PART II

QoS ROUTING IN IP/MPLS NETWORKS

This Part gives a clear analysis of what is the research problem tackeld in the Thesis as well as a broad overview of most recent and significant contributions existing in the literature focussing on the introduced problem. Along the different Chapters of this Part the proposed routing mechanism is formally stated, illustrated with different network examples, enhanced to include bandwidth constraints and finally evaluated in several network scenarios.

.

Chapter 4

The Routing Inaccuracy Problem in IP/MPLS Networks

Routing algorithms select routes based on the network state information contained in the network state databases, called *TED*s when including *TE* attributes. These *TED*s must be updated in order to include the network evolution, therefore modifying the network state information. Routing protocols must include an updating mechanism devoted to collect, distribute and maintain accurate network state information on each node along the network. However, maintaining accurate network state information cannot be always guaranteed, therefore generating inaccurate network state information. Separating two different components in the updating distribution process, namely the number of nodes and links able to generate update messages and the frequency at which these nodes and links generate these messages, leads to an initial classification of the origins of routing inaccuracy.

In a large, connection-oriented packet-switched network scenario it is extremely difficult to guarantee that the information contained in the network state database on

each node has been perfectly updated, due to the network dynamics and the huge amount of signalling messages needed to maintain accurate state information in all network elements. Some proposals exist, such as the *Private Network-to-Network Interface* (*PNNI*) [46] introduced by the ATM Forum to distribute network state information in a scalable form, based on a hierarchical process that allows the network to aggregate network state information based on the hierarchical level context.

However, this aggregation process implicitly introduces a loss of information, since information about physical nodes and links is not distributed. This loss of information introduces a certain degree of inaccuracy in the link state information, which cannot be easily measured since it depends on the aggregation method in use. For example, if an entire network is aggregated to a single virtual node, the available bandwidth of this virtual node can be obtained as either an average of the individual bandwidths or the best individual bandwidth or the worst individual bandwidth. In each case some information is lost, but the degree of inaccuracy is different. In fact, the larger the number of hierarchical levels the larger the inaccuracy.

Another important factor to be considered when analysing the causes of routing inaccuracy is the frequency at which update messages are sent throughout the network. In highly dynamic networks (plenty of *LSP*s are being constantly established/released) when link state changes are frequent, it is impractical to keep the link state databases correctly updated due to the large number of signalling messages needed to successfully perform this process. These messages generate an unnecessary reduction of available bandwidth in all links and consume processing cycles in all nodes. In order to reduce the frequency of these link state advertisements it is important to properly define when a node should trigger a message to inform the network about changes in one or more of its directly connected links. Hence, in order to reduce the number of update messages some triggering policies have to be defined; in this way routing accuracy varies depending on the values assigned to the triggering policies parameters. On the one hand, having the network perfectly updated leads to an increment in the number of update messages (signalling overhead), and on the other hand reducing the overhead leads to an increase in

routing inaccuracy. In fact, a trade-off exists between having accurate link state information and reducing signalling overhead.

Based on the mechanism used to trigger the update messages, the above-mentioned triggering policies can be classified as follows [47]:

1 *Threshold based updates*: This policy is based on a threshold (*tv*). If B^i_{adv} is the last advertised bandwidth of the link *i* and B^i_{real} is the current real bandwidth of that link, then an update message is triggered when

$$\left\| \frac{B^i_{adv} - B^i_{real}}{B^i_{adv}} \right\| > tv \qquad (1)$$

2 *Class based triggers*: Two examples are most representative, *Equal class based* updates and *Exponential class based* updates. Both are based on link partitioning. This partitioning is made either using a fixed bandwidth size *Bw* (*Equal class*), or adding to this value another constant value *f* (*Exponential class*). As a result, the total link capacity is divided into several equal size classes {*(0,Bw), (Bw, 2Bw),* . . .} or into several unequal size classes {(*0,Bw), (Bw, (f+1)Bw), ((f+1)Bw, (f*^{2}*+f+1)Bw),* . . .}. For these two policies, an update message will be sent only when the link capacity variation produces a change of class.

3 *Time based triggers*: In this case, updates are sent at fixed intervals. Also known as *hold-down timers*, these are usually implemented along with one of the above-described mechanisms, in order to reduce the number of update messages they generate.

In [48] a moving average technique is introduced as a substitute for the hold-down timers, with the objective of reducing the number of update messages. The authors show that by monitoring and filtering bandwidth utilization an average bandwidth utilization value can be computed, and this value can then be used to trigger update messages. Simulation results exhibit a more efficient reduction in signalling overhead when using this moving average technique.

In summary, in spite of the fact that the use of triggering policies to define when update messages must be sent reduces signalling overhead, it introduces a certain degree of routing uncertainty.

Some different sources of routing inaccuracy have been analysed so far. Now, the effects of this inaccuracy on the path selection process are described.

All routing algorithms rely on the accuracy of network state information to optimise the path selection process. When the information contained in the network state databases is not perfectly updated the routing process could potentially select a path that is unable to support the traffic requirements. In this case routing is done under inaccurate network state information; in other words, a certain degree of uncertainty exists in the routing information used to select the optimal path. As a consequence of this routing uncertainty, the connection blocking probability increases and the route selection process is definitely either badly performed or in the best case non-optimally performed. As it has been stated earlier, the problem of performing the routing decision under inaccurate routing information is known as the routing inaccuracy problem.

Having a certain degree of inaccuracy in the network state information is not of vital importance in traditional IP networks. This sentence must be correctly understood. The potential effects of having inaccuracy are the same in IP routing and in QoS routing. However, since in traditional IP routing only topological information, which does not suffer constant changes, is considered in the path selection process, usually link state databases may be updated by periodically (seconds or even minutes) sending update messages. Thus, because of the very low probability of changes in network topology, the degree of accuracy of the link state information is wholly dependent on the length of time between update messages. In other words, the risk of routing inaccuracy is very low.

A different situation occurs when more dynamic attributes are included in the network state information and changes in these parameters are often expected. This case appears when a certain QoS routing algorithm is implemented in a large, connection-oriented packet-switched network to select a path for a traffic flow requiring a certain QoS. In this case, since changes in the QoS parameters are

constantly expected, the number of update messages needed to keep the network state databases perfectly up-to-date is substantially greater. In fact, if the QoS parameter is the link available bandwidth, every time a connection is established or released this parameter will be modified. As a consequence, a large number of update messages flow throughout the network, which implies a non-desirable signalling overhead. The reduction of this overhead requires implementing additional mechanisms (i.e., triggering policies) in the routing protocol, which leads to an unfortunate consequence: network state information is not perfectly updated, so network state databases do not represent a current picture of the network.

In this case, the computations realized using inaccurate network state information could yield erroneous results. Subsequently, when this routing uncertainty is used to select the most suitable path through which traffic will be transported, it is possible that the selected route might not really have enough available resources to support the traffic requirements. Consequently, a certain incoming *LSP* demand that is initially allocated to a particular path would be rejected in the path set-up process, increasing the blocking probability.

At this point, it is important to note that the effects produced in the path selection vary depending on both the nature of the routing inaccuracy introduced and the QoS parameter required. First, the nature of the routing inaccuracy fundamentally affects the path selection process assuming that the inaccuracy is due to a triggering policy component. In this case, when the triggering policy in use does not include a hold-down timer it is possible to locate the current value of the QoS parameter into a range of values based on the last advertised value. This can be done because the details of the triggering policy's performance may be known allowing an analytical model to be generated to represent the routing inaccuracy effects. However, when the triggering policy includes a hold-down timer, since the time between two consecutive updates is usually large to reduce signalling overhead, the degree of inaccuracy is much larger and it is very difficult to make an appropriate estimation. These two triggering policy cases must be treated differently and different approaches must be implemented.

Second, the effects produced by routing inaccuracy in path selection will be different depending on the QoS parameter concerned: bottleneck requirements (bandwidth), or additive requirements (delay). Therefore, traffic flows with bandwidth requirements and traffic flows that impose an upper bound on the end-to-end delay should be handled differently. In fact, for traffic flows with bandwidth requirements, the optimal path could be found by extending existing standard routing algorithms with some modifications. For traffic flows with delay bounds, however, the problem is more difficult and several intractable issues appear when looking for the optimal path. Nevertheless, in both cases the objective is to find the optimal path that supports the incoming requirements considering that the routing information does not reliably represent the current network state. In summary, new routing algorithms, which account for routing inaccuracy in the path selection process, have to be sought.

This Thesis focuses on the routing inaccuracy problem when selecting paths with bandwidth requirements. A new routing mechanism is proposed to address the effects produced in global network performance because of having inaccurate routing information produced by using a certain triggering policy to reduce the signalling overhead needed to keep network state information perfectly updated in the edge nodes.

The network shown in Figure 3 illustrates the effects of selecting paths with bandwidth requirements under inaccurate network state information on the connection blocking probability.

Black link values (on the left) represent the residual available bandwidth of each link according to the network state information contained in the *TED*s on the edge nodes. Suppose that an incoming *LSP* demand reaches LSR1 demanding an *LSP* to LSR4 with 5 units of bandwidth. LSR1 selects two possible routes as shown in the Figure. Assume that for instance the shortest one is selected. Hence, a *Path* message is sent from LSR1 to LSR4 along LSR2 and LSR3, to set up the *LSP*. If there are enough resources a *Resv* message is sent back from LSR4 to LSR1 establishing the *LSP*. Now, traffic may be sent from LSR1 to LSR4 along the established *LSP*. Assuming that the updating mechanism is based on a certain triggering policy, it is

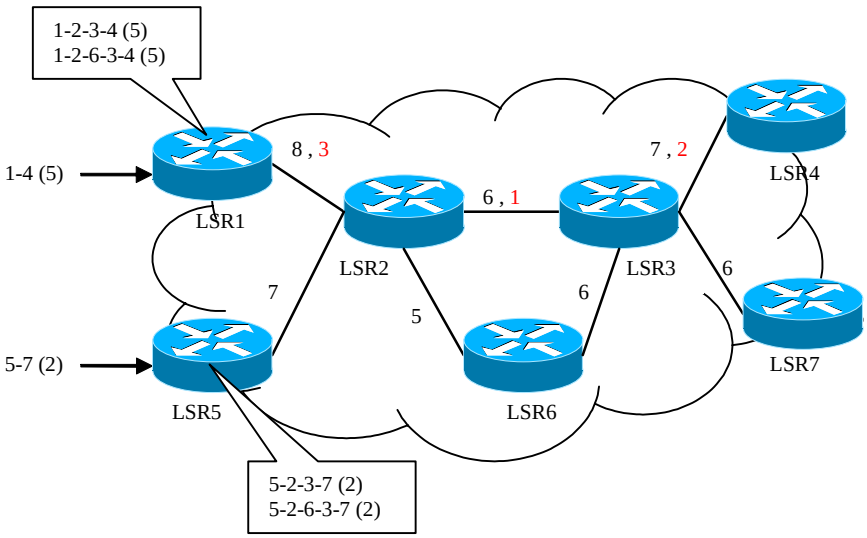


Figure 3. Routing inaccuracy effects in an IP/MPLS scenario

perfectly reasonable that update messages are not sent yet. Hence, in black (on the left) there are the last advertised residual bandwidth values of each link as read in the *TED*s and in red (on the right) there are the real bandwidth availability values once the first *LSP* has been established. Suppose that an incoming request reaches LSR5 demanding an *LSP* to LSR7 with 2 units of bandwidth. In this case the network state information from which LSR5 will select the path does not perfectly represent the real network state, which is the real bandwidth availability. LSR5 selects two possible routes and (as previously done) selects the shortest one. Hence, LSR5 sends a *Path* message from LSR5 to LSR7 through LSR2 and LSR3. When this *Path* message reaches LSR2 this node detects that there is not enough residual bandwidth on the output link to LSR3 to cope with the 2 units of required bandwidth. In this situation, LSR2 drops the *Path* message, so blocking the incoming *LSP* demand. The rejected *LSP* might be established if a routing algorithm considering inaccurate network state information was applied.

Chapter 5

Review of Existing Solutions

Several proposals dealing with finding a routing mechanism to improve routing performance under inaccurate routing information can be found in the literature. These works may be classified according to the following four approaches: (1) based on finding the path with the highest probability of having enough available resources to cope with the incoming requirements; (2) based on defining specific parameters, which reconcile the effects of having inaccurate routing information in the routing process; (3) based on improving the accuracy of network state information; and (4) based on addressing the routing inaccuracy effects introduced by the state aggregation in hierarchical networks.

Solutions following the first, probabilistic, approach include routing inaccuracy in the path selection process in order to find a path that guarantees the availability of enough resources to cope with the incoming requirements. The source node thus associates a certain stochastic behaviour to the QoS performance parameter, which is represented by a certain probability distribution function, so that an analytical model

incorporating this probability can be included in the routing process. It is important to realize that these probability distribution functions are not sent by the network nodes throughout the network as part of the link state advertisement: the source node builds these probability functions based on the received link state information.

Regarding the type of probability distribution function used, a first option would be to consider an exponential distribution, but due to the different sources of inaccuracy it would be very optimistic to suppose that this model would be useful for all the inaccuracy models. Finally, it is also important to take into account that the probability distribution function which models the routing inaccuracy can be artificially modified so that it includes any constraint desired to optimise the network performance (e.g. network load, paths successfully routed, etc.). A good example of routing algorithms based on this approach is *Safety Based Routing* [49], which is described later. Other significant contributions can be found in [50] and [51].

Examples of solutions following the second approach include *Ticket Based Probing* [52], a distributed multipath routing scheme based on probing multiple feasible paths simultaneously, and the *BYPASS Based Routing* mechanism introduced in this Thesis.

Solutions following the third approach lie in modifying the mechanism used to compute and utilize the link state attributes of each link in order to improve their accuracy. This set of solutions encompasses the works presented in [53] and [54] which are also described later on.

Finally, there are works that specifically deal with the effects that state aggregation introduces in the routing information. In this fourth group it is important to mention the crankback mechanism included in the *PNNI* protocol [46] to cope with the routing inaccuracy problem in an ATM scenario, which is presented in the final part of this Chapter. Other documents [55], [56], [57], [58], [59] present new aggregation techniques that reduce link state inaccuracy instead of substantially reducing the amount of data needed in the state.

Most significant routing mechanisms introduced in the literature fully thought to deal with the routing inaccuracy problem are now in detail described.

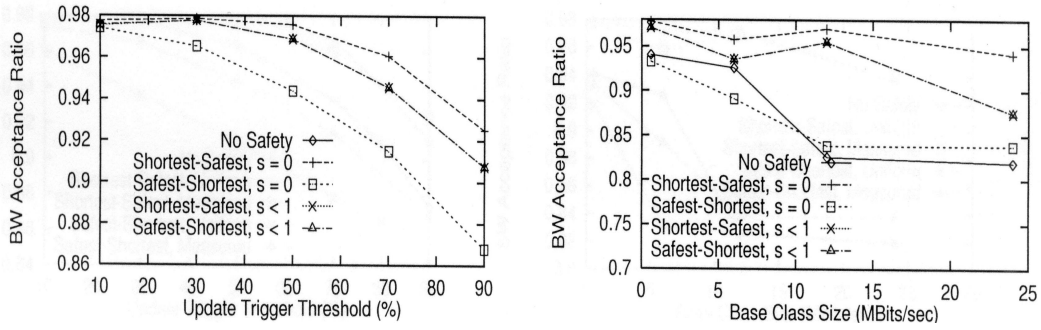
5.1 Safety Based Routing

Safety Based Routing (*SBR*), proposed by G.Apostopoulos et al in [49], assumes explicit routing with bandwidth constraints and on-demand path computation in such a way that a single path is computed when an incoming request reaches the source node. *SBR* is based on computing the probability that a path can support an incoming bandwidth request. This value, named safety (*S*), gives a measure of the probability that the total bandwidth required (b_{req}) is indeed available on the sequence of links composing each path (i.e., identifying those links having a higher probability of supporting the incoming request). This probability can be then used to classify every link, and obviously to find the safest path: the path having the best chance of supporting b_{req}.

Since the safety of each link is considered regardless of the other links in a path, the *safety* of a path (*S*) is computed as the product of every link present in that path. Once *S* has been computed it is included in the path selection process as a new parameter to be taken into account; thereby including a component of routing inaccuracy in the route selection process. As a result, the path that has larger probability *S* of having enough available capacity to provide the required bandwidth is selected.

Two different routing algorithms based on combining *S* with the number of hops are inferred from the *SBR* mechanism, namely the *Safest-Shortest Path* and the *Shortest-Safest Path*, the shortest-path being that path with the minimum number of hops. The *Safest-Shortest Path* algorithm selects that path with the largest *safety* among the shortest paths, while the *Shortest-Safest Path* algorithm selects the paths with largest *safety* and if more than one exists the shortest one is chosen.

To test these algorithms, in [49] a performance evaluation study is done by simulation. From this study it results that in terms of blocking probability, the *Shortest-Safest Path* (*SSP*) algorithm is the most effective for any of the triggering policies that were evaluated as shown in Figure 4.



(a) (b)

Figure 4. Bandwidth acceptance ratio in *SBR*: (a) Threshold; (b) Exponential classes

5.2 Explicit QoS Routing under Inaccurate Network State Information

This is one of the most known works published by R.Guerin and A.Orda in [50]. This work addresses the problem of selecting guaranteed QoS explicit paths under inaccurate network state information by selecting that path which has the largest probability of supporting the QoS incoming requirements. As explained earlier, the problem of selecting a path for traffic flows with bandwidth requirements and the same problem applied to traffic flows requiring end-to-end delay bounds must be separately solved, since they present a completely different degree of complexity. In this paper both the bandwidth constraint and the delay constraint are analysed.

Concerning to the bandwidth constraint the problem is to compute paths for those traffic flows with bandwidth requirements when the link state information contained in the source node can only be considered an inaccurate estimation of the bandwidth that is really available in the network components. This problem, known as *Most-Probable Bandwidth Constrained Path* (*MP-BCP*) can easily be solved by using a standard version of the *Most Reliable Path* (*MRP*) algorithm, which is based on computing the shortest paths for properly selected link weights. This solution efficiently addresses the problem of selecting paths for traffic flows with bandwidth requirements, but is not a good method for traffic flows requiring guaranteed end-to-end delay bounds, known as *Most-Probable Delay Constrained Path* (*MP-DCP*). In order to make the routing inaccuracy problem tractable when the objective is to guarantee that the end-to-end delay must be within a certain fixed delay threshold, the authors assume several approaches and generate two different models.

First, a "*rate based*" model is introduced. This model achieves the delay bound by ensuring a minimum service rate to the traffic flow. The main advantage of this model is that once the delay is mathematically represented, it is noticed that the end-to-end delay bound only depends on the available bandwidth on each link. Although this case and the bandwidth constraint case can be similarly addressed, they are not exactly the same, since the accumulative effect associated with delay is not produced in the bandwidth scenario. It is shown that the problem is intractable, although some solutions are presented for particular cases. Assuming that the obtained solution is not optimal but near-optimal, an algorithm named *QP* is introduced. However, this approach has the disadvantage that some constraints must be added, mostly regarding using schedulers that allow rates to be strictly guaranteed.

The second model is the "*delay-based*" model, which tries to guarantee a global delay in the selected path by concatenating the local delays associated with each node/link along the path. Therefore, in order to determine the end-to-end delay bounds, the information to be flooded throughout the network must be modified to include local delay, in such a way that the link state databases contain local delay information. Depending on the reliability of this information, the path selection process can be more or less affected, which explains why path computation is much more difficult in this model than in the *rate-based* model. Once the problem is mathematically defined in the paper, the authors conclude that although they have found tractable solutions for some cases, these cases are relatively limited; thus it is desirable to find a tractable general solution that can cope with most network conditions. In order to achieve this authors present a simplification, which reduces one general end-to-end delay guarantee problem to several minor problems. In fact the simplification is based on splitting the end-to-end constraint to a local delay constraint on each link.

5.3 QoS Routing with Uncertain Parameters for End-to-End Delay Constrained Traffic

Two different constraints, bandwidth and end-to-end delay, are analysed in the paper presented by D.Lorenz and A.Orda in [51]. The first constraint is defined as polynomial solvable, and the second is defined as computationally intractable. The

paper focuses on traffic with end-to-end delay requirements, providing new solutions and analysing their viability.

As previously defined, when end-to-end delay is the traffic flow QoS constraint, the inaccuracy is primarily due to the delay on each link, in such a way that a stochastic behaviour must be assigned as a parameter that represents the uncertainty. The paper considers a uniform distribution of the delay around the last advertised value, assuming the range to be defined by the threshold value used in the update process. However, the paper does not consider the exact mechanism used to obtain the delay probability distribution function. Hence, the main objective is to find the path most likely to cope with the delay constraint, which is named the *MP Problem*.

The complexity of the *MP Problem* depends on how the end-to-end delay constraint is split into several local end-to-end delay constraints. This process is not simple and the optimal partition is not easy to find. Hence, the initial *MP Problem* is extended to the *OP-MP Problem*, the *Optimally Partitioned MP Problem*. The authors find a solution for the *MP Problem* and the solid contribution of this work is that the solution is efficient for a wide class of probability solutions. This is achieved by defining a particular family of probability distributions (including normal and exponential distributions), where the family selection criterion is based on having a certain convexity property. Specifically, the paper extends an enhanced solution for a particular component of this family of distributions, the uniform distribution, generating an alternative algorithm for addressing the *OP Problem*, which is described in [50].

Once the authors present some solutions to deal with the *Optimal Partition* (*OP) Problem*, they go on to analyse the *OP-MP Problem*. The solution presented is based on using dynamic programming methods. In fact, the solution uses a modification of the *Dynamic-Restricted Shortest Path Problem* (*D-RSP*). The *RSP* problem is a well-known problem that aims to find the optimal path that minimizes the cost parameter among all the paths that satisfy the end-to-end delay constraint. Although the *RSP Problem* is NP-hard [60] authors propose a pseudopolynomial solution from which a new algorithm, *Dynamic-OP-MP*, is inferred. The main difference between the *Dynamic-OP-MP* algorithm and the *D-RSP* algorithm is the cost computation

method. As in the *OP Problem*, the *MP-OP Problem* is analysed in detail when a uniform distribution exists, yielding the *Uniform-OP-MP* algorithm.

Finally, an approach to obtaining a fully polynomial solution to deal with the *OP-MP Problem* is proposed. As in the last case, this approach is based on making some modifications to the *D_RSP* algorithm, such that the result is a non-optimal approximation (discrete solution) that introduces a bounded difference in terms of cost and success probability regarding the optimal solution. Basically, this solution interchanges the cost and delay roles in the algorithm.

5.4 Ticket-Based Distributed QoS Routing Scheme

The *Ticket-Based Distributed* QoS Routing mechanism was proposed by Chen and Nahrstedt in [52]. The authors focus on the NP-complete delay-constrained least-cost routing, and generate a routing algorithm designed to find the low-cost path, in terms of satisfying the delay constraint, by using only the available inaccurate or imprecise routing information. In order to achieve this purpose, the authors initially suggest a simple imprecise state model that defines which information must be stored in every node: connectivity information, delay information, cost information, and an additional state variable. Named delay variation, this additional state variable represents the estimated maximum change of the delay information before receiving the next update message.

It has to be noted that the imprecise model is not applied to connectivity and cost information, in order to simplify the process. The authors justify this assumption by saying that the global routing performance is not significantly degraded.

Then the authors propose a multipath distributed routing scheme, *ticket based probing*. *Ticket based probing* sends routing messages, *probes*, which are routed from a source *s* to a destination *d* according to the (imprecise) network state information available at the intermediate nodes, in order to find a low-cost path that fulfils the delay requirements of the *LSP* request. Each *probe* carries at least one ticket in such a way that by limiting the number of tickets, the number of probes is limited as well. Moreover, since each probe searches a particular path, the number of searched paths is also limited by the number of tickets. In this way, the trade-off

between signalling overhead and global routing performance may be controlled. Finally, based on this ticket based probing scheme, the authors suggest a routing algorithm to address the NP-complete delay-constrained least-cost routing problem, the *Ticket Based Probing* algorithm (*TBP*). Three algorithms are simulated in [52]: the flooding algorithm, the *TBP* algorithm and the shortest-path algorithm. Simulation results are represented using three parameters, the success ratio, the average message overhead, and the average path cost. The results obtained show that the TBP algorithm exhibits a high success ratio and a low-cost path, satisfying the delay constraint with minor overhead, and tolerating a high degree of inaccuracy in network state information.

5.5 Centralized Server Based QoS Routing

Unlike previous solutions, the solution proposed in [53] does not attempt to enhance the routing process assuming inaccurate network state information; rather, it seeks to eliminate the inaccuracy. In fact, the authors propose centralized server based QoS routing schemes which lead to both the elimination of the overhead needed to exchange network state update messages and the achievement of higher routing performance by utilizing accurate network state information in the path selection process. In this scenario, routing is handled by a route server and several

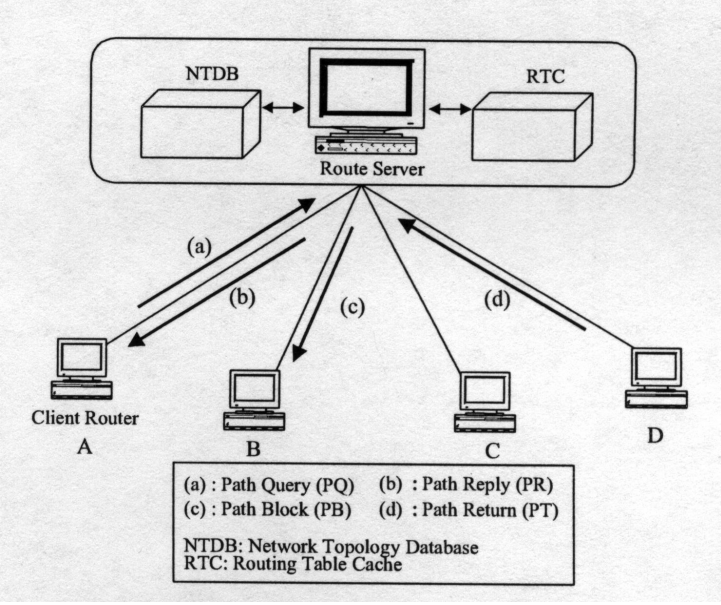


Figure 5. Centralized Server based QoS routing architecture

routers acting as clients: client routers send routing queries for each incoming request to the route server as shown in Figure 5. The route server stores and maintains two data structures: the *Network Topology Data Base* (*NTDB*), which contains link state information for each link in the network, and the *Routing Table Cache* (*RTC*), which stores the computed path information.

Although the main idea is derived from the mechanism suggested in [61], these new schemes use a different network state information collection method. Instead of collecting the link state information from the other routers, in this new approach the proposed router server updates and maintains a link state database as the paths assigned to or returned back from a certain flow. The main aspects to be considered in this centralized scheme are: (1) the processing load and storage overhead required at the server; (2) the protocol overheads to exchange the router queries and the replies between the server and the remote routers that act as clients; and (3) the effects produced when the server becomes either a bottleneck point or a single point of failure. The authors suggest various alternatives to reduce the loads and overheads, but leave the last aspect for future work.

There are two algorithms used to compute the path: a modification of Dijkstra's algorithm, and the Bellman-Ford algorithm with QoS extensions. Thus, assuming the existence of a certain locality in the communication pattern, a large number of source-destination pairs are expected to be unused. For this reason the authors use a path caching approach to reduce the path computation overhead. There are two parameters that limit the size of the *RTC*: *K*, defined as the maximum number of entries in the *RTC* (source-destination pairs), and *n*, defined as the maximum number of paths for each source-destination pair.

The server based QoS routing schemes introduced in this document are evaluated by simulation. Two types of results are obtained, one to show the effects of these schemes on path computation overhead, and the other to compare the proposed schemes with other QoS distributed routing schemes. The simulation results show that a simple path-caching scheme substantially reduces path computation overhead when considering locality in the communication pattern. Furthermore, according to

the simulation results the proposed schemes perform better than distributed QoS routing schemes while keeping a similar overhead.

5.6 A localized QoS Routing Approach

Document [54] focuses on localized QoS routing. The main advantage of using localized approaches for QoS routing is that no global network state information exchange among network nodes is needed, hence reducing the signalling overhead. In fact, path selection is performed in the sources nodes according to their local view of the global network state. However, the main difficult in implementing any localized QoS routing scheme is how the path selection is done based only on the local network state information collected in the source nodes. In order to address this problem the authors present a new adaptive proportional routing approach for designing localized QoS routing schemes. This approach provides an idealized proportional routing model, where all paths between a source-destination pair are disjoint and their bottleneck link capacities are known. However, this situation is not usually the case, and to address the effects produced by this deviation from reality, the concept of the *virtual capacity* of a path is introduced. The *virtual capacity* concept provides a mathematically sound way to deal with links shared among multiple paths. Based on this concept a theoretical scheme, *Virtual Capacity based Routing* (*VCR*), is described. Simulation results obtained show how the *VCR* scheme adapts to traffic load changes by adjusting traffic flows to the set of predefined alternative paths. However, the authors describe two significant difficulties related to *VCR* implementation, which suggest a new, easily realizable implementation of the *VCR* scheme: *Proportional Sticky Routing* (*PSR*). The *PSR* scheme can be viewed as operate in two stages: proportional flow routing and computation of flow proportions.

PSR proceeds in cycles of variable lengths. During each cycle, any incoming request can be routed along a certain path selected among a set of eligible paths, which initially may include all candidate paths. A candidate path becomes ineligible depending on the maximum permissible flow blocking parameter, which determines how many times this candidate path can block a request before becoming ineligible. When all candidate paths become ineligible a cycle ends and all the parameters are

reset to start the next cycle. An eligible path is finally selected to route the traffic depending on its flow proportion: the larger the flow proportion, the larger the chances of being selected. Simulation results show that the PSR scheme is simple, stable and adaptive, and the authors conclude that it is a good alternative to global QoS routing schemes.

5.7 Crankback and Fast Rerouting

The crankback and fast rerouting mechanisms were included in the ATMF *PNNI* [46] to address the routing inaccuracy problem due to fast changes in resource availability, and due to information condensation in a hierarchical network structure.

Assuming that a hierarchical network is divided into different peer groups, the mechanism is based on sending a *Release* message in the reverse path from the node that cannot forward the set-up message (i.e., the node that detects the set-up message blocking) to a node able to compute an alternate path in the same Crankback Level (i.e., the peer group level). The *Release* message contains a *Crankback Information Element* (*IE*), which indicates the blocked link and the *Crankback Level*. When a node in the reverse path receives a *Release* message it may either compute an alternate path or further crankback the message. This decision depends on alternate path availability and the routing information contained in that node. In this way, blocked links can be avoided by using an alternate path. When the crankback process returns back to the source node demanding the connection and this node also fails to find a path to the destination, the connection request is blocked or rejected. Therefore, the crankback mechanism does not guarantee a successful connection set-up, while consuming both much CPU-time in the nodes as control data.

This mechanism can be implemented over both a datagram network, where path selection and fast rerouting are dynamically performed by the nodes; and an explicitly routed network, where the selected path and the alternate routes are computed by the source nodes.

Chapter 6

The BYPASS Based Routing Mechanism

In this Chapter we describe a new QoS routing mechanism, *BYPASS Based Routing* (*BBR*) [62], which aims to reduce the effects of the routing inaccuracy problem when considering bottleneck requirements, the increase of the bandwidth blocking ratio and non-optimal path selection, in an *IP/MPLS* scenario. The main target is to obtain a routing mechanism that improves the behaviour obtained when applying other existing solutions. Assuming that the (*Safety Based Routing*) *SBR* [49] is the best solution to address the effects produced in global network performance when performing the path selection process under inaccurate available bandwidth information, the *BBR* mechanism is proposed as a new solution to improve *SBR*'s behaviour. The main concept of the *BBR* mechanism is the dynamic bypass concept, which is based on computing more than one feasible route to reach the destination. *BBR* instructs the ingress node to compute both the working route and a number of *bypass-paths:* paths that bypass those links that potentially might not be able to cope

with the incoming traffic requirements. Nevertheless, as we discuss below, only those paths that bypass links that truly lack enough available bandwidth to support the required bandwidth are set up.

Note that the idea of the *BBR* mechanism is derived from protection switching for fast rerouting, discussed in [63]. However, unlike the use of protection switching for fast rerouting, in this proposal both the working and the alternative path (*bypass-path*) are simultaneously computed but not set up. Alternative paths are only set up when required.

There are three main components in the *BBR* mechanism: to decide which links should be bypassed, to compute the *bypass-paths*, and to decide when *bypass-paths* must be used. These three components may be summarized as follows:

1) ***Obstruct-Sensitive Links***: A new policy must be added in order to find those links (*Obstruct-Sensitive Links*, *OSL*s) that might not be able to support the traffic requirements associated with an incoming *LSP* demand. This policy must guarantee that whenever a set-up message sent along the explicit route reaches a link that does not have enough residual bandwidth to support the required bandwidth, this link had previously been defined as an *OSL*.

2) ***Working path selection*:** Using *BBR*, two different routing algorithms can be initially analysed. These algorithms are obtained from the combination of the Dijkstra's algorithm (in terms of hopcount) and the *BBR* mechanism. Therefore, two different strategies may be applied:

   - The *Shortest-Obstruct-Sensitive Path* (*SOSP*), computing the shortest path among all the paths that have the minimum number of *Obstruct-Sensitive Links*.
   - The *Obstruct-Sensitive-Shortest Path* (*OSSP*), computing the path that minimizes the number of *Obstruct-Sensitive Links* among all the shortest paths.

3) ***Bypass-paths*, calculation and utilization:** Once the working path is selected the *BBR* computes the *bypass-paths* needed to bypass those links in the working path defined as *OSL*. When the working path and the *bypass-paths* are computed, the working path set-up process starts. Thus, a signalling

message is sent along the network following the explicit route included in the set-up message. When a node detects that the link through which the traffic must flow does not have enough available bandwidth to support the required bandwidth, it sends the set-up signalling message along the *bypass-path* that bypasses this link. Thus, the set of bypassed links must always be defined as *OSLs* so that a feasible *bypass-path* exists. Moreover, it is important to note that the *bypass-path* nodes are included in the set-up signalling message as well (i.e., *bypass-paths* are also explicitly routed). A possible option to implement the mechanism used to set up the *bypass-paths* is described later in this Chapter. In order to minimize the set-up message size, *bypass-paths* are removed from the set-up message when the links that they are intended to bypass have been traversed.

6.1 Description of BYPASS Based Routing

Let *G(N,L,B)* describe a defined network, where *N* is the set of nodes, *L* the set of links and *B* the bandwidth capacity of the links. Suppose that a set *P* of source-destination pairs *(s,d)* exists, and that all the *LSP* requests occur between elements of *P*. Let b_{req} be the bandwidth requested in an element *(s,d)* $\hat{I}$ *P*.

***Rule 1*:** Let $G_r(N_r,L_r,B_r)$ represent the last advertised residual graph, where N_r, L_r and B_r are the remaining nodes, links and residual bandwidths respectively at the time of path set-up. Let L^{os} be the set of *OSLs* (l^{os}), where the elements of L^{os} are determined depending on the triggering policy in use. Therefore,

- Threshold policy: Let b^i_r be the last advertised residual bandwidth for a link l_i, and let *tv* be the threshold value. This link l_i is defined as an *OSL*, l^{os}_i if

$$l_i = l^{os}_i \mid l^{os}_i \in L^{os} \Leftrightarrow b_{req} \in (b^i_r(1\text{-}tv), b^i_r(1+tv)] \; . \qquad (2a)$$

- Exponential class policy: Let $B^i_{l\_j}$ and $B^i_{u\_j}$ be the minimum and the maximum bandwidth values allocated to class *j* for a link l_i. So, l_i is an *OSL*, l^{os}_i if

$$l_i = l^{os}{}_i \mid l^{os}{}_i \in L^{os} \Leftrightarrow b_{req} \in (B^i{}_{l\_j}, B^i{}_{u\_j}] \,. \qquad (2b)$$

Rule 2: Let L^{os} be the set of *OSLs*. Let i_j and e_j be the edge nodes of a link $l^{os}{}_j$ $\hat{I}$ L^{os}. Let l_k be a link adjacent to $l^{os}{}_j$. The edge nodes of the *bypass-paths* to be computed are

$$(i_j, e_j) \qquad \Leftrightarrow \quad l_k \notin L^{os} \qquad (3a)$$

or

$$(i_j, e_k) \text{ and } (i_k, e_k) \qquad \Leftrightarrow \quad l_k = l^{os}{}_k \in L^{os} \,. \qquad (3b)$$

In this way two or more adjacent *OSLs* could be bypassed by a single *bypass-path*, as shown in Figure 6.

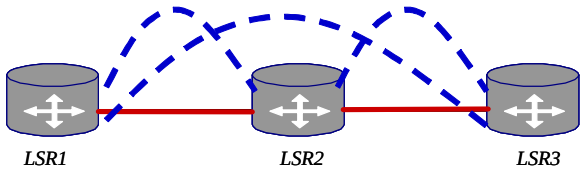


Figure 6. Rule 2 illustration

Three different situations exits: (1) Link LSR1-LSR2 is defined as an *OSL* but the adjacent link LSR2-LSR3 is not an *OSL*. In this case, if LSR1 detects that the requested bandwidth is unavoidable on the output link, forwards the set-up message along the *bypass-path* to LSR2; (2) Link LSR1-LSR2 is not defined as an *OSL* but link LSR2-LSR3 does. In this case, the *bypass-path* from LSR2 to LSR3 is used to send the set-up message when is needed; (3) Link LSR1-LSR2 and link LSR2-LSR3 are defined as *OSL*s. In this situation if LSR1 detects that requested bandwidth is unavoidable on the link of the primary path, it sends the set-up message along the *bypass-path* from LSR1 to LSR3 instead of LSR1-LSR2.

In accordance with these rules, a brief description of the *BBR* mechanism is presented in Figure 7. Steps 4 and 5 should be explained in detail. Once a link is defined as an *OSL*, the *BBR* mechanism computes the *bypass-path* that bypasses this link. The *bypass-paths* are computed according to *SOSP*, the shortest path among those paths minimizing the number of *OSLs*. Other criteria could be used to select the

BYPASS BASED ROUTING (BBR)

Input*:* The input graph $G_r(N_r,L_r,B_r)$. The *LSP* request is between a source-destination pair *(s,d)* and the bandwidth requirement is b_{req}.

Output: A route from *s* to *d* with enough *bypass-paths* to bypass the routing inaccuracy effects in the *obstruct-sensitive links.*

Algorithm:

1. Mark those links that are defined as *obstruct-sensitive link (OSL)* according to Rule 1.
2. Depend on the algorithm to be used, *OSSP, SOSP:*

   SOSP *(shortest-obstruct-sensitive path)*:
   - Compute the weight of a link *l* as
     $$w(l) = 1 \Leftrightarrow l \in L^{os}, \qquad w(l) = 0 \Leftrightarrow l \notin L^{os}$$
   - Apply Dijkstra's algorithm to select the paths $p \hat{I} P$ that minimize the number of *OSLs* by using *w(l)* as the cost of each link
   - If more than one path *p* exists selects the path that minimizes the number of hops

   OSSP *(obstruct-sensitive-shortest path)*:
   - Apply Dijkstra's algorithm to select the paths $p \hat{I} P$ that minimize the number of hops by using *w(l)* =1 as the cost of each link.
   - If more than one path *p* exists selects the path that minimizes the *OSLs.*
3. Compute a *bypass-path* for all the *OSLs* included in the selected path according to Rule 2.
4. Decide which *bypass- paths* must be used in accordance with real available resources in the path setup time.
5. Route the traffic from *s* to *d* along the setup path.

Figure 7. BYPASS Based Routing mechanism

bypass-paths, such as to simply apply *OSSP* or to maximize the residual available bandwidth. These different approaches are left for further studies

There are two main factors contributing to the complexity cost of *BBR*. First, selecting the shortest path by using a binary-heap implementation of the Dijkstra algorithm introduces a cost of $O(L \cdot \log N)$. Second, additional cost is introduced by the *bypass-path* computation. Assuming that the *bypass-path* cannot include a network element which is also included in the working path, G(V, E) represents the reduced network, where $V < N$ and $E < L$. Hence, a factor of $O(E \cdot \log V)$is added in order to compute one *bypass-path*. In addition, since a variable number M of *bypass-paths* may be computed along a working path, the cost is $O(M(E \cdot \log V))$. However, $\hat{M}$ being an upper bound of the number of computed *bypass-paths* along a working path, the complexity reduces to $O(\hat{M}(E \cdot \log V))$: effectively to $O(E \cdot \log V)$. Hence, the complexity is $O(L \cdot \log N) + O(E \cdot \log V)$. This expression may be finally reduced, considering that the *bypass-paths* are computed based on a reduced graph. Therefore, the complexity is $O(L \cdot \log N)$.

6.2 *Bypass-path* Signalling

At this point a new solution for signalling *bypass-paths* is introduced. This solution is borrowed from [64], where a solution for establishing and signalling alternative paths for fast rerouting in *IP/MPLS* networks is suggested. Assuming that the *Resource Reservation Protocol* (*RSVP*) is used as the *Label Distribution Protocol* (*LDP*), there are two main messages (*Path* and *Resv)* involved in the path set-up process. Hence, while the *Path* message flows downstream from the source node to the destination node requesting the label allocation, the *Resv* message flows generally upstream from the destination node to the source node establishing the path by allocating the label on each node along the selected path. A new object containing the intermediate node addresses along the path (the *Explicit Routing Object* (*ERO*) [13]) is added to the *Path* message when using explicit routing. The *ERO* is unique and will be made up of a set of sub-objects with the addresses that the *Path* message must follow to reach the destination node.

The *BBR* mechanism assumes that the source node detects *OSL*s and then a *bypass-path* route is computed for each one of these links. These *bypass-paths* are then sent along with the working path in the *Path* message, and each intermediate node selects which route use for the incoming *Path* message depending on its real resource availability. Since only those links defined as *OSL*s must be associated with a *bypass-path*, a new field is added to the *ERO* sub-objects to signify an *OSL*, and a *bypass-path* should be explicitly signalled for this link as well. In practice, the node upstream of the link defined as an *OSL* is the node where the link is marked as an *OSL*. This new field, named *Protect* (P) to maintain the nomenclature used in [64], consists of one bit that is set ‘true’ when the output link is an *OSL*. Therefore, we can say that a node is *protected* when the output link on which the traffic must flow has been identified as an *OSL*. In Figure 8 the format of an IPv4 address for a sub-object of the *ERO* including the *Protect* field is shown.

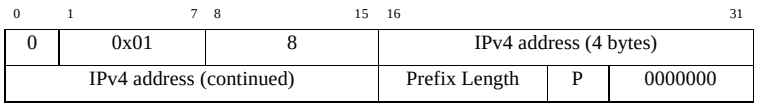


Figure 8. IPv4 sub-object of the *ERO*

Once a link is defined as an *OSL*, the *Path* message must carry the computed *bypass-path* that could be used by the protected node to bypass the link when there are not enough resources to cope with the traffic requirements. The addresses of the intermediate nodes on this *bypass-path* must also be explicitly piggybacked in the *Path* message. Therefore, the sub-objects of the *ERO* must be again modified, adding the new *Bypass-Path Address* (*BPA*) field. It is a variable length field, always a multiple of 32 bytes, which is only useful when the node has been defined as protected. In this case, the *BPA* field provides the *n* addresses that make up the *bypass-path*, from the protected node to another node along the working path, according to the routing algorithm's decision. The *BPA* field addition is shown in Figure 9. Two cases are depicted: in (a) the node that receives the *ERO* (@IPv4-i) is not a protected node (*P* = 0), so the output link where the traffic will be sent is not an *OSL*; in (b) the node is protected (*P* = 1) and the *BPA* field is shown. In this case the *BPA* consists of two IPv4 addresses (@IPv4-1 and @IPv4-2), which explicitly configures the *bypass-path* that bypasses the link directly connected to this protected node.

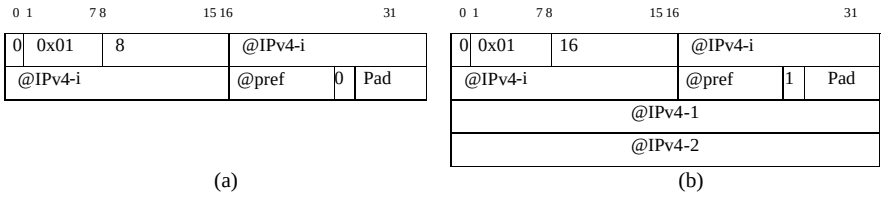


Figure 9. IPv4 sub-object of the *ERO*: (a) non-protected node; (b) protected node

It is important to note that a different situation may be found. In fact, even though a node is defined as protected, the existence of a *bypass-path* is not always guaranteed. This case matches the scenario in which there is not another feasible route between the edge nodes of an *OSL*. In this situation, the *P* field is set but the *length* field is left unchanged. This weakness of the *BBR* mechanism is addressed in Chapter 8.

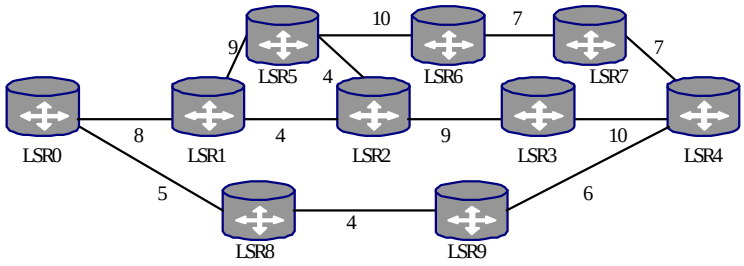


Figure 10. Network topology used to illustrate the *BBR* behaviour

6.3 Example for Illustrating BBR Behaviour

Before analysing the suggested algorithms in a large topology, we can test *BBR* performance in the simple topology shown in Figure 10, which shows the network topology, where the number associated with each link shows the residual units of bandwidth available. An *Exponential class* triggering policy is used, with $f = 2$ and $Bw = 1$ (as used in [49]), such that the resulting set of classes on each link are {(0,1], (1,3], (3,7], (7,15],…}. Moreover, an incoming *LSP* request is assumed to demand b_{req} of 4 units of bandwidth between LSR0 and LSR4.

In order to compare the *BBR* mechanism with other related work, the *Shortest-Safest Path (SSP)* algorithm presented in [49], which computes the path based on maximizing the "*probability of success*" of supporting the bandwidth requirements, is chosen as the sample routing algorithm. Thus, the algorithms tested in this example are *SSP*, *WSP* (as explained in Chapter 3, it selects the widest path among the shortest ones in the pruned graph which only contains links supporting the requested bandwidth), *SOSP*, and *OSSP*. The shortest path algorithm (SP) implemented in the OSPF routing protocol is used as a routing algorithm that does not consider routing inaccuracy when selecting the path. Table 2 describes the link QoS parameters used to compute the path, where B_t, *Class* and *S* are the bandwidth, class and *safety* associated with each link. The *S* value has been computed according to the expressions found in [49]. Note that *S* represents the probability that the requested amount of bandwidth is indeed available on a given link. Using this information the *BBR* mechanism is applied.

Table 2. Link QoS attributes

Link	B_t	*Class*	*S*
0-1	8	7,15	1
1-2	4	3,7	0,75
2-3	9	7,15	1
3-4	10	7,15	1

Link	B_t	*Class*	*S*
1-5	9	7,15	1
5-2	4	3,7	0,75
5-6	10	7,15	1
6-7	7	3,7	0,75

Link	B_t	*Class*	*S*
7-4	7	3,7	0,75
0-8	5	3,7	0,75
8-9	4	3,7	0,75
9-4	6	3,7	0,75

Table 3 shows different possible routes from LSR0 to LSR4, including the number of hops *H*, the number of *Obstruct-Sensitive Links OSL*, the minimum last advertised residual bandwidth b_r^{min}, and the *safety* parameter *S* for each path. Different paths are selected depending on the algorithm in use, as shown in Table 3. It is important to note that the *SOSP* and the *SSP* algorithms select the same route. Assuming that this is the normal *SOSP* behaviour, the blocking probability obtained by both algorithms will be the same. However, although the *SOSP* and the *SSP* algorithms select the same route, the key difference between the algorithms is the use of *bypass-paths* when needed. In fact, the *OSL* definition is a different manner of representing the *safety* of a link, which provides rerouting capabilities to some intermediate nodes. Hence, *bypass-path* utilization will reduce the blocking probability. Moreover, the *OSSP* algorithm exhibits a different behaviour (i.e., selects a different path) in the example. No prediction about bandwidth blocking results in comparison with the *SSP* algorithm can be made in advance, due to the possibility of *bypass-path* utilization.

Table 3. Feasible routes and selected paths depending on the algorithm in use

Id	*Route (LSR)*	*H*	*OSL*	b_r^{min}	*S*
a	0-1-2-3-4	4	1	4	0.75
b	0-1-5-6-7-4	5	2	7	0.56
c	0-1-5-2-3-4	5	1	4	0.56
d	0-8-9-4	3	3	4	0.42

Alg	*Path*
SP	d
WSP	d
OSSP	d
SOSP	a
SSP	a

Once feasible routes have been computed, the *bypass-path* selection process starts. If the *SOSP* algorithm is in use there is only one *OSL* in the route *a*, which can be bypassed by the path LSR1, LSR5 and LSR2. However, when the *OSSP* algorithm is in use, the process is much more complex since there are some *OSLs* that cannot be bypassed, e.g. link LSR8-LSR9. In this case *BBR* cannot be applied. A method for bypassing *OSLs* that do not have *bypass-paths* between their edge nodes

must be sought. One approach to address this case is to find different edge nodes for the *OSL* (the *BYPASS Discovery Process*), which is analysed in Chapter 8. Currently, as pointed out above, the *bypass-paths* are always computed by minimizing the number of *OSLs*.

Finally, after computing the *bypass-paths*, a path set-up message is sent along the working path. Each node checks the real available link bandwidth, and depending on this value the set-up message is sent through either the working path or the *bypass-path*.

6.4 Performance Evaluation

In this section we compare by simulation the *SOSP* and the *OSSP* algorithms inferred from the *BBR* mechanism with the *WSP* and the *SSP* algorithms. We exclude *SWP* due to its worse performance shown in [65]. The parameters used to measure the algorithms' behaviour are the routing inaccuracy and the blocking ratio.

- ***Routing Inaccuracy:*** This parameter represents the percentage of paths that have been incorrectly selected, defined as:

$$\text{routing inaccuracy} = \frac{\text{number of paths incorrectly selected}}{\text{total number of requested paths}} \qquad (4)$$

  A path can be incorrectly selected because of two factors. The first is an *LSP* request rejection when a route with enough resources is available to support that demand. The second factor is the blocking of an *LSP* that was initially routed by the ingress node but later rejected due to insufficient bandwidth in an intermediate link.

- ***Blocking Ratio*****:** We use the bandwidth-blocking ratio defined as:

$$\text{bandwidth blocking ratio} = \frac{\sum_{i \in \text{rej\_LSP}} \text{bandwidth}_i}{\sum_{i \in \text{tot\_LSP}} \text{bandwidth}_i} \qquad (5)$$

  where *rej_LSP* is the set of blocked demands, and *tot_LSP* is the set of all requested *LSP*s.

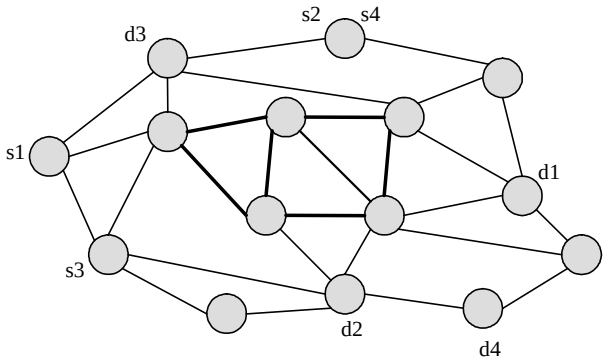


Figure 11. Network topology used in the simulations

The simulations are performed over the network topology shown in Figure 11, borrowed from [24], using the ns2 simulator extended with *MPLS* and *BBR* features. Two link capacities are used: 622 Mb/s, represented by a light line; and 2.5 Gb/s, represented by a dark line. The source nodes (s) and the destination nodes (d) are those shown in Figure 11. Every simulation requests 2000 *LSP*s from s_i to d_i, which arrive following a Poisson distribution, where the requested bandwidth is uniformly distributed between 1 Mb/s and 5 Mb/s. The holding time is randomly distributed with a mean of 60 sec. The *Threshold based triggering policy* and the *Exponential class based triggering policy* with $f = 2$, are implemented in the simulation. Let *n_bp* be the number of *bypass-paths* that may be computed per route. In order to reduce the computational cost, in these simulations three *bypass-paths* may be computed per route, so *n_bp* = 3.

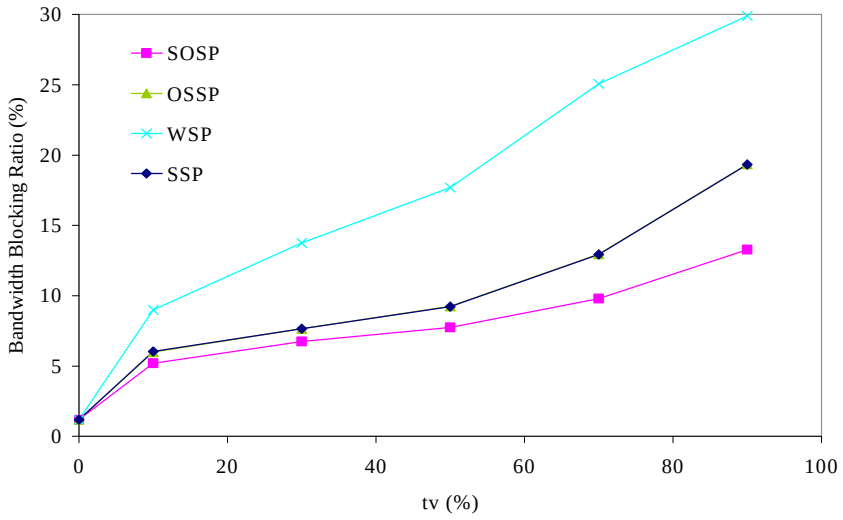


Figure 12. Bandwidth Blocking Ratio for the Threshold triggering policy

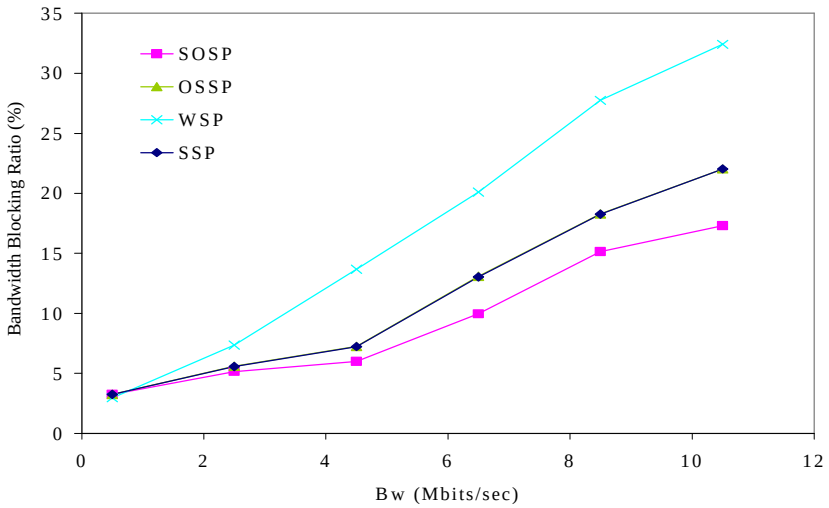


Figure 13. Bandwidth Blocking Ratio for the Exponential class triggering policy

All the results are obtained with a 95% confidence interval after repeating the experiment 10 times, with each simulation lasting 259 seconds. Figure 12 and Figure 13 show the bandwidth-blocking ratio for the *Threshold* and the *Exponential class* triggering policies respectively. The x-axis represents the threshold value *tv* and the base class size *Bw* for both triggering policies respectively. Remind that these values are used by the triggering policies to trigger the update messages.

The algorithms derived from the *BBR* mechanism (*OSSP* and *SOSP*) perform better than *WSP*. In addition, while *OSSP* yields similar results to *SSP*, *SOSP* is substantially better than the *SSP* performance. Specifically, for the *SOSP* algorithm

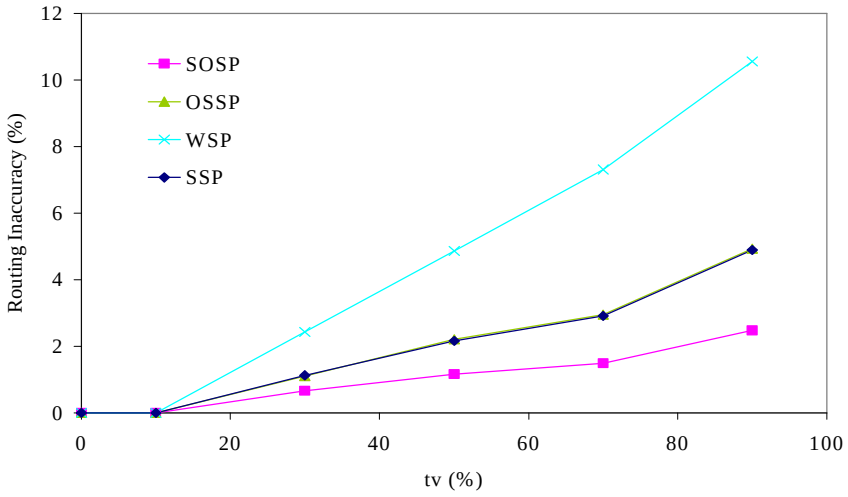


Figure 14. Routing Inaccuracy for the Threshold triggering policy

the Threshold value can be increased 10% while keeping the same bandwidth blocking ratio as *SSP*.

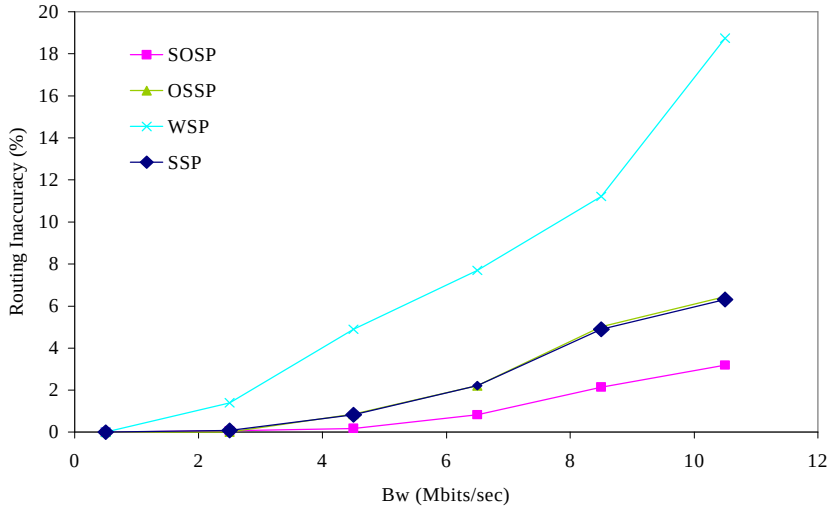


Figure 15. Routing Inaccuracy for the Exponential class triggering policy

Figure 14 and Figure 15 represent the routing inaccuracy for both triggering policies Threshold based and Exponential class based respectively. The *SOSP* algorithm presents the best behaviour, computing a lower number of incorrect routes.

The *OSSP* algorithm behaves better than the *WSP* algorithm although does not substantially improve the *SSP* behaviour.

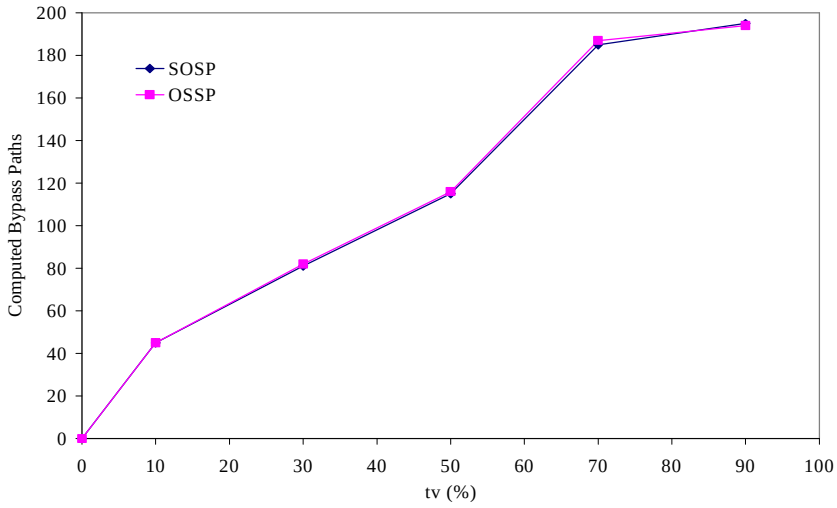


Figure 16. Computed *bypass-paths* for the Threshold triggering policy

The cost of the *BBR* mechanism for both triggering policies in terms of the number of computed *bypass-paths* is shown in Figure 16 and Figure 17. Remind that in order to reduce the computational cost in these simulations, *n_bp* = 3.

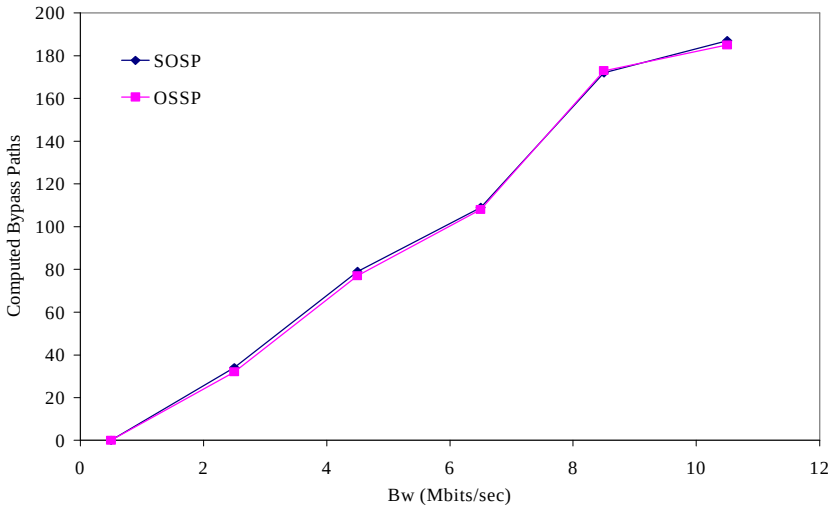


Figure 17. Computed *bypass-paths* for the Exponential class triggering policy

Both figures show that the cost is similar for both algorithms derived from the *BBR* mechanism. It reinforces the conclusion that *SOSP* behaves better than the *OSSP* algorithm. *SSP* and *WSP* do not incur the cost depicted in these figures. Note, however, that this cost is low given the benefits provided by the *BBR* mechanism.

In summary, as a numeric example we take a *tv* value of 70% and analyse the results provided by the *BBR* mechanism and those provided by the *SSP* algorithm, showing that the bandwidth blocking ratio presented by *SOSP* (9.7%) is substantially lower than that provided by *SSP* (12.9 %). Regarding routing inaccuracy, *SOSP* (1.49%) yields a lower number of incorrectly selected paths compared to *SSP* (2.91%). In both cases *OSSP* presents results similar to *SSP*, and *WSP* presents the worst behaviour. This is due to the fact that *WSP* does not consider routing inaccuracy when selecting a path. Finally, for *tv* = 70% the number of *bypass-paths* computed by the *BBR* mechanism during the simulation is close to 180 for both the *SOSP* and the *OSSP* algorithms. That means an *LSP* computation overhead of about 9%, but not in signalling, as explained earlier.

Chapter 7

Applying the BBR Mechanism under Bandwidth Constraints

In the previous Chapter the behaviour of the *BBR* was analysed and its benefits were also shown by simulation. Now, in this Chapter an enhancement of the *BBR* mechanism [66] to optimise bandwidth allocation is proposed. In fact, the algorithms inferred from the *BBR* mechanism presented so far do not explicitly include the available bandwidth in the path selection process; instead, they only consider a range of bandwidth values (those included in the *OSL* definition). The *BBR* enhancement presented here is based on balancing the path length and the residual bandwidth. *SOSP* being the best performing *BBR* algorithm, a simpler initial approach for selecting routes in accordance with certain bandwidth constraints is based on including the residual bandwidth in the *SOSP* algorithm. In this way *SOSP* is only modified when the final selection includes more than one path. In this case, the route is not randomly selected but the widest is chosen. We call this algorithm *Widest-Shortest-Obstruct-Sensitive-Path* (*WSOSP*).

WSOSP is just an initial approach, where the number of hops has more weight than bandwidth capacity in the route selection process. Therefore, in order to balance the path selection process, avoiding those paths that are both widest but too long and shortest but too narrow, a new algorithm is suggested. We call this algorithm *Balanced-Obstruct-Sensitive-Path* (*BOSP*). The *BOSP* algorithm is based on extending the shortest path algorithm with the number of *OSL*s, but unlike previous algorithms based on the *BBR* mechanism already described in the last Chapter, a new parameter is added to each feasible route between source and destination node pair. This parameter, F_p, represents the relation between the maximum residual bandwidth and the number of hops along a path $p \in P$:

$$F_p = n\left[\max\left(\frac{1}{b_r^i}\right)\right], \ldots\ldots\ldots i=1..n \qquad (6)$$

where n is the number of hops and b^i_r is the available residual bandwidth on link i in the path p.

In this way, by using F_p as the cost of each link, network load and network occupancy are balanced in the path selection process.

BALANCED OBSTRUCT SENSITIVE PATH ALGORITHM (BOSP)

Input: The input graph $G(N_r,L_r,B_r)$. The *LSP* request is between a source-destination pair *(s,d)* and the bandwidth requirement is b_{req}.

Output: An optimized and balanced route from *s* to *d* with enough *bypass-paths* to bypass the routing inaccuracy effects in the *obstruct-sensitive links*.

Algorithm:

1. Mark those links that are defined as *OSL* according to Rule 1
2. Compute the weight of a link *l* as
   $w(l)= 1 \Leftrightarrow l \in L^{os}, \qquad w(l)= 0 \Leftrightarrow l \notin L^{os}$
3. Apply Dijkstra's algorithm to select the path that minimizes the number of *OSLs* by using *w(l)* as the cost of each link
4. If more than one exists compute the cost F_p of each path
   $$F_p = n\left[\max\left(\frac{1}{b_r^i}\right)\right] \qquad i=1..n$$
5. Select the path that minimizes F_p
6. Determine the edge nodes pair *(i,e)* of the *OSLs* existent in the selected path
7. Compute the *bypass-paths* for each element *(i,e)* according to Rule 2
8. Decide which *bypass-paths* must be used in accordance with real available resources in the path setup time
9. Route the traffic from *s* to *d* along the setup path

Figure 18. *BOSP*: The enhanced *BBR* mechanism

In Figure 18 the *BOSP* algorithm is briefly described. Again, if more than one possible *bypass-path* exists, the route that minimizes the number of *OSL*s is chosen. The complexity of *BOSP* can be represented in the same manner used to represent the complexity of the initial algorithms inferred from the *BBR* mechanism, *SOSP* and *OSSP*.

As a summary, Figure 19 includes a flowchart depicting all the algorithms inferred from the *BBR* mechanism, i.e., the *SOSP*, *OSSP*, *WSOSP* and *BOSP* algorithms.

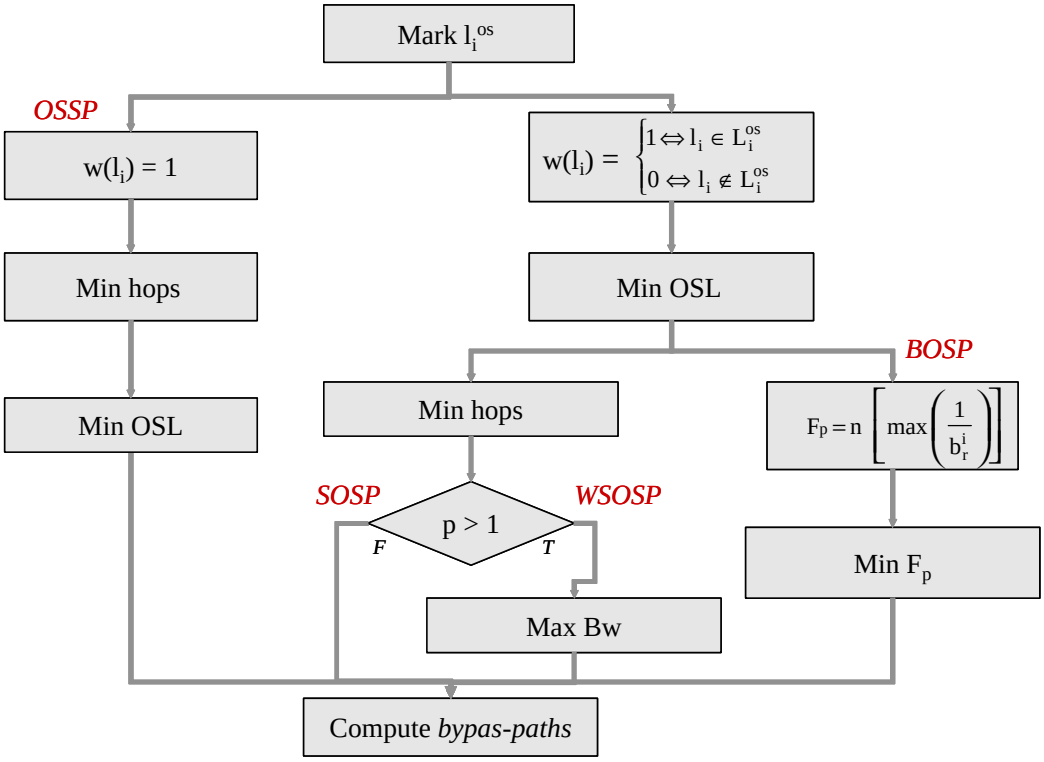


Figure 19. Routing algorithms inferred from the *BBR* mechanism

7.1 Example Illustrating the BOSP Behaviour

The topology shown in Figure 20 is used to test the performance of *BBR*. This test supposes an incoming *LSP* request demanding b_{req} of 4 units of bandwidth between LSR0-LSR7. Moreover, the triggering policy used is the Exponential class based policy, with $f = 2$ and $Bw = 1$ (as used in [49]).

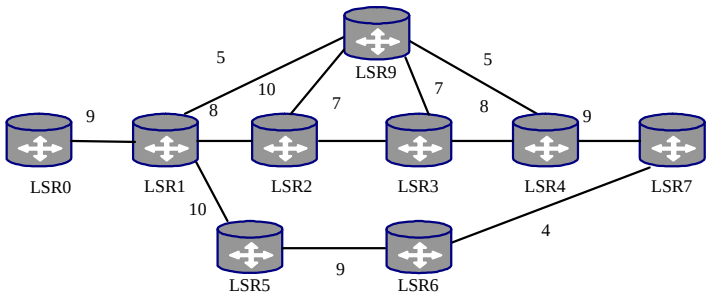


Figure 20. Network topology used to illustrate the *BOSP* algorithm

Table 4 shows all the different routes between *LSR0* and *LSR7*. *ID* is defined as a path identifier; *H* represents the number of hops along the path; b_r^{min} is the minimum residual bandwidth along the path; *N_OSL* is the number of *Obstruct-Sensitive Links*, F_p is the cost, and *Algorithm* represents the algorithms that select each path. The two new routing algorithms based on the *BBR* mechanism, *WSOSP* and *BOSP*, select the paths identified by *b* and *a* respectively.

Table 4. Link QoS attributes

ID	*PATH (LSR)*	*H*	b_r^{min}	*N_OSL*	F_p	*Algorithm*
a	0-1-2-3-4-7	5	7	1	0.71	BOSP
b	0-1-5-6-7	4	4	1	1	SOSP,WSOSP
c	0-1-2-9-3-4-7	6	7	1	0.85	
d	0-1-9-3-4-7	5	4	2	1.25	
e	0-1-9-2-3-4-7	6	4	2	1.5	
f	0-1-2-9-4-7	5	4	2	1.25	
g	0-1-9-4-7	4	5	2	0.8	WSP

Once the path has been selected, the *BBR* mechanism computes a *bypass* path for each *OSL*. When the *BOSP* routing algorithm is used, *a* is the route selected and one *OSL* exists, as shown in Table 4. Then, the *BBR* mechanism computes the edge nodes of said *OSL* that is {LSR2-LSR3}. In order to compute the *bypass-path*, it is possible to use any of the parameters shown in Table 5 (*H*, b_r^{min}, F and *N_OSL*). In this case, *N_OSL* is the parameter used to compute the *bypass-path*s. Therefore, {LSR2-LSR9-LSR3} is the *bypass-path* selected to bypass the *OSL*. When *WSOSP* is implemented, the edge nodes to bypass are {LSR6-LSR7}. However, in this case it is not possible to find a path that bypasses this link in the network topology. Hence, there are again some cases in which the dynamic bypass concept cannot be applied.

Table 5. Possible *Bypass-paths*

ID	L^{os}	*Edge LSRs*	*Bypass_path (LSR)*	*H*	b_r^{min}	F_p	*N_OSL*
a	2-3	2-3	2-9-3	2	7	0.28	1
b	6-7	6-7	--	-	--	--	--

7.2 Performance Evaluation

The ns/2 simulator used earlier, extended to implement the new algorithms, has been used to evaluate the performance of the proposed *BBR* enhancement. A set of simulations have been performed on different scenarios to evaluate the suggested *WSOSP* and *BOSP* algorithms in comparison with the already existing *WSP*, *SSP*, *SOSP* and *OSSP* algorithms. All the results have been obtained with a 95% confidence interval after repeating the experiment 10 times. The *BBR* mechanism has been evaluated in three different scenarios.

1) Scenario 1:

The simulations were carried out over the network topology shown in Figure 11. Every simulation requests 2000 *LSP*s, which arrive following a Poisson distribution where the requested bandwidth is uniformly distributed between 1 Mb and 5 Mb and the holding time is randomly distributed with a mean of 120 seconds. The triggering

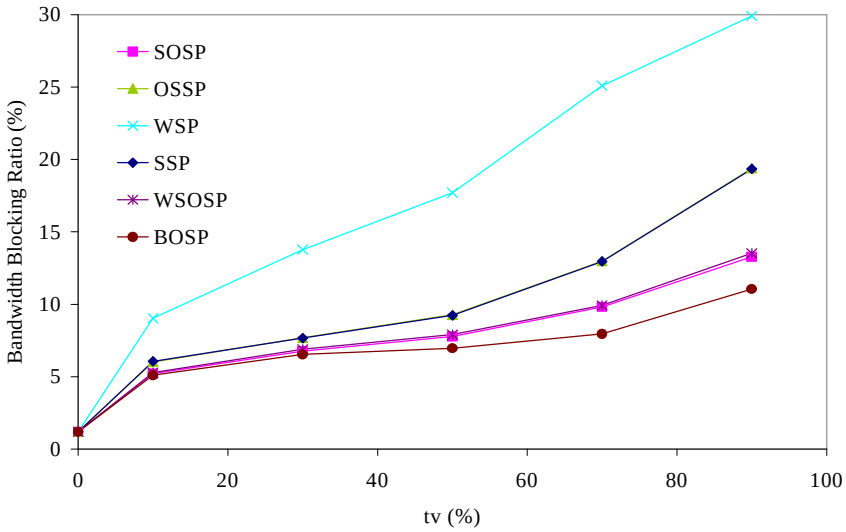


Figure 21. Bandwidth Blocking Ratio for Scenario 1(Threshold triggering policy)

policies used in these simulations are Threshold and the Exponential class (with f = 2). The results presented have been obtained after repeating 300 seconds of simulation 10 times. The parameters used to measure the algorithms' behaviour are the routing inaccuracy and the bandwidth blocking ratio, that is, the same used to evaluate the basic *BBR* mechanism.

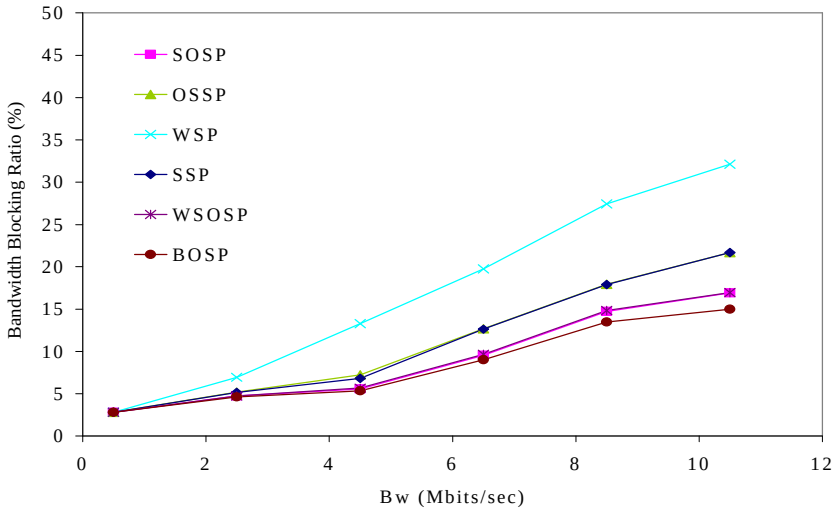


Figure 22. Bandwidth Blocking Ratio for Scenario 1 (Exponential class triggering policy)

Figure 21 and Figure 22 show the bandwidth blocking ratio tested for both the Threshold and Exponential class triggering policies respectively. Note that the best performance is obtained using *BOSP*, whereas *WSOSP* exhibits similar results to *SOSP*, and in both cases, *BOSP* and *WSOSP* present better results than *WSP*. In the worst conditions (the threshold *tv* of the triggering policy is 90%), the bandwidth blocking ratio obtained by *BOSP* (11%) substantially improves those obtained by *WSOSP* (13.5%), *SOSP* (13.3%) and *SSP* (19.3%). Recall that by increasing the threshold value the number of update messages flooded throughout the network is reduced.

Regarding routing inaccuracy behaviour, Figure 23 and Figure 24 show again that *BOSP* exhibits better results for both triggering policies. The number of paths incorrectly selected is extremely low even for large values of the threshold and the base class size.

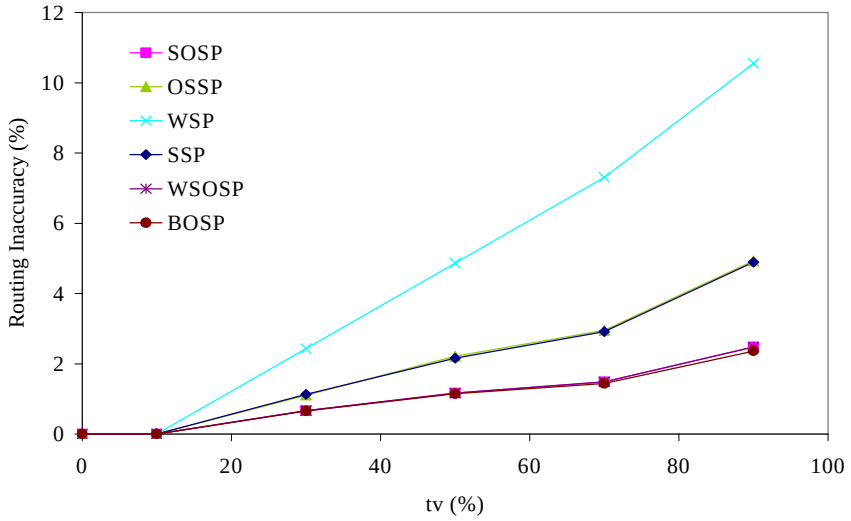


Figure 23. Routing Inaccuracy for Scenario 1 (Threshold triggering policy)

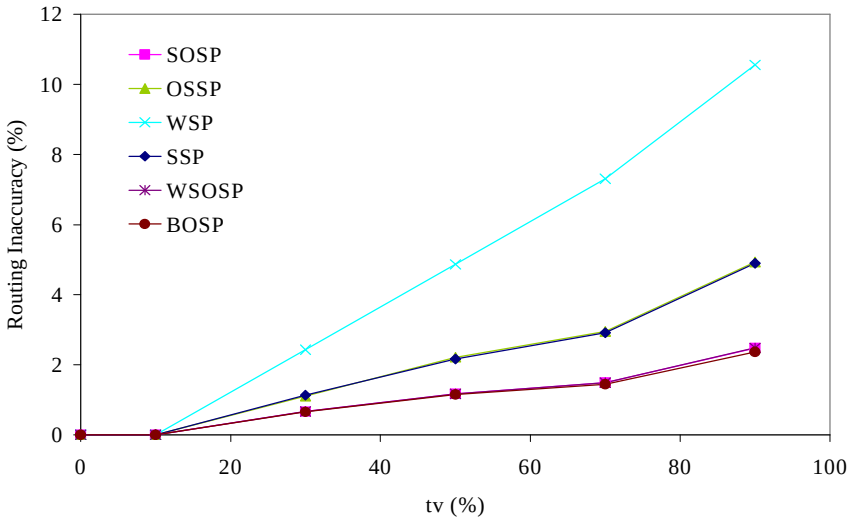


Figure 24. Routing Inaccuracy for Scenario 1 (Exponential class triggering policy)

Finally, Figure 25 and Figure 26 show the cost of using the *BBR* mechanism. As in previous Chapter the number of *bypass-paths* computed per route, *n_bp* is limited to 3. The number of computed *bypass-paths* grows when either the threshold or the base class size value increases. This is logical, since the number of *OSLs* grows when the amount of flooding messages decreases. It is important to observe that when *BOSP* is applied, not only do the bandwidth blocking ratio and the routing

inaccuracy decrease, but the cost decreases as well. This is due to the optimisation achieved in the path selection process.

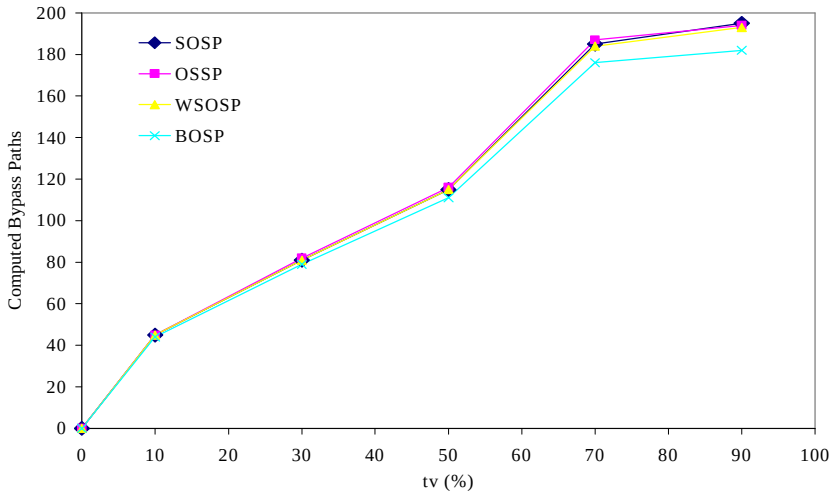


Figure 25. Computed *bypass-paths* for Scenario 1 (Threshold triggering policy)

Hence, the *BOSP* is the *BBR* algorithm which presents the best performance in terms of blocking probability, number of routes incorrectly selected and the cost because of computing *bypass-paths*.

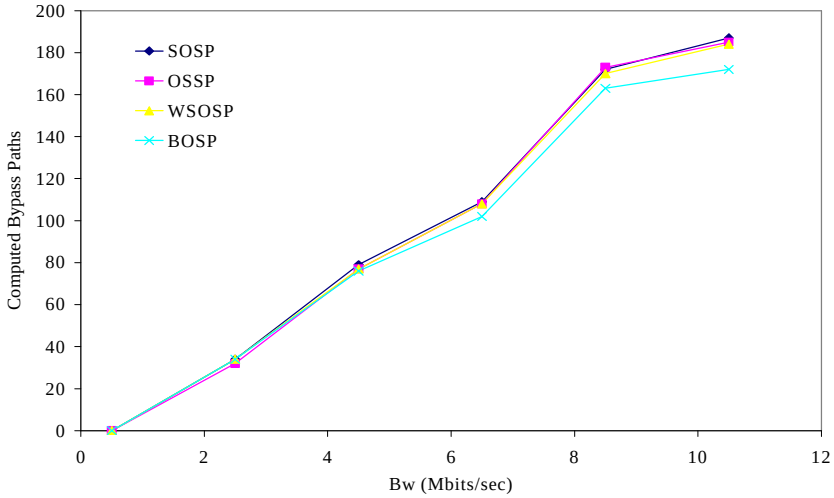


Figure 26. Computed *bypass-paths* for Scenario 1 (Exponential class triggering policy)

It is worth to notice that all simulations performed in Chapter 6 and Chapter 7 limit the number of computed *bypass-paths* per route to 3. As already explained, this is a restriction imposed to reduce the cost. It might be quite interesting to find out

which is the restriction imposed in the reduction obtained in the bandwidth blocking ratio because of limiting the number of *bypass-paths* computed per route. This evaluation is performed in two parts. On the one hand, the impact of limiting computed *bypass-paths* to a fixed value (1,2 and 3) on both bandwidth blocking probability and cost is analyzed by simulation in Chapter 8.

On the other hand, unlike the solution applied so far to reduce the cost, a different solution to reduce the cost of the *BBR* mechanism is to make the *n_bp* value dependent on the network load. When network is not highly loaded, a low number of links defined as *OSL* is expected, therefore a low number of *bypass-paths* per route might be computed. However, when network is heavily loaded a high number of *bypass-paths* per route are needed to cover *OSL* definition. Assuming that the larger the network load, i.e., the larger the number of *bypass-paths* computed, the lower the probability that *bypass-paths* might be really used, since they are also unavailable, the probability that a *bypass-path* may be used decreases as network load increases. Therefore another solution to reduce the cost of the *BBR* mechanism is to make the number of computed *bypass-paths* per route dependent on the network load. This approach is evaluated in the Scenario 2.

2) Scenario 2:

In this case, the *BBR* mechanism is tested over a realistic network topology to verify its influence on global network performance. The *ISP* topology shown in Figure 27 is a popular topology used in many old and recent QoS routing studies and is typical of the nationwide network of a US based ISP. There are three source nodes computing routes for any other destination node (except themselves).

Assuming the *SOSP* and the *BOSP* be those routing algorithms inferred from the *BBR* mechanism presenting lower blocking probability, the algorithms evaluated are the *WSP*, *SSP*, *SOSP* and *BOSP*. Unlike previous simulations where the number of computed *bypass-paths* per route is limited by fixing the *n_bp* value, in these simulations the cost is reduced by making the *n_bp* value dependent on the network load in the range from 0 *bypass-paths* per route (network highly loaded) to 5 *bypass-paths* per route (network not loaded). Once more, the bandwidth blocking ratio and the routing inaccuracy are the parameters evaluated. In previous simulations, both

parameters present similar behaviour for both the Threshold and the Exponential class triggering policies. Assuming this as a constant behaviour, in the following simulations only the Threshold triggering policy is analyzed.

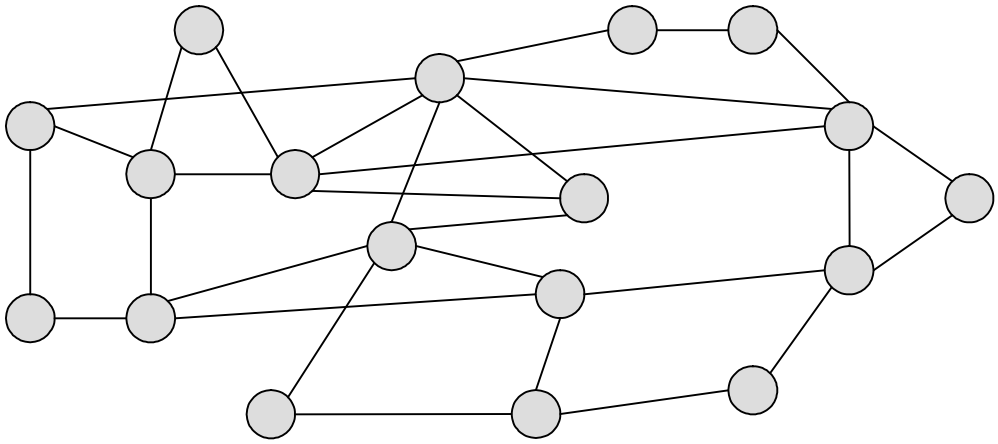


Figure 27. The *ISP* topology used in simulations

All links are assumed to be bi-directional and of the same capacity of 622 Mb/s. Every simulation requests 2500 *LSP*s which arrive following a Poisson distribution, where the requested bandwidth is uniformly distributed between 1Mb/s and 5Mb/s. The holding time is randomly distributed with a mean of 120 sec. Simulations run during 250 sec.

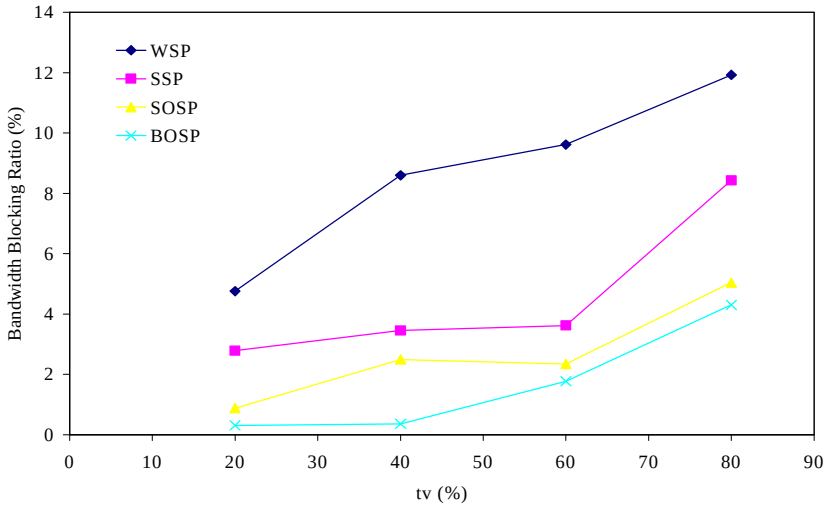


Figure 28. Bandwidth Blocking Ratio for Scenario 2 (Threshold triggering policy)

Figure 28 exhibits the bandwidth blocking ratio obtained by simulation. Again, while the *SOSP* behaves better than the *SSP*, the *BOSP* provides the lower blocking probability. It is worth to notice that the larger difference in the obtained blocking reduction when using the *BOSP* algorithm compared to the *WSP* and *SSP* algorithms is obtained for *tv* = 60% instead of *tv* = 80%.

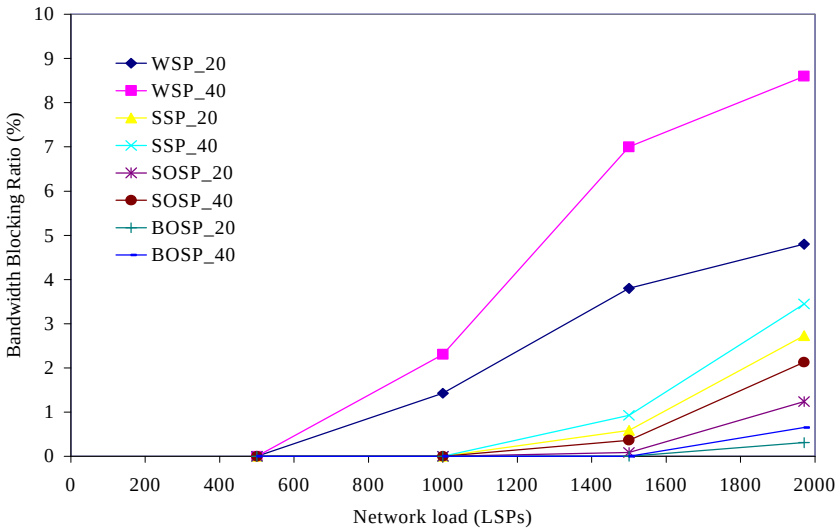


Figure 29. Bandwidth Blocking Ratio and network load for Scenario 2 (*tv* = 20%, *tv* = 40%)

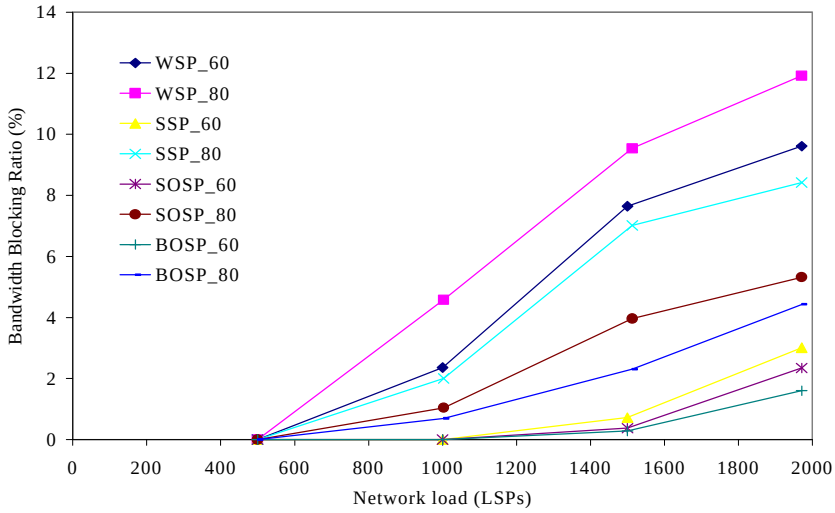


Figure 30. Bandwidth Blocking Ratio and network load for Scenario 2 (*tv* = 60%, *tv* = 80%)

The evolution of the Bandwidth blocking as a function of network load in terms of established *LSP* connections is analysed in next Figures. Meanwhile Figure 29

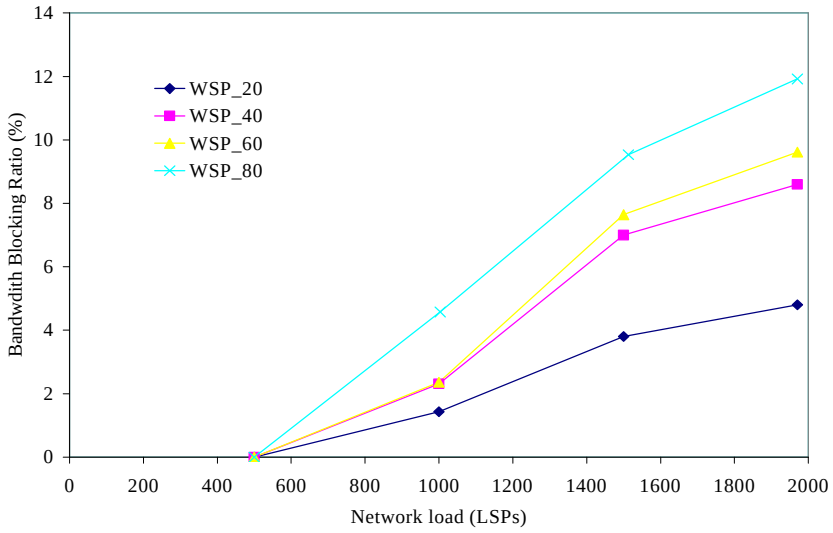


Figure 31. *WSP* algorithm behaviour as a function of the network load (Scenario 2)

exhibits the blocking behaviour for *tv* = 20% and *tv* = 40%, Figure 30 represents the blocking behaviour for *tv* = 60% and *tv* = 80%. Both Figures show the benefits in terms of bandwidth blocking reduction obtained when applying algorithms inferred from the *BBR* mechanism for any *tv* value. It is worth to notice that for low network load and medium values of *tv* the *SOSP* performs almost better than the *BOSP* algorithm. Therefore, the *BOSP* algorithm gives larger benefits on bandwidth blocking reduction for high network loads and medium values of *tv*.

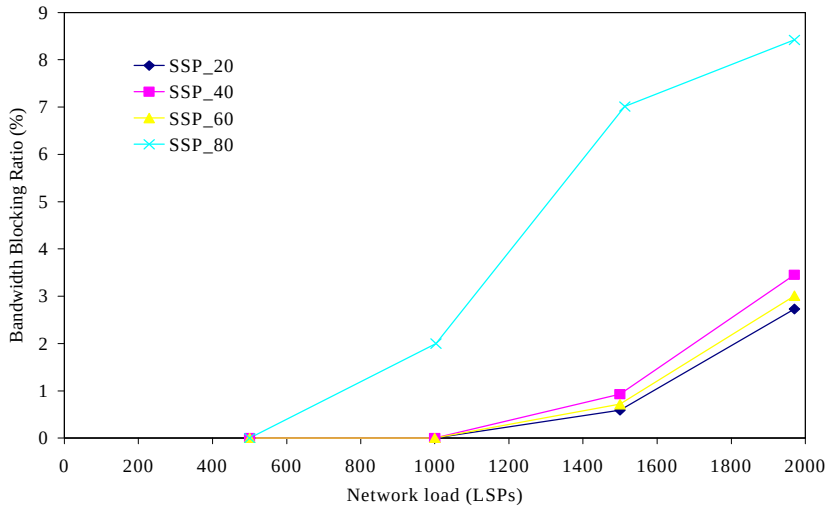


Figure 32. *SSP* algorithm behaviour as a function of the network load (Scenario 2)

Next figures draw the behaviour of each algorithm as a function of the network load in terms of number of established *LSP*s for each threshold value. In this way it is possible to independently observe the evolution in the bandwidth blocking depending on the *tv* value on each algorithm. The *WSP*, *SSP*, *SOSP* and *BOSP* are shown in Figure 31, Figure 32, Figure 33 and Figure 34 respectively.

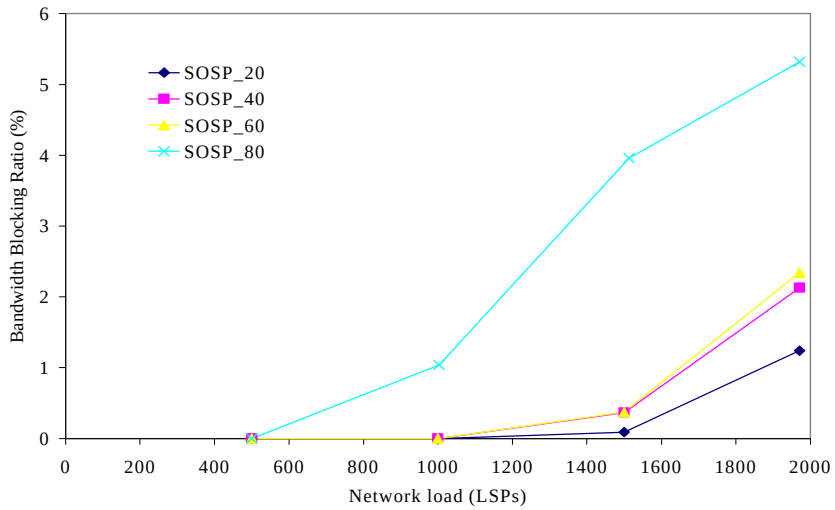


Figure 33. *SOSP* algorithm behaviour as a function of the network load (Scenario 2)

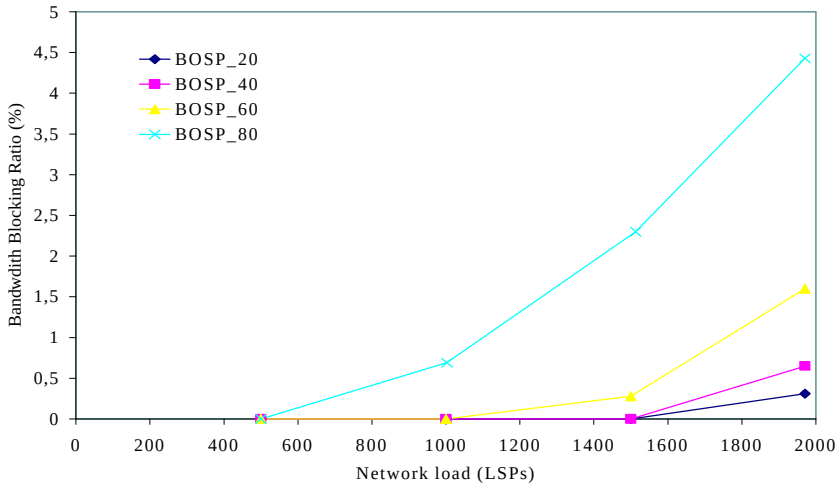


Figure 34. *BOSP* algorithm behaviour as a function of the network load (Scenario 2)

The number of routes incorrectly selected is shown in Figure 35. Again, the *BOSP* is the algorithm that computes the lower number of incorrect routes. As previous

simulations, *BOSP* and *SOSP* algorithms present better behaviour compared to the *SSP* and the *WSP*.

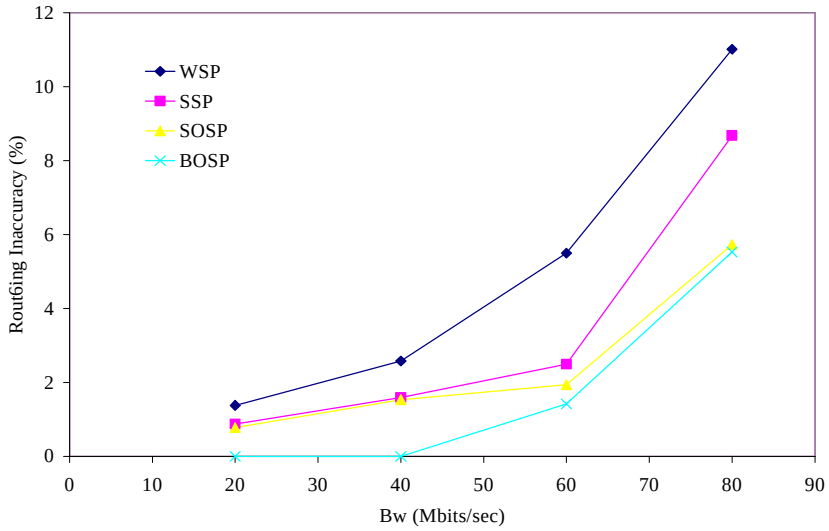


Figure 35. Routing Inaccuracy for Scenario 2 (Threshold triggering policy)

Figure 36 draws the cost of using the *BBR* mechanism. Remind that in these simulations *n_bp* is not a constant value instead it is network load dependent. The total number of computed *bypass-paths* is substantially reduced in comparison with values obtained in previous simulations. However, someone could argue that the reduction obtained in the cost might negatively reduce the *BBR* introduced benefits.

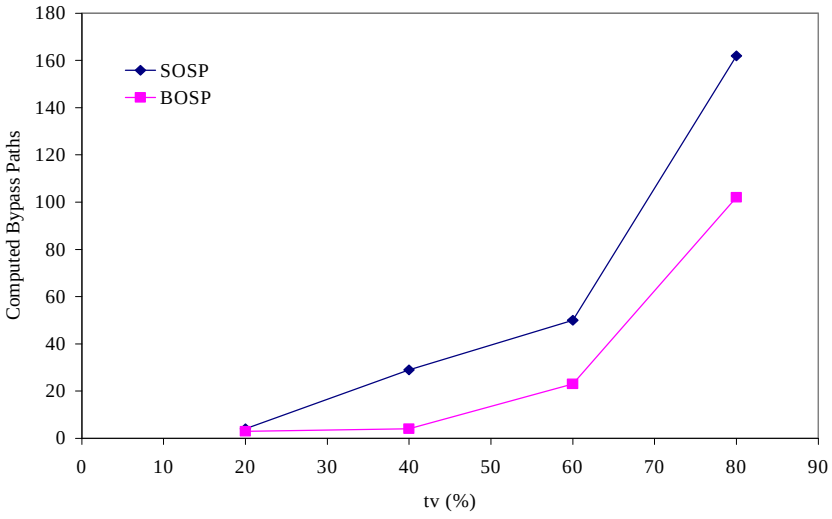


Figure 36. Computed *bypass-paths* for Scenario 2 (Threshold triggering policy)

Table 6 compares the reduction in the bandwidth blocking obtained by the *BOSP* under three different situations: when *n_bp* = 3, when *n_bp* is not limited (a ∞ number of *bypass-paths* might be computed per route) and when *n_bp* is network load dependent.

Table 6. Cost analysis

n_bp	BW_{WSP}	BW_{SSP}	*Cost (bypass-paths)*	*Cost (time)*
3	8.24	4.62	282	14.30%
Not limited	15.11	7.99	3178	161.23%
Network load dependent	7.62	4.13	102	5.71%

The values BW_{WSP} and BW_{SSP} stand for the difference in the bandwidth blocking obtained when comparing the *BOSP* to the *WSP* and the *SSP* respectively. The cost is represented in terms of both the total number of computed *bypass-paths* and the impact of computing these *bypass-paths* on the computational time. The second situation, when *n_bp* is not limited, is really not affordable because of the extreme cost. Analysing the first and the last situation, while similar results in bandwidth blocking are obtained the cost is significantly reduced when distributing *n_bp* proportionally to the network load.

3) Scenario 3:

This simulation is performed to analyse the impact on the *BBR* performance depending on the link capacity. Simulations are performed over the network topology shown in Figure 27 although some variations are considered. In this case all links are assumed to be bi-directional with a capacity of 2.5 Gb/s. Every simulation requests 10500 *LSP*s which arrive following a Poisson distribution, where the requested bandwidth is uniformly distributed between 1Mb/s and 5Mb/s. The holding time is randomly distributed with a mean of 120 sec. 10 simulations have been carried out each one lasting 295 sec. Again, the same parameters are evaluated and the same algorithms are implemented, *WSP*, *SSP*, *SOSP* and *BOSP*.

There are two conclusions that can be extracted after analysing the bandwidth blocking shown in Figure 37. First, while all the algorithms suffer from an increment in the bandwidth blocking, the *BOSP* algorithm is still maintaining a quasi linear behaviour. Second, there is a large difference in the bandwidth blocking obtained by

the *BOSP* algorithm in comparison with the other algorithms. In fact, while in previous simulation the bandwidth blocking difference obtained by the *BOSP* compared to the *SSP* is 3.99 %, in this simulation this difference is 5.7%.

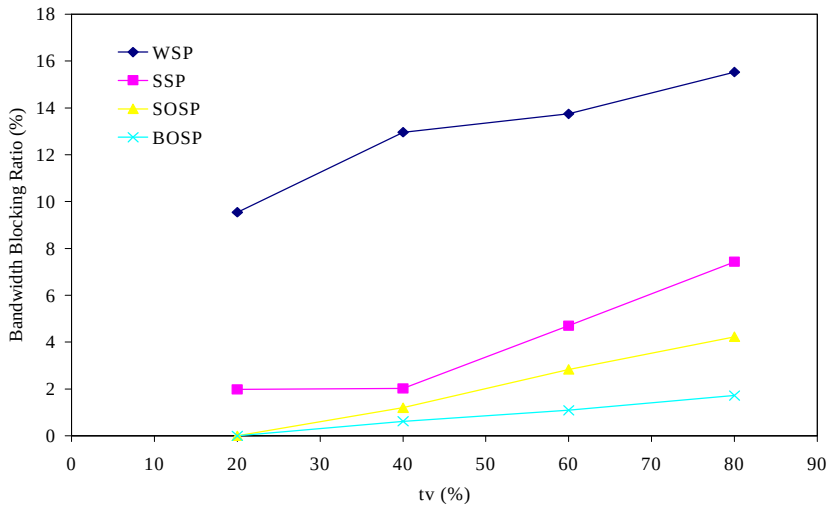


Figure 37. Bandwidth Blocking Ratio for Scenario 3 (Threshold triggering policy)

Figure 38 shows the impact on the blocking as a function of the network load in terms of number of established *LSP*s for values of *tv*= 20% and 40%. In Figure 39 *tv* values are 60 % and 80%.

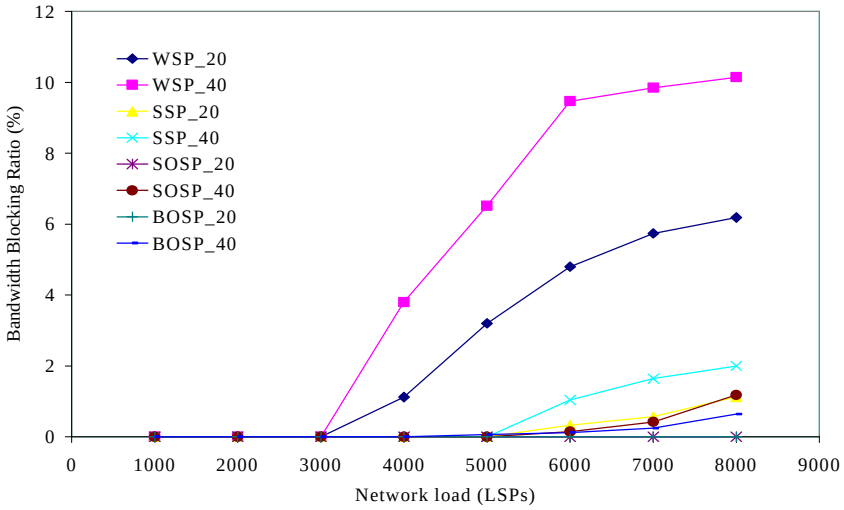


Figure 38. Bandwidth Blocking Ratio and network load for Scenario 3 (*tv* = 20 %, *tv* = 40%)

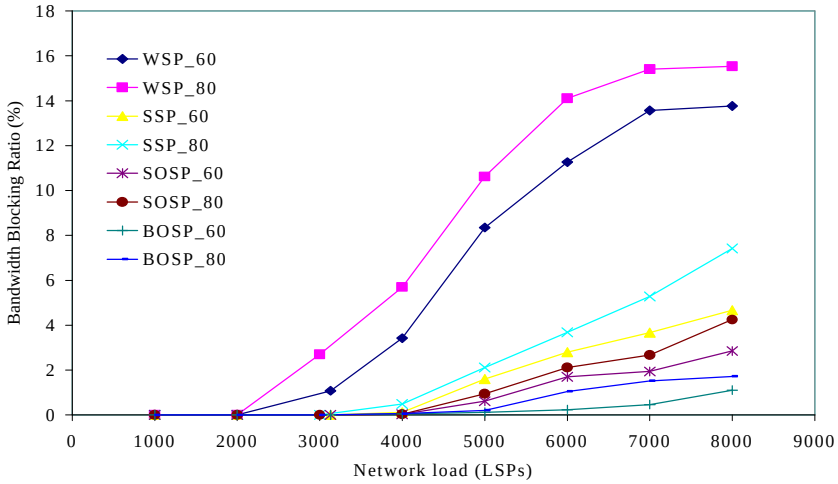


Figure 39. Bandwidth Blocking Ratio and network load for Scenario 3 (*tv* = 60%, *tv* = 80%)

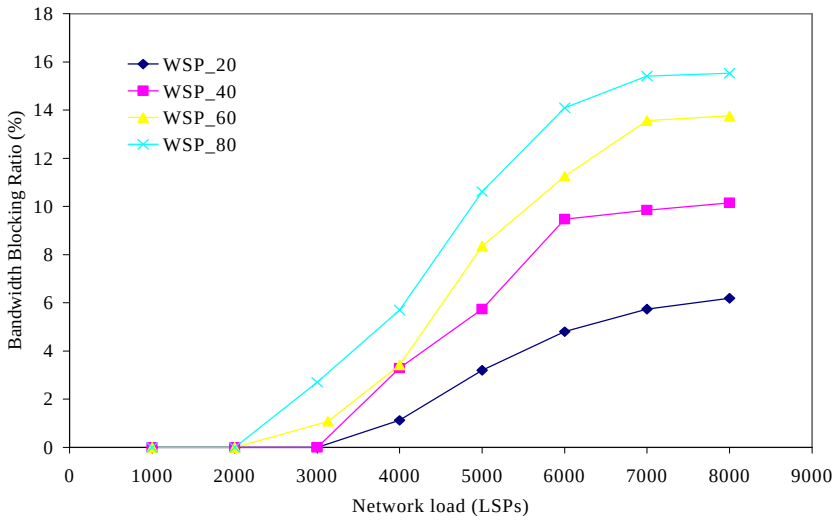


Figure 40. *WSP* algorithm behaviour as a function of the network load (Scenario 3)

Again, next figures draw the behaviour of each algorithm as a function of the network load in terms of number of established *LSP*s for each threshold value. Figure 40, Figure 41, Figure 42 and Figure 43 show the bandwidth blocking evolution for the *WSP*, *SSP*, *SOSP* and *BOSP* algorithms respectively.

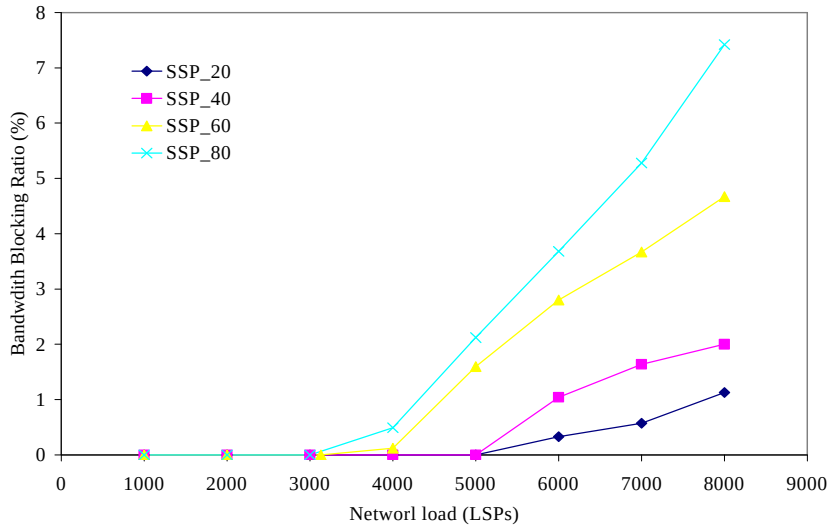


Figure 41. *SSP* algorithm behaviour as a function of the network load (Scenario 3)

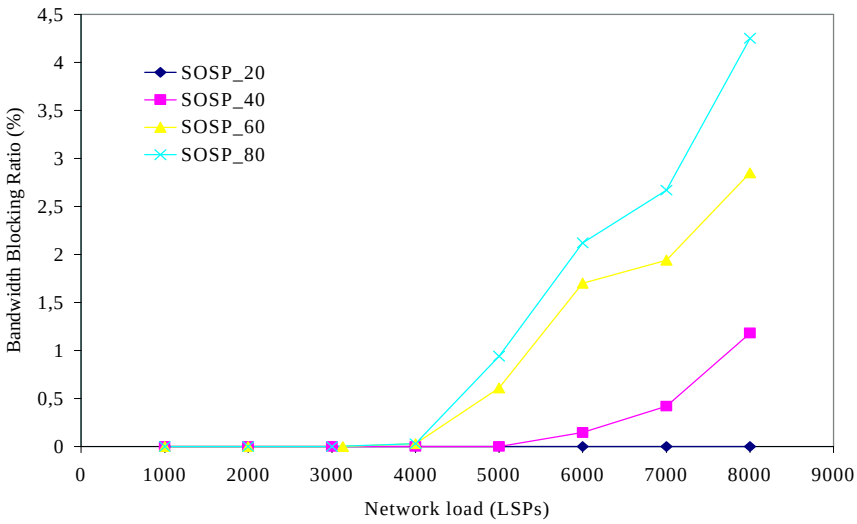


Figure 42. *SOSP* algorithm behaviour as a function of the network load (Scenario 3)

Figure 44 shows the routing inaccuracy for all the evaluated algorithms when applying the Threshold triggering policy. The *SOSP* and the *BOSP* algorithms exhibit a very similar behaviour.

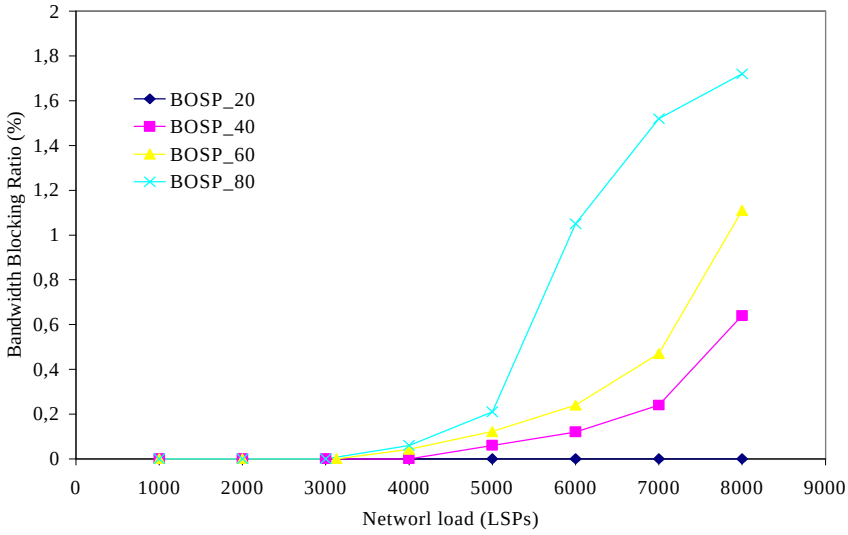


Figure 43. *BOSP* algorithm behaviour as a function of the network load (Scenario 3)

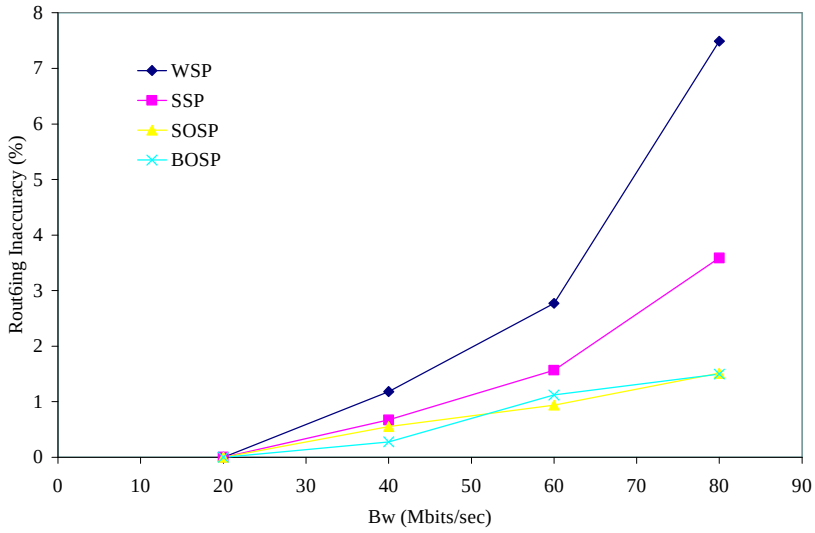


Figure 44. Routing Inaccuracy for Scenario 3 (Threshold triggering policy)

Finally, Figure 45 plots the cost in terms of computed *bypass-paths*. Only a negligible increment in the number of computed *bypass-paths* is produced. Therefore, while *BBR* mechanism is regardless of links size in terms of the number of routes incorrectly selected, the obtained bandwidth blocking ratio suffers from lower reduction.

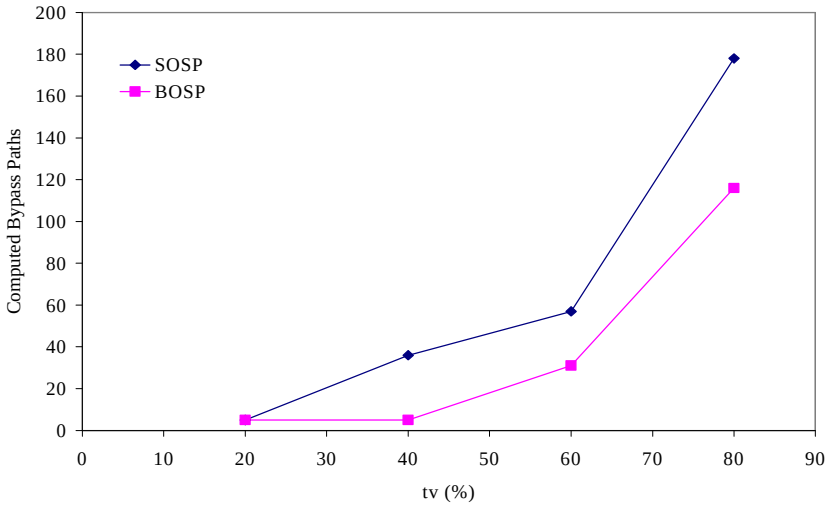


Figure 45. Computed *bypass-paths* for Scenario 3 (Threshold triggering policy)

Summarizing, in this Chapter it has been extensively proved that the larger the number of computed *bypass-paths* the lower the blocking probability reduction, but also the larger the cost. It has also been shown that when the *n_bp* value is dependent on the network load the cost of applying the *BBR* mechanism, specifically the *BOSP* algorithm, is substantially reduced while variation on the obtained bandwidth blocking is hardly significant.

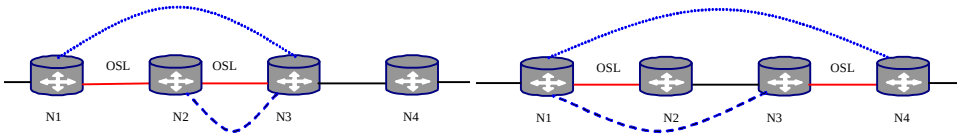
The next Chapter presents the *BYPASS Discovery Process*. Moreover, it is also analyzed the evolution in the blocking ratio reduction as a function of the number of computed *bypass-paths* per route.

Chapter 8

BYPASS Discovery Process

The *BYPASS Discovery Process (BDP)* appears as a solution to extend the *BBR* applicability. In fact, as already stated in previous Chapters, there are some network scenarios where the *BBR* mechanism cannot be applied since a *bypass-path* cannot be found. This occurs because the *bypass-path* definition requires that the edge nodes of the *bypass-path* to be computed must perfectly match the edge nodes of the *OSL* that bypasses. Therefore, whenever an alternative and disjoint route between the edges nodes of that link defined as an *OSL* cannot be found, the *BBR* mechanism cannot be applied. In this case, instead of rerouting the set-up message along the *bypass-path* that should have been computed the set-up message will be blocked and dropped when the selected route lacks enough bandwidth.

The *BDP* addresses this problem adding a modification in the *BBR* mechanism which extends the mechanism used to select *bypass-paths* guaranteeing a higher number of computed *bypass-paths*. Let i_j and e_j be the edge nodes of a link $l^{os}_j \hat{I} L^{os}$.



(a) (b)

Figure 46. *BDP* process

Let e_{j+k} be the *k* node adjacent to e_j downstream along the working path. Then, *BDP* computes *bypass-paths* in accordance with the following rules:

- Look for an alternative and disjoint route between the (i_j,e_j) pair (as done in the normal *BBR* mechanism).
- If there is not a feasible disjoint route between the (i_j,e_j) pair, then look for a route between the (i_j,e_{j+k}) pair, for $1 \leq k \leq d$, being e_{j+d} the destination node.

Three main aspects must be analyzed when applying the *BDP*. Firstly, feasible bypass routes cannot include any node belonging to the working path but the egress (destination) node. Secondly, Rule 2 in the *BBR* definition is still meaningful. Therefore, when two adjacent nodes are defined as *OSL*s the *BDP* computes a *bypass-path* to bypass both links and then another *bypass-path* to bypass only the second link, as shown in Figure 46 (a). Finally, consider the scenario shown in Figure 46 (b) where links N1-N2 and N3-N4 are defined as an *OSL* (if link N2-N3 is also defined as an *OSL* this case matches Rule 2). Suppose that the dash line stands for a possible *bypass-path* to bypass link N1-N2. In this situation, to reduce the possible number of *bypass-paths* used, an optimal *bypass-path* would be that drawn by the dot line from N1 to N4. Then, a *bypass-path* must also be computed to bypass link N3-N4 (not drawn in Figure 46 (b)).

8.1 Example to Illustrate the BDP Performance

Figure 47 is used to ease the *BDP* understanding. The performance of the *SOSP* and the *BOSP* (the two *BBR* algorithms better performing) algorithms when both include the *BDP* are analyzed. Suppose that update messages are sent according to the Exponential Class triggering policy with f=2 and Bw=1. Assuming that an

incoming *LSP* request arrives at LSR0 demanding b_{req} of 4 units of bandwidth between LSR0 to LSR4, Table 7 shows different routes from LSR0 to LSR4 including the parameters used by the *SOSP* and the *BOSP* algorithms to select the path.

Table 7. *BBR* Process when including the *BDP*

Id	*Route (LSR)*	*H*	*OSL*	b_r^{min}	F_p
a	0-1-2-3-4	4	1	4	1
b	0-1-5-6-7-4	5	2	7	0.71
c	0-1-5-2-3-4	5	1	6	0.83
d	0-8-9-4	3	3	4	0.75

BBR	*SOSP*	*BOSP*
1th step	Mark OSLs	Mark OSLs
2on step	a,c	a,c
3th step	a	c
4th step	1-2 (1,5,2)	5-2 (5,6,7,4)

It is also shown in detail in Table 7 the four steps realized by the *BBR* mechanism to select paths. Specifically the *SOSP* and the *BOSP* algorithms are depicted. The 4th step, *bypass-path* selection, includes the *BDP* mechanism proposed in this Chapter.

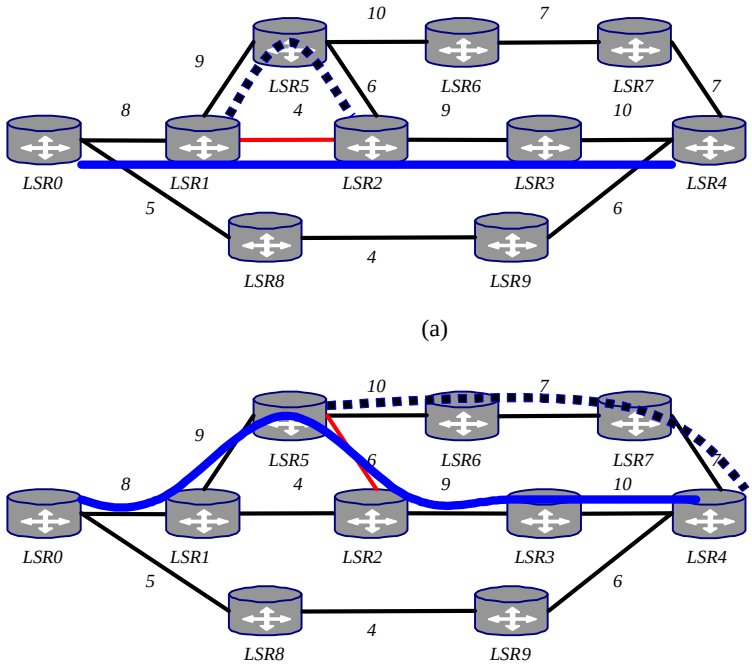


Figure 47. *BDP* performance: illustrative example

According to the *SOSP* behaviour, Figure 47 (a), a *bypass-path* exists (LSR1, LSR5, LSR2) to bypass the edges nodes of the link defined as an *OSL* that is LSR1-LSR2, represented by a dash line. However, when applying the *BOSP* algorithm, Figure 47 (b), there is not a *bypass-path* that directly bypasses the edge nodes of the

link defined as *OSL*, namely LSR5-LSR2. In this case, if the *BDP* mechanism is not implemented, a *bypass-path* might not be computed. *BDP* allows the *BBR* mechanism to compute a *bypass-path* (to bypass the *OSL* between nodes LSR5 and LSR2) from LSR5 to LSR4 (destination node), represented by a dash line. This *bypass-path* will be used if there are insufficient bandwidth in the link defined as an *OSL* when the *Path* message reaches LSR5, hence improving the *BBR* applicability and so reducing the bandwidth blocking ratio.

8.2 Performance Evaluation

In this Section we compare by simulation the benefits introduced when including the *BDP* in the *BBR* mechanism. Being the *BOSP* the best algorithm inferred from the *BBR* mechanism, in terms of blocking ratio and optimal path selection, the algorithms evaluated are the *Shortest-Safest Path* (*SSP*), and the *BOSP*. In order to clearly identify the improvement obtained by the *BBR* mechanism when the *BDP* is also applied, the *BOSP* algorithm is evaluated in both situations, i.e., when it does not include the *BDP* (named *BOSP*) and when includes the *BDP* (named *B/BDP*). Moreover, in order to evaluate the impact on the blocking probability because of the number of *bypass-paths* that can be computed per route, different simulations are carried out as a function of *n_bp*. Remind that *n_bp* has been defined as a bundle showing the maximum number of *bypass-paths* that can be computed on each working path. The notation used in the figures to denote the *n_bp* values is *BOSP(n_bp)* and *B/BDP(n_bp).* The simulations are performed over the network topology shown in Figure 11, using the ns/2 simulator extended with *MPLS, BBR* and *BDP* features. We use two link capacities, 622 Mb/s represented by a light line and 2.5 Gb/s represented by a dark line. Every simulation requests 2500 *LSP*s which arrive following a Poisson distribution where the requested bandwidth is uniformly distributed between 1 Mb/s and 5 Mb/s. The holding time is randomly distributed with a mean of 120 seconds. The Threshold and the Exponential class (with $f = 2$) triggering policies are evaluated. Results have been obtained after repeating 300 seconds of simulation 10 times. As previous simulations, the parameters used to measure the algorithms behaviour are the routing inaccuracy and the bandwidth

blocking ratio. The bandwidth blocking ratio for the Threshold and Exponential class triggering policies are depicted in Figure 48 and Figure 49 respectively.

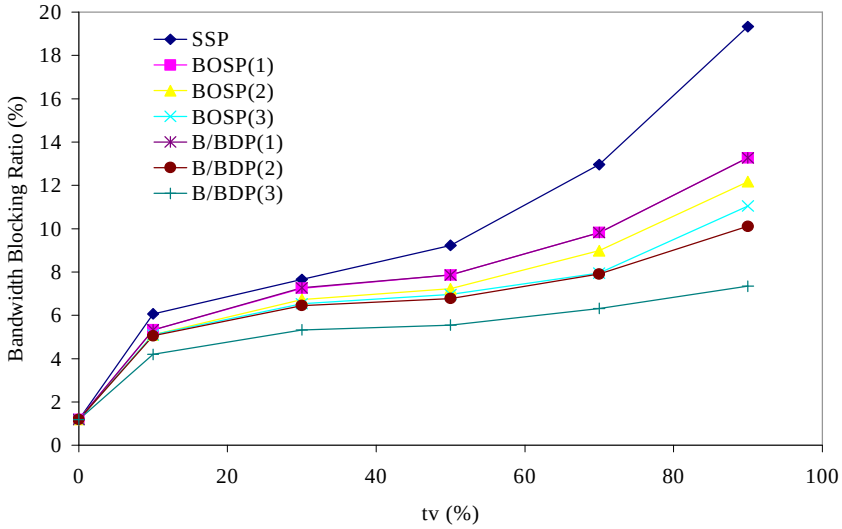


Figure 48. Bandwidth Blocking Ratio for the Threshold triggering policy

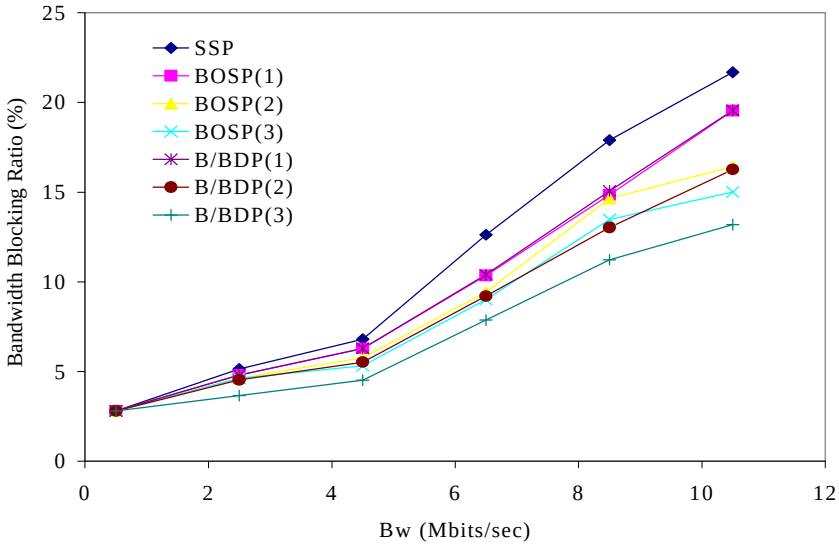


Figure 49. Bandwidth Blocking Ratio for the Exponential class triggering policy

Focusing, for instance, on the Threshold triggering policy a lower blocking is obtained when the *BBR* mechanism is applied compared to the *SSP* algorithm. Several conclusions can be obtained when analyzing Figure 48 in detail. Firstly, the lowest blocking is obtained when the *BDP* is applied, i.e., by the *B/BDP* algorithm. Secondly, although applying the *BDP* when *n_bp* = 1 hardly affects on the blocking

ratio, we can conclude that the effects produced when including the *BDP* in the *BBR* mechanism are completely dependent on the *n_bp* value. In fact, whereas a similar blocking is obtained for the *BOSP(1)* and the *B/BDP(1)* algorithms, (13.27 % and 13.3 % respectively) a significant reduction of 2 % is obtained when *n_bp* = 2. Increasing the *n_bp* values leads to obtain an even more significant blocking reduction. Hence, the larger the number of computed *bypass-paths* per route the lower the blocking, that is, the blocking depends on the computational cost introduced in the network. Lastly, the larger the threshold value, i.e., the inaccuracy, the larger the improvement in the blocking reduction. In the worst conditions (the threshold *tv* of the triggering policy is 90 %), the bandwidth blocking ratio obtained when applying the *BDP* process substantially improves that obtained by only applying the *BOSP*. For instance, when *n_bp* = 3, the obtained reduction in the blocking ratio is about 3.75 %.

Figure 50 and Figure 51 depict the number of routes incorrectly selected for the Threshold and the Exponential class triggering policies respectively. A similar behaviour is obtained for both triggering policies. As expected, the number of routes incorrectly selected decreases with the number of computed *bypass-paths*. Focusing again on the Threshold triggering policy, in the worst conditions (*tv* = 90 %), a reduction of 1.3 % is obtained when applying the *B/BDP(3)* algorithm compared to the *BOSP(1)* algorithm.

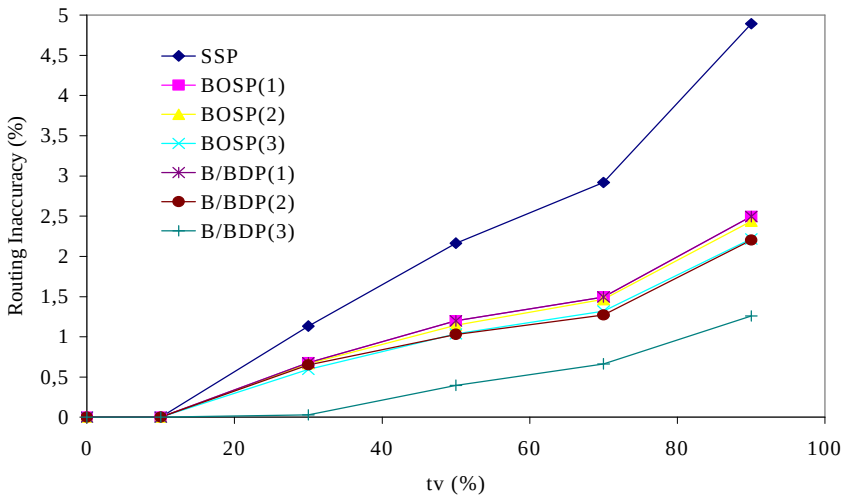


Figure 50. Routing Inaccuracy for the Threshold triggering policy

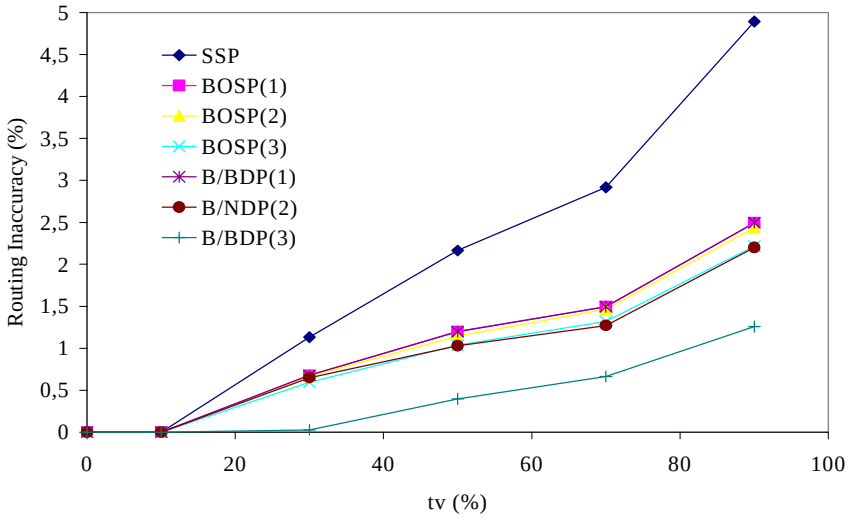


Figure 51. Routing Inaccuracy for the Exponential class triggering policy

Finally, Figure 52 and Figure 53 show the cost of applying the *BBR* mechanism with and without *BDP* capabilities. Moreover, it is also drawn the variation in the cost produced as a function of the *n_bp* value.

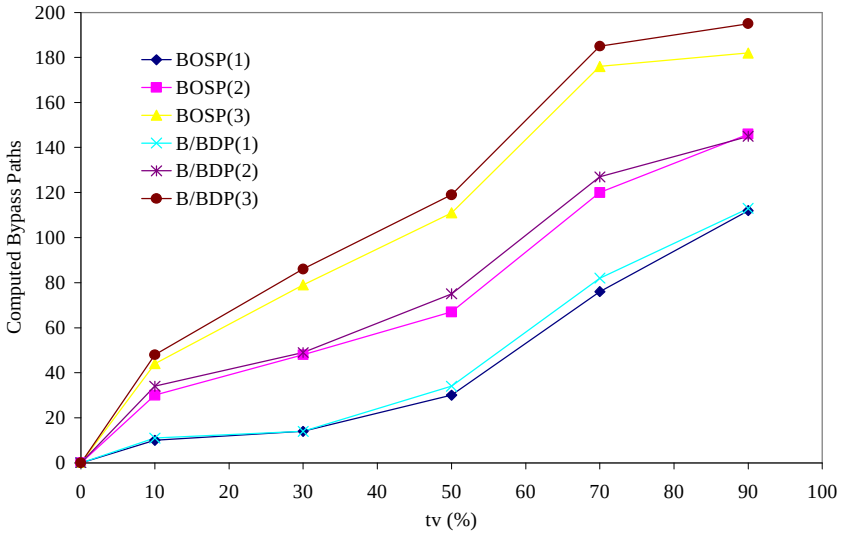


Figure 52. Computed *bypass-paths* for the Threshold triggering policy

Definitely, we can conclude that including the *BDP* in the *BBR* mechanism substantially improves global network performance.

In this Chapter the obtained bandwidth blocking reduction has been evaluated depending on two factors. On the one hand, it has been shown that introducing the

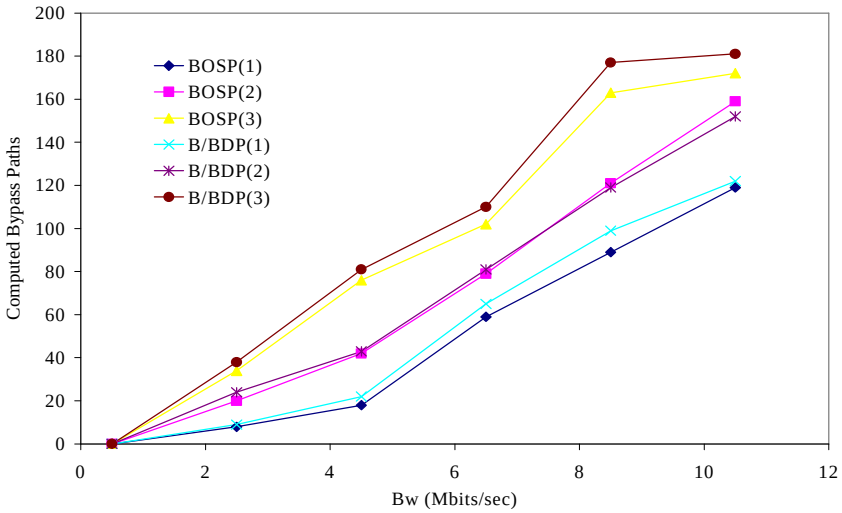


Figure 53. Computed *bypass-paths* for the Exponential class triggering policy

BDP process in the *bypass-path* computation significantly reduces the bandwidth blocking. This is because a high number of *bypass-paths* may be computed to bypass links defined as an *OSL*. On the other hand, different values of the *n_bp* parameter have been considered. Simulations have been performed for *n_bp* values of 1, 2 and 3. This means that only 1, 2 or 3 *bypass-paths* per route can be computed and used respectively. As expected, the larger the *n_bp* value the larger the blocking reduction, the larger the cost (in terms of total number of computed *bypass-paths*) and the lower the signalling overhead, i.e., number of update messages flooded throughout the network.

Summarising, the *BBR* mechanism is proposed to address the routing inaccuracy problem also introduced in this Part. Some routing algorithms are inferred from the *BBR* mechanism when considering either hopcount or load balance. *BBR* performance is enhanced both by adding the *BDP* process and by fixing the computational cost according to the network load. Results obtained by simulation show the benefits on global network performance when using the *BBR* mechanism.