A Prototype Implementation for the IntServ Operation over DiffServ Networks

Werner Almesberger (*), Silvia Giordano(*),
Roberto Mameli (**), Stefano Salsano (**), Fabrizio Salvatore (**)

(*) Institute for Computer Communication and Applications (ICA)
Ecole Polytechnique Federale de Lausanne (EPFL) - http://icawww.epfl.ch/
(**) CoRiTeL – Consorzio di Ricerca sulle Telecomunicazioni - http://www.coritel.it/

Abstract

Combining Integrated Services ([1], [5]) QoS mechanisms for end-to-end signaling at hosts and Differentiated Services ([2], [10]) QoS mechanisms in the network core is one proposal for overcoming the limitation of either mechanisms when applied alone ([8]). In this paper some of the open issues in [8] are discussed and a concrete proposal is presented, which maps the IntServ Controlled Load Service into the DiffServ Expedited Forwarding "Per Hop Behavior". The proposed approach has been implemented under the Linux Operating System. The implementation allows verifying the correct functional behavior and provides a platform where more complete performance tests are possible.

I. Introduction

The integration of voice, data, and video services modified the target of networking technologies. Now, instead of simply providing best effort delivery, the network has also to address integration of services and Quality of Service (QoS) support. The applications should be therefore provided with means to describe the service they want to receive from the network.

In the Internet, two major mechanisms have been studied and developed: Integrated Services and Differentiated Services. The approach chosen by the IETF IntServ Working Group [1] is per-flow based and it relies on the RSVP protocol [4]: it offers QoS guarantees, but it also implies scalability problems. On the other side, the approach chosen by the IETF DiffServ Working Group [2] is based on flow aggregates: it allows an efficient implementation inside the network, but it doesn't provide any real service guarantees.

One logical evolution adopted by the IETF is an architecture where RSVP guarantees an end-to-end service and the network is responsible for forwarding the packets using a DiffServ mechanism (see [8]). The network core is thus seen as a single trunk, removing from it all the signaling and the state management required by RSVP.

In this paper, we present an implementation of this combined solution under the Linux operating system. We focus on the case of the Controlled Load [6] service specification for IntServ and Expedited Forwarding [12] for DiffServ. The study of other services is a matter of ongoing work.

The structure of this document is as follows. In Section 2 the rationale for the interworking of IntServ and DiffServ and the high level scenario are provided. In Section 3 we describe the main aspects of the Linux implementation of the IP traffic control and of the IntServ and the DiffServ architectures. In Section 4 we introduce our proposal of implementation of the combined IntServ-DiffServ architecture and we provide a deeper analysis of some open architectural issues. Section 5 describes the scenarios used in our trials, and the related IntServ and DiffServ services. Finally, we present our results and we discuss some open issues and future work.

This work has been partly supported by the European Commission under projects ELISA (ACTS AC310) and DIANA (AC319)

II. The interoperation scenario between IntServ and DiffServ

The "Integrated Services" architecture has been proposed in the IETF [5] as an extension of the services already available in the Internet through the definition of two service classes, the "Guaranteed Service" (GS) [7] and the "Controlled-Load Service" (CLS) [6]. In this solution, the QoS is managed on a per-flow basis; a flow is identified with a granularity that allows the distinction of single application streams. Each flow is submitted to a proper admission control and, in case of acceptance, it is entitled to receive the QoS specified by the associated service class. In particular, the GS class guarantees an assured level of bandwidth and a firm end-to-end delay bound. On the other hand, the CLS class provides only a qualitative service (no firm quantitative guarantees) and is intended for applications that can tolerate a certain amount of delay. With such a solution, the user needs the capability to signal per-flow QoS requirements to the network; this is done in the control plane by a proper reservation protocol ([3], [4]). As a consequence of the per-flow management and signaling, the IS architecture raises some serious scalability concerns.

The "Differentiated Services" architecture [10] exploits flow-aggregation and manages aggregates based on the information contained in the DS byte. Each packet belonging to a certain DS-aggregate is marked according to a PHB (Per-Hop Behavior) and treated accordingly in the DS network. Currently only two PHBs have been standardized: the "Expedited Forwarding" PHB [12] and the "Assured Forwarding" PHB [13].

The interoperation of these two approaches seems to be a promising solution to provide end-to-end QoS in a scalable way [8]. The basic idea is to use the DiffServ approach in the core network and the RSVP/IntServ in the access network. In this scenario, a key-role is played by interworking devices, called Edge Devices (ED), placed at the borders between these domains (see Figure 1).

As shown in Figure 2, in the control plane the ingress ED receives RSVP PATH messages from the RSVP sources, stores the "PATH state" and forwards the messages towards the destination. The DiffServ routers in the core simply ignore the RSVP messages and forward them transparently (i.e. without processing them). When the PATH message reaches the egress ED, that is again RSVP capable, it is interpreted and forwarded toward the final destination. Note however that the concept of ingress or egress ED depends

upon the flow's direction; an ED can be contemporarily ingress ED for a flow and egress ED for another one (in the opposite direction). The same procedure applies to RSVP RESV messages, which are received by the egress ED and sent upstream to the ingress ED. Upon the reception of the first RESV message related to a given flow the ingress ED performs a flow admission control procedure related to the DiffServ cloud. If the admission procedure is successful, the ingress ED sends back the RESV message towards the sender host.

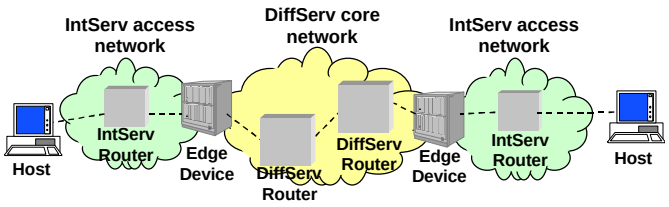

Figure 1: IntServ/DiffServ Interoperation scenario

On the other side, in the user plane, the ingress ED receives the IP packets related to a given flow, maps them into the appropriate DiffServ class, and forwards them into the DiffServ network towards the Egress ED. The routers within the DiffServ cloud route the IP packets toward the Egress ED with DiffServ-based scheduling (i.e. without changes with respect to a classical DiffServ router). Finally the egress ED behaves as a normal RSVP router and manages IP packets on a per flow basis.

In a generic scenario, the access to the DiffServ network is operated by a device called Border Router [8]. This is not the ED when the IntServ and the DiffServ networks belong to separate administrative domains. For simplicity, in the following, we consider that the ED can also play the role of the Border Router.

In Section 4 we describe how we solved two main issues in order to provide a complete solution: the mapping between IntServ services and DiffServ traffic classes and the admission control procedures in the ingress ED.

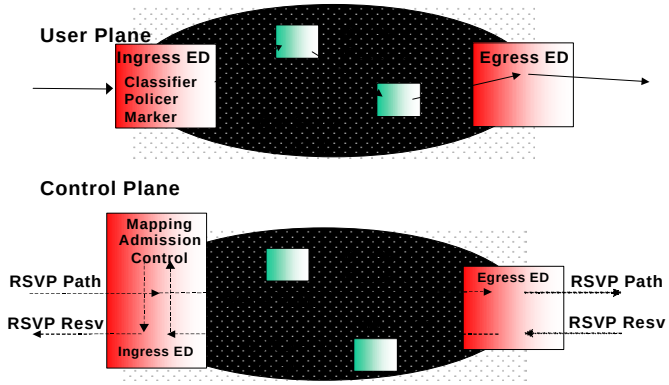

Figure 2: Support of RSVP on a DiffServ network

III. IP Traffic Control, Integrated and Differentiated Services under Linux

A *Traffic Control Mechanisms under Linux*

Recent Linux kernels offer a wide variety of traffic control functions [15]. The kernel parts for traffic control and several user-space programs to control them, implemented by Alexey Kuznetsov, cover the mechanisms required for supporting both IntServ and DiffServ.

Figure 3 shows roughly how the kernel processes data received from the network, and how it generates new data to be sent on the network.

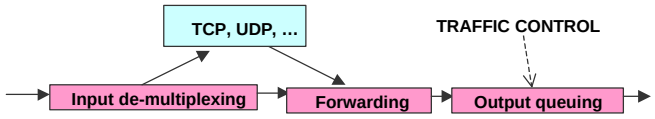

Figure 3: Processing of network data.

"Forwarding" includes the selection of the output interface, the selection of the next hop, encapsulation, etc. Once all this is done, packets are queued on the respective output interfaces. Here traffic control comes into play, e.g. deciding in which order the packets should be sent on the net, delaying or dropping some portions of traffic.

Once traffic control has released a packet for sending, the device driver picks it up and emits it on the network.

The traffic control code in the Linux kernel consists of the following major conceptual components:

- queuing disciplines
- classes (within a queuing discipline)
- filters
- policing

(For further details see [15]).

B *Integrated Services: RSVP under Linux*

The RSVP demon (RSVPd) we used is Alexey Kuznetsov's port to LINUX of the ISI code [17].

Figure 4 shows a simplified view of the functionality within RSVPd. This implementation supports:

- interfaces to application
- signaling
- functionality of reservation styles (GS, CLS) to link-layer-specific parameters
- interface to traffic control (in the kernel)

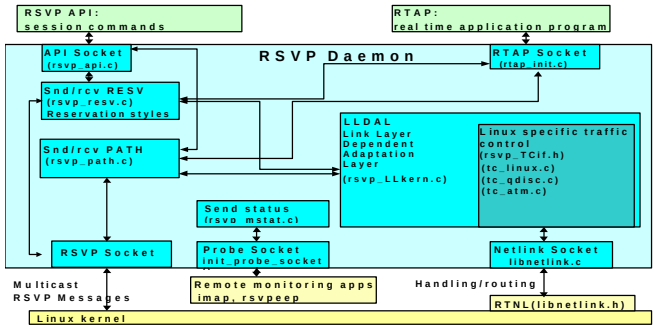

Figure 4: Block Diagram of the RSVPd

C *Differentiated Services under Linux*

The Linux implementation of DiffServ [16] supports basic classification and DS field manipulation required by DiffServ nodes. The design enables configuration of the PHBs that are being defined by the DiffServ WG. The implementation allows maximum flexibility for node configuration and for experiments with PHBs, while still maintaining a design that does not unnecessarily sacrifice performance.

Figure 5 shows the general structure of the forwarding path in a DiffServ node.

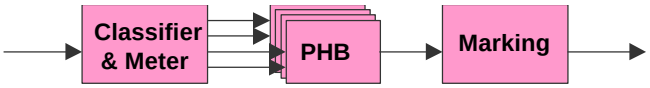

Figure 5: General DiffServ forwarding path.

The prototype implementation of DiffServ requires the addition of three new traffic control elements to the kernel: (1) the queuing discipline "sch_dsmark" to extract and to set the DSCP[1], (2) the classifier "cls_tcindex" that uses this information, and (3) the queuing discipline "sch_gred" that supports multiple drop priorities and buffer sharing. The current implementation supports:

1. Classification
2. Marking
3. Cascaded meters
4. PHBs Implementation (e.g. EF, AF, …)
5. Shaping
6. End systems functionality

(For further details see [16]).

IV. Combining Integrated and Differentiated Services under Linux

A *Mapping of IntServ flows into DiffServ classes.*

The process of interworking between the IntServ domain and the DiffServ domain can be conceptually structured in five different levels.

1) **Architectural level**: The possible mapping between the services defined in the IntServ architecture (i.e. GS and CLS) and the DiffServ traffic classes and Service Level Specifications must be defined.

2) **Network provisioning/management level:** In order to provide the required level of service the DiffServ network must be provisioned: the needed resources must be allocated (e.g. the share of link bandwidth). Hence the network devices must be configured. In particular for the IntServ/DiffServ interworking, an ED must be configured in order to support a set of the possible service mappings. The proper admission control procedures and traffic control mechanisms must be initialized.

3) **Flow admission control level**: The actual mapping decision takes place in the ED. Typically a DiffServ traffic class is selected and the availability of resources is checked.

4) **Traffic control mechanisms level:** The result of the admission control is the configuration of per-flow traffic control elements in the ED: classifiers, policers, and schedulers can be set-up as needed.

5) **Packet level:** During a flow's lifetime, the packets are classified, policed, shaped, and marked as needed.

1. Architectural issues

According to [8], the mapping at the architectural level is still an open issue.

Table 1 shows a list of possible options for mapping IntServ service types into DiffServ classes; it is not intended to be exhaustive, since many other mappings can be envisioned. Once a specific option is selected, the actual mapping of traffic parameters from the IntServ traffic specification to a specification suitable for the DiffServ domain must be defined. The specification in the DiffServ domain obviously depends on the DiffServ traffic class, Service Level Specification and admission control rules. Note also that the described scenarios are "static": the mapping depends only on the IntServ service types; more complex "dynamic" scenarios are under study. In such scenarios the mapping can depend also on "policy" and administrative information (i.e. different users can have different contractual QoS levels) and on the requested and available resources (i.e. a different DiffServ traffic class can be selected depending on the "size" of the request and/or on the load of the network).

The Guaranteed Service requires that the DiffServ network can provide a transport with bounded maximum delay and virtually no loss. The bound to the maximum delay must be known by the ingress ED and it is needed to update the advertised delay parameters in the ADSPEC object of the PATH messages. The Expedited Forwarding PHB is obviously the candidate transport mechanism to support the GS, but there are still several issues to be solved in order to implement GS with a DiffServ mechanism. In particular an end-to-end characterization of the EF PHB in a DiffServ network is needed: if the maximum delay is considered as the deterministic worst-case delay, it could easily exceed the performance target with few hops.

The Controlled Load Service seems to be more easily tractable, as there are no strict guarantees to be provided by the network. Any DiffServ transport mechanism that can provide a given throughput and reasonable low delay and jitter is suitable. Of course the EF PHB is again suitable, but in this case also the AF PHB or even a priority level in the Class Selector Compliant (CSC) PHB could suit. The difference is that for the EF some simple engineering rules enable to define end-to-end services with guaranteed throughput. For the AF or CSC options the provisioning or admission control mechanisms should make sure that the amount of reserved resources can always provide the required throughput and avoid congestion phenomena.

TABLE 1 - OPTIONS FOR THE MAPPING OF INTSERV INTO DIFFSERV

	IntServ	DiffServ
Option 1	GS	EF
	CLS	EF
Option 2	GS	EF
	CLS	AF
Option 3	GS	EF
	CLS	CSC
Option 4	GS	Not supported
	CLS	EF

[1] The value of DS field that identifies the used PHB.

In the current implementation, we considered only "static" options, and we concentrated on the CLS to EF mapping (Option 4). We based our choice only on the "qualitative" considerations we have given before. A "quantitative" comparison between the different options is out of the scope of this work. In our implementation the GS flows are not supported and an admission control error is generated at the IntServ level.

The mapping of traffic parameters is as follows. A Virtual Leased Line (VLL) with a given bandwidth, reflecting the maximum amount of EF traffic allowed, is defined between two end-points[2] as the concatenation of several hops supporting the EF PHB. We assume that the set of VLLs is statically provisioned, therefore each ingress ED is assigned a given amount of bandwidth VLL_MAX for each possible egress ED.

The ED uses the CLS *FLOWSPEC* parameters (see [4]) to compute the DiffServ "Effective EF" (EEF) rate. The admission control procedure checks if the total of EEF rates exceeds the VLL_MAX. In our implementation we use the average rate as DiffServ EEF rate. Note, however, that in order to get better link utilization other combinations of *FLOWSPEC* parameters can be used.

2. ADMISSION CONTROL ISSUES

In the static scenario proposed, the ingress ED performs a local admission control based on the statically assigned VLL_MAX. This can lead to an under-utilization of network resources.

Enhanced solutions can foresee centralized devices (e.g. an Admission Control Server) in order to perform a more dynamic network-wide resource allocation. Different levels of integration of the centralized device with the flow admission control procedure are possible. One simple option is to exchange "offline" information with the EDs in order to periodically reconfigure the DiffServ network (e.g. change the VLLs sizes in the table stored locally in each ED – see also 3).

A more complex option is to perform a truly centralized admission control procedure, where a centralized server, called Bandwidth Broker, manages the overall network resources in real time. The architecture and the protocols needed to support the remote admission control procedure are under study in [14]. This last approach raises of course scalability concerns, as examined in [9].

B Design and implementation report.

1. HANDLING OF RSVP SIGNALING IN THE RSVPD

The signaling flows are already described in Section 2. The role of the RSVPd in one direction ("ingress" ED) is to forward the received PATH into the DS network, after creating the corresponding PATH state. In the other direction ("egress" ED) to create the PATH state and forward the message to the next IntServ hop. When the egress ED receives a RESV message, the RSVPd checks the available resources on the outgoing interface (performing IntServ admission control), then it creates a reservation state and forwards the RESV to the previous IntServ hop (the ingress ED). The RSVPd also sets up the proper traffic control mechanisms (classifiers, schedulers) to enforce the reservation. Note that the behavior of the egress ED is exactly the same as in a pure IntServ scenario.

On the contrary, the RSVPd in the ingress ED recognizes when a RESV message comes from a remote ED and enforces a mapping into the appropriate DiffServ PHB with the proper admission control.

In the Linux implementation the RSVPd can differentiate its behavior depending on the interface from which it receives the RESV. The RSVPd will handle all the tasks related to the IntServ-DiffServ interworking when the RESV comes from an interface connected to the Diff-Serv network.

2. SETTING UP TRAFFIC CONTROL MECHANISMS

The actual mapping of an IntServ flow into DiffServ traffic is performed by setting up an appropriate Multi-Field (MF) classifier matching the reserved flow characteristics (retrieved from the FlowSpec in the RESV message). This classifier will filter the packets belonging to this flow into a DS class statically configured on the ED outgoing interface towards the DS domain, where they will be also marked setting the DS-field according to the PHB codepoint of the class itself.

The configuration of these classifiers is dynamic: they are created when a flow reservation is set up and automatically deleted when it is torn down.

3. INTERFACING RSVPD WITH TRAFFIC CONTROL

In the RSVPd, according to the RFC2205 [3], the RSVP/Traffic Control Interface is as independent as possible from the underlying Link-Layer technology (see Figure 6). The semantics of those functions does not involve low level management of single flows: they only specify operations in terms of high level objects contained in RSVP messages, such as the FlowSpec or the FilterSpec. The translation between requests made by RSVP through this API to the proper setting of Traffic Control objects (e.g. schedulers and classifiers) is made up by lower level Link-Layer mechanisms, taking into account the Admission Control constraints.

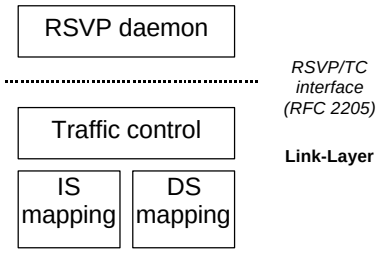

Figure 6: Interaction between RSVP daemon and Traffic Control

As previously marked, the Admission Control mechanism on the ED is based on some information about the entire DS domain, such as the VLL sizes (see 2). At the purpose the API defined in [3] should be slightly modified: the *TC_AddFlowspec* function, invoked each time a new reservation is set up, now needs the Egress ED IP address to perform the Admission Control (this information is retrieved from the *NEXTHOP* field in the RESV messages). This modify is necessary to maintain the Admission Control in the Traffic Control section at the Link-Layer level also in this

[2] In this case the ingress ED and the egress ED.

new IS/DS interoperation scenario (as already implemented in the original RSVPd for the IS scenario). Moreover, at the physical interface towards an IS domain, we maintained the original CBQ class structure at Link-Layer (see Figure 7):

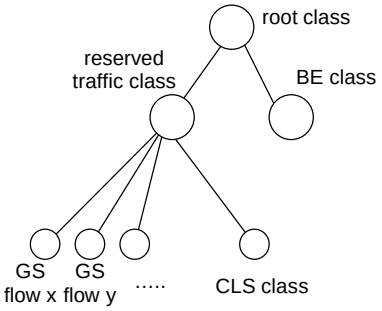

Figure 7: Traffic control CBQ classes for the IS interfaces

Whenever a new GS flow reservation request is accepted a new class is added as a leaf of the aggregate RSVP traffic class, while all CLS flows are aggregated together in a separate class.

In the current implementation we keep the CBQ scheduler on the physical interface towards the DiffServ domain to implement the DS PHBs. Moreover now we have only two CBQ classes, the higher priority class reserved for the aggregate EF traffic and the lower priority class where all the remainder traffic is collected as BE (see Figure 8). The EF class is defined rate-bounded and its rate is equal to the sum of the VLLs towards the set of different destinations (egress EDs); furthermore, its outgoing packets are marked with the DSCP 0xb8, in compliance with RFC 2598 [12]. On the other side the BE class leaves all its outgoing packets unmarked.

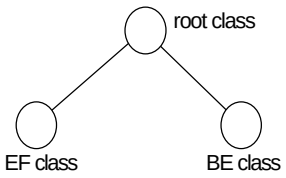

Figure 8: Traffic control CBQ classes for the DS interface

4. CONFIGURING THE EDGE DEVICE

One of the key aspects of an ED implementing interoperation between RSVP/IntServ and DiffServ is the possibility to associate each physical interface with a different configuration. The ED will have a set of "classical" IntServ interfaces on one side and a set of DiffServ interfaces on the other side. The ED must behave as ingress ED for the flows coming from one of the IntServ interfaces and as egress ED for the flows coming from the DiffServ ones. In the first case, it must manage the traffic aggregates according to the DiffServ paradigm. In the second, it must carry on the reservation on the outgoing interface according to the IntServ paradigm. This is configured at the RSVPd startup by means of the *rsvpd.conf* file. For example, in order to configure the interface eth1 as a DiffServ interface, the file rsvpd.conf must contain the line:

```
interface eth1/ip4 diffserv
```

Finally, note that an ED can be equipped with N interfaces, part of them facing the DiffServ domain and the remaining ones the IntServ domain. This implies that the ED could also act as a DiffServ router receiving and forwarding DiffServ native aggregate flows. Currently, our implementation does not address this issue: the pre-marked DS traffic would be treated as best effort in order not to affect the traffic coming from the IntServ cloud.

V. TRIALS SCENARIOS AND RESULTS

In Figure 9 the configuration of the lab trials is depicted. Basically two types of tests have been accomplished: functional tests and performance tests. The former ones have been made in order to check the proper working of the modified RSVP daemon. In particular we focused on the ED, checking the correct classification and marking of the outgoing packets towards the DS domain. At the purpose, we used the RTAP application (included in the RSVPd package, see [17]) to set up the reservations between the sender and the receiver, and the LTG tool [18] to send UDP flows matching the reservation parameters. To check the correct working of the ED we activated the Tcpdump tool [19] on the outgoing interface towards the DS region.

On the other side, the performance tests are finalized to evaluate the behavior of the kernel traffic control elements and the processing overhead introduced by these QoS mechanisms on an ED; these tests are currently ongoing.

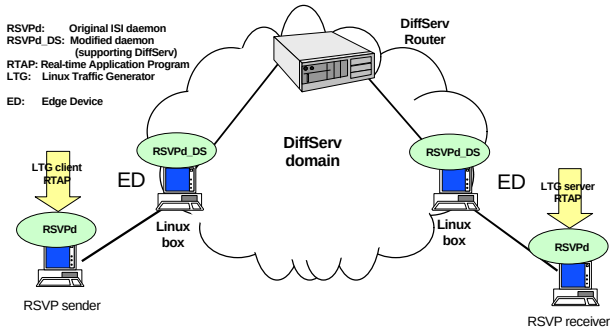

Figure 9: Lab testbed scenario

A similar testbed was used also in a demonstration at Telecom99 [20]. We used the ARMIDA[3] [21] application to transmit a video stream in the presence of disturbing traffic with and without QoS. The aim was to visually compare the same video when the traffic does not receive any QoS support from the networkand and when the traffic is mapped onto EF class and separated from the BE traffic.

VI. CONCLUSIONS

We have shown a possible implementation of an architecture that combines IntServ and DiffServ approaches

[3] ARMIDA is a MPEG-4 compliant client-server platform that supports RSVP reservation style. It provides Video Streaming feature, potentially requiring a large amount of bandwidth with strict QoS bounds to satisfy audio/video requirements. It also supports dynamic QoS renegotiation [22].

in Linux. We have also illustrated several configuration examples and the demonstration scenario we proposed at Telecom99. Future work will focus on the integration of other service classes and PHBs, as well as on the introduction of a more advanced Admission Control procedure.

ACKNOWLEDGEMENTS

The authors would like to thank

- Roberto Cocca for his support in the implementation and testing of the Edge Device.
- Piergiorgio Cremonese and Martin Potts for their collaboration in the definition and realization of the Telecom99 demonstration.

REFERENCES

[1] IETF Integrated Services Working Group, http://www.ietf.org/html.charters/IntServ-charter.html

[2] IETF Differentiated Services Working Group, http://www.ietf.org/html.charters/DiffServ-charter.html

[3] R. Braden et al., "Resource Reservation Protocol (RSVP) - Version 1 Functional Specification", IETF RFC 2205, September 1997.

[4] P.White, "RSVP and Integrated Services in the Internet: A Tutorial", IEEE Communications Magazine, May 1997.

[5] R. Braden et al., "Integrated Services in the Internet Architecture: an Overview", IETF RFC 1633, June 1994.

[6] J. Wroclawski, "Specification of the Controlled-Load Network Element Service", IETF RFC 2211, September 1997.

[7] R. Guerin et al., "Specification of Guaranteed Quality of Service", IETF RFC 2212, September 1997.

[8] Y. Bernet et al., "Integrated Services Operation Over DiffServ Networks", IETF draft draft-ietf-issll-DiffServ-rsvp-03.txt, September 1999. Available via http://www.ietf.org/html.charters/issll-charter.html

[9] S. Salsano et al., "Supporting RSVP in a Differentiated Service Domain: an Architectural Framework and a Scalability Analysis", ICC'99, Vancouver, Canada, June 1999.

[10] S. Blake et al., "An Architecture for Differentiated Services", IETF RFC 2475, December 1998.

[11] K. Nichols et al., "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", IETF RFC 2474, December 1998.

[12] V. Jacobson et al., "An Expedited Forwarding PHB", IETF RFC 2598, June 1999.

[13] F. Baker et al.,"The Assured Forwarding PHB group", IETF RFC 2597, June 1999.

[14] S. Hares et al., "A discussion of Bandwidth Broker requirements for Internet2 QBone Deployment", version 7, August 1999. Available via http://www.internet2.edu/qos/qbone/QBBAC.shtml

[15] W. Almesberger et al., "Linux Traffic Control – Implementation Overview", November 1998. Available via http://lrcwww.epfl.ch/linux-diffserv/

[16] W. Almesberger et al., "Differentiated Services on Linux", IETF draft-almesberger-wajhak-diffserv-linux-01.txt, June 1999. Available via http://icawww1.epfl.ch/linux-diffServ/

[17] University of Southern California - Information Sciences Institute (ISI), the RSVP Project. Available via http://www.isi.edu/div7/rsvp/

[18] "Linux Traffic Generator (LTG)", CoRiTeL, July 1999. Available via ftp://ftp.coritel.it/pub/ftp/ltg1_1

[19] "Tcpdump Network Tool", Network Research Group (NRG), LBL. Available via http://www-nrg.ee.lbl.gov/

[20] Telecom99, ITU, Geneva, 10-17 October 99, http://www.itu.int/telecom-wt99/index.html

[21] P. Cremonese, S. Giordano, "An example of dynamic QoS negotiation", in proceedings of the IMSA'99, Nassau.

[22] S. Giordano, J-Y. Le Boudec, "The Renegotiable Variable Bit Rate Service", in proceedings of the IFIP ATM '99, Antwerp.