International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:1, No:2, 2007

Evaluating Performance of Quality-of-Service Routing in Large Networks

V. Narasimha Raghavan, M. Venkatesh, T. Peer Meera Labbai, and Praveen Dwarakanath Prabhu

***Abstract*—**The performance and complexity of QoS routing depends on the complex interaction between a large set of parameters. This paper investigated the scaling properties of source-directed link-state routing in large core networks. The simulation results show that the routing algorithm, network topology, and link cost function each have a significant impact on the probability of successfully routing new connections. The experiments confirm and extend the findings of other studies, and also lend new insight designing efficient quality-of-service routing policies in large networks.

***Keywords*—**QoS, Link-State Routing, Dijkstra, Path Selection, Path Computation.

I. Introduction

THE performance and implementation trade-offs of QoS routing depend on the interaction between a large and complex set of parameters. For example, the underlying network topology not only dictates the number of candidate paths between each pair of nodes, but also affects the overheads for computing routes and distributing link-state information. The effects of inaccurate link state information depend on the amount of bandwidth requested by new flows. In this paper, the performance issues are investigated through a systematic study of the scaling characteristics of QoS routing in large backbone networks. Also, a detailed model of QoS routing that parameterizes the path selection algorithm, and link-cost function, based on the proposed QoS extensions to OSPF, as well as the results of previous performance studies is constructed. The model permits a realistic evaluation of large backbone networks and the routing of the longer-lived traffic flows that are likely to employ QoS routing. Finally, new routing algorithms have been proposed that reduce computation and memory overheads by basing path selection on a combination of few QoS parameters, dealt in detail below. Finally, the paper concludes with a list of guidelines for designing efficient quality-of-service routing policies in large backbone networks.

II. Link Cost Computation

For better QoS, the ability to specify "path metrics" or "link-cost metrics" that permit routers to compute QoS paths to destinations and forward packets incrementally is needed. Link state routing is a lowest cost algorithm. Cost, when said simply is a weighted value based on a variety of factors such as a) security levels b) traffic or c) state of the link. It is just used for weighting and should not be confused with transmission fees paid. There are various methods to calculate link costs. A simple method is to assign a cost of 1 to all links. It neither distinguishes between links on a latency basis nor on capacity basis or on current load. Second method is to use queue length as routing metric. Here the disadvantage is that either bandwidth or delay is not taken into consideration. Third one assigns weight to each link derived from average delay experienced by packets recently sent over that link. Here the disadvantage is congested link advertise high cost, traffic moves off, leaving it idle. Then it advertises low cost, attracting back traffic. (i.e., instability) Another method considers utilization. But in this work utilized bandwidth is considered for link cost computation. The formula mentioned above is used to calculate cost values for each link. Since link costs are based on utilized bandwidth, which is QoS parameter, better QoS paths will be selected. Further, routing takes into consideration the resource availability. Changes in resources are quickly reflected. The cost function mentioned above involves two link cost parameters. Those are briefly discussed below.

A. Path Cost Determination

It is hoped that the integrated services Internet architecture would allow providers to charge for IP flows based on their QoS requirements. A QoS-based routing architecture can aid in distributing information on expected costs of routing flows to various destinations via different domains. Clearly, from a provider's point of view, there is a cost incurred in guaranteeing QoS to flows. This cost could be a function of several parameters, some related to flow parameters, others based on policy, and similarly here it uses utilized bandwidth. From a user's point of view, the consequence of requesting a particular QoS for a flow is the cost incurred, and hence the selection of providers may be based on cost. A routing scheme can aid a provider in distributing the costs in routing to various destinations, as a function of several parameters, to other providers or to end users. .

B. Link Cost Parameters

Since in the worst case all users can compete for the same link at the same time, a necessary condition is that the link cost to exceed the total number of tokens in the system when the link utilization approaches unity. Among many possible cost function with this property $c(t) = a / (1 - u(t))$ *is one*, where a is the fixed cost associating with using the link, and $u(t)$ is the link utilization at time t. But this is very sensitive when utilization reaches unity. Connection Blocking Probability is an important QoS measurement. The connection blocking probability is defined as the probability that there are

not enough resources between inlet and outlet of the switch to assure the quality of all existing as well as new connection. An optimal algorithm will maximize network throughput, so minimizing the network connection blocking probability. Fine-grain cost metrics are much less useful, and can even degrade performance, in the presence of stale link-state information. With a careful selection of the exponent α, the path-selection algorithm can reduce the number of cost levels *C* without increasing the blocking probability. Smaller values of *C* reduce the space and time complexity of the route computation, allowing the QoS-routing algorithm to scale to larger network configurations.

C. Number of Cost Levels (C)

The experiments evaluate a link-cost function with a large number of cost levels, limited only by machine precision. With such fine-grain cost information, the path selection algorithm can effectively differentiate between links to locate the "cheapest" shortest-path route. Figures evaluate the routing algorithm over a range of cost granularity and link utilization. To isolate the effects of the cost function, the routing algorithm does not attempt to prune (seemingly) infeasible links before invoking the shortest-path computation in this experiment. The *C* cost levels are distributed throughout the range of link utilizations by setting $u\min = 0$. Compared to the high blocking probability for static routing ($C = 1$), larger values of *C* tend to decrease the blocking rate. Fine-grain cost metrics are less useful, however, when link state information is stale. For example, having more than four cost levels does not improve performance. Although fine-grain cost metrics help the routing algorithm distinguish between links, larger values of *C* also limit the number of links that the routing algorithm considers which can cause route flapping. In fact, under stale information, small values of *C* can sometimes outperform large values of *C*, but this crossover only occurs once the update period has grown so large that QoS routing has a higher blocking probability than static routing. The appropriate number of cost levels depends on the update period and the connection-bandwidth requirements, as well as the overheads for route computation. Larger values of *C* increase the complexity of the Dijkstra shortest-path computation without offering significant reductions in the connection blocking probability.

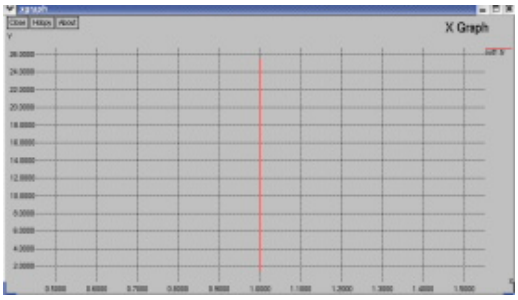

Fig. 1 Cost vs. Link Utilization for α = 0 (static routing)

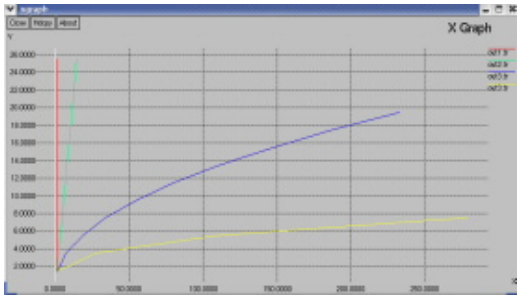

Fig. 2 Cost vs. Link Utilization for α values 0,1,2,3

D. Link-Cost Exponent (α)

To maximize the utility of coarse-grain load information, the cost function should assign each cost level to a critical range of link utilizations. Under fine-grain link costs (large *C*), the exponent α does not have a significant impact on performance; values of $\alpha \geq 1$ have nearly identical performance. Other experiments (not shown) confirm that these results hold across a range of link state update periods, from very frequent updates to a period equal to 40 times the mean connection inter arrival time. This implies that large values of α do not introduce much extra route flapping. This also has important implications for path selection algorithms, since it suggests that widest shortest-path and cheapest shortest-path should have similar performance under stale link-state information. However, the choice of exponent α plays a more important role in cost-based routing with coarse-grain link costs. When α is too large, the link-cost function concentrates most of the cost information in a very small, high-load region. For large α and small *C*, some of the cost intervals are so narrow that the arrival or departure of a single connection could change the link cost by one or more levels. For example, when $\alpha = 8$ and $C = 10$, the link-cost function has four cost levels in the 90–100% range. This sensitivity exacerbates route flapping and also limits the routing algorithm's ability to differentiate between links with lower utilization. The selection of α is actually more sensitive when the QoS-routing algorithm has accurate knowledge of link state.

III. QoS Path Selection

"Quality of Service" is a set of service requirements to be met by the network while transporting a flow. It is usually taken to mean that the network gives the user some kind of performance-related guarantee. Examples of QoS include guarantees on n/w delay and guarantees on throughput.

A. Metrics

The process of selecting a path that can satisfy the QoS requirements of a new flow relies on both the knowledge of the flow's requirements and characteristics, and information about the availability of resources in the network. In addition, for purposes of efficiency, it is also important for the algorithm to account for the amount of resources the network has to allocate to support a new flow. In general, the network

prefers to select the cheapest path among all paths suitable for a new flow, and it may even decide not to accept a new flow for which a feasible path exists, if the cost of the path is deemed too high. Accounting for these aspects involves several metrics on which the path selection process is based. They include:

- **Link available bandwidth**: As mentioned earlier, we currently assume that most QoS requirements are derivable from a rate-related quantity, termed "bandwidth." We further assume that associated with each link is a maximal bandwidth value, e.g. the link physical bandwidth or some fraction thereof that has been set aside for QoS flows. Since for a link to be capable of accepting a new flow with given bandwidth requirements, at least that much bandwidth must be still available on the link, the relevant link metric is, therefore, the (current) amount of available (i.e., unallocated) bandwidth. Changes in this metric need to be advertised as part of extended LSAs, so that accurate information is available to the path selection algorithm.
- **Link propagation delay**: This quantity is meant to identify high latency links, e.g., satellite links, which may be unsuitable for real-time requests. This quantity also needs to be advertised as part of extended LSAs, although timely dissemination of this information is not critical as this parameter is unlikely to change (significantly) over time.
- **Hop-count**: This quantity is used as a measure of the path cost to the network. A path with a smaller number of hops (that can support a requested connection) is typically preferable, since it consumes fewer network resources. As a result, the path selection algorithm will attempt to find the minimum hop path capable of satisfying the requirements of a given request. Contrary to bandwidth and propagation delay, hop count is a metric that does not affect LSAs, and it is only used implicitly as part of the path selection algorithm.

B. Metric Selection and Representation

There are some considerations in defining suitable link and node metrics. First, the metrics must represent the basic network properties of interest. Such metrics include residual bandwidth, delay and jitter. Since the flow QoS requirements have to be mapped onto path metrics, the metrics define the types of QoS guarantees the network can support. Alternatively, QoS-based routing cannot support QoS requirements that cannot be meaningfully mapped onto a reasonable combination of path metrics. Second, path computation based on a metric or a combination of metrics must not be too complex as to render them impractical. In this regard, it is worthwhile to note that path computation based on certain combinations of metrics (e.g., delay and jitter) is theoretically hard. Thus, the allowable combinations of metrics must be determined while taking into account the complexity of computing paths based on these metrics and the QoS needs of flows. A common strategy to allow flexible combinations of metrics while at the same time reduce the path computation complexity is to utilize "sequential filtering". Under this approach, a combination of metrics is ordered in some fashion, reflecting the importance of different metrics (e.g., cost followed by delay, etc.). Paths based on the primary metric are computed first (using a simple algorithm, e.g., shortest path) and a subset of them are eliminated based on the secondary metric and so forth until a single path is found. This is an approximation technique and it trades off global optimality for path computation simplicity (The filtering technique may be simpler, depending on the set of metrics used. For example, with bandwidth and cost as metrics, it is possible to first eliminate the set of links that do not have the requested bandwidth and then compute the least cost path using the remaining links.)

Now, once suitable link and node metrics are defined, a uniform representation of them is required across independent domains - employing possibly different routing schemes - in order to derive path metrics consistently (path metrics are obtained by the composition of link and node metrics). Encoding of the maximum, minimum, range, and granularity of the metrics are needed. Also, the definitions of comparison and accumulation operators are required. In addition, suitable triggers must be defined for indicating a significant change from a minor change. The former will cause a routing update to be generated. The stability of the QoS routes would depend on the ability to control the generation of updates. With interdomain routing, it is essential to obtain a fairly stable view of the interconnection among the ASs.

C. Metric Hierarchy

A hierarchy can be defined among various classes of service based on the degree to which traffic from one class can potentially degrade service of traffic from lower classes that traverse the same link. In this hierarchy, guaranteed constant bit rate traffic is at the top and "best-effort" datagram traffic at the bottom. Classes providing service higher in the hierarchy impact classes providing service in lower levels. The same situation is not true in the other direction. For example, a datagram flow cannot affect a real-time service. Thus, it may be necessary to distribute and update different metrics for each type of service in the worst case. But, several advantages result by identifying a single default metric. For example, one could derive a single metric combining the availability of datagram and real-time service over a common substrate.

D. Path Selection

There are two major aspects to computing paths for QoS requests. The first is the actual path selection algorithm itself, i.e., which metrics and criteria it relies on. The second is when the algorithm is actually invoked.

The topology on which the algorithm is run is a directed graph where vertices consist of routers and networks (transit vertices) as well as stub networks (non-transit vertices). When computing a path, stub networks are added as a post-processing step. The optimization criteria used by the path selection are reflected in the costs associated with each

interface in the topology and how those costs are accounted for in the algorithm itself. As mentioned before, the cost of a path is a function of its available bandwidth. As a result, each interface has associated with it a metric, which corresponds to the amount of bandwidth that remains available on this interface. This metric is combined with hop count information to provide a cost value, whose goal is to pick a path with the minimum possible number of hops among those that can support the requested bandwidth. When several such paths are available, the preference is for the path whose available bandwidth (i.e., the smallest value on any of the links in the path) is maximal. The rationale for the above rule is the following: we focus on feasible paths (as accounted by the available bandwidth metric) that consume a minimal amount of network resources (as accounted by the hop-count metric); and the rule for selecting among these paths is meant to balance load as well as maximize the likelihood that the required bandwidth is indeed available.

The standard routing algorithms are typically single objective optimizations, i.e., they may minimize the hop-count, or maximize the path bandwidth, but not both. Double objective path optimization is a more complex task, and, in general, it is an intractable problem. Nevertheless, because of the specific nature of the two objectives being optimized (bandwidth and hop count), the complexity of the above algorithm is competitive with even that of standard single-objective algorithms. Before proceeding with a more detailed description of the path selection algorithm itself, we briefly review the available options when it comes to deciding when to invoke the algorithm. The two main options are: 1) to perform on-demand computations, that is, trigger a computation for each new request, and 2) to use some form of pre-computation. The on-demand case involves no additional issues in terms of when computations should be triggered, but running the path selection algorithm for each new request can be computationally expensive. On the other hand, pre-computing paths amortizes the computational cost over multiple requests, but each computation instance is usually more expensive than in the on-demand case (paths are computed to all destinations and for all possible bandwidth requests rather than for a single destination and a given bandwidth request). Furthermore, depending on how often paths are recomputed, the accuracy of the selected paths may be lower. In this case, clearly, an important issue is when such pre-computation should take place. The two main options we consider are periodic pre-computations and pre-computations after a given (N) number of updates have been received. The former has the benefit of ensuring a strict bound on the computational load associated with pre-computations, while the latter can provide for a more responsive solution.

Path computation by itself is merely a search technique, e.g., Shortest Path First (SPF) is a search technique based on dynamic programming. The usefulness of the paths computed depends to a large extent on the metrics used in evaluating the cost of a path with respect to a flow. Each link considered by the path computation engine must be evaluated against the requirements of the flow, i.e., the cost of providing the services required by the flow must be estimated with respect to the capabilities of the link. This requires a uniform method of combining features such as delay, bandwidth, priority and other service features. Furthermore, the costs must reflect the lost opportunity of using each link after routing the flow.

E. Path Computation Algorithm

As explained in the metric selection part, combination of two metrics is good and to have reduced computational complexity, 'sequential filtering' can be used. The various combinations of metrics used here are:

- Bandwidth and delay
- Hop count and delay
- Cost and bandwidth
- Cost and delay

For example take bandwidth and delay. Here, set of possible paths is found first using bandwidth. Then a subset of it is eliminated using the second metric, delay. This process of elimination proceeds until it ends up with a single path. Also here bandwidth is the primary metric and delay forms the second metric. Similarly for hop count and delay, possible paths are sorted out in ascending order of hop count. Then path with minimum hop count is checked for, whether it satisfies bandwidth or not. If it satisfies then it is the best path. If not the process continues with the next minimum hop count path. Third one finds the minimum cost path, and then it checks for whether the minimum cost path satisfies the bandwidth requirement. Similarly for the last combination, minimum cost path is found. If two such paths exist, then path with minimum delay is selected out of them.

Fig. 3 shows how much QoS path selection stands out when compared with Dijkstra's and Floyd's path selection algorithms. It takes less time to reach each destinations compared to others, shown in blue color. Floyd's behaves better for destinations with more hops.

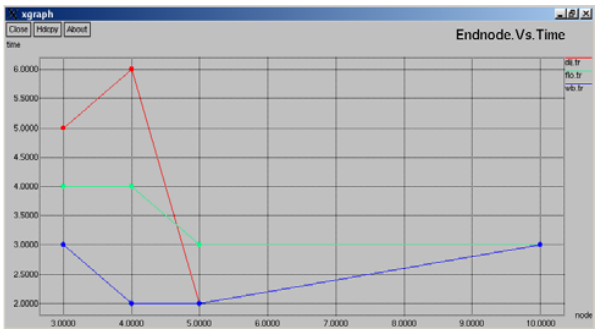

Fig. 3 QoS based Algorithm outperforms others

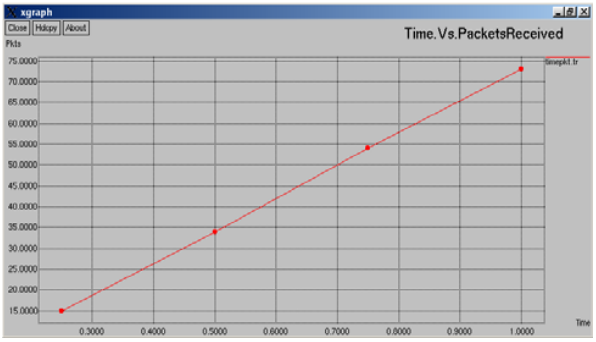

Fig. 4 Traffic Characteristics

Fig. 4 shows the characteristics of traffic generator used. As mentioned above, exponential traffic generator is used. Figure below (5) shows how link state routing behaves under various loads. Next, Fig. 6 compares how dijkstra's algorithm and QoS parameter based algorithm behaves under various loads.

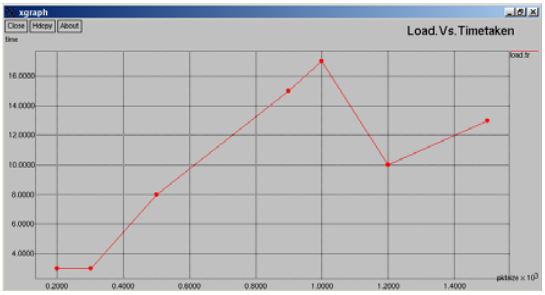

Fig. 5 Link State Routing under various loads

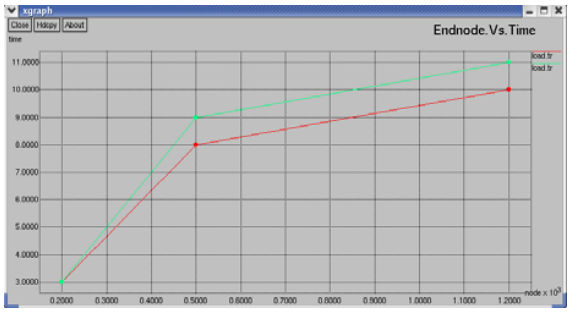

Fig. 6 Dijkstra's and QoS based algorithm under various loads

Here also QoS parameter based algorithm behaves better than its counterpart. On proving it, here further effort is laid in analyzing in depth, other possible path computations mentioned earlier. Following figure (7) shows how various QoS parameter based path computations behave under various loads.

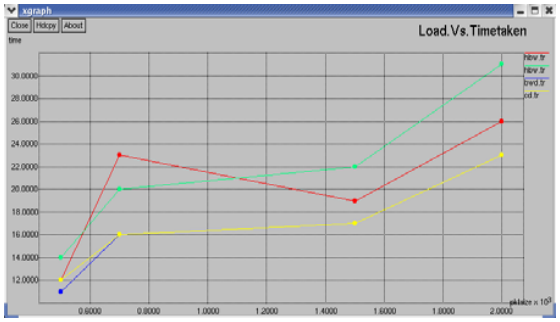

Fig. 7 QoS parameter based path computation algorithms under various loads

IV. Performance Objectives

One common objective during path computation is to improve the total network throughput. In this regard, merely routing a flow on any path that accommodates its QoS requirement is not a good strategy. In fact, this corresponds to uncontrolled alternate routing and may adversely impact performance at higher traffic loads. It is therefore necessary to consider the total resource allocation for a flow along a path, in relation to available resources, to determine whether or not the flow should be routed on the path. Such a mechanism is referred to in this document as "higher level admission control". The goal of this is to ensure that the "cost" incurred by the network in routing a flow with a given QoS is never more than the revenue gained. The routing cost in this regard may be the lost revenue in potentially blocking other flows that contend for the same resources. The formulation of the higher-level admission control strategy, with suitable administrative hooks and with fairness to all flows desiring entry to the network, is an issue. The fairness problem arises because flows with smaller reservations tend to be more successfully routed than flows with large reservations, for a given engineered capacity. To guarantee a certain level of acceptance rate for "larger" flows, without over-engineering the network, requires a fair higher-level admission control mechanism.

V. Future Work

Analysis can be done on various update policies to bring out it's impact. Further work can be done towards designing efficient Quality-of-Service routing policies.

References

[1] Apostolopoulos.G, Williams.D, Kamat.S, Guerin.R, Orda.A, and Przygienda.T,(1998) "QoS routing mechanisms and OSPF extensions." Request for Comments 2676".

[2] Apostolopoulos.G, Guerin.R, Kamat.S, and Tripathi.S, (1998) "Quality-of-service based routing: A performance perspective," in *Proceedings of ACM SIGCOMM*, (Vancouver, Canada), pp. 17–28.

[3] Cherkassky.B.V, Goldberg A.V. and Radzik.T (1996), "Shortest-path algorithms: Theory and Experimental Evaluation", *Mathematical Programming, vol*. 73, pp. 129-174.

[4] Chen.S and Nahrstedt.K, (1998) "An overview of quality of service routing for next-generation high-speed networks: Problems and solutions," *IEEE Network Magazine*, vol. 12, pp. 64–79.

[5] Cormen.T.H, Leiserson.C.E, and Rivest.R.L, (1990) *Introduction to Algorithms*. Cambridge, MA (New York): MIT Press (McGraw-Hill).
[6] Crawley.E, Nair.R, Rajagopalan.B, and Sandick.H, (1998) "A framework for QoS-based routing in the Internet." Request for Comments 2386".
[7] Floyd.S and Jacobson.V, (1994) "Synchronization of periodic routing messages," *IEEE/ACM Transactions on Networking*, vol. 2, pp. 122–136.
[8] Ma.Q and Steenkiste.P, (1997) "Quality-of-service routing for traffic with performance guarantees," in *Proc. IFIP International Workshop on Quality of Service*, (Columbia University, New York), pp. 115–126.
[9] Ma.Q and Steenkiste.P, (1997) "On path selection for traffic with bandwidth guarantees", in *Proceedings of IEEE International Conference on Network Protocols*, (Atlanta, GA).
[10] Matta.I and Shankar.A.U, (1996) "Dynamic routing of real-time virtual circuits," in *Proceedings of IEEE International Conference on Network Protocols*, (Columbus, OH), pp. 132–139.
[11] Pornavalai.C, Chakraborty.G, and Shiratori.N, (1997) "QoS based routing in integrated services packet networks," in *Proceedings of IEEE International Conference on Network Protocols*, (Atlanta, GA).
[12] Shaikh. A, Rexford.J, and Shin.K.G, (1998) "Efficient precomputation of quality-of-service routes," in *Proceedings of Workshop on Network and Operating System Support for Digital Audio and Video*, pp. 15–27.
[13] Whang.Z and Crowcroft.J, (1996) "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 1228–1234.
[14] Zhang.Z, Sanchez.C, Salkewicz.B, and Crawley.E.S, (1997) "Quality of service extensions to OSPF or quality of service path first routing (QOSPF)".