unbalanced offered load: the source/destination pairs S_2-T_2 and S_3-T_3 offer to the network a traffic load which is four times higher than that offered by the pair S_1-T_1

In this simple topology only one path is available to route connections between S_1-T_1 and S_3-T_3, while connections S_2-T_2 can choose between two different paths.

This case evidentiates the main limitation of MIRA that does not consider the information about the total load offered to the network. Since the links (1,2),(2,3) and (8,9) are critical for S_2-T_2, the route selected by MIRA follows the path with the minimum number of critical links (5-8-9-6 in the example). Unfortunately this interferes with the path (7-8-9-10) that carries the high load of S_3-T_3. This choice will penalize the performance as shown in Figure 4.

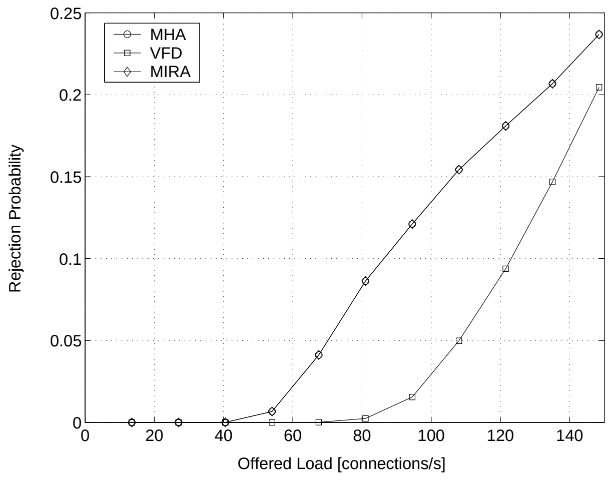

Figure 4: Connection rejection probability versus the average total offered load to the network of Figure 3

VFD achieves the best performance since it exploits the information on the unbalanced load. The performance of MHA and MIRA are exactly the same. In fact MIRA operates for the connections between S_2-T_2 the same path selection of MHA, since the path (5-8-9-6) is shorter than (5-1-2-3-6).

The second network considered is shown in Figure 5 where a balanced traffic is offered at S_1 and S_2. All links have the same capacity (120 bandwidth units) and are bidirectional.

The critical links identified by MIRA are (0,1),(0,2),(0,3),(1,4),(2,4),(3,4) for connections S_2-T_2 and (1,0),(1,2),(1,4),(0,3),(2,3),(4,3) for connections S_1-T_1. This leads to have only the path (1-2-3) available for connections S_1-T_1 and the path (0,2,4) for connections S_2-T_2.

This is a very limiting way of operation that penalizes MIRA. As shown in Figure 6, VFD can reach a more balanced routing using all the available paths with no limitation.

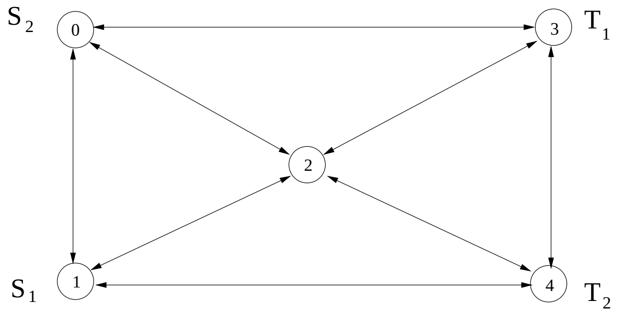

Figure 5: Network Topology with a large number of critical links

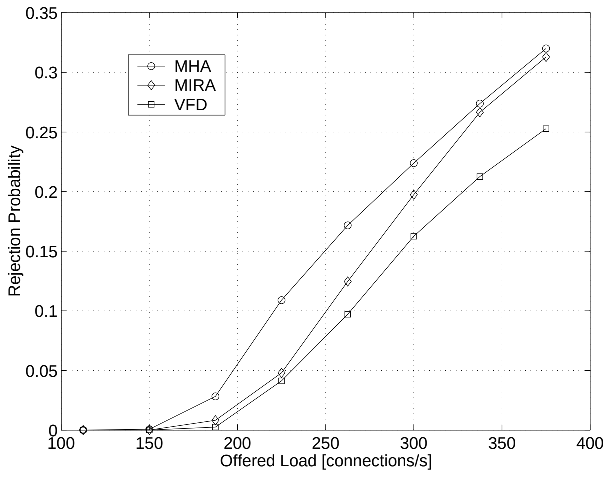

Figure 6: Connection rejection probability versus the average total offered load to the network of Figure 5

As third scenario we have considered the network shown in Figure 7 that was proposed in [8] and represents a more realistic scenario. All links are bidirectional. Those marked by heavy solid lines have a capacity of 480 bandwidth units while the others have a capacity equal to 120 bandwidth units. The performance for the case of balanced offered traffic, considered in [8], are shown in Figure 8.

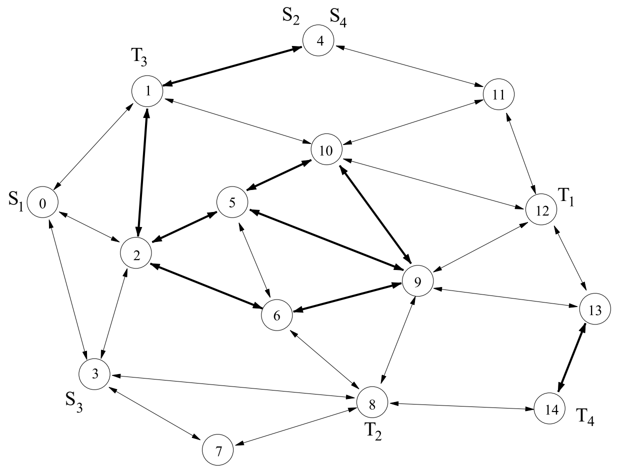

Figure 7: Network Topology with a large number of nodes, links, and source/destination pairs

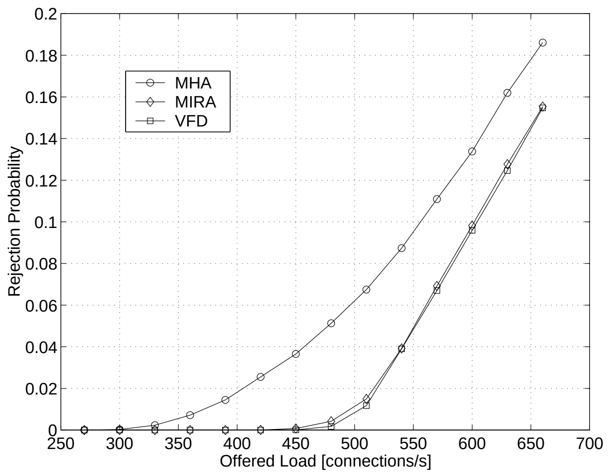

Figure 8: Connection rejection probability versus the average total offered load to the network of Figure 7

VFD and MIRA achieve almost the same performance. However, VFD presents some advantages at low rejection probabilities since it starts rejecting connections at an offered load 10% higher than MIRA. We have measured that a rejection probability of 10^{-4} is reached at an offered load of 420 connections/s by MIRA as opposed to 450 connections/s for VFD.

If we consider on the same topology an unbalanced load where for instance traffic S_1-T_1 is four times the traffic of the other sources, the improvement in the performance obtained by VFD is much more significant. The curves shown in Figure 9 confirm that the unbalanced situations are more demanding on network resources with respect to the balanced case as the rejection probability for the same given offered load is much higher. In these more critical network operations VFD has proved to be more effective providing improvements of the order of 20%.

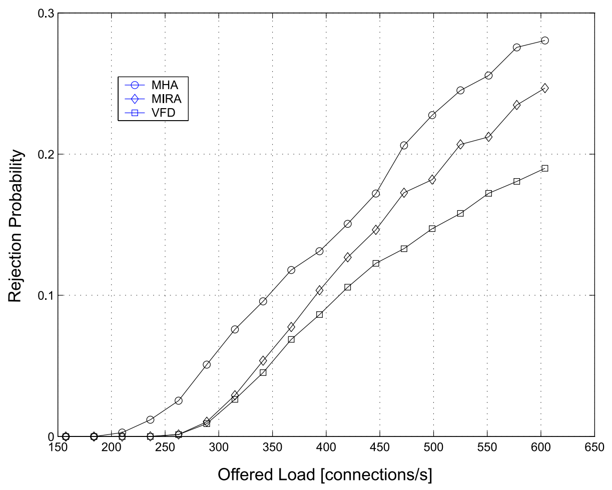

Figure 9: Connection rejection probability versus the average total offered load to the network of Figure 7, where the traffic between S_1-T_1 is four times higher than the traffic produced by the other pairs

5 Conclusions

We have proposed a new QoS routing scheme, called Virtual Flow Deviation (VFD), which exploits the informations about the ingress and egress nodes of the network and the traffic statistics.

As a key innovation with respect to existing QoS routing algorithms, VFD performs a path selection based not only on the current state of the network, but also on an estimate of its future evolution. This goal is achieved by routing a set of *virtual* calls together with the current call using the Flow Deviation algorithm to ensure an optimal disposition of all the flows. Virtual calls represent an estimate (based on traffic statistics measured at each ingress node) of the calls which are likely to be offered to the network during the current connection lifetime, thus interfering with it.

VFD allows to achieve lower connection rejection rates than existing algorithms, expecially in more critical network operations whith unbalanced traffic offered at the ingress nodes.

We have shown that this new algorithm allows to reduce remarkably the blocking probability in most scenarios with respect to previously proposed schemes.

References

[1] S. Chen and K. Nahrstedt. An Overview of Quality-of-Service Routing for the Next Generation High-Speed Networks: Problems and Solutions, 1998.

[2] Z.Wang and J.Crowcroft. QoS Routing for Supporting Resource Reservation. In *IEEE JSAC*, Sept.1996.

[3] A.Orda. Routing with End to End QoS Guarantees in Broadband Networks. In *IEEE INFOCOM'98*, Mar.1998.

[4] B.Awerbuch et al. Throughput Competitive On Line Routing. In *34th Annual Symp. Foundations of Computer Science*, Palo Alto, CA, Nov.1993.

[5] E.Rosen, A.Viswanathan, and R.Callon. Multiprotocol Label Switching Architecture. In *RFC 3031*, January 2001.

[6] D.O.Awduche, L.Berger, D.Gain, T.Li, G.Swallow, and V.Srinivasan. Extensions to RSVP for LSP Tunnels. In *Internet Draft draft-ietf-mpls-rsvp-lsp-tunnel-04.txt*, September 1999.

[7] R.Guerin, D.Williams, and A.Orda. QoS Routing Mechanisms and OSPF Extensions. In *Proceedings of Globecom*, 1997.

[8] Murali S. Kodialam and T. V. Lakshman. Minimum Interference Routing with Applications to MPLS Traffic Engineering. In *Proceedings of INFOCOM (2)*, pages 884–893, 2000.

[9] L.Fratta, M.Gerla, and L.Kleinrock. The Flow Deviation Method: An Approach to Store-and-forward Network Design. In *Networks 3*, pages 97–133, 1973.