The Prediction Approach in QoS Routing

Eva Marín-Tordera, Xavier Masip-Bruin,
Sergio Sánchez-López, Jordi Domingo-Pascual

Advanced Broadband Communications Center
Universitat Politècnica de Catalunya, UPC
Vilanova i la Geltrú, Barcelona, Catalunya, Spain
{eva, xmasip, sergio, jordid}@ac.upc.edu

Ariel Orda

Department of Electrical Engineering
Technion I.I.T., Haifa, Israel
ariel@ee.technion.ac.il

***Abstract*—** Usual QoS routing algorithms involve the periodic update of network state information in all the network nodes. Based on this knowledge the QoS routing algorithms select the 'best' route. It has been shown in the literature that the performance of these QoS routing algorithms strongly depends on the frequency of updating. We propose a new QoS routing mechanism called Prediction-Based Routing based on predicting the availability of links and routes regardless from the network state information. Consequently, update messages are not required, hence reducing signalling overhead and providing a major enhancement in terms of scalability. We show that the PBR is a viable option compared with usual QoS routing algorithms from the point of view of performance, complexity ad signalling overhead.

Keywords-component; QoS Routing, Prediction-Based Routing, Routing Inaccuracy.

I. INTRODUCTION

Scalability is one of the main challenges in QoS routing. There are many issues impacting on the scalability problem, such as the signaling overhead and the route computation schemes.

Concerning to the signaling overhead, a significant amount of the existing signaling messages is because the update procedure required to keep the network state databases correctly updated. Assuming source routing, QoS routing algorithms seek for the "optimal" route between source-destination node pairs based on the network state information obtained from the network state databases on such source nodes. The signaling overhead may be reduced by limiting the amount of updating messages. Unfortunately, reducing such updating messages leads to have inaccurate network state information. As a consequence routing is done according to outdated network state information, so increasing the blocking probability. There are many contributions in the literature proposing routing mechanisms that take into the unavoidable presence of such inaccuracies [1][2][3][4][5][6]. Closely related to the inaccuracy problem [7] and [8] propose a new algorithm named 'proportional routing', aiming to removing the update messages.

Concerning to the routing computation schemes, a major tool that has been explored in several previous studies to address the scalability problem is that of precomputation [9]. For example, the well-known hot potato routing scheme [10] predicts the best route to a destination node based on the delay information coming from that node. In [11] it is proposed to predict the future traffic load in a link through past measured samples of the traffic load in that link. In [12], a dynamic variant of hot potato routing is presented. All these contributions target to predict the incoming traffic load.

In this paper we propose the *Prediction-Based Routing* (PBR) mechanism. Despite we named our mechanism with the same name used in a previous work [19], it is worth to notice that they address a very different problem, proposing a cost function to predict the average queuing delay. Our PBR addresses the scalability problem by both, proposing a new computation scheme and reducing the signaling overhead. In short, unlike the previously mentioned precomputation schemes, they all predicting the incoming traffic load, the PBR focuses on predicting link and route availability. Moreover, the PBR mechanism also significantly reduces the signaling overhead because update messages are not required. Similar to the 'proportional routing', proposed in [7][8], in the PBR the routes are selected without taking into account network state information. However, in 'proportional routing' the route selection is based on flow blocking statistics collected locally, whereas in the PBR the route is predicted to be blocked or not based on both new tables, named Prediction Tables, and local information. The *Prediction-Based Routing* (PBR) mechanism has been already presented in [13] as a *Routing and Wavelength Assignment* (RWA) mechanism in the context of optical transport networks. The positioning papers in [14] [20] briefly introduce the PBR in the context of IP/MPLS networks.

The main objective of this paper is to describe in depth the behavior of the PBR mechanism. Based on the obtained results we justify how a heuristic mechanism such as the PBR correctly assigns the routes based on both the training of the Prediction Tables (PT), and the local network state information. We propose to apply the PBR mechanism for both, off demand and on demand route computation. In the first case, off demand computation, the routes are computed previously to the connection request. However, in the second case the routes are dynamically computed when a connection request reaches the source node. Inferred from the application of the PBR mechanism, off demand and on demand respectively, we propose two QoS routing algorithms, the *Predictive Selection of Route Fixed Alternate* (k-PSR_FA) and the *Predictive Selection of Route On demand* (k-PSR_R). While in the k-PSR_FA, the algorithm selects the route between a set of precomputed routes (we name these routes

fixed alternate), in the k-PSR_R, the algorithm dynamically calculates and selects the routes.

We also study in this paper the impact of the number of feasible routes on the PBR performance. In [15] and [16] it was described the problem involved when increasing the routes to be selected, since more feasible routes does not always imply better performance. This is due to the cost involved in using longer alternate routes.

This paper is organized as follows. In Section II the PBR mechanism and the algorithms inferred from the PBR mechanism are deeply described. In Section III we present a performance evaluation and, finally, in Section IV we conclude the paper.

II. PREDICTION-BASED ROUTING IN IP/MPLS NETWORKS

The *Prediction-Based Routing* is based on the well-known ideas of branch prediction developed in the context of computer architectures [17]. In this area, the main target boils down to find out whether a branch instruction will be taken or not before being processed. This is done to speed up the processor. The concepts used in branch prediction can be applied to a network scenario whenever substantial changes are included. The main components of our proposal are: the routing register, the prediction tables and the PSR algorithms.

A. PBR off demand: the k-PSR_FA algorithm.

The PBR mechanism presented in [14] is based on choosing the possible routes between different fixed alternate routes. That is, in the work done so far the route is chosen between 2 (k in general) static (fixed) and previously computed (precomputed) routes. The main reason motivating the use of fixed precomputed routes is to limit the number of Prediction Tables in the sources nodes; using fixed alternate routes we are limiting the number of Prediction Tables.

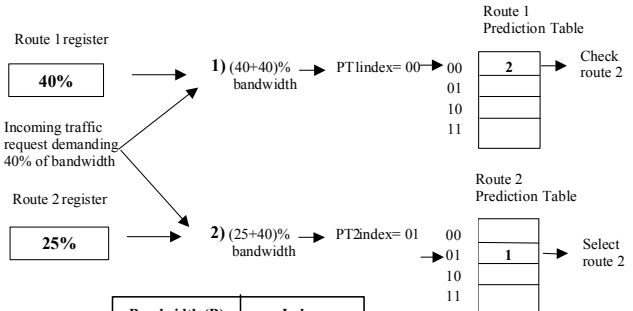
Unlike branch prediction where the history of prediction outcomes is stored in a register, in a network scenario it is necessary to keep the network state from the point of view of the source node. In order to achieve it, the PBR mechanism registers the amount of bandwidth that every source node allocates to every route originated on such a source node. For simplicity of exposition, we assume that the information about both available and used bandwidth is expressed in terms of a percentage of the total capacity of the end-to-end route. There is one register per route on every source node. These route registers are updated with information about assigned bandwidth from the point of view of these source nodes. One of the main characteristics of the PBR mechanism is that the register's updating process is achieved without distributing update messages. Because of the removal of these update messages, the bandwidth allocated in the route registers of the source nodes does not reflect the precise bandwidth assignment values.

The information about assigned bandwidth is used to access some tables (termed prediction tables, or PTs); hence it should be digitalized in order to constitute a proper table index. As an example, if we employ a single bit for digitalizing the bandwidth information, we can assign '0' to the index when the used bandwidth in the route is bigger than or equal to 50%, otherwise we assign '1'. Table in Fig.1 shows the index values for two bits.

Source nodes include one prediction table for every feasible route. Every route register has its corresponding PT. The PTs have different entries, each keeping the information about a different pattern by means of a two-bit counter. The use of two values to account for the availability or the unavailability has been widely studied in the area of branch prediction in computer architecture. As shown in [17] a two-bit counter provides significantly better accuracy than a one-bit counter. It is also shown that counters of more than two bits do not provide significantly better results; this is due to the "inertia" that can be built up with a large counter. A two-bit counter admits four values, namely 0, 1, 2 and 3. The prediction is done by reading the value of the two-bit counter, as follows. If the value is 0 or 1, the prediction result is to select the route associated with this counter; otherwise the prediction outcome is that this route is unavailable and should not be selected.

The number of entries of the prediction tables depends on the number of bits of the route registers. For example, if the route registers keep information about the used bandwidth in the route within two bits, then the number of entries of the prediction tables is 4.

Based on the PBR off demand mechanism, we propose the k-PSR_FA (*Predictive Selection of Route Fixed Alternate*) algorithm, being k the number of feasible routes. Fig.1 illustrates an execution of the algorithm. We assume that there are two precomputed shortest routes between every source-destination nodes pair, and that the assigned bandwidth is codified by two bits. In Fig.1 we depict the handling of a new request that demands 40% of bandwidth. We also assume that these shortest routes are link disjoint, if possible. Otherwise the shortest routes should share the minimum number of links. This is done because if the first route is predicted to be blocked, then the prediction is effectively to use a completely different route, since the source node does not know the identity of the link blocking the first route. Generally, the k-PSR_FA algorithm checks the k shortest routes in a computed order, according to the availability of their links. The information about the availability of the links does not represent the current picture of the network. Indeed, without updating, every node only knows how routes and links have



Bandwidth (B)	Index
75%<=B	0
50%<=B<75%	1
25<=B<50	2
B<25%	3

Figure 1. 2-PSR_FA performance, bandwidth codified with 2 bits.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

been used in the past. This information dictates the order by which the PTs are checked. Getting back to Fig. 1, the last information upon the first route is a used bandwidth of 40%. This used bandwidth is incremented by the requested bandwidth, i.e. 40%+40%. If the resulting figure is lower than 100 %, then the PT of the first route is checked, that is the counter of the corresponding entry is read; otherwise the next PT would be checked. In our example, the total bandwidth is 80% (>75%), so that the index used to access the first PT is 00. With this index, the PT of the first route is accessed and the counter is read. According to Fig.1, the value obtained after accessing the PT is 2, hence the decision made by the prediction process is to avoid the first route. Hence, the second route is examined. In this second route, the used bandwidth is 25%, so that the resulting figure is 40%+25%=65%. This means an index of 01. The PT of the second route is accessed with this index, obtaining a value of 1. According to this counter value, the algorithm selects this second route. We point out that the algorithm checks both the counter value of the PT and the availability of the node's output links towards each of the two routes, because the nodes always have updated information on the availability of their output links.

In Fig. 2 we present a short summary of the k-PSR_FA algorithm, for k=2. We call the functions that check the availability of route 1 and route 2 as Check(Route1), and Check(Route2), respectively. In the example, after checking the PTs of both routes, if the algorithm still has not selected any route according to the prediction, the algorithm will select the route by only checking the availability of the node's output links. These functions are termed CheckF(Route1) and CheckF(Route2), respectively.

The route registers at the source node are updated with the information about the used bandwidth for the source node in every route. In the example above, when the algorithm selects the second route, the new bandwidth used by this node in this second route will be 65%. It is important to note that this used bandwidth is just the value known by the node, which might be substantially different from the real bandwidth occupation. This is because, due to the lack of update messages, bandwidth changes produced by other source nodes allocating bandwidth on links of the same route are not reported

An important issue to be considered is that only the PT of the selected route is actually updated (or trained). Hence, if the connection is established, the corresponding counter on the PT is decreased, otherwise (i.e., the connection is blocked) the counter is increased. In our example, if the connection is successfully established, the counter of the entry 01 in the PT of route 2 will be 0, but if the connection is finally blocked the counter will be 2. The attempt of selecting the route by just checking the output availability when no route is assigned is done to unblock the PT counters. Indeed, if the route is selected and the connection can be established, then the corresponding PT counter of route 1 or route 2 is decreased, hence unblocking it.

B. PBR on demand: the k-PSR_R algorithm

As it was exposed earlier the potential problem of a PBR on demand mechanism is the amount of memory required by both the number of PTs and the size of the PTs. Remember that in the source nodes there is a PT for every possible route to every possible destination. In addition a large number of PTs negatively impacts on the computational cost.

```
New request demanding an X% of bandwidth.
Check(Route 1):
    The new bandwidth is added to the bandwidth kept in the route1 register (Y%).
    The total bandwidth is X+Y%.
        If (X+Y)% <=100% the PT of the first route is checked
            If(PT counter<2) and there is availability in the output link the
            algorithm selects the route1
            Else Check(Route 2).
        Else Check(Route 2)
Check(Route 2)
    The new bandwidth is added to the bandwidth kept in the route2 register (Z%).
    The total bandwidth is X+Z%.
        If (X+Z)% <=100% the PT of the second route is checked
            If (PTcounter<2) ) and there is availability in the output link the
            algorithm selects the route2
            Else CheckF(Route 1)
        Else CheckF(Route 1)
CheckF (Route 1):
    The new bandwidth is added to the bandwidth kept in the route1 register (Y%).
    The total bandwidth will be X+Y%.
        If (X+Y)% <=100%
            If there is availability in the output link the algorithm selects the
            route1
            Else CheckF(Route 2).
        Else CheckF(Route 2)
CheckF (Route 2):
    The new bandwidth is added to the bandwidth kept in the route1 register (Z%).
    The total bandwidth will be X+Z%.
        If (X+Z)% <=100%
            If there is availability in the output link the algorithm selects the
            route2
            Else No route is assigned
        Else No route is assigned
```

Figure 2. Summarizing the 2-PSR_FA algorithm

We address the problem of the memory requirements by means of both, reducing the PT size, and reducing the number of PTs. First, the PT size is reduced so that there is only one entry of two bit counter in every PT. As a consequence, it is not necessary to codify the requested bandwidth in a certain number of bits, since the algorithm does not consider it in the route selection (because there is only one entry on each PT). For every new connection request the corresponding PTs of the possible routes are accessed and read, independently of the requested bandwidth. This is done to both, limit the necessary amount of memory required, and simplify the execution of the algorithm. Second, the algorithm is able to calculate all the possible routes and then check all the possible PTs. However, to reduce even more the memory requirements we add a new parameter, R. R is the number of statically precomputed shortest routes. Then, in the source nodes, there is R PTs for every source-destination pair of nodes.

Despite the fact that the number of PTs has been reduced as well as their size, we are aware that a significant computational cost is needed to access all the feasible PTs. Hence, to reduce this computational cost we propose to limit the number of routes to be compared, varying the parameter k; being k the dynamically k-shortest routes with two-bit counter lower than 2 and with output link availability. We name the routing algorithm inferred from the PBR on demand mechanism, *Predictive Selection of Route On Demand* (k-PSR_R). In short, the k-PSR_R algorithm checks the k-

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

```
For(i=1 to R)  (R can be=all possible routes){
      While(CheckedRoutes<=k){
      If(two-bit_counter(Route(i)<2) and there is output link availability{
      CheckedRoutes++;
          If(Length(Route(i)<Length(AssignedRoute)) AssignedRoute=Route(i);
          If(Length(Route(i)==Length(AssignedRoute)){
              CheckLocalLinkAvailability:
              If(LocalLinkAvailability(Route(i))> LocalLinkAvailability(AssignedRoute))
                                                  AssignedRoute=Route(i);
                                                  }
```

If any route is assigned run the same algorithm without checking two-bit_counter values:

```
For(i=1 to R)  (R can be=all possible routes){
      While(CheckedRoutes<=k){
      If there is output link availability{
      CheckedRoutes++;
          If(Length(Route(i)<Length(AssignedRoute)) AssignedRoute=Route(i);
          If(Length(Route(i)==Length(AssignedRoute)){
              CheckLocalLinkAvailability:
              If(LocalLinkAvailability(Route(i))> LocalLinkAvailability(AssignedRoute))
                                                  AssignedRoute=Route(i);
                                                  }
```

Figure 3. Summarizing the k-PSR_R algorithm

shortest routes with two-bit counters lower than 2 and with output link availability between the first R shortest routes.

Once we have fixed the problem of the memory requirements we explain the k-PSR_R algorithm. The k-PSR_R algorithm looks, for every new connection request, the possible routes and reads the two-bit counter values as follows. Once the routes are calculated they are checked according to their length in number of hops. The first, shortest route, is checked. If its corresponding two-bit counter is lower than 2 and the corresponding output link has enough available bandwidth the route is provisionally selected. In any case, if the first route is selected or if it is not selected, the next, second route, is checked. If the second route has its two-bit counter lower than 2, the same hop length than the first, and output link availability, this second route is compared with the first. If the second route has more available bandwidth, this second route is now provisionally selected. This process finishes when k possible routes are considered (k shortest routes with two-bit counter lower than 2 and output link availability) or when R routes are checked. See in Fig. 3 a summary of this k-PSR_R algorithm. In order to make understanding easier we compare the k-PSR_R algorithm with the *Widest Shortest Path* (WSP) [18]. The k-PSR_R algorithm runs similar than the WSP but with two differences. The first is that the algorithm selects the widest shortest route between the routes with counter lower than 2 and output link availability. That is, it selects the widest shortest route in a graph where the routes with two-bit counters larger than 1 or no output link availability are pruned. The second difference is that k-PSR_R uses the local information on the source node about the link availability of the routes. This local information stands for the amount of bandwidth allocated by those connections originated by such a source node.

As in the k-PSR_FA algorithm, if the k-PSR_R algorithm does not select any route, the routes are checked as explained above but eliminating the restriction of two-bit counters lower than 2. See also summary in Fig. 3.

The k-PSR_R algorithm updates (or trains) the two-bit counters of the PTs according to the following. If the connection can be established the two-bit counter corresponding to that route is decreased, otherwise, the connection is blocked, the two-bit counter is increased.

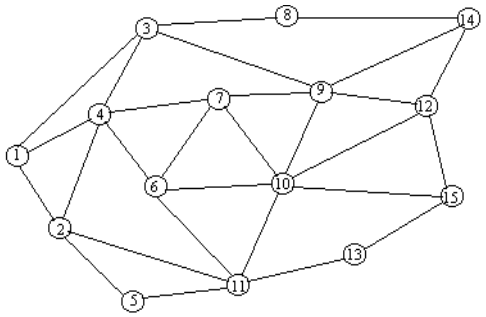


Figure 4. Topology used in the simulations

III. PERFORMANCE EVALUATION

In order to evaluate our proposal we compare the performance of the PBR mechanism with a well-known QoS routing algorithm the WSP. For every new incoming request, the WSP dynamically selects the route with the largest amount of available bandwidth among the shortest (i.e., minimum-hop) ones. All the performed simulations are obtained by applying both PSR algorithms and the WSP algorithm on the NSF topology, depicted in Fig. 4. We assume that in our simulations nodes 1, 2, 11, 12, 14 and 15 in Fig. 4 are source and destinations nodes. Connection arrivals are assumed to be Poisson, and all the links have the same available bandwidth, which is normalized to 100%. Each arriving connection requires a certain percentage of the total bandwidth. The holding and arrival times of the incoming requests are measured in units of time. All the connection requests have a averages holding time of 10 units and an average arrival time of 10 units. In order to change the traffic load, we change the average requested bandwidth (that is, the average value of all the requested bandwidths) demanded by the incoming requests from 10% to 25 %. We carry out three set of simulations. The first targets to find out the optimal number of bits needed to codify the bandwidth requirements in the k-PSR_FA algorithm. The second targets to evaluate the PSR performance, comparing it with the WSP algorithm. And finally we evaluate the impact of the parameters R and k on the k-PSR_R algorithm performance.

A. *Number of bits to codify the requested bandwidth in the k-PSR__FA algorithm*

As it is exposed in section II.A, the k-PSR_FA algorithm uses the bandwidth codification in the process of selection of the route. In this set of simulations we want to evaluate the impact on the k-PSR_FA performance when the number of bits used to codify the bandwidth changes. Notice that the length of the route registers and the number of PT entries depend on the number of bits used to codify the bandwidth. For example if the number of bits used to codify the bandwidth is 3, the route registers will have a length of 3 bits, and the PTs will have 8 entries each one, but if the number of bits is 0 (bandwidth is not codified) there will not be route registers and the PTs will have only one entry. We present in Table I the percentage of blocked connection, for the 4-PSR_FA algorithm for 0 (bandwidth is not codified), 1, 2 and

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

3 bits to codify the requested bandwidth, and for 10%, 15%, 20% and 25% of average requested bandwidth. We can see that for 10%, 15% and 20% the best results are for 0 bits; only for 25% the best results are for 2 bits. We obtain similar results for 2-PSR_FA. On average, for our range of traffic load the best results are usually for 0 and 2 bits. For simplicity and taking into account that 0 bits implies that there are not route registers, and only one PT of one two-bit counter per route is required in the source nodes, in the rest of the performance evaluation we only present results for 0 bits for the k-PSR_FA algorithm.

TABLE I. 4-PSR _FA % OF BLOCKED CONNECTIONS VS THE NUMBER OF BITS TO CODIFY THE REQUESTED BANDWIDTH.

Average Requested Bandwidth	Number of Bits 0	1	2	3
10%	0.3314%	0.3314%	0.3321%	0.40262%
15%	1.2682%	1.5041%	1.3434%	1.6959%
20%	3.9550%	4.8036%	5.5262%	5.2469%
25%	12.1375%	11.8983%	11.2713%	12.9306%

B. PSR algorithms performance

We compare the two PSR algorithms, k-PSR_FA and k-PSR_R, with the WSP algorithm. In the case of k-PSR_FA, we assume that the shortest routes are link-disjoint. Accordingly, for coherence when comparing the performance of all the algorithms we simulate also two WSP versions, WSP with off demand route calculation, named k-WSP_FA, with k link-disjoint routes, and WSP with on demand route calculation, named k-WSP_R.

In Fig. 5 we present results of the percentage of blocked connections versus the time between updating (in units of time) for 10%, 15%, 20% and 25% of average requested bandwidth. In these simulations we assume k=2 and k=4 for the k-WSP_FA and k-PSR_FA algorithms, and k=All and R=All for the k-WSP_R and k-PSR_R algorithms. In the off demand algorithms that use precomputed routes (k-WSP_FA and k-PSR_FA) the routes have been manually selected. For 2-FA, the two routes are the two shortest link disjoint. For 4-FA, the first 3 routes are the shortest link disjoint, while the fourth shares the minimum number of links with the other 3, because there are not 4 link disjoint routes in the topology simulated. Remember that the PSR algorithms do not vary their performance with the updating time because they do not need update messages.

From the obtained results we can conclude that both PSR algorithms outperform the WSP algorithms when the network state updating time is bigger than 5-10 units of time. Moreover, in some case the PSR outperforms the WSP even when updating is every unit of time. Remember that updating every unit of time, even every 5 or 10 units of time, is unaffordable from the point of view of the signaling overhead. On the other hand, the 4-PSR_FA algorithm outperforms in almost all the cases the k-PSR_R algorithm, except for 25% of requested bandwidth. This effect is also observable in the WSP algorithms. This can be explained because more routes to select does not always imply better performance [15]. The 4-PSR_FA algorithm only selects among 4 routes, but these routes has been previously and manually selected, being link disjoint the first three and sharing the minimum number of links the fourth. From this observation we argue that the selection of the fixed alternate routes is as important as the routing algorithm as stated in [8].

C. Adjusting parameters of the k-PSR_R algorithm

As it is exposed in section II.B we introduced the parameter R in the k-PSR_R algorithm to reduce the amount of required memory. R is the number PTs of one two-bit counter per source-destination pair in the source nodes. We also

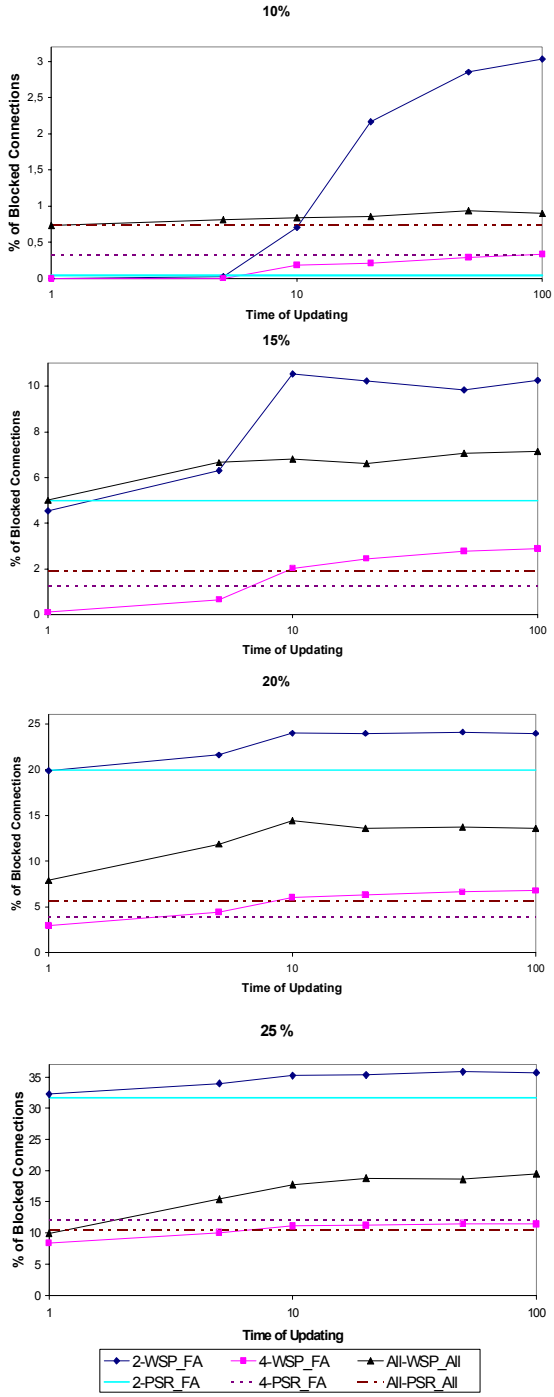


Figure 5. PSR versus WSP for traffic load of 10%, 15 %, 20% and 25% of average requested bandwidth.

This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE ICC 2006 proceedings.

introduced the parameter k to reduce the computational cost. The k-PSR_R algorithm selects the route between the k-shortest with availability in their output link and the two-bit counter lower than 2. In Table II we present results of the k-PSR_R algorithm being the R parameter, either all the possible routes, 100 routes, or 10 routes; and being the k parameter either R, 4 or 2. We observe that the results are the same if we consider all the possible routes, R=all, or if we consider R=100. Even for 10% of average bandwidth we also obtain the same results when considering only R=10 routes. On the other hand, in general, reducing k from all the possible routes to 4 and 2, the percentage of blocked connections decreases (except for 25%). The two above observations mean that we can reduce the number of PTs, R, without increasing the blocked connections. In addition, the results are in general better when reducing k. From these last observations together with the good results of the off demand algorithm, k-PSR_FA, we argue that it is possible, for every network topology and traffic characteristics, to find an optimal combination of the R and k parameters and also, as it is presented in [8], to find the best fixed precomputed routes to be selected.

TABLE II: % OF BLOCKED CONNECTIONS OF THE K-PSR_R VARYING R AND K

FOR 10% OF TRAFFIC LOAD

R/k	All	4	2
All	0.7068%	0.1600%	0.1667%
100	0.7068%	0.1600%	0.1667%
10	0.7068%	0.1667%	0.1667%

FOR 15% OF TRAFFIC LOAD

R/k	All	4	2
All	1.9201%	1.7468%	1.6201%
100	1.9201%	1.7468%	1.6201%
10	1.8934%	1.7468%	1.6201%

FOR 20% OF TRAFFIC LOAD

R/k	All	4	2
All	5.5937%	4.6736%	3.8269%
100	5.5937%	4.7136%	3.8269%
10	4.8000%	4.5070%	4.3002%

FOR 25% OF TRAFFIC LOAD

R/k	All	4	2
All	10.5140%	12.8875%	10.2940%
100	10.5140%	12.8875%	10.2940%
10	12.3408%	11.9008%	11.7074%

IV. CONCLUSIONS

We have presented the PBR mechanism, a precomputation approach based on prediction, to address the scalability problem of QoS routing. Our proposal is based on predicting routes availability not according to the network state information but according to its capacity of learning (training of the PTs). One of the main characteristics of this approach is that update messages are not needed, thus reducing significantly the signaling overhead. Simulation results show that the algorithms inferred from the PBR mechanism behave better than a standard QoS routing algorithm.

Also we have shown the importance of finding and storing the best precomputed routes when using off demand route calculation, to improve the performance of the routing algorithms. Our future work will be aimed to develop algorithms that dynamically store the best routes depending on the network and traffic characteristics.

ACKNOWLEDGMENTS

This work was partially funded by the MCyT (Spanish Ministry of Science and Technology) under contract FEDER-TSI2005-07520-C03-C02, the IST project E-Next under contract FP6-506869 and the CIRIT (Catalan Research Council) under contract 2005-SGR00481.

REFERENCES

[1] R.A.Guerin, A.Orda, "QoS routing in networks with inaccurate information: theory and algorithms", IEEE/ACM Transactions on Networking, Vol.7, nº.3, pp. 350-364, June 1999.

[2] D.H.Lorenz, A.Orda, "QoS routing in networks with uncertain parameters", IEEE/ACM Transactions on Networking, Vol.6, nº.6, pp.768-778, December 1998.

[3] G.Apostolopoulos, R.Guerin, S.Kamat, S.K.Tripathi, "Improving QoS routing performance under inaccurate link state information", Proc. ITC'16, 1999.

[4] S.Chen, K.Nahrstedt, "Distributed QoS routing with imprecise state information", Proc.7th IEEE International Conference of Computer, Communications and Nettworks, 1998.

[5] X.Masip, S.Sánchez, J.Solé, J.Domingo, "QoS Routing Algorithms under Inaccurate Routing Information for Bandwidth Constrained Applications", Proc. IEEE ICC 2003.

[6] T.Korkmaz, M.Krunz, "Bandwidth-Delay Constrained Path Selection Under Inaccurate State Information", IEEE/ACM Transactions on Networking, Vol.11, nº3, June 2003.

[7] Nelakuditi, Zhang, Tsang and Du, "Adaptative Proportional Routing: A Localized Approach", in IEEE/ACM Transactions on Networking (ToN) december 2002.

[8] Nelakuditi, Zhang and Du,, "On selection of candidate paths for proportional routing," in Computer Networks 44, 2004.

[9] A. Orda and A. Sprintson: "Precomputation Schemes for QoS Routing", IEEE/ACM Transactions on Networking vol. 11(4), pp.578-591, 2003.

[10] P. Baran, "On Distributed Communications Networks", IEEE Transactions on Communications, pages 1-9, 1964.

[11] T.Anjali, C.Scoglio, J.de Oliveira, L.C. Chen, I.F. Akyldiz, J.A. Smith, G.Uhl, A.Sciuto, "A New Path Selection Algorithm for MPLS Networks Based on Available Bandwidth Estimation", QofIS 2002.

[12] C. Busch, M. Herlihy, R.Wattenhofer, "Routing without Flow Control", ACM Symposium on Parallel Algorithms and Architectures, 2001.

[13] E. Marín-Tordera, X.Masip-Bruin, S.Sánchez-López, J.Solé-Pareta, J.Domingo-Pascual, "The Prediction-Based Routing in Optical Transport Networks", accepted for publication in Computer Communications Special Issue.

[14] E.Marín-Tordera, X.MasipBruin, S.Sánchez-López, "Prediction-Based Routing in IP/MPLS Networks",Infocom 2005 Student Workshop, 2005.

[15] Mitra and Seery, "Comparative Evaluations of Randomized and Dynamic Routing Strategies for Circuit-Switched Networks", IEEE Trans. on Communications, pp. 102-116, 1991.

[16] Kelly, "Routing and Capacity Allocation in Networks with Trunk Reservation", Mathematics of Operation Research, Volume 15, Issue 4, November 1990.

[17] J.E. Smith, "A study of branch prediction strategies", In Proc. of 8th International Symposium in Computer Architecture, Minneapolis 1981.

[18] R. Guerin, A.Orda and D. Williams, "QoS Routing Mechanism and OSPF Extensions", in Proceedings of 2nd Global Internet Miniconference (joint with GLOBECOM'97) 1997.

[19] Y.G. Kim, A. Shivari and P.S. Min, "Predition-Based Routing through Least Cost Dealy Constraint", 18th IPDPS, IPDPS'04

[20] E.Marín-Tordera, X.MasipBruin, S.Sánchez-López, A.Orda, J.Domingo-Pascual, "Prediction-Based Routing in IP/MPLS Networks", IV Workshop in GMPLS Networks, Girona Spain 2005.