Performance Analysis of IP Resource Reservation Protocols

prepared for the lecture "Computer System Performance Analysis and Benchmarking" by Prof. Thomas Stricker, Laboratory for Computer Systems, Swiss Institute of Technology (ETH), Zürich, Switzerland

Author: Károly Farkas

Institut für Technische Informatik und Kommunikationsnetze
ETH-Zentrum, Gloriastrasse 35, CH-8092 Zürich, Switzerland
Tel: +41 1 63 25447, Fax: +41 1 63 21035
E-Mail: farkas@tik.ee.ethz.ch

Abstract - this project work investigates the scalability limitations of IP resource reservation protocols using RSVP and Boomerang as examples. The memory and processing time consumption of signaling message primitives were measured as a function of the total number of concurrent reservation sessions on PC based routers running Linux. The signaling handling algorithm was also analyzed and critical operations were identified.

The results show that CPU time is a more significant scalability concern than the router memory and furthermore, that the former is very dependent on the implementation and the complexity of the signaling algorithm. Thus, for example the same Linux PC can handle almost twenty times more reservation sessions per second with Boomerang protocol than with RSVP.

15th February, 2002

Background

This project work was part of an ongoing research activity supported by Telia Reseach Sweden to elaborate a benchmarking system for measuring the performance of signaling based resource reservation protocols in Internet. Implementations of the measured protocols (i.e., RSVP and Boomerang) were given thus my contribution was to work out the benchmarking system and to make performance measurements over the above-mentioned protocols using this system.

This report is organized as follows: Section 1 contains an introduction to my work describing the measured signaling protocols. Section 2 describes the benchmarking framework by defining the investigated factors and the measurement scenario. Section 3 provides the measurements results together with considerations and finally, conclusions are drawn in Section 4.

Acknowledgements: I would like to thank the design and implementation work of Krisztián Németh, Gábor Fehér and István Cselényi and the support of Telia Research Sweden.

1 Introduction

It is not so unpopular any more to mention signaling in the context of Differentiated Services (DS) [1] as it was some months ago. Aggregation of reservation sessions, flow identifiers and signaling messages give a realistic hope in demolishing scalability limitations. However, there remain still some points in the network where micro-flows shall be differentiated from each other (e.g., border nodes) and therefore it is important to scrutinize the scalability limitations of per flow resource reservations in such network nodes.

There are several factors, which influence scalability. In this work the resource demand was investigated in signaling-aware routers in terms of memory and processing power for the RSVP [3] and Boomerang [4] IP resource reservation protocols, which have working implementation for measurements and public source code. However, the same benchmarking framework could be applied to other resource reservation protocols, such as ST-II, Yessir, Ticket or DRP. A summary of resource reservation protocols and comparative evaluation can be found in [5].

1.1 The RSVP Protocol

The Resource Reservation Protocol (RSVP) is a receiver oriented signaling protocol, which initiates soft reservation states in network nodes, which are on the path of the possible multicast datagram transfer. There are two main signaling messages defined in RSVP, both are sent end-to-end between the sender and the receiver host(s). The *Path* message is issued by the sender host(s) and forwarded hop-by-hop towards the receiver(s). *Path* sets up the address of previous hop at each node and conveys the sender's traffic specification and optionally network specific information from involved nodes. The receiver can calculate the amount of available resources and the end-to-end path characteristics using this information.

By sending a *Resv* message as a reply to the received *Path*, the receiver initiates a reservation setup. Since the previous hops are captured in each hop thanks to *Path* message, the *Resv* message travels on the same route as the *Path* even for asymmetrical routes. *Resv* specifies the amount of resources to reserve in each router en route. *Path* & *Resv* messages are also used for refreshing the soft reservation states in the routers, which are otherwise removed after a certain time.

There are three more RSVP messages. The *Path Tear* and *Resv Tear* messages are used to tear down a certain reservation state, while the *Confirmation* message is used for acknowledging the receiver that the reservation was successful.

1.2 The Boomerang Protocol

Boomerang is a lightweight, sender oriented IP resource reservation protocol. Since it is described in details in [4] just a short overview is given here.

The Boomerang protocol can be used for specifying and reserving network resources for uni- or bi-directional traffic streams between two IP nodes. Traffic stream can mean a single data flow or a flow aggregate (e.g., DS behavior aggregate or flows between subnets), where up- and downstream routes can differ. Reservation setup is made by a single message handshake (currently the ICMP – Internet Control Message Protocol – Ping message) and kept alive by periodically repeated Boomerangs refresh established soft reservation-states (see Figure 1). Unlike RSVP, the same protocol message is used for refreshing or triggering changes in a reservation state or tearing it down. The Initiating Node that is responsible for keeping track of the reservation session periodically generates signaling messages. Boomerang messages are forwarded hop-by-hop in the network up to the Far-End Node, from where they are returned back to the Initiating Node.

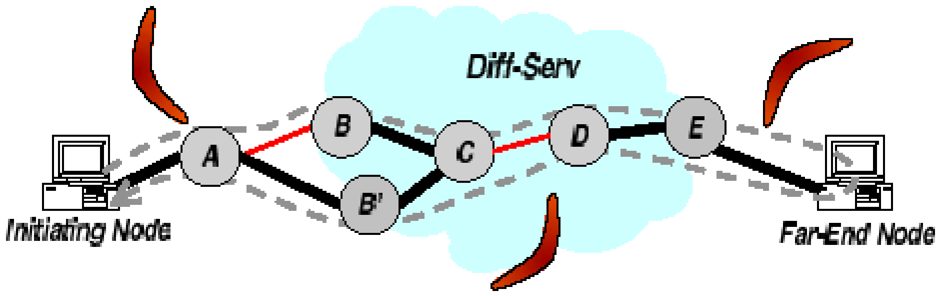


Figure 1. Reservation Setup with the Boomerang Protocol

The Initiating Node can describe the required service quality in various ways, by specifying a DSCP (DiffServ Code Point) and peak information rates for the forward and reverse directions or by providing an IntServ-like [2] object. In the prototype implementation Boomerang protocol messages are wrapped into *ICMP Echo* / *Echo Reply*-s which are supported in the majority of network nodes and hosts, giving good chance for quick penetration of the protocol in current Internet.

2 BENCHMARKING FRAMEWORK

2.1 Router Resources

The main scalability concern opposed to signaling-based resource reservation at microflow scale is that this paradigm places a load on signaling-aware routers proportionally to the number of reserved flows. This load can be quantified by measuring the *processing time* and *memory requirement* per signaling message types as a function of concurrent reservation sessions [8]. The following equations can be derived from simple heuristics:

$$t = \tau + \alpha n, \qquad n < N_t$$

where t is the message processing time, n denotes the number of reserved flows in the router, while τ and α are message specific constants and N_t represents the scalability limit above which the linear approximation is not valid. With similar notions, the memory demand can be modeled with the following equation:

$$m = \mu + \beta n, \qquad n < N_m$$

The constants τ and α are larger in case of a more complex signaling handling algorithm, thus it is worth to analyze the related source code and perform the tests for each signaling primitive separately. On the contrary, constants μ and β are independent from the atomic signaling primitives and they are determined solely by the memory allocation scheme of the implementation and size of reservation states.

Other performance metrics of the reservation protocol, such as *reservation setup time*, *signaling overhead* on links and *number of signaling messages per reservation* can be retrieved from simple calculations by using the results of these measurements and the protocol specifications as input.

2.2 Critical Handler Operations

Two critical operations were considered in the signaling handling software (referred to as handler), which is running in signaling-aware routers.

Every time, when a signaling-aware router receives a signaling message, it should check in the record of existing reservation sessions whether the message refers to a new flow or triggers some change in an established reservation or perhaps just refreshes it. The performance of this *reservation session lookup* operation depends on the data structure used for storing the reservation states and the algorithm that looks up the stored entry.

If the signaling message refers to a previously unknown reservation, the handler must *create a new reservation state* (i.e., new entries). The CPU time requirement depends on the size of the data entries that the handler has to fill out and on some additional procedures, like initializing refresh timers.

The third critical operation is *setting up the internal traffic control* mechanisms in the router in order to maintain the service level of each individual reservation. This operation is performed if the received message triggers some change in the reservation state or tears it down. While SoftWare-RSVP (see Section 2.3) uses the operating system's built in traffic control routines, the SoftWare-Boomerang (see Section 2.3) uses own mechanism that is bounded to the protocol and not easy to detach it so this critical operation was not investigated.

2.3 Measurement Scenario

Measurements were carried out to obtain the memory and processing time consumption of the different protocol primitives. The test configuration is shown in Figure 2. Resource reservation was performed in the *Router*, which connected *Host 1* and *Host 2*. Signaling messages on ingress and egress ports of the router were intercepted and logged together with time stamps by *Sniffer*, which was connected to both side of the router via hubs. Pentium II PCs were used with 300 MHz CPU and 64 MB RAM, running Linux as operating system with a kernel version that supports QoS (Quality of Service) [6].

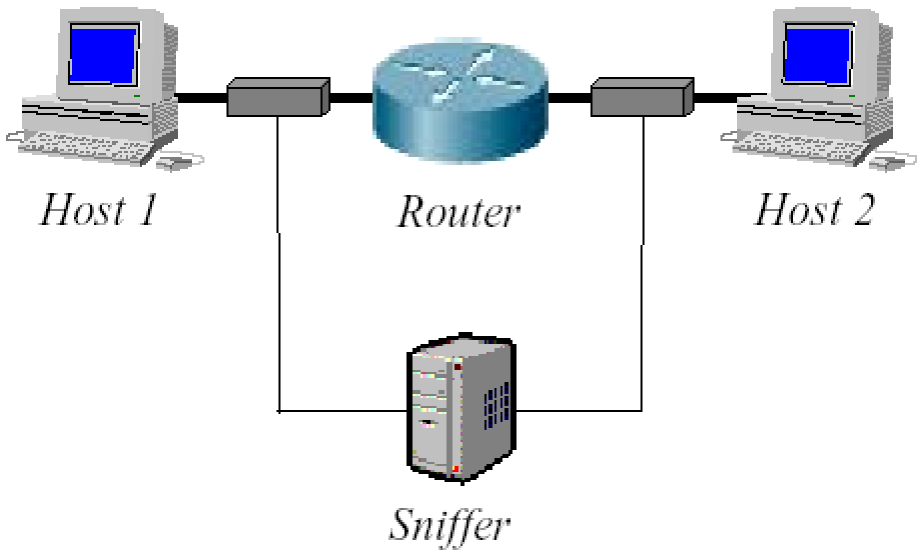


Figure 2. Measurement Scenario

Since the performance of signaling handling is very dependent on the actual implementation, thus two different QoS routers running two kinds of reservation protocols have been examined in this work. The router configurations are shown in Table 1.

Table 1. Investigated Router Configurations

Protocol	HW	SW
SoftWare-RSVP	PC	Linux, user space
SoftWare-Boomerang	PC	Linux, kernel space

For SoftWare-Boomerang measurements Telia's prototype implementation and source code [4], while for SoftWare-RSVP [7] the source code of the public ISI (Information Sciences Institute) implementation were used.

During the measurements the interesting issue to study was how the number of concurrent reservation sessions does affect the resource usage. Thus, increasing number of reserved flows was set up and maintained in the router, and each measurement was repeated 30 times. The goal was to benchmark the load caused by the signaling, so there was no data traffic utilizing the reserved resources: only setup, tear down and the periodically sent refresh messages loaded the router.

3 RESULTS

3.1 Measurement of Memory Consumption

In order to determine the memory usage as a function of the number of concurrent flow reservations, increasing number of reserved flows was set up and it was measured how much memory is allocated in case of SoftWare-RSVP and SoftWare-Boomerang.

In case of SoftWare-RSVP, simply the *proc* file system was used, which can report how much dynamic memory is reserved for the processes. However, in case of SoftWare-Boomerang, which runs as a kernel module and has no process entry, certain routines had to be placed into the module that displayed the memory allocations. The measurement results are shown in Figure 3.

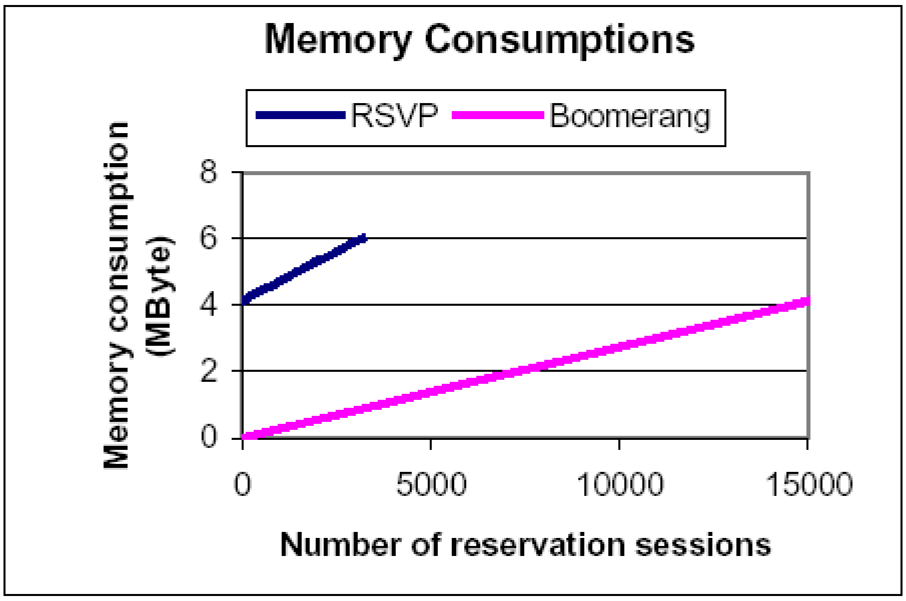


Figure 3. Memory Consumption

The memory allocation measurements revealed that the memory consumption is linear in case of both protocols. However, while SoftWare-Boomerang takes 288 bytes, SoftWare-RSVP needs 614 bytes to record the details of one reserved flow. Notice, that SoftWare-RSVP's initial memory consumption (μ) is higher than SoftWare-Boomerang's, because latter is a kernel module and takes its initial memory allocation from the kernel space, while SoftWare-RSVP is a user mode process with large arrays and a built-in client interface.

It is also visible in the figure that the investigated implementation of RSVP does not scale above to 3200 reservation sessions. This limitation was due to the receiver endpoint, after that amount of reserved flows it hung up. The figure shows the result of the first 15,000 reservation sessions for SoftWare-Boomerang, although, up to 60,000 flows were measured where the curve still had the same linear behavior.

3.2 Measurement of Message Processing Time

The time interval elapsed between the arrival and departure of a signaling message to/from the router was measured by capturing the signaling traffic before and after the router, and logging the absolute time, using *Sniffer*. Apart from the time of actual message processing, this interval includes the time, which the packet spends in the forwarding plane, e.g., the time of classification and routing.

These additional factors are expressed in the τ constant. For this measurement the *tcpdump* program was used that can display the packet arrival times in microseconds.

As said earlier, the processing time was examined as a function of the already set-up flow number. To achieve this, first the desired number of flows were set up and then a test program was launched, which set up and shortly afterwards tore down one test reservation, repeating this several times with a longer break between the *setup* and *tear down* pairs. Test flows were identified by their destination port numbers, so that the packet capture program could filter them out making it possible to focus on the test messages only.

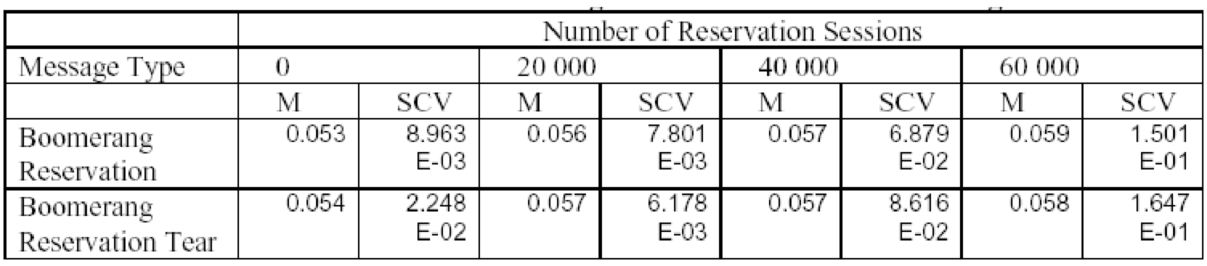
RSVP refresh messages were not measured, because they are generated between neighboring RSVP routers, so an incoming refresh message does not immediately leave on the outgoing port, instead it is sent out when the router decides that the connection requires it.

Table 2 and Table 3 show a small part of the results, where the measured processing time values were characterized with their median value *M* and with the scaled coefficient of variation *SCV* [8]:

$$SCV = \frac{Var(X)}{E^2(X)}$$

where *X* is a random variable. These results are referred to later in this report. It can be seen that the processing time median values and their scaled coefficient variance always increase when the number of maintained reservations are increased in the router, but the different protocols and implementations give different steepness. Notice, that in case of SoftWare-Boomerang the scale of reservation sessions is larger by two order of magnitude, than in case of the other examined protocol.

Table 2. Measured Processing Time in case of SoftWare-Boomerang



	Number of Reservation Sessions							
Message Type	0		20 000		40 000		60 000	
	M	SCV	M	SCV	M	SCV	M	SCV
Boomerang Reservation	0.053	8.963 E-03	0.056	7.801 E-03	0.057	6.879 E-02	0.059	1.501 E-01
Boomerang Reservation Tear	0.054	2.248 E-02	0.057	6.178 E-03	0.057	8.616 E-02	0.058	1.647 E-01

Table 3. Measured Processing Time in case of SoftWare-RSVP

	Number of Reservation Sessions							
Message Type	0		200		400		600	
	M	SCV	M	SCV	M	SCV	M	SCV
RSVP Path	15.44	9.90 E-07	15.57	1.601 E-05	15.72	3.102 E-05	15.89	3.458 E-05
RSVP Path Tear	0.169	2.22 E-03	0.151	1.052 E-01	0.151	1.650 E-01	0.156	4.095 E-01
RSVP Reservation	8.607	2.59 E-06	9.081	1.939 E-05	9.583	1.996 E-05	10.06	3.938 E-05
RSVP Reservation Tear	8.078	4.58 E-06	8.553	4.124 E-05	9.025	3.399 E-05	9.521	1.196 E-04
RSVP Confirmation	0.243	9.67 E-04	0.257	1.450 E-02	0.241	4.276 E-02	0.240	1.388 E-01

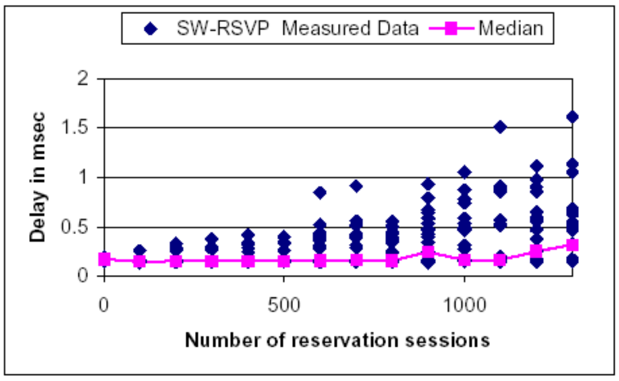
3.3 Analysis of Algorithmic Complexity

The available source code was analyzed in order to estimate how much time the router does spend on critical operations.

3.3.1 Reservation Session Lookup

According to the source code *bucket hash* algorithm is used in ISI's RSVP implementation, while *modified binomial tree* is implemented in Boomerang prototype. The efficiency of the lookup routines was estimated based on the processing time measurements on similar functions. Investigating the source code of the ISI RSVP implementation, the *RSVP Path Tear* message processing should be the shortest in processing time, because it consists solely a lookup and freeing of session entry. The throughput time of *Path Tear* message begins around 170µs and slightly raises with the number of reservations, while the variance of the measured values increases steeper as it can be seen on Figure 4 (notice, that the scales on the axis are different in case of the different protocols). This phenomenon can be explained by the increasing size of hash table.

In case of SoftWare-Boomerang, the reservation tear message was investigated, which is functionally the same as the corresponding RSVP message, although it also cleans the associated queue. The result indicates that the processing time of SoftWare-Boomerang's tearing begins at 53µs and increases very slowly. In case of 60,000 reserved sessions, the reservation tear takes still 58µs only. However, the variance of processing time did not show a noticeable change while the number of sessions was increasing. It is worth to mention that the measured processing time is affected by other factors, as well, for instance the kernel performs numerous operations on the received packet while giving it to the protocol handler routines. That is why the simple forwarding time was also measured, where QoS protocols were not running on the router. In this case the forwarding of a single packet, like a Boomerang or RSVP message, takes about 40μs.



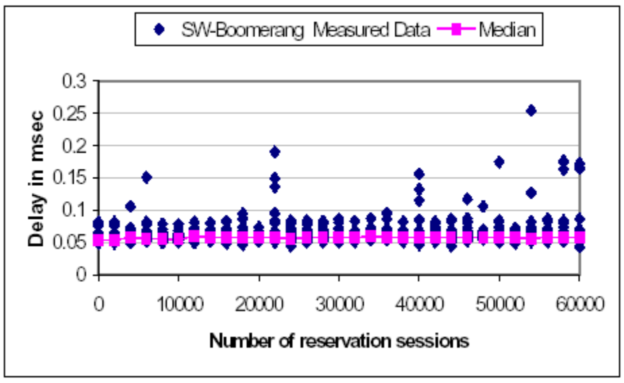


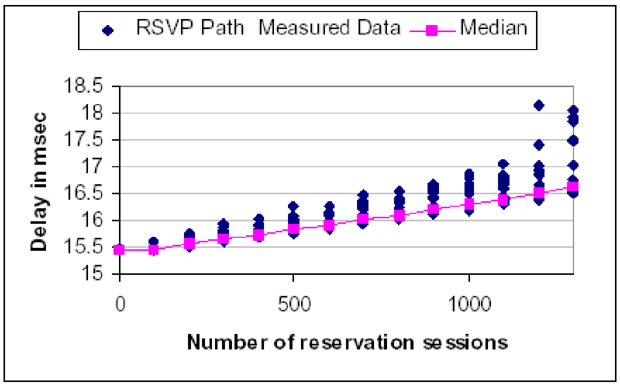
Figure 4. Teardown Messages for SoftWare-RSVP (Path Tear) and SoftWare-Boomerang

The difference of the two lookups is based on mainly the data structure and the lookup algorithm. SoftWare-Boomerang takes smaller structures and uses modified binomial tree algorithm, which costs about 13μs for a flow, if we don't take into account the 40μs packet forwarding time, and increases very slowly. The bucket hashing algorithm and large data structures used by RSVP results in larger processing time, around 140μs without packet forwarding, and it increases faster than it was at the previous protocol.

3.3.2 Creation of New Reservation State

According to the source codes, new reservation states are created in case of RSVP *Path* and *Resv* messages. Unlike this, the Boomerang prototype stores reservation states in lookup entries and no state information is required for handling a state machine of signaling.

RSVP *Path* and *Path Tear* messages show the biggest difference, which is around 15ms (see Figure 5 – notice, that the scales on the axis are different). The difference arises directly from the creation of a new reservation state. The rest of the functionality, like reservation session lookup, can be found in both message handling routines.



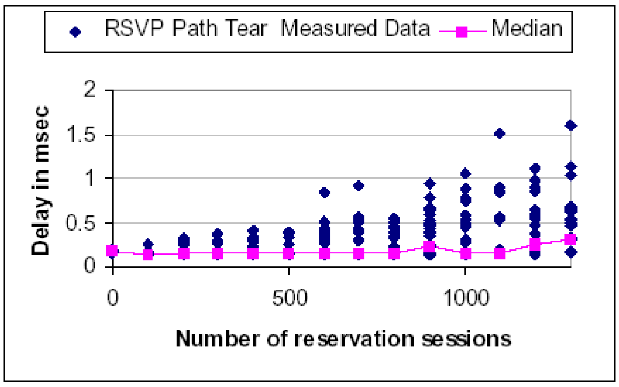


Figure 5. RSVP Path and Path Tear Message Processing Time

4 CONCLUSIONS

This work investigated different factors, which significantly contributes to the scalability limitations of signaling based resource reservations in Internet.

It has been shown that the reservation lookup scheme is an important factor in scalability. Lookup of reservation session is three times longer for the SoftWare-RSVP than for SoftWare-Boomerang.

The most unexpected result of the measurements is that the reservation state creation is the most time consuming event in case of SoftWare-RSVP. It takes about 15ms, while in case of SoftWare-Boomerang the cost of reservation state creation is almost nothing.

Additional measurements show that signaling intensity – the number of atomic signaling messages received by the router per second – is also an important scalability factor. In case of SoftWare-Boomerang we could scale up to 3600 messages per second without having lost signaling messages. On the other hand SoftWare-RSVP has saturated already by 30 messages per second.

It should, however, be noted that apples cannot be fairly compared to pears; therefore the difference of configuration should not be neglected while looking at the results. The SoftWare-RSVP setup is capable of handling multicast sessions and the handler run as a Linux, user-space routine. On the other hand, the prototype in SoftWare-Boomerang is designed for unicast traffic and the signaling handler run as a Linux kernel module. The comparison here was only to show the scalability consequences of different implementation approaches.

REFERENCES

[1] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[2] R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview", Internet RFC 1633, July 1994.

[3] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP) Version1 Functional Specification", IETF RFC 2205, Proposed Standard, September 1997.

[4] D. Ahlard, J. Bergkvist, I. Cselényi, "Boomerang Protocol Specification", Internet Draft, June 1999; http://www.ietf.org/internet-drafts/draft-bergkvist-boomerang-spec-00.txt

[5] Paul White and Jon Crowcroft, "A Case for Dynamic Sender-Initiated Reservations in the Internet", Journal of High Speed Networks, Special issue on QoS Routing and Signaling, Vol 7 No 2, 1998.

[6] "Differentiated Services on Linux", Internet Draft, June 1999; http://www.ietf.org/internet-drafts/draft-almesberger-wajhak-diffservlinux-01.txt

[7] USC Information Sciences Institute (ISI) implementation of RSVP: http://www.isi.edu/div7/rsvp/

[8] Raj Jain, "The Art of Computer Systems Performance Analysis", John Wiley & Sons, New York, 1991.