Intra-Domain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis

G. Apostolopoulos
georgeap@cs.umd.edu
U. Maryland
College Park
Maryland, MD 20742

R. Guérin
guerin@ee.upenn.edu
U. Pennsylvania
200 S. 33rd Street
Philadelphia, PA 19104

S. Kamat
sanjayk@dnrc.bell-labs.com
Lucent Bell Laboratories
101 Crawfords Corner Road
Holmdel, NJ 07733

A. Orda
ariel@ee.technion.ac.il.edu
Technion
I.I.T.
Haifa, 32000 Israel

S. K. Tripathi
tripathi@engr.ucr.edu
Bourns College of Eng.
U. of CA, Riverside
Riverside CA 92521

Abstract

Constraint-based routing gradually becomes an essential enabling mechanism for a variety of emerging network services such as virtual private networking and QoS support. A number of recent works have recognized its significance and investigated many aspects of the operation of constraint based routing and in particular its variant that is concerned with determining paths for requests with specific QoS requirements, known as *QoS routing*. In this work we build on previous results on the cost of QoS routing and investigate the performance/cost trade-offs involved in the operation of a representative QoS routing architecture, elaborate on the constituents of this cost, and identify the main methods for containing the cost that QoS routing incurs on routers. Our results show that the cost of QoS routing is not excessive and that there indeed exist operational configurations, that can achieve reasonable performance gains with only a minimal increase in processing cost when compared to conventional best-effort routing.

1 Introduction

Routing in IP networks has so far been based on a *best-effort* service model, in the sense that established routing protocols build routing tables with the only objective of minimizing the *administrative cost* of each path. Emergence of multiple service offerings in IP networks, some of which require service level assurances such as reliability, bounds on bandwidth availability, delay and packet loss, etc., have sparked interest in developing routing protocols for IP networks, that are capable of accommodating such constraints, i.e., *constraint-based routing*. A constraint based routing protocol can generate paths that satisfy certain specified constraints, such as routing policy constraints or Quality of Service (QoS) constraints. Routing with QoS constraints is also known as QoS routing, and is a longstanding research topic that has recently received renewed attention [1, 27, 25, 24, 26] (see [2] for an overview). In particular, some of the recent focus has been directed towards assessing the impact of deploying QoS routing protocols in IP networks, as well as developing techniques that achieve the best cost-performance trade-off for the constraints of most relevance in IP networks, e.g., available bandwidth. Initial efforts have been primarily directed at the *intra-domain* version of the problem, and this is what we focus on in this paper.

The main benefits of QoS routing are three-fold: it enables creation of virtual circuit-like services over IP networks, improves user satisfaction by increasing chances of finding a path that meets the QoS constraints, and improves network utilization by finding alternate paths around congestion spots. However, these benefits come at the cost of developing new routing protocols or enhancing existing ones, and of incurring potentially higher communication, processing and storage overheads. As a result, an important concern is whether or not QoS routing benefits are worth its cost. Answering such a question requires both experimentation and quantitative analysis, to better assess the costs and benefits of QoS routing in practice. The purpose of this article is to briefly present the scope of possible approaches to the problem of QoS routing and discuss design decisions that impact the routing performance and cost trade-off of QoS routing. In particular, we assess the nature and magnitude of the cost of QoS routing, identify the control parameters available to modulate it, and study the impact of such modulation on both cost and performance. For this purpose, we draw on recent results based on simulations and data derived from a reference implementation. These results demonstrate the feasibility of QoS routing in the intra-domain context.

The rest of this paper is structured as follows. In the next section, we provide some general background information on QoS routing, together with a brief outline of the specific model we focus on in this paper. Section 3 discusses the costs associated with each of the main components of QoS routing, and the controls available to modulate them. Section 4 discusses the methods used for measuring the cost of QoS routing. Section 5 presents experimental results illustrating the magnitude and the range of these costs for each component. This provides initial insight into their respective scale and the efficiency of the different mechanisms available to modulate them. Section 6 presents additional numerical results on the performance of QoS routing, that illustrate the trade-off that exists between lower cost and greater performance. Section 7 concludes the paper.

2 Scope and Goals of QoS Routing

2.1 Control Path and Data Path Interactions

A first step towards understanding the impact as well as the complexity of introducing QoS routing in an IP network, is to identify where it departs from the traditional best effort routing model. Routers running best effort routing protocols exchange routing information, which they use to construct their Routing Information Base (RIB), i.e., a collection of routes to all destinations. The routes in the RIB are then used to generate the router's Forwarding Information Base (FIB), which is structured for efficient look-up. The FIB is the *data path* instantiation of the RIB, which in turn can be viewed as part of the router *control path.* As packets arrive in the data-path, a destination based look-up of the FIB determines the outgoing interface on which packets are forwarded. Figure 1 depicts a simple functional block diagram for this model, for the case of a link state protocol such as OSPF [5]

The introduction of QoS routing preserves most of the structure of Figure 1, but it nevertheless introduces a number of small changes. For example, in order to allow determination of paths that satisfy more complex constraints, the RIB now will have to contain more detailed information about the network state, e.g., amount of unreserved bandwidth, delays and so on. This information may be quite dynamic, implying an increase in the amount of control traffic needed to keep the RIBs up to date.

Another place where QoS routing departs somewhat from the model of Figure 1 is in how the routing information is translated into forwarding state. The main difference is that we typically go

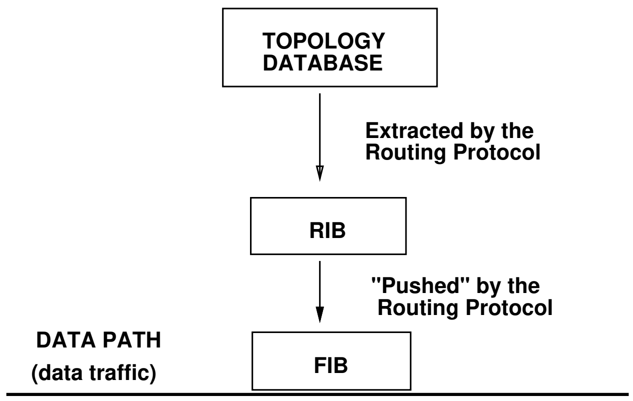


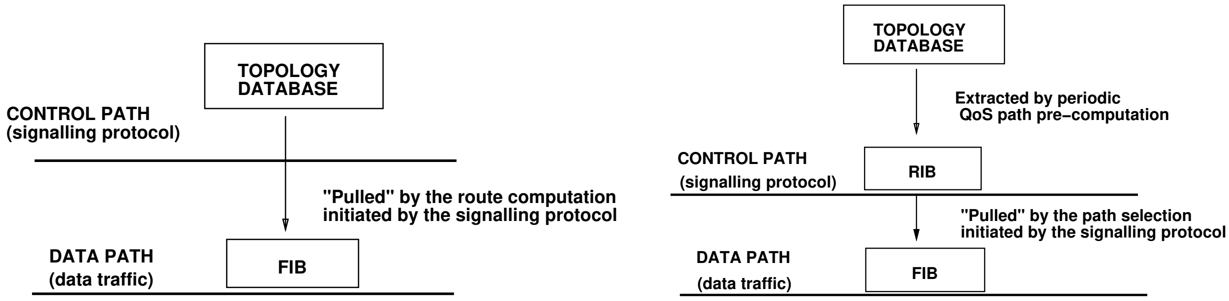
Figure 1: Data flow under conventional routing

from a *push* model used in best effort routing (the routing protocol pushes the content of the RIB into the FIB), to a *pull* model where QoS routes are selectively inserted in the FIB. The insertion of a QoS route in the FIB is usually triggered when a signaling protocol attempts to establish a QOS path for some traffic. The arrival of such a signaling message generates a query to QoS routing to find an appropriate path that meets the traffic requirements. If such a path is found, the FIB is suitably modified, so that when packets arrive in the data-path, they get forwarded along the corresponding path. RSVP [3] and the MPLS path set-up protocol [11] are examples of potential signaling protocols, and extensions to support QoS routing have been investigated for both [33, 32].

Figure 2 depicts a simple functional block diagram for this model, where the arrival of a signaling message is the trigger for the creation of the necessary forwarding state (FIB update). Figure 2(a) shows the case where a path, and hence the associated forwarding state, is computed anew for each request, while Figure 2(b) considers the case where this information is pre-computed and stored in the RIB from where it is then extracted on arrival of the signaling message. There are benefits to each approach in terms of cost-performance trade-off, and we discuss them further later in the paper.

The structure of the FIB itself is also affected by QoS routing. Forwarding, instead of being based only on the destination address (or address prefix), can be based on various classification criteria. Reservation signaling protocols define classification criteria for the FIB entries. RSVP, for example, uses a FIB entry based on the 5 tuple (source address, source port, destination address, destination port, protocol number). MPLS, on the other hand, uses *labels* to determine how to forward data traffic. Each data packet is pre-pended (at appropriate points, e.g., at an ingress router) with a label, and this label is used as the key for looking up the FIB and determining how to forward the packet. The advantage of a label based approach is that it can minimize the added complexity in the FIB,

To summarize, constraint-based routing, and in particular QoS routing, affects the traditional structure of a router along multiple dimensions. The data path is impacted because of the need for a larger number of forwarding entries with a different structure than the ones used for best effort routing. Similarly, the control path is also affected, not only because of the additional processing load induced by the signaling messages used to request and set up paths, but also because of the added processing and information exchange required in order to compute QoS routes. In this paper, we focus primarily on the latter aspect, although we briefly discuss the former in Section 3.5.



(a) On-demand path computation (b) Path pre-computation

Figure 2: Data flow under QoS routing

2.2 QoS Routing Models

The previous discussion identified some of the changes required to a traditional router architecture in order to support QoS routing. The cost of these changes will in turn largely depend on the frequency and granularity of the actions they correspond to, i.e., updates to the RIB and FIB. As a result, there exists a broad range of models and design points for QoS routing, that represent different performance goals and operating environments.

2.2.1 QoS Routing for Traffic Engineering

At one extreme, QoS routing is aimed primarily at *traffic engineering* (see [37] for an example), and its operation is then characterized by a long time scale (long term traffic variations) and a coarse granularity of the traffic flows it handles (traffic aggregates). In such an environment, the goal of "QoS routing" is maximization of network performance, e.g., minimize delay, in the presence of slowly changing traffic patterns. This is achieved through continuous measurements of traffic patterns and the computation of paths on which to route traffic aggregates so as to optimize various performance measures. One such instance of QoS routing is the optimized-multipath routing of [34, 35]. This work proposes extending conventional routing protocols by monitoring network performance (most notably utilization of links and delay) and dynamically shifting traffic across pre-computed paths in order to optimize performance.

In the context of a traffic engineering application, the different paths computed by QoS routing are either pre-established or change only infrequently, so that when coupled to the long term nature of the traffic monitoring function, the overall cost of such an approach is typically low. On the other hand, the *guarantees* that such a traffic engineering based approach can provide are limited. This is because the use of traffic aggregates and the focus on network wide traffic optimization when computing paths make it difficult to provide explicit guarantees to individual flows.

2.2.2 QoS Routing for Dynamic Requests

At the other end of the spectrum of possible QoS routing solutions, QoS routes are computed for each *request*, where requests explicitly express their resource requirements. Handling of such explicit requests differs from the implicit allocation of resources inferred from the traffic engineering measurements. In particular, requests are likely to be more frequent, and the granularity of resource allocation smaller, e.g., each individual flow may require reservation of resources. As a result, the goal of QoS routing in this environment is also somewhat different. It still targets optimizing network resources, but now under the constraints of satisfying individual request requirements rather than a general measure of network performance. It should be noted that while this provides *stronger* guarantees, it incurs a potentially much higher cost. This is not only due to the greater overhead in updating network state and computing QoS paths for each request, but also due to the signaling overhead to set up individual paths for each request.

In other worlds, this type of QoS routing represents a “worse case scenario”, i.e., the flavor of QoS routing with the highest cost. In the present study we concentrate on this type of QoS routing, since its cost should be an upper bound on that of other simpler QoS routing solutions. A good understanding of this cost will allow conclusions to be drawn about the feasibility of various QoS routing options.

This being said, many previous works have been devoted to investigating various aspects of QoS routing, and it is important to acknowledge their contributions. The two main environments where such investigations have been conducted parallel the ones used in traditional best-effort routing, i.e., link state and distance vector protocols. The majority of those investigations have been in the context of extensions to link state protocols [16, 17, 18, 15, 4, 1, 26, 31], even if several proposals have also explored distance vector algorithms [26, 25, 22, 28, 29, 24, 30]. Most of these works deal primarily with requests with bandwidth requirements, the focus of this paper, even if a number of papers also address the more complex cases of additive constraints, e.g., delay, [16, 17, 18], and multiple constraints, e.g., bandwidth and delay, [23, 24, 19, 20, 21, 22, 15].

3 Evaluating The Cost Of QoS Routing

3.1 Routing Architecture

In order to proceed with our evaluation of the feasibility of QoS routing when used to handle individual requests, we focus on a specific approach, namely a solution based on a link state protocol aimed at unicast connections requesting bandwidth guarantees. As mentioned in the previous section, such an approach is representative of most of the proposals previously put forward. Furthermore, unicast connections and bandwidth guarantees are likely to be the more important initial targets for QoS routing, and therefore represent an appropriate first step for our investigation.

Once the choice of a link state approach has been made, the next task is to design a protocol that incorporates the requirements of QoS routing. While it is possible to design a brand new protocol, there are many basic functions, e.g., neighbor discovery, flooding of updates, etc., that QoS routing shares with traditional best effort, link state routing protocols. As a result, it is typically easier to extend an existing protocol rather than to develop a new one. This is the basis of the proposal of [4], that describes extensions to OSPF to support QoS routing. Substantial experimental data is available for this proposal; which makes it a suitable choice for the QoS routing architecture to be used in our investigation.

In the rest of this section, we review the core functional components of a QoS routing protocol and their contributions to its cost. These can be broken down into three major categories: 1. Protocol; 2. Processing; and 3. Storage. For each, we outline the impact of QoS routing and its cost implications. In addition, we briefly discuss the issue of the signaling and packet forwarding costs introduced by QoS routing. This aspect was briefly mentioned earlier, and while it is not the focus of this paper, we summarize available alternatives, results, and their implications.

3.2 Protocol Overhead

One basic requirement for supporting QoS routing is to track the availability of network resources, i.e., link bandwidth, so that this information is available to the path selection algorithm. The Link State Advertisements (LSAs) flooded by OSPF already carry administrative cost metrics for each link, and there is a provision for advertising multiple cost metrics using Type of Service (ToS) fields. As a result, a simple solution (proposed in [4]) is to build on this existing mechanism, and use the ToS fields to encode and flood information such as available link bandwidth and propagation delay. As an added benefit, because the existing OSPF update mechanism triggers the simultaneous flooding of updates for all links on a router, the processing cost of QoS updates will be amortized over multiple links. However, while the distribution of QoS updates can be accomplished with minor modifications to OSPF, additional mechanisms are still needed to determine when updates are to be sent. In particular, routers need to track available bandwidth on their interfaces, and determine when it has changed sufficiently to warrant a new update. The latter is particularly important as it plays a major role in both the protocol overhead and the performance of QoS routing.

The decision of when to flood QoS updates is the responsibility of a triggering function, whose design involves various trade-offs between performance and cost. In particular, a sensitive triggering function that advertises every change in resource level, provides the most accurate information for computing paths, but its communication overhead is often not acceptable. A simple method for bounding the communication overhead, is to rely on a timer to limit the frequency of advertisements. This clearly provides direct control over the volume of updates, but may not ensure timely propagation of significant changes. Another alternative is, therefore, to rely on the magnitude of changes as the primary criterion for triggering updates. For example, a threshold based method triggers a new advertisement whenever the change in available resources exceeds a certain percentage of the previously advertised value. Alternatively, resources may be partitioned into ranges or classes, with new advertisements being issued for each class boundary crossing. Such methods provide some control on the trade-off between information accuracy and volume of updates. However, periods of rapid traffic fluctuations may still trigger frequent flooding of updates and as a result cause transient control overloads, so that threshold or class based triggering functions are often complemented with a (short) timer (which we will call *hold-down* timer) to enforce a minimum spacing between consecutive updates. In the paper, we evaluate the efficiency of various combinations of thresholds and timers in controlling the volume of QoS updates, and illustrate the effect that lowering update cost has on routing performance.

Besides the triggering mechanism, other factors also influence the volume of update traffic. For example, the network topology and connectivity determine the actual number of link state update packets that are flooded on each link. Similarly, the characteristics of requests affect the level of activity in the network, i.e., number and duration of requests, and consequently the frequency of update generation. We attempt to also explore the sensitivity QoS routing to these additional factors by varying network size, topology, and traffic patterns.

3.3 Processing Requirements

There are several aspects of QoS routing that introduce different or additional processing requirements from traditional best effort routing. The two major ones are discussed below.

- **Path Computation and Selection:** Path computation is the component whose implementation differs most from its best effort counter-part. QoS routes are computed based on request characteristics, e.g., how much bandwidth is required, and the resource information provided in the topology database. Differences with the best effort model are along two axes: The algorithms used to compute routes, and the conditions that trigger algorithm execution. The latter is a major factor in the computational overhead associated with QoS routing, as well as the in quality of the routes being computed. As mentioned in Section 2.1 and illustrated in Figure 2, paths can either be computed *on demand*, i.e., for each new request, or pre-computed. An on-demand approach has the benefit of being able to always use the most recent information. However, if requests arrive too frequently, this approach may prove costly even if the algorithm is of relatively low complexity. As a result, it is desirable to explore alternatives of lower computational complexity. One possibility is to rely on *path caching* [9, 12], which seeks to reduce computational complexity by reusing previously computed paths. Yet another approach is to *pre-compute* a QoS routing table in a manner similar to the way a best-effort routing table is pre-computed. However, since the amount of bandwidth requested is not known in advance, such a routing table needs to pre-compute and store multiple alternative paths to each destination, potentially for all possible values of bandwidth requests.

An efficient approach to solving this problem, using a modified version of the Bellman-Ford shortest path computation algorithm is proposed in [4]. Specifically, for each destination d, i.e., entry in the best effort routing table, the algorithm generates an ordered list of sets of paths $S_d = < S_d^1, S_d^2, \ldots, S_d^n >$. The set S_d^h contains all paths from the router to d with path length of at most h and the largest available bandwidth. The QoS routing table generated by the algorithm can thus be conceptually viewed as a matrix, with each row associated with a particular destination, and each column corresponding to a given hop count and containing information about the highest bandwidth path available to the destination. As a result, the table contains a series of paths with increasing bandwidth and hop-count paths for each destination.

In terms of processing load, there are pros and cons with both an on-demand and a pre-computation approach, and either can yield a lower processing load based on operating conditions. Executing the on-demand path selection algorithm is simple since it only involves traversing the topology database and determining a single QoS path. The main contributors to the cost of a single path computation are then the network topology and the relative distance between the destination and the source, although the size of the request and the levels of available bandwidth on network links also have some minor impact on the computational complexity of the algorithm. The more important factor in determining the overall cost of on-demand path computation is the frequency of new requests.

In contrast, path pre-computation is mostly insensitive to the frequency of new requests, and primarily depends on how often the QoS routing table is recomputed, which as opposed to the arrival rate of new requests is a parameter that the router can control. Clearly, frequent re-computations improve accuracy and, therefore, routing performance, but this comes at the cost of a substantial increase in processing load. This is because building a complete QoS routing table is typically more complex than computing a single path, and it further involves the additional cost of de-allocating and re-allocating memory. In addition, when paths are pre-computed, an additional step, i.e., path selection, is required to retrieve a suitable path when an incoming request needs to be routed. A suitable path is one with sufficient bandwidth, and it is retrieved from the QoS routing table by searching column by column the row associated with the selected destination. The search stops at the first entry with an available bandwidth larger than the requested value. Note that this means that the cost of path selection will depend to some extent on the requested amount of bandwidth.

- **Processing Link State Updates:** Generating and receiving a link state update involves accessing the link state database to extract information or insert newly received information. In addition, the generation of a link state update requires that a packet be assembled and transmitted. Given the greater frequency of updates that QoS routing requires, these are likely to be important contributors to the increased cost of QoS routing.

3.4 Storage Costs

There are two areas where QoS routing affects storage costs. The first is the extension of the topology database in order to accommodate link resource availability information. In the context of OSPF, this is easily accomplished through minor modifications to the existing topology database. This extension is facilitated by the fact that resource updates are themselves communicated using existing OSPF mechanisms, i.e., flooding of extended LSAs as described above. As a result, both processing of resource updates and their inclusion in the topology database, can be added with minimal modifications.

The QoS routing table itself, when used, also affects storage costs. The size of the QoS routing table depends to a large extent on specific implementation details such as the exact data structures used. However, it is also affected by parameters of QoS routing such as the operation of the triggering policy, that can influence the number of distinct paths that the path computation algorithms generates. For example, there may exist, for each destination, a large number of distinct paths with incrementally different bandwidth values, hence contributing to a larger QoS routing table.

In the rest of the paper, we ignore the storage cost dimension of QoS routing, as from our experience we have found QoS routing to only have a small impact on this factor. In particular, the increase in the size of the topology database is minor, and although a QoS routing table, when used, can be large, we do not anticipate it to be a problem for the storage capabilities of modern systems. A detailed discussion of the storage cost of a specific implementation of QoS routing, and its comparison to that of best effort routing, can be found in [10].

3.5 Signaling and Packet Forwarding Costs

As discussed in Section 2.1, QoS routing also requires additional forwarding and signaling support, and this does not come for free. The impact on forwarding, i.e., the structure of the FIB, is mostly in terms of the potentially greater number of entries that need to be stored and searched. The added complexity this imposes depends in part on the packet classification approach used in the FIB. The use of a "native" layer 3 classification, e.g., as with RSVP classifiers, can impose a substantial overhead if it needs to be performed at a very fine granularity. However, various proposals have been made, e.g., see [39], that can provide efficient and flexible layer 3 classification methods for a large number of flows. Alternatively, a "layer 2" lookup approach based on negotiated labels can also be used to handle the forwarding requirements of QoS routing. For example, this would apply to an MPLS environment, where different labels would be assigned to enable forwarding along different QoS paths. As a result, we do not expect the forwarding cost to be a major inhibitor of QoS routing, although clearly there would be limitations imposed by the size of the FIB feasible with different technologies.

The other, and probably more significant cost component, is the signaling needed to install the required FIB states in the nodes used by a QoS path. This signaling involves both additional network traffic and processing. The load induced by signaling will clearly depend on the protocol used, but it could be substantial in the presence of a large number of short-lived flows. However, there are various approaches that can be used to mitigate this impact. For example, QoS routing may be reserved for *long-lived* flows as suggested in [38]. This should help lower the frequency of signaling requests. Alternatively, QoS routing may be performed only at the level of some aggregates as is suggested in the various models mapping Integrated Services flows onto Differentiated Services "pipes," e.g., [45, 44]. This would also contribute to reducing the number of QoS requests. Finally, signaling protocols themselves can be made more efficient (see [36, 40, 41, 42, 43] for examples), and even existing protocols, while certainly contributing added complexity, may not be so expensive as to make QoS routing unaffordable. For example, the measurements of [36] indicate a 1 msec setup time for RSVP, when running on a low performance processor, i.e., a 33MHz 68040 CPU. Given the progress in processor technology, an improvement by a factor of one order of magnitude and maybe two is possible.

As a result, while the cost of establishing the reservations that go hand in hand with QoS routing is certainly a topic that requires further investigation, it does not appear to represent a "show-stopper" for QoS routing, i.e., one that would make its deployment completely impractical. In that context, it remains important to assess the other costs and associated complexity of upgrading the (IP) routing infrastructure to support QoS routing. Exploring these issues is what the rest of this paper is about.

4 Evaluation Approach

In order to get a better handle on the actual costs of QoS routing, it is necessary to evaluate and measure them in a realistic environment. This environment should be as close as possible to the actual setting of a real router, while allowing tuning of the various parameters mentioned earlier. The latter is important in order to sample a reasonable range of operational points. In this section, we briefly describe and justify the methodology we follow to evaluate the operational cost of QoS routing.

A key factor that affects the cost of QoS routing is network size. In order to assess how cost varies with network size, we rely on a sample topology whose size (number of routers in the network) can be easily and systematically changed. Specifically, we use topologies that are constructed by repeating a basic building block consisting of 4 routers and 5 transit networks as shown on the right hand side of Figure 3. The left hand side of Figure 3 shows a sample topology obtained by concatenating two building blocks along each dimension, i.e., a 2×2 mesh topology. In our experiments we will use instances of this topology ranging from 1×1 to 8×8. Furthermore, links are chosen to have a capacity of 45 Mbits/second and a propagation delay of 1 millisecond. The topology used in this work is meant to represent an *intra-domain* network. As a result, the overall size of the network is limited. Using larger topologies does not make sense since, in these cases, completely different routing architectures must be used, due to the scaling limitations of flooding.

As discussed in Section 3, the operational cost on QoS routers also depends on the traffic that the network has to handle as well as the settings of the parameters of the QoS routing protocol, i.e., path pre-computation frequency and threshold for triggering new link state updates. In order to evaluate the dependency of our results on the characteristics of the request traffic we use different combinations of duration and arrival rates as well as traffic patterns. In all experiments, we generate a workload assuming that requests arrive at each node independently, according to a Poisson distribution, and have exponentially distributed duration. The amount of requested bandwidth is uniformly distributed between a minimum value of 64 Kbit/sec and a maximum value that varies. We use two different workloads. The base one consists of requests that can have bandwidth requirements up to 6 Mbits/sec and average duration of 3 minutes. This corresponds to a "typical" workload, with requests large enough to represent video traffic (MPEG-2 for example) and duration

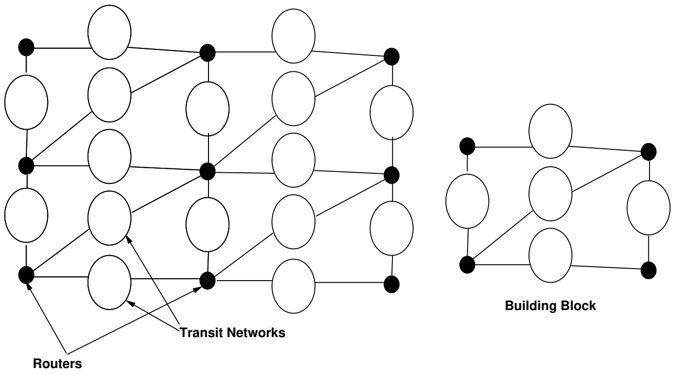


Figure 3: The variable size mesh topology

values derived from the experience of telephony networks. Our second workload consists of “small” requests, up to 1 Mbit/sec Duration and arrival rates are scaled appropriately so that the overall offered load to the network remains about the same. In particular, we maintain the same average duration for all types of traffic (3 minutes) and only vary the arrival rates. We also vary the distribution of traffic, and consider both uniform and non-uniform traffic. With uniform traffic all source-destination pairs are equally likely, while with non-uniform traffic some pairs of nodes have increased traffic levels. These nodes in our tests are the four corner nodes of the mesh topology. Each of these nodes has increased traffic levels with the node in the opposite corner.

In our experiments, we assume that all the incoming traffic has QoS requirements. This means that there is no best-effort traffic, sharing network resources with the QoS traffic. This, although not realistic, does not affect the validity of our results. As discussed in [6], the coexistence of best-effort and guaranteed traffic raises additional issues; ensuring adequate performance for best-effort traffic may require modifications to the path computation algorithms. These modifications though will not seriously affect the overall cost of the QoS routing architecture since they do not change the nature of either path computation or link state update processing and propagation. Handling best-effort traffic does not incur any additional overhead in terms of maintaining forwarding state or QoS route table entries. All best effort traffic can follow a default path, which is the one that would have been computed by the conventional routing protocol. Consequently, a single FIB and QoS route table entry per destination would be sufficient for handling best-effort traffic.

In our cost measurement experiments we used specific sets of values for the QoS routing protocol parameters. In particular, when path pre-computation is used, path pre-computations are performed periodically, for period values of .5, 5, 10, and 20 seconds. We also consider the case where paths can be computed on-demand, in which case the pre-computation period need not be specified. Similarly, a range of different update triggering threshold values between 10% and 80% were used. A value of 10% provides very precise link state information but corresponds to a relatively large volume of update traffic. A value of 80% substantially reduces the amount of update traffic, at the cost of greater inaccuracy in link state information. Finally, hold-down timers were used in some cases to further control the volume of link state updates generated. They took various values between 0 and 200 seconds. We believe that these combinations of parameter settings provide a reasonably comprehensive coverage of different operational environments.

4.1 Measurement Methodology

The process of QoS routing (and consequently its cost components) can be reduced to a set of basic operations. These operations are 1) path computation (either pre-computation or on-demand) and the associated path selection in the case of pre-computed paths and, 2) reception and generation of a link state update. In the following we discuss the costs of these individual operations, as well as what happens when a real router performs a mix of the above operations as part of its usual routing processing. The methodology we follow is essentially that used in [10].

In [10] we presented how the cost of these individual operations can be measured using a sample implementation of QoS routing extensions to the OSPF routing protocol. The three functional components of QoS routing (link state update triggering, extensions to the link state database, and path computation) were implemented as extensions to the `gated` implementation of the OSPF routing protocol. Then, in a sequence of experiments involving a single and two interconnected routers, the cost of the above operations was measured. In all experiments, the test systems used were IBM Intellistations with a Pentium Pro 266 MHz processor, 64 Mbytes of real memory, 3.4 Gbytes of disk, running FreeBSD 2.2.5-RELEASE and `gated 4.0` software. The dependency of the path computation costs on the topology was captured by artificially populating the topology database so that it reflected an arbitrary topology.

Measuring the cost of individual QoS routing operations alone is, however, not sufficient to provide a complete assessment of the impact of QoS routing on a router's operation. This is because, while such atomic measurements provide an accurate estimate of the intrinsic cost of QoS related operations, they do not fully capture the many dependencies and interactions that take place in a real router, i.e., when and how often operations occur and overlap with each other.

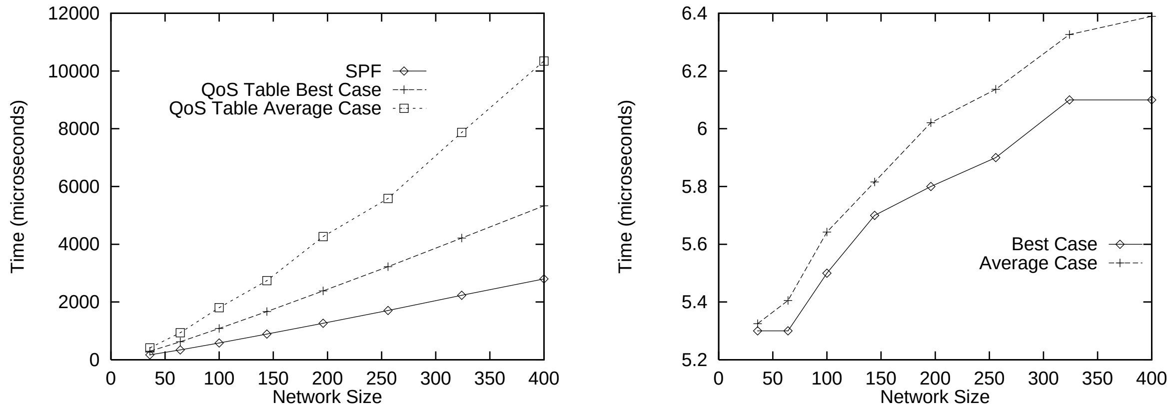
Nevertheless, as discussed in [10], it is possible to obtain a "reasonable" estimate of the operational cost of QoS routing by combining simulation and experimental data. Specifically, while performing a simulation run, we generate a log that contains the time at which each of the following operations occurs at every router: a) generation of an LSA, b) reception of an LSA, c) initiation of path pre-computation (when applicable), and d) initiation of path selection. The timing information contained in these logs, together with the experimental data on the cost of each operation (derived from the stand-alone experiments) enables us to recreate and compute the processing load at the router. In particular, we compute both "instantaneous" (over 50 millisecond intervals) processing load, and average processing load over the duration of the simulation experiment.

5 Evaluating The Cost of QoS Routing

In this section, we present measurements of the cost of the different components involved in implementing QoS routing. This discussion focuses only on evaluating these costs and does not address their relation to routing performance. Section 6 will address in detail the issue of how cost and performance are related.

5.1 Processing Cost of Path Pre-computation

The path pre-computation approach incurs the cost of periodic computation of the QoS routing table and the additional cost of accessing the table to select a path for a given request. Varying the size of the mesh topology allows us to observe how both computation and path selection costs vary with increasing network size. The time needed for computing the QoS routing table for different topology sizes is shown in Figure 4(a), which also shows, for comparison purposes, the computation time of the standard OSPF routing table. Note that for QoS routing, the pre-computation times



(a) Processing time for path computation

(b) Cost of path selection

Figure 4: Path pre-computation costs

include the cost of de-allocating the previous QoS routing table. The exact cost of computing the QoS routing table depends on the bandwidth availability on the network links, since different combinations of values can result in a different QoS routing table and, therefore, a different cost for computing the table. Consequently, we show results for a “best” case, where only a single path (bandwidth value) is maintained in the QoS routing table, for each destination, and an average case where there are multiple paths to each destination based on a random assignment of link bandwidths. Averaging is performed over all destinations, assuming they are all requested with equal probability. Figure 4(b) also displays the cost of path selection, which is incurred for each request when path pre-computation is used. As can be seen from Figure 4(b), this cost is small and will typically be negligible when compared to other cost components.

5.2 Processing Cost of On-Demand Path Computation

When a path is computed for each new request, the computation cost is also a function of the destination of the incoming request, as it affects the number of iterations the path computation algorithm will go through. Figure 5 shows the cost of on-demand computation for different path lengths and topologies. As the path length increases, the cost of computing paths on-demand approaches[1] that of computing the QoS routing table. For all topologies, the cost of finding a path of a given length is the same, thus indicating that the main factor that determines the cost of computing an on-demand path is its hop length, i.e., accessing the topology database to retrieve link metrics information only has a minor impact.

5.3 Link State Advertisement Generation and Reception

The time required for generating and receiving LSAs was measured by connecting two routers running OSPF with the QoS routing extensions. The machines were configured so as to form an OSPF adjacency, and then exchange LSAs between them. The OSPF hold down mechanisms were disabled, in order to get accurate measurements of the LSA generation and reception costs. According to these measurements, either generating or receiving an LSA takes about 900 microseconds in our test system.

[1] With remote destinations, paths to all destinations are usually computed before the destination is reached.

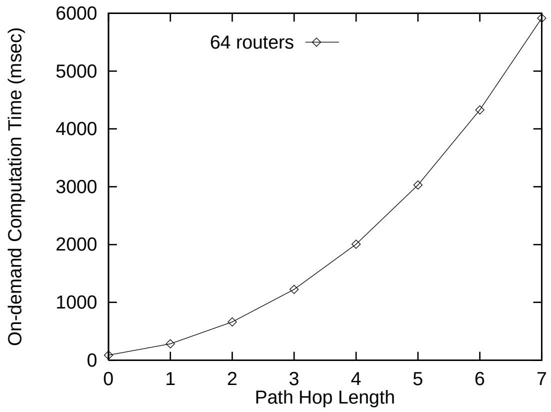


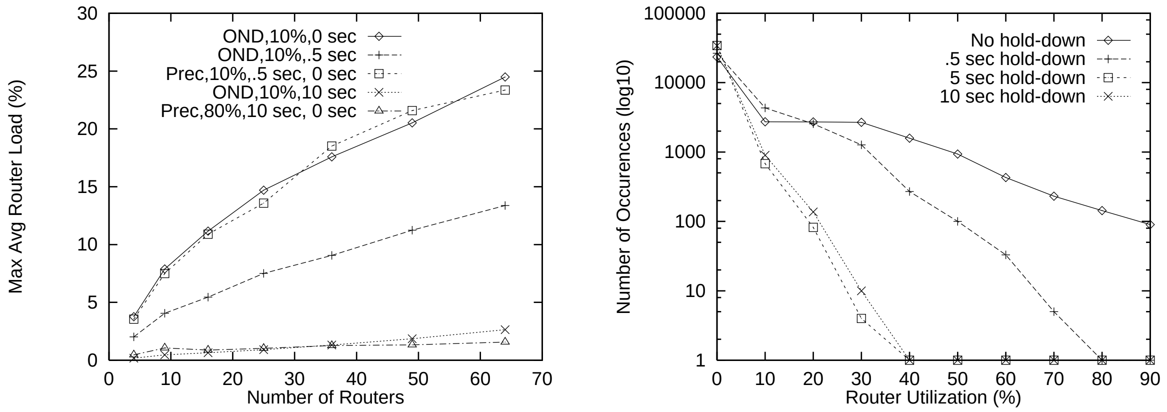
Figure 5: Cost of on-demand path computation

5.4 Operational Costs

In this section, we show how the above costs shape the overall processing cost that is incurred in a router that participates in a larger network under some realistic load conditions. In particular, following the method described in section 4.1 we combine the costs incurred by both path computation and flooding into a single value for processing load. This value depends on the settings of path pre-computation period and update triggering thresholds as well as the traffic characteristics, i.e., traffic arrival rates, and patterns. In the following figures, unless otherwise noted, the maximum average load observed over all routers in the network is shown.

In the legends of the figures we follow a simple convention. First, we indicate the type of path computation (i.e., on-demand or pre-computation). The next number corresponds to the setting of the update triggering threshold, when applicable. In the case of path pre-computation, the pre-computation period is indicated next. Finally, the last number gives the value of the hold-down timer, with a value of 0 indicating that no hold-down timer was used. Confidence intervals were computed for all the results reported in the figures. For routing performance, the 95% confidence interval was found to be less than .05%, while it was less than 5% for processing costs and link update traffic. In all the experiments reported here, the average request arrival rate was one request every 6 seconds. Consequently, when comparing on-demand and path pre-computation, if paths are pre-computed every n seconds, then up to $n/6$ on-demand computations are replaced by a single path pre-computation.

In Figure 6(a), we show how the maximum observed average load scales with increasing topology size, for various combinations of threshold and hold-down timer settings. In particular, we show a high cost case, where a small update triggering threshold is used (10%) without hold-down timer, and a lower cost case where the update triggering threshold is again 10% but a hold-down timer of 10 seconds is used. In both cases, path pre-computations are performed very frequently (every .5 seconds) so that both on-demand and path pre-computation have similar routing performance. The cost of both on-demand and pre-computation is similar in both cases. A key finding from the figure, is that the use of a hold down timer can drastically reduce the total processing cost. This indicates that processing update traffic is the major component of the processing cost of QoS routing. Indeed, we verify this in Figure 7(a) where we compare the contribution of the path computation cost to the overall cost of QoS routing, for both on-demand and path pre-computation. The figure shows results (note the logarithmic scale) for a high cost case, where no hold-down timer is used. From



(a) How router load changes with topology size

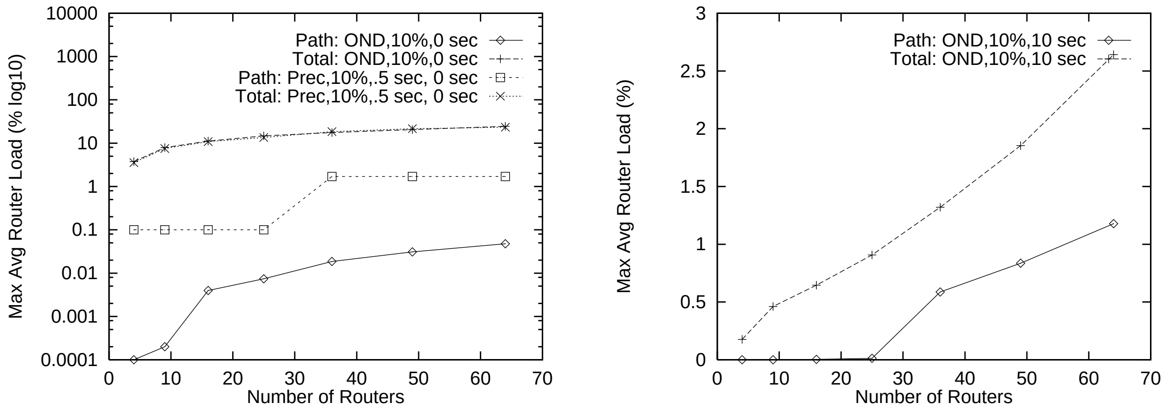
(b) Distribution of instantaneous router utilization

Figure 6: Average and instantaneous router load

the figure, we see that the contribution of path computation to the overall cost of QoS routing is several orders of magnitude smaller than the total processing cost, which is dominated by the update processing cost. The figure also shows, that in the pre-computation case, the cost of path computation is higher than that of on-demand. This is primarily because of the small period of path pre-computation used in this case (.5 seconds), and is meant to illustrate the impact of this parameter.

The values shown in Figure 7 correspond to average load values, but it is also of interest to see variations in the processing load over shorter time intervals. For that purpose, Figure 6(b) plots the distribution of the "instantaneous" processing load, i.e., the processing load measured over a single 50 millisecond period. From the figure, we see that when no hold-down timers are used, the tail of the distribution can extend to large values. However, the figure also shows that the use of hold-down timers can essentially eliminate such transient overloads. In our experiments, even a hold-down timer of .5 second was sufficient to achieve drastic reductions in the tail of the distribution of the processing load. This is also reflected in the average router load, which is also significantly reduced as can be seen in Figure 6(a). As a result, when hold-down timers are used, the average load provides a reasonable measure of the actual processing load at our router. In particular, the results of Figure 6 show that our sample router should be able to handle, i.e., operate with an average load below 75%, networks of size well in excess of the maximum recommended size for an OSPF area (about 400 routers).

Here we should note that although our results indicate that update processing is the dominant contributor to the cost of QoS routing, it does not mean that we can ignore the cost of path computation altogether and, for example, assume that we can always use on-demand path computations. Specifically, when the volume of update traffic is reduced as a result of a hold-down timer or a large update triggering threshold, path computation is not an insignificant contributor to the total cost anymore, and controlling it becomes important. This is illustrated in Figure 7(b) where we show the cost contribution of the path computation for on-demand path computation when a hold-down timer of 10 seconds results in a drastic reduction of the update traffic. In this case, processing cost accounts for almost 50% of the total cost of QoS routing. In such a context, on-demand path computations can end-up generating substantial processing loads, especially in environments where



(a) Contribution of path processing to router load

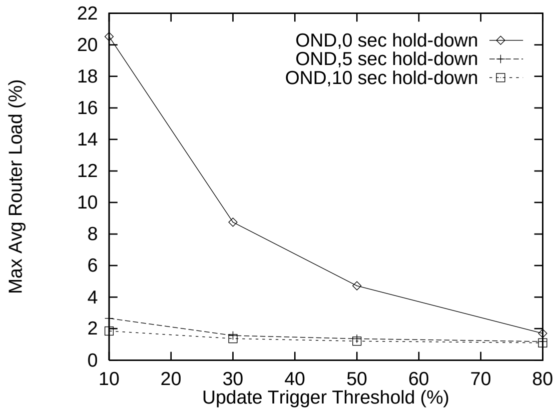
(b) Contribution of path processing to router load when a large hold-down timer is used

Figure 7: Router load

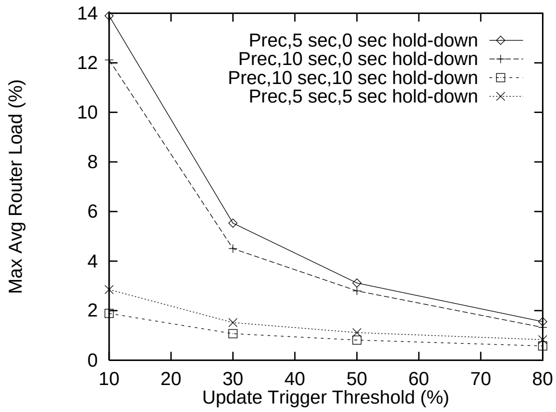
the arrival rate is high. Path pre-computation provides an option for lowering the cost of path computation in such cases. Furthermore, when link metrics are not updated very frequently, as is the case when mechanisms for controlling the volume of update traffic are used, on-demand path computation can be wasteful as it often recomputes the same path multiple times. This means that path pre-computation can indeed provide a more efficient and lower cost alternative in those instances. As we will see in Section 6, this is achieved with only minimal impact on performance.

Our next set of experiments is aimed at a better understanding of how processing load is affected by other parameters such as link state update thresholds, hold-down timers, and frequency of pre-computation of the QoS routing table (when applicable). We report these results in Figure 8 for a 7×7 mesh topology. Part (a) of the figure is devoted to the on-demand path computation case, while part (b) reports on the pre-computation case. From the figures, we see that reducing the sensitivity of the triggering of updates, can significantly reduce processing costs in both the on-demand and the pre-computation cases, especially when there is no hold-down timer. This is consistent with our earlier finding regarding the dominance of update processing in contributing to the overall processing load of our test router. As expected, for large settings of the hold-down timer, the dependency of the processing load on the triggering threshold is minimal since the hold-down timer itself limits the frequency of update traffic. As mentioned earlier, in the pre-computation case, processing costs also depend to some extent on the pre-computation period. This dependency is meaningful mostly for large hold-down timers, where the contribution of path computation to the total processing cost becomes significant. Finally, we notice that for small or no hold-down timers, the cost of on-demand path computation is higher than that of pre-computation. The reason for this difference is the lower routing performance of path pre-computation, which is caused by the relatively coarse period for pre-computing the routing table. This coarse period means that routing cannot take advantage of many of the updates that are generated. As a result, the paths provided by periodically recomputing the QoS routing table are not able to accommodate as many requests as when paths are computed on demand. This difference in the number of accepted requests then effectively translates into a lower rate of link state updates.

Another component of the cost of QoS routing is the additional link traffic that corresponds

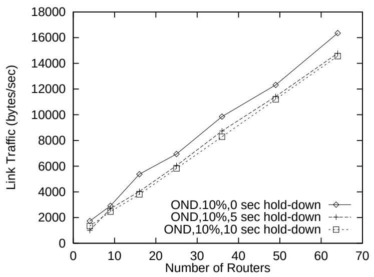


(a) On-demand

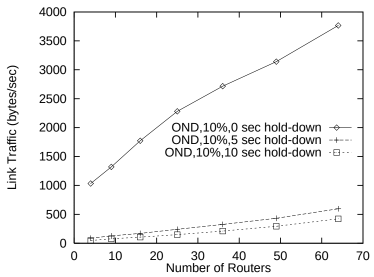


(b) Pre-computation

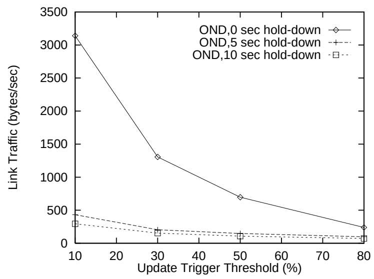
Figure 8: Maximum router load



(a) Maximum update traffic



(b) Average update traffic



(c) Average update traffic

Figure 9: Link traffic due to link state updates

to link state update traffic, i.e., the bandwidth consumed to transmit them. Figure 9(a) shows for the (Non-uniform, 1 Mbit/sec) workload, the variation of the worst case link traffic observed over the test network, i.e., the maximum value of update traffic over all links, as the network size increases. The link traffic does not appear to be sensitive to the value of the hold-down timer. This is mainly because we report maximum values of link traffic across all links. Even with a large hold-down timer, there exist some instant where updates from multiple nodes are synchronized, and result in a large volume of update traffic. If instead of the maximum values we plot the average value of update traffic over a particular link, we expect to see a strong dependency of the average update traffic on the hold-down timer. This is illustrated in Figure 9(b), which uses a worst case scenario with a 10% update triggering threshold and no hold-down timer, in order to magnify this dependency. As expected, using a hold down timer (or reducing the update triggering threshold) results in lower update traffic values. Finally, Figure 9(c) shows how the average link traffic for a single link of the 7×7 topology varies as a function of the update threshold. In all cases, except for very large threshold values, a hold-down timer can drastically reduce the volume of update traffic.

5.5 Conclusions on the Cost of QoS Routing

Based on the experiments we conducted and the results reported in this paper, we believe that the operational cost of QoS routing is well within the capabilities of current processors. Even the relatively low power machine we used as our router appears to be able to handle networks with hundreds of elements. While the average utilization of the router remained low, there were occasional incidents of overload, resulting from concurrent link state update triggering. The cost of processing and generating link state updates turned out to be the major component of QoS routing cost, with path computation making up only a small percentage of the total cost. Since link state update processing is the major component of the cost of QoS routing, decreasing the volume of update traffic can significantly reduce it. Both hold-down timers and larger triggering thresholds are capable of drastically reducing the cost of QoS routing. Finally, the bandwidth consumed by transmitting link state updates appears to be minimal, practically negligible for the link sizes used in our experiments.

Obviously, the above conclusions need to be revisited based on the actual benefits, i.e., performance gains achieved by QoS routing, especially when operating in configurations with relatively infrequent link state updates. This is the topic of the next section.

6 Routing Performance

The previous section dealt with the cost of QoS routing in a variety of configurations and under various parameter settings. Different methods for containing the cost of QoS routing were identified, such as reducing the sensitivity of the update triggering threshold or using a hold-down timer. However, while these methods are successful at controlling costs, this is achieved by reducing the accuracy of the available link state information and consequently affects routing performance. It is only after evaluating the magnitude of this impact, that we can reach a conclusion on the effectiveness and relative merit of these cost controlling methods. The rest of this section is devoted to investigating this issue.

Investigating routing performance is somewhat different from the investigation of the previous section, where what we needed was primarily a simple and systematic method to vary network size and study its impact on cost. For that purpose, the `mesh` topology provided a convenient solution that enabled us to easily increase network size and measure the corresponding increase in cost. Furthermore, experiments with both the mesh and a number of other simple topologies confirmed that network size rather than the exact shape of the topology was the dominant factor affecting cost across network configurations. The situations is, however, somewhat difference when routing *performance* is the aspect of interest. In particular, performance is known to be relatively sensitive to the relation between traffic patterns and network topology. In addition, characteristics of the topology such as the number of equal length paths it offers, the amount of link sharing between those paths, etc. are also all factors that influence routing performance. This means that results are likely to be much more sensitive to the choice of network topology.

In that context, it is unlikely that the `mesh` topology is an appropriate choice for investigating routing performance. This is not to say that it may not be applicable in certain environments, but it has a number of characteristics that are not really representative of typical topologies. For example, it exhibits a symmetry that is unlikely to be present in most networks, and because of this symmetry it offers an unusually large number of equal cost paths that can bias routing performance (see [31] for a discussion of this issue). As a result, and in order to better assess the impact of different hold-down timers and update triggering methods on routing performance, we use the more "realistic" `isp` topology, shown in Figure 10. When evaluating routing performance,

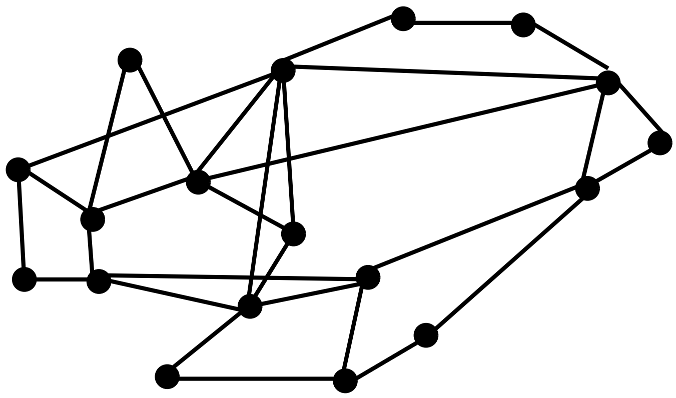


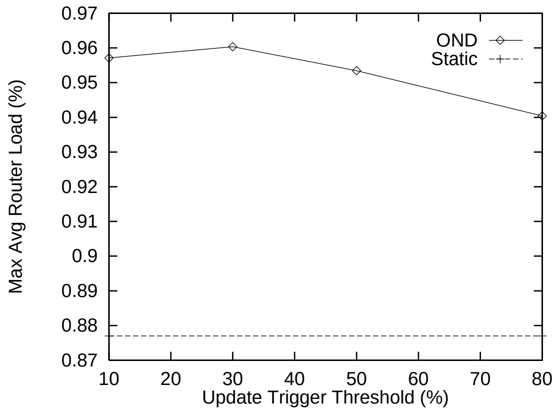
Figure 10: The ISP topology

we also use a best effort routing algorithm as a benchmark, and its performance is compared to that of QoS routing for different scenarios. The best effort routing algorithm we use, relies on a static shortest path routing algorithm that is insensitive to resource availability in the network or to request requirements. We measure routing performance using the bandwidth blocking ratio, first introduced in [1]. The bandwidth blocking ratio is defined as the percentage of the offered traffic that is successfully routed.

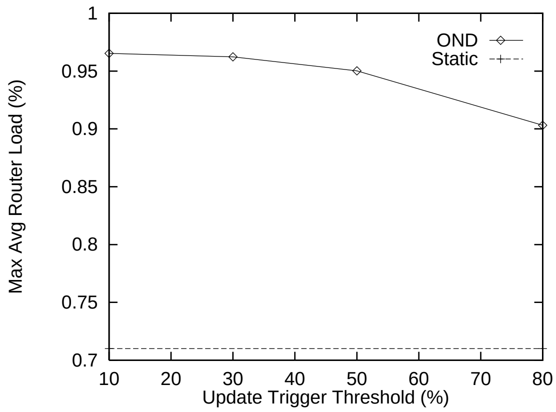
6.1 Effects of Traffic Patterns

We expect the effectiveness of QoS routing to also depend on the characteristics of the network traffic. In particular, the distribution of source-destination pairs, e.g., uniform vs. hot-spot traffic, and how well the traffic matches the network topology and link capacities, can both have a significant effect on the performance of QoS routing. Indeed, when the input traffic closely matches the network capacity, even a static routing algorithm should perform reasonably well, and the performance improvements achievable by QoS routing will be limited to taking advantage of short term imbalance on link loads. In contrast, when there is a substantial mismatch between the offered traffic and the routes used by static routing, e.g., because of a transient overload between a small set of sources and destinations, we expect the potential performance improvements achievable by QoS routing to be much more substantial.

In our experiments, we explore both the above scenarios. Figure 11 compares the performance of QoS routing and static best effort routing for uniform and hot-spot traffic, assuming request sizes of up to 1 Mbit/sec. As expected, the benefits of QoS routing (on-demand path computation, without hold-down timer) are more pronounced in the case of hot-spot traffic (shown in part (b) of the figure). For the particular scenario we have chosen, QoS routing achieves a 25% reduction of the bandwidth blocking rate when compared to static routing, compared to only 9% for the case of uniform traffic shown in part (a) of the figure. The routing performance improvements are indicative even if they depend on the particular experimental scenario used. In the next section, we explore further how the performance advantage of QoS routing behaves as we vary some of its parameters such as triggering threshold and hold-down timer value. This provides additional information on the relationship that exists between performance gains and costs.



(a) Uniform, 6 Mbit/sec



(b) Non-uniform, 6 Mbit/sec

Figure 11: Improvement over static routing

6.2 Effects of Information Inaccuracy

In section 5.4, we showed how the cost of QoS routing can be controlled with a combination of reduced sensitivity of the triggering of link state updates and hold-down timers. Although these methods can drastically reduce the cost of QoS link state update processing, they also introduce inaccuracies in the routing information available to the path computation algorithms and can, therefore, potentially reduce the performance of QoS routing. Similarly, path pre-computation can introduce additional inaccuracies when pre-computations are less frequent than traffic updates, e.g., the pre-computation period is larger than the time between consecutive updates. In order to investigate these issues, we proceed to compare the performance of on-demand and pre-computed paths while varying a number of system parameters. Specifically, Figure 12 compares the routing performance of periodic path pre-computation and on-demand path computation for the cases of uniform and non-uniform traffic of 6 Mbits/sec in the `isp` topology, and for different combinations of trigger thresholds and hold-down timers. Two values of hold-down timer are used, 60 and 200 seconds. The average request arrival rate in these experiments is about one requests every 20 seconds.

As expected, routing performance drops as a result of both increasing update triggering threshold and hold-down timer, but the effect of the hold-down timer is more pronounced. For large values of the hold-down timer, the setting of the update triggering threshold does not impact routing performance significantly, while for 0 hold-down timer, there is a substantial drop in routing performance with increasing update triggering threshold. For all values of hold-down timer, on-demand path computation performs better than pre-computation, although the differences decreases as the hold-down timer value increases. The path pre-computation periods have been selected to be the same as the values of the hold-down timer, i.e., 60 and 200 seconds.

Overall, reducing the sensitivity of the update triggering policy or introducing a hold-down timer can affect routing performance significantly, as shown by our experiments with the `isp` topology. Although there is considerable loss of routing performance when routing information becomes inaccurate, still, the benefits of QoS routing when compared to static routing are retained to a large degree. In general, the effects of the hold-down timer setting are more pronounced that those of reducing the update triggering threshold, When there is no hold-down timer, a larger update triggering threshold results in only a moderate loss in routing performance.

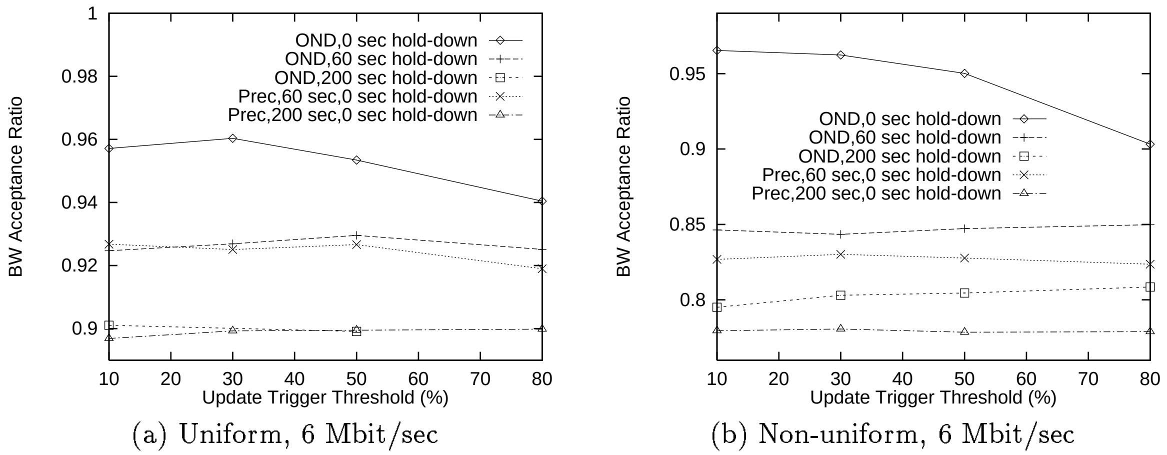


Figure 12: Comparison of on-demand and pre-computation

7 Conclusions

In this paper, we attempted to provide some insights into the issue of QoS routing cost, its different components, and how it is affected by various parameter settings. The focus was on a link state based intra-domain QoS enabled routing protocol, as this is likely to represent the first setting in which QoS routing is deployed. An important conclusion of the study is that the processing of the increased update traffic that QoS routing generates, is the dominant contributor to the cost of QoS routing. If left unchecked, it has the potential to severely limit the scalability of QoS routing, and therefore the range of environments where it would prove useful.

Fortunately, a number of techniques, e.g., coarser update thresholds and hold-down timers, are available to effectively control the volume of update traffic, and therefore its processing cost. The paper explored their effect on both cost reduction and performance. The latter aspect is of significance as it determines the trade-off involved when controlling update traffic. Further cost reduction can be achieved through the use of path pre-computation, and the paper also explored the efficiency of such approaches. In particular, it illustrated that pre-computing paths incur a small drop in performance when compared to on-demand routing, as soon as some cost control mechanisms (e.g., coarse threshold or large hold-down timer) are in effect.

Overall, the paper found that the cost of QoS routing was reasonable and, for the representative set of configurations that were tested, within the capabilities of a moderately powerful processor. This held even for fairly large networks. Thus, the findings of this paper seem to indicate that with the help of cost control techniques such as hold-down timers and path pre-computation, it is possible to achieve effective QoS routing for large networks.

References

[1] Q. Ma and P. Steenkiste, “On Path Selection for Traffic with Bandwidth Guarantees.” In Proceedings IEEE International Conference on Network Protocols, Atlanta, Georgia, October 1997.

[2] S. Chen, and K. Nahrsted, "An Overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions", IEEE Network, November/December 1998, pp. 64-79

[3] The RSVP project, `http://www.isi.edu/div7/rsvp/`.

[4] R. Guérin, A. Orda, and D. Williams, "QoS Routing Mechanisms and OSPF Extensions." In Proceedings GLOBECOM'97, Phoenix, AZ, November, 1997.

[5] J. Moy, OSPF Version 2, Internet Request for Comments, RFC 2178, July 1997.

[6] Qingming Ma and Peter Steenkiste, "Supporting Dynamic Inter-Class Resource Sharing: A Multi-Class QoS Routing Algorithm". Proceedings of IEEE INFOCOM '99, New York, NY, March 1999.

[7] G. Apostolopoulos, R. Guérin, S. Kamat, and S. K. Tripathi, "Improving QoS Routing Performance Under Inaccurate Link State Information." Proceedings of ITC'16, Edinburgh, Scotland, June, 1999.

[8] A. Shaikh, J. Rexford, and K. Shin, Dynamics of quality-of-service routing with inaccurate link-state information, University of Michigan Technical Report CSE-TR-350-97, November 1997

[9] G. Apostolopoulos, R. Guérin, S. Kamat, and S. K. Tripathi, "On Reducing the Processing Cost of On-Demand QoS Path Computation.", Journal of High Speed Networks, Vol. 7, no. 2, pp. 77-98

[10] G. Apostolopoulos, R. Guérin, and S. Kamat, "Implementation and Performance Measurements of QoS Routing Extensions to OSPF". Proceedings of IEEE INFOCOM'99, New York, NY, March 1999.

[11] E. C. Rosen, A. Viswanathan, and R. Callon, "Multiprotocol Label Switching Architecture", *draft-ietf-mpls-arch-04.txt*, Internet Draft, February 1999, work in progress

[12] M. Peyravian and A. D. Kshemkalyani, Network Path Caching: Issues, Algorithms and a Simulation Study, Computer Communications, vol. 20, pages 605-614, 1997.

[13] "Interim Inter-Switch Signaling Protocol Version 1," ATM Forum, *af-pnni-0055.000*, March 1996.

[14] G. Malkin, "RIP Version 2", Internet RFC 2453, November 1998

[15] V. P. Kompella, J. C. Pasquale, and G. C. Polyzos, "Multicast Routing for Multimedia Communication", IEEE/ACM Transactions on Networking, June 1993.

[16] J. Moy, "Multicast Extensions to OSPF", Internet Request of Comments, RFC 1584, March 1994.

[17] L. Kou, G. Markowski, and L. Berman, "A Fast Algorithm for Steiner Tree", Acta Informatica 15, 1981, pp. 141-145.

[18] H. Takahasi, and T. Matsuyama, "An Approximate Solution for Steiner Tree Problem in Graphs", Mathematica Japonica, 1980.

[19] V. P. Kompella, J. C. Pasquale and G. C. Polyzos, "Two Distributed Algorithms for Multicast Multimedia Communication", ICCCN'93, San Diego, CA, June 1993, pp. 343-349.

[20] Q. Zhu, M. Parsa, J.J. Garcia-Luna-Aceves, "A Source-Based Algorithm for Delay-Constrain Minimum Cost Multicasting", IEEE INFOCOM'95, Boston, MA, April 1995.

[21] R. Widyono, "The Design and Evaluation of Routing Algorithms for Real Time Channels", ICSI TR-94-024, University of California at Berkley, International Computer Science Institute, June 1994.

[22] Q. Sun, and H. Langerdorfer, "A New Distributed Routing Algorithm with End-to-End Delay Guarantee". Proceedings of 2nd Workshop on Protocols for Multimedia Systems, October 1995, pp. 452-458.

[23] G. N. Rouskas, and I. Baldine, "Multicast Routing with End-to-End Delay and Delay Variation Guarantees", IEEE JSAC, vol. 15, Apr. 1997, pp. 346-356.

[24] S. Chen, and K. Nahrstedt, "Distributed Quality of Service Routing in High Speed Networks Based on Selective Probing". Proceedings of LCN'98.

[25] H. F. Salama, D. S. Reeves, and Y. Viniotis, "A Distributed Algorithm for Delay-Constrained Unicast Routing", Proceedings of IEEE INFOCOM'97, Kobe, Japan, April 1997.

[26] Z. Wang, and J. Crowcroft, "QoS Routing for Supporting Resource Reservation", IEEE JSAC, September 1996.

[27] S. Chen, and K. Nahrstedt, "On Finding Multi-Constrained Paths". Proceedings of IEEE ICC'98, June 1998.

[28] I. Cidon, R. Rom, and Y. Shavitt, "Multi-Path Routing Combined with Resource Reservation". Proceedings of IEEE INFOCOM'97, Kobe, Japan, April 1997, pp. 92-100.

[29] K. G. Shin, and C.-C. Chu, "A Distributed Route Selection Scheme for Establishing a Real-Time Channel". Proceedings of 6th IFIP International Conference on High Performance Networking, September 1995, pp. 319-329.

[30] S. Chen, and K. Nahrsted, "Distributed QoS Routing with Imprecise State Information". Proceedings of ICCCN'98, October 1998.

[31] G. Apostolopoulos, R. Guérin, S. Kamat, and S. K. Tripathi, "QoS Routing: A Performance Perspective." Proceedings of SIGCOMM'98, Vancouver, Ontario, Canada, September 1998.

[32] D. Awduche et al. , "Extensions to RSVP for LSP Tunnels", *draft-ietf-mpls-rsvp-lsp-tunnel-02.txt*, Internet Draft, work in progress, March 1999.

[33] Jamussi et al, "Constraint-Based LSP Setup using LDP", *draft-ietf-mpls-cr-ldp-01.txt*, Internet Draft, work in progress, February 1999.

[34] C. Villamizar, "OSPF Optimized Multipath (OSPF-OMP)", *draft-ietf-ospf-omp-02* , Internet draft, work in progress, February 1999.

[35] C. Villamizar, "IS-IS Optimized Multipath (ISIS-OMP)", *draft-ietf-isis-omp-02* , Internet Draft, work in progress, February 1999.

[36] P. Pan, and H. Schulzrinne, "YESSIR: A Simple Reservation Mechanism for the Internet". Proceedings of NOSSDAV'98, Cambridge, UK, June 1998.

[37] D. Awduche et al., "Requirements for Traffic Engineering Over MPLS", *draft-ietf-mpls-traffic-eng-00.txt*, Internet Draft, work in progress, October 1998.

[38] A. Shaikh, J. Rexford, and K. G. Shin "Load-Sensitive Routing of Long-Lived IP Flows. To appear in Proceedings of SIGCOMM'99, Boston, MA, September 1999. (Available from `http://www.acm.org/sigcomm/sigcomm99`).

[39] T. V. Lakshman and D. Stiliadis, "High speed policy-based packet forwarding using efficient multi-dimensional range matching." Proceedings of SIGCOMM'98, Vancouver, Ontario, CANADA, September 1998.

[40] W. Almesberger, J.-Y. Le Boudec, and T. Ferrari. "Scalable resource reservation for the Internet". Proceedings of PROMS-MmNet'97, Santiago, Chili, November 1997.

[41] P. Pan, H. Schulzrinne, and R. Guérin. "Staged Refresh Timers for RSVP," *draft-pan-rsvp-timer-00.txt*, Internet Draft, work in progress, November 1997.

[42] L. Berger, D.-H. Gan, and G. Swallow. "RSVP Refresh Reduction Extensions," *draft-berger-rsvp-refresh-reduct-01.txt*, Internet Draft, work in progress, April 1999.

[43] Lan Wang, A. Terzis, and L. Zhang. "A Proposal for reducing RSVP Refresh Overhead using State Compression," *draft-wang-rsvp-state-compression-00.txt*, Internet Draft, work in progress, April 1999.

[44] "Aggregation of RSVP for IPv4 and IPv6 Reservations," *draft-baker-rsvp-aggregation-01.txt*, Internet Draft, work in progress, June 1999.

[45] R. Guerin, S. Blake, and S. Herzog. "Aggregating RSVP Based QoS Requests," (*draft-guerin-aggreg-rsvp-00.txt), Internet Draft, work in progress, November 1997. (Available from* `http://pender.ee.upenn.edu/~guerin/publications/draft-guerin-aggreg-rsvp-00.txt`).