DESIGN AND IMPLEMENTATION OF A NEW ROUTING SIMULATOR

Peng Zhang, Raimo Kantola, Zhansong Ma
Laboratory of Telecommunication Technology,
Helsinki University of Technology
Otakaari 5A, Espoo, Finland
Tel: +358 9 4515454
email: {pgzhang@tct.hut.fi, raimo.kantola@tct.hut.fi, zhangsong@tct.hut.fi **}**

Discrete-event simulation, quality of service (QoS), QoS routing (QoSR), constraint based routing, performance analysis

ABSTRACT

In this paper, we present the design and implementation of a new QoS Routing Simulator (QRS) [*]. Based on the core of a public routing simulator – MaRS [**], we have developed QRS by designing and implementing new QoS-related components, i. e., resource reservation (RSVP), resource management (RM), QoS routing algorithms, traffic scheduling and real-time traffic workload. QRS allows users to configure the parameters of a QoS guaranteed network, where the dynamics of QoS routing algorithms as well as traffic management algorithms can be investigated. We also present some simulation results obtained by using QRS.

1 INTRODUCTION

Routing protocols play an important role in the wide-area networks. They are responsible for maintaining the network topology image, choosing routes for data packets and yielding good performance in terms of delay and throughput. Routing protocols in the current Internet (e.g., OSPF), usually characterize the network with a single metric such as hop-count or delay and use shortest-path algorithms for path computation. These protocols do not use alternate paths with acceptable but non-optimal cost to route traffic.

QoS routing selects routes with requirements for additional routing metrics, e.g., delay, and available bandwidth. An objective of QoS routing scheme is to aid in the efficient utilization of network resources by improving the total network throughput. Moreover, QoS routing provides flexibility in support for various service requirements by customers [Chen and Nahrstedt 1998]. QoS routing also provides support for alternate routing. If the best existing path cannot admit a new flow, the associated traffic can be forwarded in an adequate alternate path. QoS routing algorithms can prevent traffic shifting from one path to another "better" path only if the current path meets the service requirements of the existing traffic.

However, many issues related to QoS routing still remain open. The dynamics of QoS routing in the networks, that is, how QoS routing algorithms impact traffic behavior and network performance is unclear yet. Moreover, we need to study the strategy of implementing QoS routing and the potential algorithms [Zhang 1999]. In this paper, we introduce a new QoS routing simulator - QRS. QRS allows users to arbitrarily configure the network topologies and network parameters, log and save the selected parameters. QoS routing components calculate the routes demanded by the resource reservation component. In QRS, we implement such QoS-related components as resource reservation, resource management and traffic scheduling.

The rest of the paper is organized as follows. In section 2, we present the structure of QRS. We describe the design and implementation of new components in section 3. In section 4, we use QRS to examine the network performance and present some results. Some conclusions are given in the final section.

2 STRUCTURE OF QRS

QoS routing in the Internet is regarded as a missing part in the evolution of Internet. Also, both public and commercial simulators all lack the function of QoS routing. In order to reduce the programming work, we examined the existing three simulators which were available to us: BONES[Cadence 1998], NS2 [Fall and Varadhan 2000] and MaRS[Alaettinoglu et al. 1994]. BONES is a commercial package simulator suitable for universal traffic engineering and network evaluation; NS2 is a public software designed for simulating a network with traffic policy components; MaRS is specially designed for simulating routing in a best-effort network.

We found MaRS most suitable for our purposes for several reasons. First, the source code was readily available, and its style of coding and documentation is good. Thus, modifications and expansions seem easy to do. Second, since we focus on routing study, we play high value on the strong routing capability of MaRS. Third, we aim to construct the prototype of QoS routing in a QoS_based network. MaRS provides the basic event engine and the framework which seem to be suitable for us to achieve the goal.

QRS has the same structure as MaRS: a simulation engine; a user interface; and a set of components which accommodate a variety of target systems and performance measures [Alaettinoglu et al. 1994]. The simulation engine manages the event list and the user interface. When QRS is started, the simulation engine initializes variables of the user interface, processes command line options (including reading any files), and then goes into the basic simulator loop. (It should be noted that only command line interface is provided at present.)

[*] This work is supported by IPANA project carried out in Laboratory of Telecommunication Technology, Helsinki University of Technology. An extended version of this paper can be found at *http://www.tct.hut.fi/~pgzhang*.
[**] MaRS is reserved for University of Maryland.

Applications		
Resource Reservation	QoS Routing	Resource Management
Forwarding, Traffic Engineering		

Figure 1 QoS Architecture in QRS

The components are for modeling the target systems and certain simulation functions. Figure 1 shows the QoS architecture in QRS and its components. We should note that as the first step, we mainly follow the IntServ [Wroclawski 1997] model in QRS because till now only IntServ model provides an integral solution from resource reservation to flow management in a large wide-area computer network. Moreover, these functions in IntServ are also needed in DiffServ [Blake et al. 1998] network. For example, RSVP [Braden et al. 1997] is probably used as the signaling protocol between Bandwidth Brokers in DiffServ networks [Mamais et al. 1999].

Thus, to develop QRS, we added several new components and modified some components of MaRS. For modeling QoS routing protocols, we add a QoS routing component, i.e., QOSPF component. For modeling resource reservation protocol, we add a resource reservation component, i.e., RSVP component. For modeling resource management, we add RM component. For modeling a QoS oriented application, we add Realtime Traffic Source/Sink component. We have modified Node component to identify real time traffic packets and to look up a QoS_based route. We have also modified Link component to accommodate traffic scheduling, i.e., Class Based Queuing (CBQ). QRS abandons routing components of MaRS, i.e., SPF, ExBF, and SEGAL, thus adopts QOSPF as the sole routing component. QRS preserves all other components of MaRS, including Link Cost Function component, Performance Monitor component and workload components. Figure 2 shows a target system of instances of five node components and six link components.

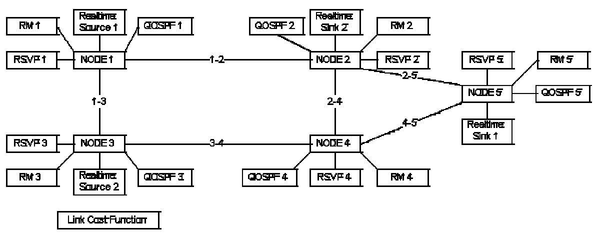

Figure 2 An example target system

Each link component is connected to exactly two node components. A node component models the physical aspects of a store-and-forward entity such as a host or a switch. Each node is connected to a QOSPF routing component, a RM component, a RSVP component and zero or more workload components. Each workload component is either a source or a sink of some types (i.e., FTP, Telnet, Simple Traffic, or Realtime Traffic). For each source component instance, there is a corresponding "peer" sink component instance. There is a single instance of a link-cost component, which is not connected to any component.

As in MaRS, components can interact with each other by calling events, by scheduling events, or in some special cases by sharing variables. Component instances interact with the simulation engine as follows. A component instance can make access component data structures to determine the displays to be updated. The simulation engine can call events of a component in respect to user interface commands (from input file in QRS).

3 DESIGN AND IMPLEMENTATION

We have several objectives in designing QRS: First of all, we aim to investigate dynamics of QoS routing in the networks; Second, we intend to develop new QoS routing algorithms and to look for the feasible ways to build up the quality guaranteed network; Third, we intend to achieve substantial knowledge in developing such a routing simulator. Especially, we want to avoid complicated policies and traffic management in QRS. Thus, we make out the general design on the basis of the above objectives and rules.

3.1 General design

As described in section 2, a QoS_based network should consist of such components as resource reservation, resource management, traffic scheduling, QoS routing, and policy control. Under IntServ model, we design and implement these components. We present the general design and interactions between components of QRS as shown in Figure 3.

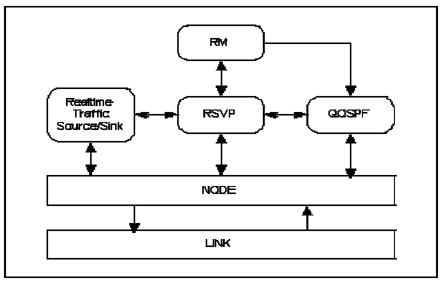

Figure 3 General design and interactions between components in QRS

Before we begin our design, we have made some pre-decisions on QRS:

First, we design and implement QoS Routing component on the basis of OSPF. So far, OSPF is the most popular routing protocol in the intra-domain environment and is supported in most routers. Therefore, it is reasonable to integrate OSPF into new QoS routing protocol [Apostolopoulos, Guerin and Kamat 1999]. Particularly, we implement QoS routing on-demand instead of pre-computation because of the considerable complexity. On the other hand, the cost of QoS routing on demand is unclear yet and needs further study. We prefer to employ QoS routing computation on the basis of request from network operator or each application with policy control, where the frequencies of request are constrained.

Second, we select QoS metrics: bandwidth and optional delay, which are possibly sufficient for most studies. RSVP broadcasts and updates these two flow metrics in the flow state table.

Third, we design and implement one specific traffic scheduling: a simplified CBQ. On one hand, we select CBQ because it is one of the most promising traffic scheduling algorithms used in future networks [Floyd and Jacobson 1995]. On the other hand, we do not implement a full set of CBQ because the full implementation would increase considerably our programming work and would decrease the simulation efficiency.

Moreover, in QRS, we specify four kinds of flows and assign them to different workload types. They are Class A, Class B, Class C and Class D. The priority decreases from Class A to Class D. Class A is the highest priority for control and singling traffic, i.e., RSVP traffic and route traffic. Class B and C are for Realtime Traffic workload, which can be assigned to an instance of Realtime Traffic workload in an input file configured by the user. Class D is the lowest priority for best effort services, i.e., FTP, Telnet and Simple Traffic workload. Traffic with higher priority will be served before traffic with a lower priority in accordance with the CBQ scheduling algorithm. There are three class levels, as shown in Figure 4.

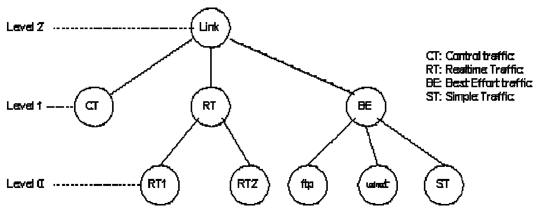

Figure 4 Class levels in QRS

Notably, there are two tables for route selection: one is the flow table for specified flows, e.g., Class B and Class C; the other is the normal routing table for best-effort services, e.g., Class D. RSVP is responsible for the operations and management of the flow table, while the route for a specified flow is obtained from querying QOSPF. QOSPF is in charge of the operation and management of the normal routing table and it broadcasts link state changes and calculates routes. However, if the connection for a specified flow is not set up, the traffic from this flow will be regarded as best effort traffic, i.e., its data packets will be forwarded to the route selected from the routing table instead of the flow table and served as the lowest priority traffic in CBQ.

3.2 Realtime Traffic Source/Sink

As discussed in the previous subsection, we assign Classes B and C to Realtime Traffic. Realtime Traffic has a QoS request for bandwidth, delay or both. It will request RSVP to set up a flow connection between source and sink. In detail, Realtime Traffic Source requests RSVP to setup a flow connection downstream, and Realtime Traffic Sink replies to reserve resource through RSVP upstream. If the flow connection is set up, the source sends data packets to the destination along the flow connection. The major functions of Realtime Traffic Source/Sink are: sending/receiving control messages through RSVP; producing data packets at Source and consuming data packets at Sink.

We prefer to use Realtime Traffic as aggregated traffic from multiple hosts/applications instead of a single host/application. Realtime Traffic produces traffic in a way similar to Simple Traffic, but without a window_based mechanism.

3.3 RSVP

We have designed and implemented a simplified RSVP version. The major function of RSVP is to setup the flow connection and update flow state periodically. To carry out the function, RSVP needs to interact with Realtime Traffic workload, QOSPF, Resource Management and Node.

We should note that we do not implement the function of local repair [Apostolopoulos, Guerin and Kamat. 1999] because it seems an additional function of RSVP and increases the processing complexity of RSVP.

3.4 QOSPF

We have developed a QoS routing component called QOSPF on the basis of SPF component of MaRS. We have designed and implemented QoS routing algorithms in QOSPF and preserved the best-effort routing of SPF.

Figure 5 shows the design of QOSPF in QRS. QOSPF consists of three core functional components, namely, (1) distribution of resource availability information; (2) topology database with resource information; (3) QoS route computation [Zhang 1999].

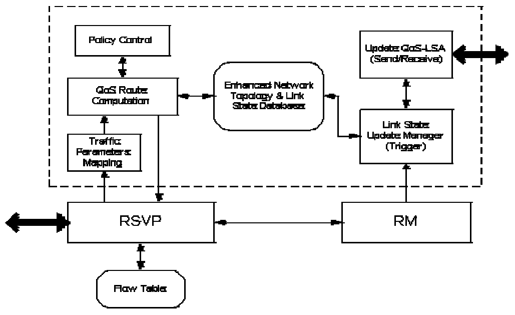
Policy Control
QoS Route Computation
Traffic Parameters Mapping
Enhanced Network Topology & Link State Databases
Update QoS-LSA (Send/Receive)
Link State Update Manager (Trigger)
RSVP
RM
Flow Table

Figure 5 Design of QOSPF in QRS

In (1), we provide two simple update algorithms. One algorithm performs update periodically. The value of the period is configured by the user. The other algorithm performs update when the variation of bandwidth exceeds a configured threshold. In the second algorithm, we also preserve the periodical update in SPF in case of deadlock. Also, link state message carries the available resource information;

In (2), we enhance local topology database by including available resource information, i.e., available bandwidth of each link.

In (3), we implement two QoS routing algorithms: the lowest cost algorithm and the widest bandwidth algorithm. Both algorithms first eliminate the link whose available bandwidth is below the required bandwidth. Then, the former calculates the lowest cost with Dijkstra’s algorithm while the latter calculates the widest bandwidth with a variation of Dijkstra’s algorithm. The first algorithm makes full use of the link cost function in SPF so that it has various derivations, e.g., least hop, lowest delay, maximum utilization, etc [Alaettinoglu et al. 1994].

3.5 Resource Management

We have designed and implemented a very simple Resource Management. At the moment, we only consider bandwidth request in RM. Its functions can be described as follows.

When a reserve request from RSVP is received, RM checks if enough bandwidth in the link is available, replies the result and informs QOSPF that available resource has changed if resource is reserved;

When a tear down message from RSVP is received, RM updates the available bandwidth and informs QOSPF that available resource has changed.

3.6 Node and Link

Since traffic loads are specified into different classes, the corresponding traffic scheduling should be supported too. In the real world, traffic scheduling should be performed at nodes. However, we implement it in the Link component instead of the Node component because of less modifications. We have also modified the event processing of Link, so that the function of traffic scheduling seems as a part of Node.

As shown in Figure 6, when a data packet arrives at the node, it gets into the Packet Classifier. If it belongs to a specified flow and its flow connection exists, Node indexes the flow table for next hop; otherwise it indexes the routing table for the next hop. Then the packet queues in the buffer to the corresponding link. According to the class the packet belongs to, the packet queues in the corresponding buffer in CBQ. And the packet is served by the CBQ mechanism and transmitted to the next hop.

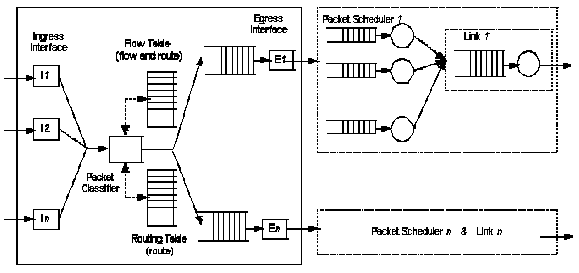

Figure 6 Traffic processing and scheduling in QRS

User can configure the parameters of CBQ, e. g., allocated bandwidth, buffer size, etc.

4 SOME RESULTS

In this subsection, we present some simulation results by using QRS to examine network performance in various network configurations. The purpose is to verify the accuracy and efficiency of QRS.

Simulation 1: Tree Topology

As shown in Figure 7, we configure the network as a tree topology with four nodes and three links. There are three pairs of workload: one pair of Realtime Traffic with Class B, called RTH; one pair of Realtime Traffic with Class C, called RTL and one pair of Simple Traffic, called BE. Traffic enters into the network through node 1 and node 4 and escapes the network from node 3. All links have the same bandwidth of 6Mbit/s. The average rates of RTH and RTL are 2Mbit/s and 3Mbit/s, respectively. The average rate of BE is 6Mbit/s. Thus, the total average rate of all workloads is larger than the link bandwidth.

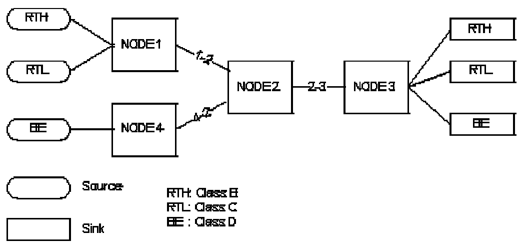

Figure 7 Tree Topology

We run simulations for a period, i.e., 10sec and log throughput of each pair of workload and total throughput. Particularly, Link 1-2 fails at a mean time 4sec and repairs at a mean time 3sec. As shown in Figure 8, the total throughput is nearly equal to the link bandwidth, that is, the link efficiency is very high. Meanwhile, RTH and RTL can obtain the bandwidth required, while BE utilizes the rest bandwidth of the link.

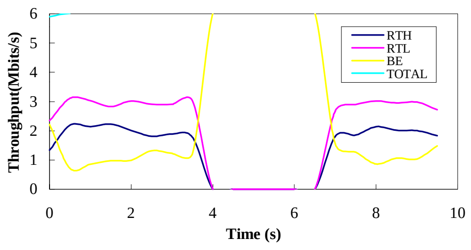

Figure 8 Throughput vs. time in Simulation 1

Simulation 2: Matrix topology

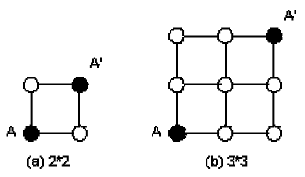

Figure 9 Matrix Topology

We use the matrix topology in Simulation 2. In this simulation, we have investigated the efficiency of QRS. As shown in Figure 9, we illustrate the two matrixes, i. e, 2*2 and 3*3. There are five pairs of workload between Node A and Node A': two pairs of RTH, two pairs of RTL and one pair of BE. The average rate of each kind of pair is same as Simulation 1. We varies the size of the matrix from 2*2 to 5*5. Then, we run the simulation for 1000msec on a Sun Ultra1, then we record the CPU running time. We also change the link state updating period in QOSPF from 10 ms to 500 ms. The results are shown in Table 1. Column 1 is the link state updating period. Row 1 is the size of the topology.

As shown in Table 1, we get the results that the running time increases with the increment of matrix size and decreases with the increment of the link state updating period. Moreover, we notice that the maximum running time is less than 4 seconds, therefore, we believe that QRS is capable of simulating a relatively large network.

Table 1 Running Time in Simulation 2

	2*2	3*3	4*4	5*5
10ms	0.53s	2.38s	2.27s	3.82s
50ms	0.42s	2.04s	1.89s	3.37s
100ms	0.38s	1.33s	1.59s	3.12s
500ms	0.31s	0.68s	0.96s	1.65s

Simulation 3: NSFNET Backbone

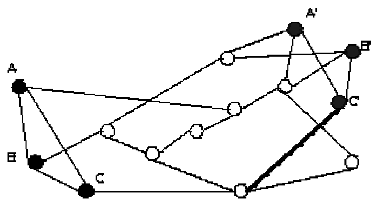

Figure 10 NSFNET Backbone

The topology used in simulation 3 is NSFNET Backbone. As shown in Figure 10, there are 14 nodes and 21 links. We configure three source nodes (A, B, C) and three sink nodes (A', B', C'). There is one Realtime Traffic workload between each source node and each sink node. There are other three Simple Traffic workload between A and A', B and B', C and C'. The average rate of each workload is set to 2Mbit/s and the link bandwidth is 6Mbit/s. The dark line shown in this figure represents the failed link in the simulation. In our configuration, each Realtime Traffic source will start up randomly. Once the previous path connection fails, each Realtime Traffic source requests for setting up a new path after a mean value of 100ms. Thus, the route for each workload might change each time. We run the simulation for 10sec and set the link failure at second 4 and link repair at second 7. Furthermore, we define the recovery time for a Realtime Traffic workload as the time between the link fails and the workload finds a new route and establishes a connection again. We have traced the route for each Realtime Traffic workload and have recorded the recovery time for the workload whose route contains the failed link.

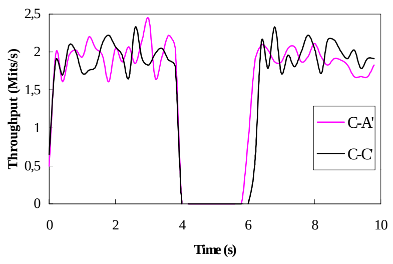

Figure 11 Link Failure in Simulation 3

As illustrated in Figure 11, two pairs of workloads (C-A' and C-C') has to change the route due to link failure. They establish the path and send packets again around second 6. We run the simulations for several times and get the mean value of the recovery time, which is nearly 2 seconds.

5 CONCLUSIONS

There are a great number of open research questions concerning routing in the next generation networks. One such open question concerns possible scenarios for the development of routing algorithms for new services into the Internet. QRS provides a simulation platform on which these questions can be examined. By using QRS, we have investigated the performance in our network configurations. The results prove the feasibility and efficiency of QRS. More simulations are needed to verify QRS in our future works. It is possible to extend QRS to accommodate many new capabilities.

References

Alaettinoglu, C., et al. 1994. "Design and Implementation of MaRS: A Routing Testbed." Journal of Interworking Research and Experience, 5, no. 1, (March): 17-41.

Apostolopoulos, G.; Guerin, R.; Kamat, S. 1999. "Implementation and Performance Measurements of QoS Routing Extensions to OSPF." Proceedings of Eighth Annual Joint Conference of IEEE Computer and Communication Societies, INFOCOM'99, (NY, March 21-25). IEEE, NY, USA, 680-688.

Blake, S., et al. 1998. "An Architecture for Differentiated Services." RFC-2475, Network Working Group, IETF. (Dec.).

Braden, R., et al. 1997. "Resource Reservation Protocol – RSVP: Version 1 Functional Specification." RFC2205, Network Working Group, IETF. (Sept.).

Cadence. 1998. "BONeS/Designer 4.0 Release Document". Internal Documents. Cadence Company. http://sourcelink.cadence.com/SL31/.

Chen, S. and Nahrstedt, K.. 1998. "An overview of Quality of Service Routing for Next-Generation High-Speed Networks: Problems and Solutions." IEEE Network, 12, no.6, (November/December): 64-79.

Fall, K. and Varadhan., K. 2000 "*ns*—Notes and Documents." http://www.isi.edu/~salehi/ns_doc/

Floyd, S.; Jacobson, V. 1995. "Link-Sharing and Resource Management Models for Packet Networks." IEEE/ACM Transactions on Networking, 3, no. 4, (August): 365-386.

Mamais, G., et al. 1999. "Efficient Buffer Management and Scheduling in a Combined IntServ and DiffServ Architecture: A Performance Study." In Proceedings of 2nd International Conference on ATM, ICATM'99, (June 21-23). 236-242.

Wroclawski, J. 1997. "The Use of RSVP with Integrated Services." RFC2210, Network Working Group, IETF. (Sept.).

Zhang, P. 1999. "Perspectives of QoS Routing in the Internet: Preliminary Study." Technical Report. Lab. of Telecom. Tech., HUT, Espoo, Finland. (Sept.).