Benchmarking of Signaling Based Resource Reservation in the Internet

István Cselényi[1], Gábor Fehér[2], and Krisztián Németh[2],

[1] Department of Network Services, Telia Research AB
Vitsandsgatan 9, S-12386 Farsta, Sweden, Phone: +46 8 713 8173
istvan.i.cselenyi@telia.se
[2] High Speed Networks Laboratory,
Department of Telecommunications and Telematics, Technical University of Budapest
Pázmány Péter sétány 1/D, H-1117 Budapest, Hungary, Phone: +36 1 463 3119
{feher, nemeth_k}@ttt-atm.ttt.bme.hu

Abstract. This paper 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 and on a commercial router. The signaling handling algorithm of the implementations were analyzed as well and critical operations were identified. Our results show that CPU time is a more significant scalability concern than the router memory and also that the former is very dependent on the implementation and complexity of the signaling algorithm. Thus the same Linux PC can handle Boomerang reservation requests several hundred times faster than RSVP requests.

1 Introduction

It is not so unpopular any more to mention signaling in the context of Differentiated Services [1] as it was some years ago [2]. 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. We investigated the resource demand in signaling-aware routers in terms of memory and processing power for the RSVP [3] and Boomerang [4] 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 us ST-II [5], Yessir [6], Ticket [7] or DRP [8]. A summary of resource reservation protocols and comparative evaluation can be found in [8].

G. Pujolle et al. (Eds.): NETWORKING 2000, LNCS 1815, pp. 643-654, 2000

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 and forwarded hop-by-hop towards the receiver(s). *Path* registers the address of previous hop at each node, conveys the sender's traffic specification and optionally network specific information from involved nodes. 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*, 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 basic RSVP messages. The *Path Tear* and *Resv Tear* messages are used to tear down a certain reservation state, while the *Resv Conf* 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][9][10] 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 loop (currently the ICMP Ping message) and kept alive by periodically repeated Boomerang messages, which establish soft reservation-states (see Fig. 1).

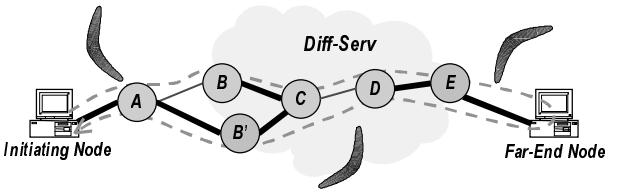

Fig. 1. Reservation Setup with the Boomerang Protocol

Unlike in 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. The required service quality can be given in various ways; by specifying a DSCP [12] and peak information rates for the forward and backward directions or by providing an IntServ-like object [13]. In our prototype implementation [4] 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 an easy penetration of the protocol in current Internet.

This paper is organized as follows: Section 2 describes our benchmarking framework by defining the investigated factors and the measurement scenario. Section 3 provides the measurement results and their analysis. Finally, conclusions are drawn in Section 4.

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, which is proportional 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. The following equations can be derived from simple heuristics:

$$t = \tau + \alpha n, \qquad n < N_t \tag{1}$$

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:

$$m = \mu + \beta n, \qquad n < N_m \tag{2}$$

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 [9].

2.2 Critical Handler Operations

We identified three critical operations 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, triggers some change in an existing reservation or 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 entry). The latency of this operation depends on the size of the entries that the handler has to fill out and on some additional procedures, like initializing refresh timers.

The signaling handler has to *set up the internal traffic control* mechanisms in the router in order to maintain the QoS of each individual reservation. This operation is not performed for refresh messages. Although the initiation of traffic conditioners includes setting of the shaper, meter or dropper, we concentrated only on setup and parameterization of the queues in the routers.

2.3 Measurement Scenario

Measurements were carried out to obtain the memory and processing time consumption of the different protocol primitives. Resource reservation was performed in the Router, which connected Host 1 and Host 2 (Fig. 2). Signaling messages on ingress and egress ports of the router were intercepted and logged together with time stamps by Sniffer running *tcpdump*, which was connected to both sides 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 [14].

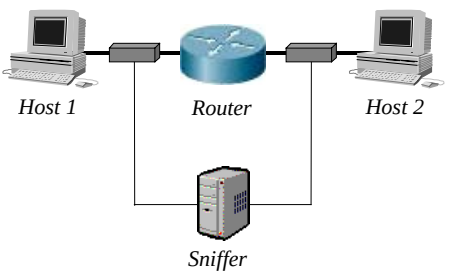

Fig. 2. Measurement Scenario

Since the performance of signaling handling very strongly depends on the actual implementation, we have examined three different QoS routers running two kinds of reservation protocols. The three router configurations are shown in Table 1.

Table 1. Investigated Router Configurations

Configuration	Hardware	Software	Protocol
SW-RSVP	PC	Linux, user space	RSVP
SW-Boomerang	PC	Linux, kernel space	Boomerang
HW-RSVP	Dedicated	No data available	RSVP

We used Telia's prototype implementation and source code for SW-Boomerang and the source code of the public ISI implementation for SW-RSVP [15] measurements. The dedicated hardware was a 3Com CoreBuilder 3500 router, which supports RSVP. We repeated each measurement 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 versus the number of concurrent reservations, we set up an increasing number of reserved flows and measured the amount of memory that is allocated in case of SW-RSVP and SW-Boomerang. We could not measure the allocated memory in case of HW-RSVP, having no access to the internal parts of the router. The measurement results are shown in Fig. 3.

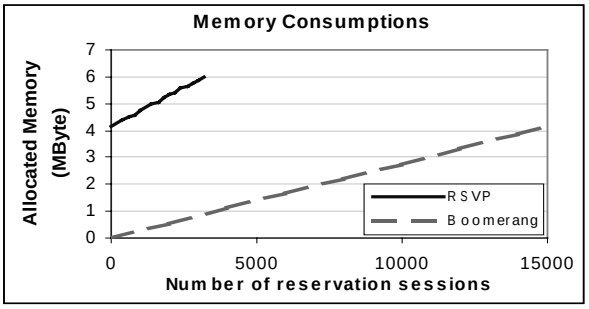

Fig. 3. Memory Consumption

The memory allocation measurements revealed that the memory consumption is linear in case of both protocols. However, while SW-Boomerang takes 288 bytes, SW-RSVP needs 614 bytes to record the details of one reserved flow, partially because

they use different traffic descriptors. Notice that SW-RSVP's initial memory consumption (μ) is higher than SW-Boomerang's, because latter is a kernel module and takes its initial memory allocation from the kernel space, while SW-RSVP is a user mode process with large arrays and a built-in application programming 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 our receiver endpoint, which could not handle more. The figure shows the result of the first 15,000 reservation sessions for SW-Boomerang, however we measured up to 60,000 flows 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 in the router was measured by capturing the signaling traffic before and after the router, and logging the absolute time. 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 (see equation 1).

For measuring the processing time as a function of established reservations, we set up the desired number of reservations and then launched a test program, which set up and shortly afterwards tore down one test reservation, repeating this 30 times with a longer pause among the setup and tear down pairs. RSVP refresh messages were not measured, because they are generated by 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, Table 3 and Table 4 show our main results, where we characterized the measured processing time values with their median value *M* and with the scaled coefficient of variation (*SCV)*.

We can see that the processing time and the SCV always increase proportionally to the number of maintained reservations in the router, but different protocols and implementations yield different slopes. The most interesting result is that the SW-Boomerang prototype can process one Boomerang reservation message within 53-59 microseconds, while the SW-RSVP implementation needs more than 15 ms, 8 ms for handling an RSVP Path, RSVP Resv message, respectively. Moreover, in case of SW-Boomerang the scale of reservation sessions is larger by two orders of magnitude, than in case of the two RSVP implementations (i.e. 60000 and 600, respectively).

3.3 Analysis of Algorithmic Complexity

We have analyzed the available source code in order to estimate the time the router spends on critical operations

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

Message Type	Number of Sessions / Median and SCV							
	0		20 000		40 000		60 000	
	M (ms)	SCV	M (ms)	SCV	M (ms)	SCV	M (ms)	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 SW-RSVP

Message Type	Number of Sessions / Median and SCV							
	0		200		400		600	
	M (ms)	SCV	M (ms)	SCV	M (ms)	SCV	M (ms)	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	0.150	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

Table 4. Measured Processing Time in case of HW-RSVP

Message Type	Number of Sessions / Median and SCV							
	0		200		400		600	
	M (ms)	SCV	M (ms)	SCV	M (ms)	SCV	M (ms)	SCV
RSVP Path	7.402	5.389 E-01	12.26	6.690 E+00	20.46	2.544 E+00	577.3	8.814 E-01
RSVP Path Tear	5.554	9.932 E-01	5.455	9.800 E+00	6.682	3.698 E+00	216.1	1.554 E+00
RSVP Reservation	11.15	5.359 E-01	13.41	1.508 E+00	16.83	3.440 E+00	262.8	1.259 E+00
RSVP Reservation Tear	8.342	1.781 E-01	18.07	6.275 E+00	35.61	2.975 E+00	70.49	4.778 E+00
RSVP Confirmation	5.385	1.170 E+00	5.366	6.903 E-02	5.448	1.644 E+01	43.30	2.008 E+00

Reservation Session Lookup

According to the source code the *bucket hash* algorithm is used in ISI's RSVP implementation, while *modified binomial tree* is implemented in Boomerang prototype. We characterized the efficiency of these lookup algorithms by comparing processing latency of different message primitives. Investigating the source code of the ISI RSVP implementation, we found that the *RSVP Path Tear* message processing should be the fastest, because it contains solely a session entry lookup and delete. The

processing time of *Path Tear* message begins around 150μs and slightly raises with the number of reservations, while the variance of the measured values increases steeper as it can be seen on Fig 4. This phenomenon could be explained by the increasing size of the hash table.

In case of SW-Boomerang, we investigated the reservation tear message, which is functionally the same as the corresponding RSVP message, although it cleans the associated queue as well. The result indicates that processing time of a Boomerang tear message starts at 53μs and increases very slowly. In case of 60 000 reserved sessions, the reservation tear takes just 58μs. The variance of processing time did not show a noticeable change while we increased the number of sessions.

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 it gives to the protocol handler routines. Thus we measured the simple forwarding time too, where there was no QoS protocol running on the router. We found that forwarding a single 100 bytes-long data packet takes about 40μs.

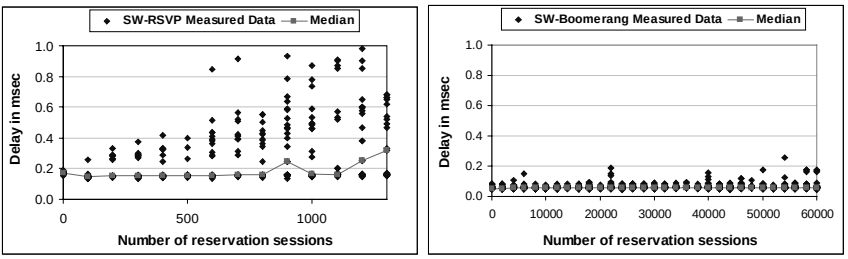

Fig. 4. Tear-down Messages for SW-RSVP (Path Tear) and SW-Boomerang

Apart from the faster execution of kernel space programs (SW-Boomerang), the reason of unequal lookup performance is the difference in data structure and lookup algorithm. SW-Boomerang takes smaller structures and uses a binomial tree lookup algorithm, which costs about 13μs per flow (since packet forwarding time is 40μs), and increases very slowly. The bucket hashing algorithm and large data structures used by RSVP results in larger processing time, around 110μs without packet forwarding, and it increases faster than for Boomerang.

Internal QoS Setup

The protocols differ in the implementation of the traffic control as well. While SW-RSVP uses the operating system's built in traffic control routines, the SW-Boomerang uses own traffic control implementation that is bounded to the protocol.

We repeated our SW-RSVP measurements with disabled traffic control operation and estimated the time spent for internal QoS setup. The difference in the message processing times expresses the time consumed by maintaining traffic queues. Fig. 5 shows the median values for reservation and path related RSVP messages with and without traffic control.

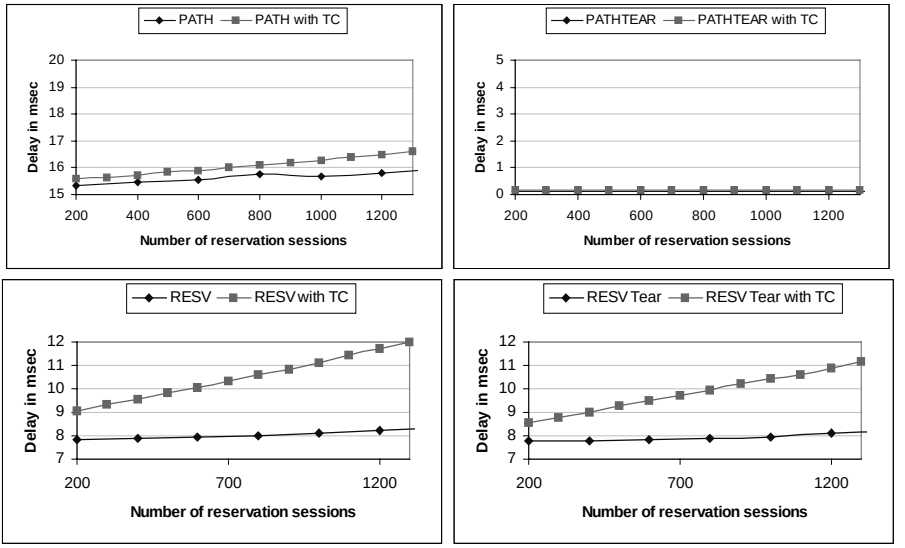

Fig. 5. RSVP Message Processing Time with and without Traffic Control

We can observe that in case of *Path* and *Path Tear* messages, there is no significant difference between the two measurement series. *Path* message is affected a little bit more than *Path Tear*, because RSVP fills out an advertisement message block that reflects the router's load calculated from traffic parameters. However in case of reservation messages, a strong difference can be revealed. The processing time was increased with 1ms (i.e. almost 15 %) and the slope of the line is larger than without traffic control. We can confirm it looking up the source code, as both types of reservation message calls the traffic control interface and the raising slope can be explained with the increasing number of queues that must be maintained.

In case of SW-Boomerang, the handler uses its own traffic control routines what simplifies the internal QoS setup thus the related CPU time is marginal. On the other hand, it should be noted that the current prototype implementation of Boomerang deals with much simpler traffic descriptor (peak rate only) than RSVP.

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 handling prototype stores reservation states in lookup entries and no state information is required for handling a state machine of signaling.

We tried to make deductions to this fact from the measured processing time values. RSVP *Path* and *Path Tear* messages show the biggest difference, which is around 15ms (see Fig. 6). When we inspected the source code, it clearly showed that 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.

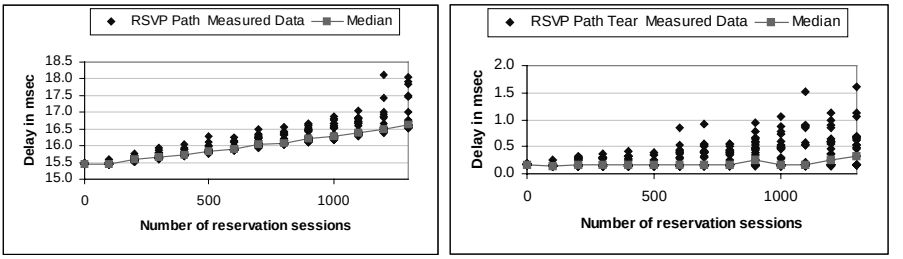

Fig. 6. RSVP Path and Path Tear Message Processing Time

3.4 Hardware RSVP Measurements

In case of HW-RSVP, we were restricted to processing time measurements, as we had no access to the internal parts of the router. The processing time measurements are presented in Table 4 and surprisingly show that HW-RSVP performs worse than SW-RSVP (see in Table 3).

In our experiments the device was not capable to scale above 600 reservation sessions. This was not a built-in limit, but above that the router's RSVP handler was not able to refresh the flows, meaning that it was not able to generate refresh messages.

If we assume that the RSVP *Path Tear* message, displayed in Table 4, mainly consist of the session lookup procedure, as we have shown it in case of SW-RVSP, then we can claim that the router device had an inefficient session lookup routine. It is thought that the device was not designed for more than few hundred sessions. We can confirm that the most time consuming part is the session lookup, during processing the *Path Tear* message, if we compare results of *Confirmation* message and assume that there is no session lookup for *Confirmation* messages in this implementation. We could not switch off the traffic control routines in the 3COM router thus could not retrieve the time factor related to queue handling. As the measured *Path* message processing time shows, it begins from a lower value than SW-RSVP, but it increases much faster.

4 Conclusions

This paper investigated different factors, which contribute to the scalability limitations of signaling based resource reservations in Internet.

We have shown that the reservation lookup scheme is an important factor in scalability. Lookup of reservation session is three times faster for SW-Boomerang than for SW-RSVP and this factor is even worse for HW-RSVP. Another result of the measurements is that the reservation state creation is the most time consuming event in case of SW-RSVP. It takes about 15ms, while in case of SW-Boomerang the cost of

reservation state creation is just 53 μs. We measured the time factor of traffic control in case of SW-RSVP and SW-Boomerang. The results indicate that internal implementation of traffic queues can speed up the processing of related messages, moreover it can decrease the impact of established reservation session on processing time. We observed that the HW-RSVP implementation has worse performance characteristics than SW-RSVP. The measured processing time of protocol messages goes exponential with the number of reservation sessions; meanwhile the scaled coefficient of variation is larger by four orders of magnitude than in case of SW-RSVP.

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 SW-Boomerang we could scale up to 3600 messages per second without having lost signaling messages. On the other hand SW-RSVP has been 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 our results. The SW-RSVP uses complex traffic descriptors and the handler run as a Linux user-space routine. On the other hand, the SW-Boomerang prototype handles only peak rate reservation running as a Linux kernel module.

Acknowledgements

We would like to thank the excellent design and implementation work of Joakim Bergkvist and the help in the measurements to Tomas Engborg, David Ahlard and Norbert Végh, all from Telia Research Sweden. We are also grateful for the help of members of the RSVP mailing list, who provided us with useful information about the implementation of RSVP protocol.

References

1 S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.
2. Y. Bernet, R. Yavatkar, P. Ford, F. Baker, L. Zhang, M. Speer, R. Braden, B. Davie, J. Wroclawski, E. Felstaine, "A Framework for Integrated Services Operation Over Diffserv Networks", Internet Draft, September 1999, <draft-ietf-issll-diffserv-rsvp-03.txt>
3. R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource Reservation Protocol (RSVP) Version 1 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. C. Topolcic, "Experimental Internet stream protocol, version 2 (ST-II)", Request for Comments (Experimental) RFC 1190, Internet Engineering Task Force, October 1990.
6. P. Pan, H. Schulzrinne, "YESSIR: A Simple Reservation Mechanism for the Internet", http://www.ctr.columbia.edu/~pan/work/yessir/yessir.ps

7. A. Eriksson, C. Gehrmann, "Robust and Secure Light-weight Resource Reservation for Unicast IP Traffic", International WS on QoS'98, IWQoS'98, May 18-20, 1998
8. 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.
9. G. Fehér, K. Németh, M. Maliosz, I. Cselényi, J. Bergkvist, D. Ahlard, T. Engborg, "Boomerang – A Simple Protocol for Resource Reservation in IP Networks", IEEE WS on QoS Support for Real-Time Internet Applications, Vancouver, Canada, June 1999
10. I. Cselényi, G. Fehér, K. Németh, "On the Scalability of Scalability of Signaling-Based Resource Reservation in Internet", submitted to the IEEE Journal on Selected Areas in Communications
11. I. Cselényi, R. Szabó, "Service Specific Information Based Resource Allocation for Multimedia Applications", accepted for publication in journal of Informatica
12. K. Nichols, S. Blake, F. Baker and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.
13. R. Braden, D. Clark, and S. Shenker, "Integrated Services in the Internet Architecture: An Overview", Internet RFC 1633, July 1994.
14. "Differentiated Services on Linux", Internet Draft, June 1999 <draft-almesberger-wajhak-diffserv-linux-01.txt>
15. University of Southern California Information Sciences Institute (USC ISI) implementation of RSVP: http://www.isi.edu/div7/rsvp/