Issues on QoS based Routing in the Integrated Services Internet

Gábor Rétvári
Department of Telecommunications and Telematics,
Budapest University of Technology and Economics (BUTE)
High Speed Networks Laboratory, QoS IT Labs
Pazmany Peter Setany 1/D, H1117, Budapest, Hungary
e-mail: retvari@ttt-atm.ttt.bme.hu

ABSTRACT

Given today's need for transmitting multimedia data over the Internet, large efforts have been made to specify and implement various service classes (Integrated Services: IntServ) over the well-known Internet Protocol (IP) infrastructure. Distinguishing service classes introduces the necessity of providing privileged treatment (QoS: Quality of Service) to a subset of data packets.

Network resources dedicated to particular IP flows must be allocated at the initial phase of a QoS communication at each network node along the forwarding path. Calculation of this forwarding path is the responsibility of a separate routing module.

While current routing protocols are not capable of considering QoS requirements when selecting paths, QoS enabled routing protocols increase the likelihood of accommodating a particular IP flow in the network according to its resource demands.

Although the components of an IntServ platform are given, attempting to establish a functional testbed fails due to numerous design and implementation reasons. A brief investigation of the issues affecting the co-existence of resource reservation and QoS routing is provided in this paper. As currently IntServ can not take advantage of QoS routing, it is restricted to utilize traditional best-effort routing functionality. Several solutions are covered in this paper to avoid this limitation.

A series of fundamental measurements are presented as well to demonstrate the efficiency of the QoS-routing enabled IntServ platform.

1. INTRODUCTION

Recently, the Internet Community realized, that given today's emerging need for multimedia communication the traditional, well-known Internet Protocol (IP) infrastructure is no longer capable of providing sufficient services. IP has proven to be very effective and robust as long as no privileged service classes are assumed, but it is completely inadequate for the introduction of QoS. Since IP is, by nature, connectionless, it is really hard to bind network resources permanently to a particular session willing to gain prioritized services. Additionally, there is a large number of various underlying lower layer mechanisms, that makes it quite difficult to adapt higher layer resource management systems. When defining new service classes in the Internet, care always has to be taken not to interfere with existing best-effort services. Furthermore, QoS communications are typically high-speed connections, thus it is worth to keep the complexity of the QoS forwarding engine rather low to facilitate rapid switching of QoS traffic. We can state, that a real connection oriented infrastructure has to be established over the connectionless, best-effort IP architecture. Currently, huge research efforts are taken to fulfill the above requirements.

The term "QoS IP network" refers herein to a network, that supports both best-effort packets and packets with QoS guarantees. The way in which network resources are split between the two classes is irrelevant, except of the assumption that each QoS capable router in the network is able to (1) dedicate some of its resources to satisfy the requirements of QoS packets, and (2) identify and advertise resources

that remain available to new QoS flows (e.g.: for Admission Control purposes).

Introducing QoS over the IP infrastructure can be viewed according to two principal approaches. One is the "Differentiated Services" model, which implies the classification of QoS flows (sequences of IP packets belonging to the same QoS session) into traffic classes to facilitate common treatment of flows with similar QoS requirements. The other approach is the "Integrated Services Internet" (IntServ) model: individual flows are handled separately. This may result in gaining finer control over QoS guarantees to be delivered for individual IP flows. However it is clear, that the cost of sophisticated resource allocations is the increased amount of state information associated with individual sessions, that have to be stored at each network node [1]. Our main goal was to establish an IntServ testbed to investigate, is increased cost worth to pay. In this document we focus on the Integrated Services Internet model.

Due to the connectionless nature of IP, a higher level signaling mechanism has to be invoked to establish signaling connection before data communication takes place. The basic component of such a signaling architecture is a resource reservation mechanism that establishes the data path for a flow at the connection initialization phase and allocates the required resources at routers along the path. Henceforward, each router along the forwarding path is capable of associating an incoming packet with a QoS session by means of packet filtering. Based on this association a QoS packet obtains prioritized treatment over best-effort data. Naturally, after the QoS communication times out, the connection has to be torn down.

The IntServ model also implies, that data forwarding paths for individual IP flows can be selected one at a time. This facilitates both the unique treatment of QoS flows from the routing perspective, and – from the traffic engineering standpoint – the introduction of some sophisticated policies to control network load with a fine granularity [2]. Each QoS capable router in the network holds proper knowledge on the actual load of the network by advertising its locally available free resources. Based on this accurate load information QoS routing module selects a suitable path for a particular QoS session according to the actual network load and the flow's traffic characteristics. Over rather meshed topologies, where multiple paths exist between a particular source and a destination, QoS routing may increase the likelihood of finding a feasible path for a QoS session, which is capable of admitting the session's traffic. Although the extra-complexity of QoS sensitive routing can be relatively high, a number of previous works presented evidence, that the increased computational cost of QoS routing is tolerable [3]. A QoS routing module is believed to be another essential component of the abstract signaling plain.

Unfortunately, despite of the fact, that QoS routing and resource reservation are deemed to be well-established mechanisms, the attempt to set up a fully functional IntServ network architecture fails due to various reasons. Principally, the inadequate design of the interface between resource reservation and QoS routing prevents the signaling connection to be established according to QoS routes. Next, our operating system environment, Linux currently lacks many crucial features that prevent us from forcing QoS traffic to QoS paths.

Although in some special scenarios a series of measurements can be accomplished to demonstrate the gains QoS routing involves owing to sophisticated routing decisions. These measurements have proven, that given today's processor capabilities QoS routing does not introduce as much complexity as it was formerly believed, however, it may drastically increase the performance provided by IntServ in a congested IP network.

In Section 2. of this document, we briefly outline, how an IntServ based QoS IP architecture is supposed to operate. In Section 3., we shall identify the crucial reasons, which prevent us from establishing an IntServ testbed with real operational system components. Also we attempt to give some solutions and outline, how a future Linux based RSVP compliant QoS forwarding engine is expected to work. Although our IntServ testbed is not fully operational, by means of manual pre-configuration some basic measurements can be carried out to demonstrate the efficiency of QoS routing. These measurements are covered in Section 4. In the last section, Section 5., we summarize the conclusions of our related work and identify the essential steps, which have to be taken in the future in order to make IntServ a really operational platform.

2. SYSTEM COMPONENTS

The introduction of QoS classes in the Internet architecture is a somewhat complicated challenge. QoS support in IP implies overall QoS support in each network layer. In this paper we focus on the mechanisms taking place in signaling plain and the network infrastructure at routers, as these are believed to be the most important building blocks of the QoS infrastructure.

2.1. Resource Reservation

As it was mentioned earlier, IP was originally invented to be connectionless. This means, that there is no easy way to associate a particular communication session with the packets constituting that session. This concept is brilliant for a network providing unreliable best-effort service. Nevertheless

the connectionless nature of IP is completely inconvenient for supporting QoS, because privileged services requires the permanent binding of some network resources to particular IP flows. Resource reservation involves the definition of the so-called session and facilitates the offering of network resources to packets associated with particular sessions. In a QoS environment the communication phase is always preceded by a signaling phase. During communication setup the applications running at Internet hosts notify the network on the characteristics of their traffic instances, and the network attempts to bind the required amount of resources to the sessions.

Managing this connection setup process is the responsibility of a separate resource reservation and setup policy. The most commonly used reservation setup protocol is RSVP (Resource reSerVation Protocol, [4]). The RSVP protocol is used by a host to request specific qualities of service from the network for particular application flows. RSVP is also used by routers to deliver QoS requests to all nodes along the path(s) of the flows and to establish and maintain state to provide the requested service. RSVP requests will generally result in resources being reserved in each node along the data path.

RSVP requests resources for simplex flows, therefore, RSVP treats a sender as being logically distinct from a receiver. RSVP operates on top of IPv4 or IPv6, occupying the place of a transport protocol in the protocol stack. However, RSVP does not transport application data but is rather an Internet control protocol, like ICMP, IGMP, or routing protocols.

In order to efficiently accommodate large groups, and to adapt to heterogeneous receiver requirements, RSVP makes receivers responsible for requesting a specific QoS. A sender describes its traffic characteristics to the receiver(s) in a Path message. Receivers respond with a Resv message, and the RSVP protocol then carries the request to all the nodes (routers and hosts) along the reverse data path to the data source(s). As a result, RSVP's reservation overhead is in general logarithmic rather than linear in the number of receivers.

Selecting an optimal path for the Path message originated at the session's sender(s) is of crucial importance, as the session's traffic will be forced to the pre-selected path. RSVP is not itself a routing protocol; RSVP is designed to operate with current and future unicast and multicast routing protocols. An RSVP process consults the local routing database(s) to obtain routes. Routing protocols determine where packets get forwarded; RSVP is only concerned with the QoS of those packets that are forwarded in accordance with routing. Dividing resource reservation and routing functions into independent modules emerges the need for a universal interface between RSVP and routing, which is able to serve the demands of any arbitrary routing protocol. As we shall see in Section 3., due to some design and implementation purposes this inter-process interface lacks some indispensable features, thus RSVP is unable to interact with QoS routing currently.

2.2. QoS based Routing

In the previous section we concluded, that selecting a proper path for an RSVP Path message at the reservation initialization phase is quite important, as this is the way a data forwarding path is assigned for the session. Each individual packet of the session must be forced to this route by the QoS forwarding engine. By selecting an optimal path for a flow, the performance of the QoS services can be increased significantly.

Suppose the situation, where there are multiple paths between a particular sender and a receiver. Best-effort routing will, regardless of the flow's resource requirements, always select the same path, which it deems to be the "cheapest" (in terms of hop count, OSPF metric, etc.). After establishing a certain number of QoS communications along the cheapest path, an immediate router may realize, that no more traffic can be admitted to this path without degrading the services delivered to the existing reservations. The cheapest path gets loaded near to its capacity, although there may exist a yet under-utilized "longer" path. Admitting further QoS traffic to this longer path obviously increases both the performance of QoS services and the overall network utilization. Therefore it seems to be remunerative to add QoS sensitivity to traditional routing mechanisms.

A good example of how to extend a conventional best-effort routing protocol with QoS capabilities is the QoSPF routing protocol (QoS extensions to OSPF, [5], [6]). Standard OSPF (Open Shortest Path First, [7]) is a unicast link-state routing protocol. This means, that each OSPF router holds knowledge on the entire topology in a routing database. Each router discovers its neighboring routers and subnetworks, and advertises its local environment to other routers in an administrative scope of the network through a reliable flooding mechanism. These advertisements are stored and updated accordingly to synchronize routing knowledge in the network. This routing architecture can be relatively easily augmented to include QoS related link metrics, namely the amount of available bandwidth at each link. Based on the collected information on the topology and the actual load state of the network, a routing table pre-computation is performed to calculate widest-shortest paths to any optional destination. The pre-computed QoS routing table comprises the widest candidate paths in increasing order of hop count (i.e., length).

There is an outgoing interface index associated with each entry in the routing table, to specify the next-hop along the path.
At QoS capable routers, upon receiving a Path message, RSVP queries QoS routing to select a feasible path from its pre-computed QoS routing table according to the flow's resource requirements (i.e.: according to the characteristics of the sender's traffic). First, QoSPF searches for an entry in the QoS routing table corresponding to the route request. Then it compares the available bandwidth of the shortest route to the destination and the requirements of the flow being processed. If the flow's bandwidth demand exceeds the available bandwidth associated with the shortest path, than a longer path is taken. If even the longest pre-computed path can not admit the flow, then an "Admission Control Error" message is returned to RSVP. Otherwise the shortest path is selected, that is capable of admitting the flow. The outgoing interface index is then returned to RSVP, which forwards the Path message in the specified direction. When the Resv message returns from the receiver on the reverse path of the Path message RSVP performs resource reservation at the proper outgoing interface. QoS routing is informed on the change of local resource availability, which triggers the re-flooding of the link state advertisements describing the link and routing information gets synchronized again. Routers, being notified on the new situation perform route pre-computation again to find optimal paths according to the actual load of the network.
The operation of the signaling protocols is depicted in Figure 1.

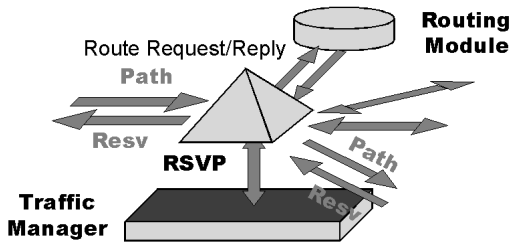

Figure 1.: Co-operation of RSVP and QoS Routing

Currently QoSPF involves a plenty of restrictions to limit QoS routing's impacts on the standard OSPF infrastructure to the lowest level. Path selection is performed in a hop-by-hop fashion, support for explicit routing is not included. The scope of QoS route computation is currently limited to a single area. Inter-operability with non-QoS aware routers is not addressed. QoSPF currently provides solely unicast routing.

Several recent research results [3] have pointed out, the potential of QoS routing for improving network utilization and the service levels provided to requests with QoS guarantees. The improvement of the service received by users is the increased likelihood of finding a path that meets their QoS requirements. Conversely, the improvement to network efficiency is usually in terms of the number of flows or the amount of bandwidth carried by the network. Despite these benefits, there remains much uncertainty regarding the additional costs of QoS routing. These added costs have two major components: *computational overhead*, due to the more sophisticated and more frequent routing computations, and a *protocol overhead* caused by the need to distribute updates on the state of the network resources that are of relevance to path selection. Previous works establish strong empirical evidence that the cost of QoS routing remains well within the capabilities of modern processors. and the protocol overhead is tolerable even for large networks. In Section 4. we present some measurements to demonstrate efficiency, QoS routing may involve.

2.3. Local traffic Control

Originally, IP was invented to co-operate with any optional lower layer mechanism, in case the link layer provides a well-defined set of functions. This means, that IP can operate over various types of link layer protocols, which may (or may not) implement QoS support in different ways. On the other hand, resource reservation functionality needs a common access interface to these lower layer QoS mechanisms. RSVP defines the LLDAL (Link-layer-dependent Adaptation Layer) routines for interfacing to a "kernel" traffic control mechanism. This supports QoS over passive media such as leased lines or (today's) shared LAN media. The module implementing LLDAL functionality is herein referred to as the so-called Traffic Manager module.
The Traffic Manager serves various purposes. Apart from being responsible for adjusting higher layer QoS functions to lower layer QoS mechanisms, it is a unidirectional interface towards QoS routing to pass local resource availability in order to facilitate the flooding of loading information. Traffic Manager also maintains a repository of existing local reservations.
QoS mechanisms at lower level involve a QoS capable forwarding engine. The forwarding engine consists of a Classifier module and Packet Schedulers associated with each QoS capable interface. The Classifier is responsible for filtering incoming packets in order to select packets belonging to a known flow. These packets have to be forced to the path determined by QoS routing: the Classifier is in charge of placing the packet into the proper outgoing interface's scheduler. Packet Schedulers maintain the

queues associated with each flow and schedule packets according to priority determined by the flow's QoS.
The Traffic Manager provides uniform access to these lower layer mechanisms for the modules implemented in the signaling plain. A special Traffic Manager module for the Linux kernel was implemented by us from scratch.

3. WHY QoS ROUTING IS USELESS CURRENTLY?

After having described the basic system components of the IntServ architecture in great detail, we present the point of this paper. Namely, we are going to identify the principal reasons, which prevent us from establishing an IntServ testbed based on the existing QoS routing and resource reservation implementations. Herein we outline a crucial design objective regarding the interface between QoS routing and RSVP, and an implementation issue concerning an unimplemented feature of our operating system environment, Linux.

3.1. Insufficient QoS routing – RSVP inter-process interface

RSVP was designed to operate independently from the routing functionality implemented in the signaling plain. This facilitates the co-operation with an arbitrary routing protocol, which installs routes into the common kernel routing table (e.g.: OSPF, RIP, BGP etc.), or implement the basic functionality of the RSVP – Routing interface (typically best-effort, multicast routing protocols, such as PIM, mrouted, etc).
In this section we shall see, that nor installing QoS routes in the kernel routing table, neither utilizing current RSVP – Routing interface facilitates the proper routing of QoS flows.
The kernel routing table is a common repository of static routes computed by best-effort routing protocols or configured manually. This repository identifies a router or a network by an address/netmask pair, and associates an outgoing interface and optionally a gateway (IP address of the next hop along the path) with it. The kernel routing table must be unambiguous, which means, that only one gateway can be ordered to a particular address/netmask pair. Attempting to install QoS routes into the kernel routing table causes ambiguity, as QoS routing may identify multiple paths to a given destination. The standard kernel routing table simply does not provide means to distinguish packets based on their belonging to a particular IP flow.
The other chance is utilizing the RSVP – Routing interface defined in several recent standards. Recall, that upon receiving a Path message referring to a yet unknown QoS flow, RSVP queries QoS routing via the RSVP – Routing interface to determine the outgoing interface the Path message has to be passed via. The following specifications define the accurate operation of this interface:
1.) "RFC 2205", the RFC describing RSVP: [4, p39]:

> "The RSVP process forwards Path messages and replicates them as required by multicast sessions, using routing information it obtains from the appropriate uni-/multicast routing process. The route depends upon the session DestAddress, and for some routing protocols also upon the source (sender's IP) address. The routing information generally includes THE LIST OF ZERO OR MORE OUTGOING INTERAFCES to which the Path message is to be forwarded."

2.) "Extended RSVP - Routing Interface" [8, p5-6]. This was created specifically for the needs of QoS based routing:

> "Routing responds to a Route_Query with a Route_Reply that identifies the OUTGOING INTERFACE(S) on which the PATH message is to be forwarded and provides a list of opaque objects that should be transmitted in the outgoing PATH message."

3.) "RSRR: A Routing Interface For RSVP". Current RSVP implementation is based on RSRRv2 (Routing Support for Resource Reservations, version 2, [9, p7]). RSRR's basic aim is to interface with multicast routing protocols:

> "Multicast routes consist of an "incoming vif" (virtual interface identifier) and an "outgoing vif bitmask". Unicast routes consist of A SINGLE BIT SET IN THE "OUTGOING VIF BITMASK" to indicate the next hop; the "incoming vif" is always zero and should be ignored by the client."

After having discussed current RSVP – Routing interface specifications, we shall see, why they are inadequate to support QoS based routing. Each of these specifications (regardless of being specified for QoS routing or multicast routing purposes) states, that the information, that routing module returns to RSVP is the list of outgoing interfaces the Path message has to be forwarded via.
RSVP Path messages holds the same IP addresses as the data packets will. After QoS routing defines the outgoing interface in a ROUTE_REPLY, RSVP attempts to forward the Path message through the given interface. Hence, RSVP has insufficient information (an interface identifier and the destination IP address, which may or may not reside on the subnet attached to the specified interface) to

identify the next-hop of the Path message. The missing point is the physical interface address of the next hop along the path, as the outgoing interface information simply does not provide any means to determine the physical address of the next-hop. Some silly ARP requests are originated ("arp who-has *destination IP address* tell *routers physical address"*), and the Path message is silently discarded, since nobody responds the ARP requests. Therefore the inaccurate design of the RSVP – Routing interface prevents the establishment of the signaling connection in the communication initialization phase.
This is a serious design issue, which was not addressed in any of the above mentioned standards on the RSVP – Routing interface. Suppose the situation of having a multi-access LAN attached to the outgoing interface of the router, where the Path message should be forwarded. There is no way to determine, which is the proper physical address of the Path message, mostly, if there are multiple routers attached to the multi-access LAN. Figure 2. shows this situation.

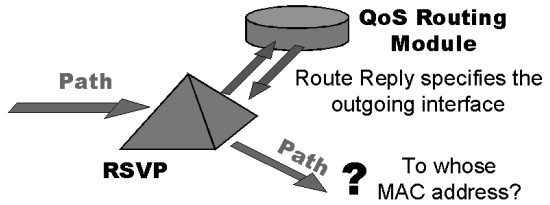

Figure 2.: Establishment of the data forwarding path

The solution is straightforward: the interface has to be completed to pass not solely the outgoing interface information, but the complete IP address of the next hop router. Furthermore, QoSPF has to be modified to store the next hop address associated with each entry in the pre-computed QoS routing table. RSVP should be updated to utilize this information in order to pass the Path message to the gateway, specified by QoS routing.

3.2. A Framework for Implementing an RSVP Classifier Module in the Linux Kernel

Nowadays Linux is the only operating system that is equipped with efficient IntServ support. There are sophisticated lower layer filtering and scheduling mechanisms implemented in recent Linux kernels. However, even Linux lacks an RSVP Classifier module that is responsible for forcing QoS packets on routes selected by QoS routing.
Currently routing is done based on a common kernel routing table. Incoming packets, regardless of their QoS or best-effort nature are forwarded in the same manner: the best match is searched in the routing table according to the destination address of the packet. Then packets are matched against RSVP filters, and inserted into the appropriate scheduler's queue. Nevertheless, in the previous section we concluded that the kernel routing table does not lend itself to install QOS routes easily, thus some modifications are required to the current routing infrastructure.
Herein we briefly outline the structure of an RSVP compliant Classifier. Although there is no such module currently available, the basic building blocks (i.e.: RSVP filtering mechanisms, Netlink Socket architecture, Netfilter) are implemented in recent kernels.
The main components of the RSVP Classifier is a packet filtering block, which filters all incoming packets by matching them against RSVP filters subsequently. A match indicates that the packet belongs to a known session, and there must be a QoS routing entry and a queue associated with it. The packet is then taken away from the standard Linux forwarding engine, and the appropriate QoS routing entry is looked up. This specifies the gateway address of the packet (next-hop's IP address) and the prioritized queue, it has to be inserted in (see Figure 3.). When a reservation is established, RSVP installs a filter and the queuing discipline for the new reservation in the kernel, and adds the proper QoS routing entry. Detailed implementation issues of the RSVP Classifier are beyond the scope of this document.

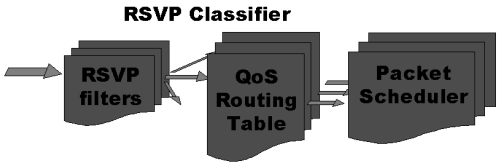

Figure 3.: RSVP Classifier in the Linux Kernel

Also it is important to note, that proper user domain applications have to be implemented to assure uniform access to lower layer QoS routing mechanisms.

4. MEASUREMENTS

The previous sections of this paper presented evidence, that RSVP's current design does not allow the co-operation with QoS routing, therefore QoS flows are restricted to best-effort routes. However, in some very special network scenarios, the above identified problems do not emerge. By means of some manual pre-configuration even the current RSVP – Routing interface can be used. Thus a series of measurements can be done to demonstrate the gains in performance and overall network utilization QoS routing may result.

4.1. Measurement Configuration

Our measurements aim to facilitate the comparison of the services a QoS flow can obtain from the network at the current state of development.

In our first measurement scenario we investigated the behavior of QoS flows in a congested best-effort network. This scenario is believed to represent the "ancient" Internet, where no prioritized treatment could be offered to a particular flow. The second scenario represents the current situation as we used RSVP without taking advantages of QoS routing. To demonstrate the benefits of QoS routing in the third scenario we used RSVP and QoSPF in the same time. Nevertheless, we have to emphasize, that these mostly unsophisticated measurements are purely for demonstration purposes.

In order to mimic a fully functional RSVP – Routing interface's functionality, some manual pre-configuration is required based on the knowledge of the paths QoSPF may select. Since there is no way to distinguish QoS packets destined to the same interface we need to send traffic directly to the proper interface of the destination. At intermediate routers appropriate *static routes* have to be added in accordance with the QoS paths, QoSPF is expected to select. Also *rp_filtering* (reverse routing for security purposes) has to be disabled to enable IP spoofing.

To facilitate the proper establishment of the signaling connection we have to avoid the discarding of Path messages with unknown next hop MAC address. A quite elegant solution is *Proxy_ARP*: a router can be told to answer any ARP request concerning a particular IP address by its own MAC address. Recall that before attempting to emit the Path message to the interface specified by QoSPF, the kernel tries to resolve the Path message's destination IP address through ARP. The next hop is set to answer these ARP requests by Proxy_ARP, thus Path messages are drawn in the right direction.

We chose the disjoint multipath network topology depicted in Figure 4. Bandwidths at network links are set in a way assuring the admission of three 10 Mbit/sec UDP flows (*QoS_short*, *QoS_longer_5_2*, *QoS_longer_6_1*) to the three feasible paths. As the scheduling architecture is quite complex, there are some inaccuracy in limiting the overall traffic to the desired rate, but it does not affect the essence of the measurements. The entire traffic travels from the *Terminal* to *Router_4*. Also we injected TCP traffic (*tcp_short*) in the network as well to investigate the service level delivered to elastic best-effort traffic in the IntServ platform. TCP traffic will always adjust its throughput to the bandwidth that remains available for best-effort communication. UDP traffic is delayed by two seconds to allow TCP to initially fill the whole bandwidth range.

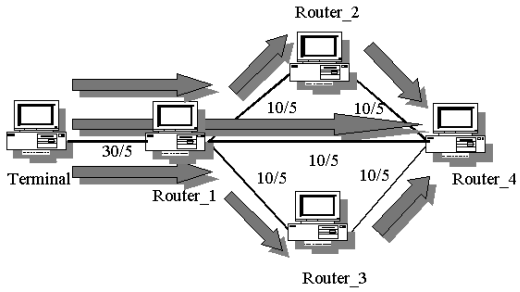

Figure 4.: Measurement configuration

4.2. Pure Best-effort Service

In the first scenario we tried to forward all the traffic over the well-known best-effort infrastructure without delivering any prioritized services to any of the sessions. The best-effort routing protocol (actually a standard OSPF) picked the same shortest path for all the traffic regardless of their QoS or best-effort nature.

Figure 5. shows the throughput of the three QoS flows and the TCP traffic in the function of elapsed time.

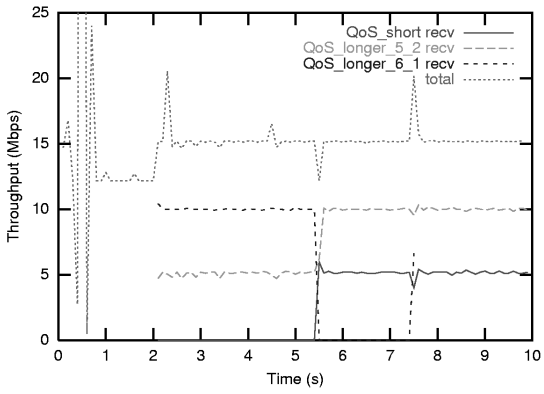

Figure 5.: Throughput in the first scenario

In this scenario, about 30 Mbit/s QoS traffic and the best-effort traffic aim to pass over the same 2 long hop path, thus, the second link of this path (*Router_1 – Router_4*) gets seriously congested. Due to the congestion each session, regardless of their required QoS obtains unpredictable and undeterministic service. There are some periods of the transmission, when the entire traffic of a QoS flow is lost.

4.3. RSVP co-operating with best-effort routing

The previous measurement aimed to represent the current Internet services. This scenario is for demonstrating the current state of development, namely RSVP, taking advantages of a best-effort routing protocol (again OSPF). Each QoS flows are limited to the same shortest path, as before. This path

can admit solely one QoS flow, therefore the QoS flow initiated first obtains real QoS (Figure 6). Other QoS flows are treated as best-effort, therefore one of them is completely lost.

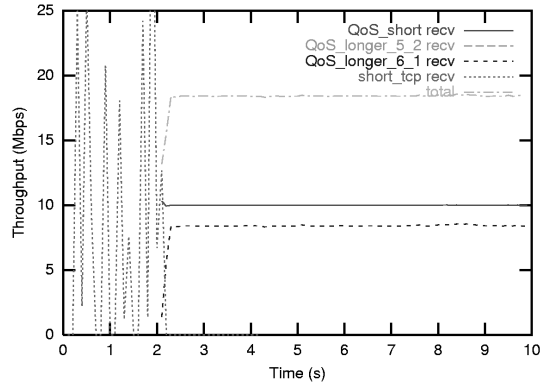

Figure 6.: Throughput in the second scenario

4.4. RSVP co-operating with QoSPF

The third scenario aims to represent the next step in the development towards a future QoS Internet. QoSPF is capable of identifying the three feasible disjoint paths for RSVP, therefore each QoS flow can be admitted in the network. 10 Mbit/sec bandwidth is allocated for each of the QoS flows along the three feasible paths, so effective QoS is offered to QoS sessions as it is depicted in Figure 7.

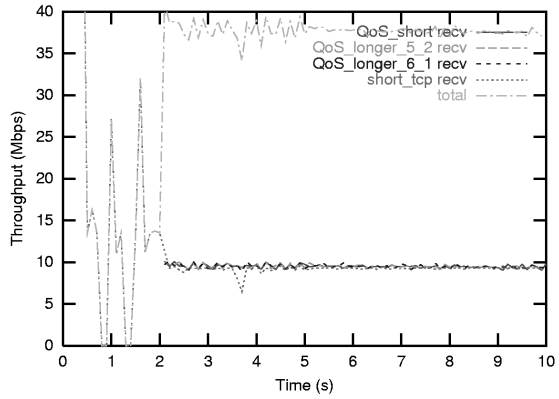

Figure 7.: Throughput in the third scenario

The gains involved by QoS routing are finally considered. First, there is about 300 percent increase in the throughput of QoS flows owing to the ability of forwarding data not only along shortest paths, but via other feasible paths as well. The total throughput also increased, this growth can be viewed as gains in overall network utilization.

Based on the above considerations, the author's view is that it is worth to utilize QoS routing over rather meshed topologies, because it can dramatically increase the performance of QoS sessions in a congested IntServ network.

5. Conclusions

This paper aims to identify the most important building blocks of an Integrated Services architecture and describe the role, these components play in delivering prioritized services for QoS sessions over the well-known IP infrastructure. Also we presented evidence, that currently QoS routing can not be used together with RSVP despite of the fact, that plenty of standards defined the co-operation. A simple solution is proposed to allow interfacing of RSVP and QoS routing. Also we pointed out, that implementing an RSVP Classifier is necessary to facilitate routing QoS packets optimally. To demonstrate the benefits of QoS routing a series measurements is presented. These measurements show, that QoS routing provides means to increase the performance of the IntServ architecture over rather meshed topologies, and therefore it is a real alternative, which is worth to consider.

REFERENCES

[1] Braden R., Clark, D., Shenker, S.: "Integrated Services in the Internet Architecture: an Overview", RFC 1633, IETF, June 1994.

[2] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick: "A Framework for QoS-based Routing in the Internet", RFC 2386, IETF, August 1998.

[3] G. Apostolopoulos, R. Guerin, S. Kamat, and S. Tripathi: "Quality of Service Based Routing: A Performance Perspective", In Proceedings of SIGCOM, pp. 17-28, Vancouver, Ontario, Canada, September 1998.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP) - Version 1 Functional Specification", RFC 2205, IETF, March 1997.

[5] G. Apostolopoulos, R. Guerin, S. Kamat, A. Orda, T. Przygienda, and D. Williams: "QoS Routing Mechanisms and OSPF Extensions", RFC 2676, IETF, August 1999.

[6] G. Apostolopoulos, R. Guerin, S. Kamat: "Design and Implementation of QoS Routing Extensions to Gated OSPF with Interface to RSVP", unpublished manuscript, December 4, 1998

[7] J. Moy: OSPF Version 2 -RFC No. 2178, IETF, July 1997.

[8] Roch Guérin, S. Kamat, E. Rosen: "Extended RSVP-Routing Interface", Internet Draft, July 1997.

[9] D. Zappala, J. Kann: "RSRR: A Routing Interface For RSVP", Internet Draft, July 1998