Available online at www.sciencedirect.com
ScienceDirect
Computer Communications 30 (2007) 1199–1204
computer communications
www.elsevier.com/locate/comcom

A lightweight implementation of RSVP-TE protocol for MPLS-TE signaling

Young Woo Lee [a], Suncheul Kim [b], Jaehyung Park [c,*], Sang Ha Kim [d]

[a] Operations Support System Laboratory, KT, 463-1, Jeonmin-dong, Yusong-gu, Daejeon 305-811, Republic of Korea
[b] Broadband Converged Network Research Division, ETRI, 162 Gajeong-dong, Yusong-gu, Daejeon 305-350, Republic of Korea
[c] Department of Computer and Information Engineering, Chonnam National University, 300 Yongbong-dong, Buk-gu, Gwangju 500-757, Republic of Korea
[d] Department of Computer Engineering, Chungnam National University, 200, Kung-dong, Yusong-gu, Daejeon 305-764, Republic of Korea

Received 14 November 2005; received in revised form 4 December 2006; accepted 7 December 2006
Available online 2 January 2007

Abstract

Traffic engineering based on an MPLS technology is being introduced for providing high quality-guaranteed services over the Internet. As one of MPLS signaling protocols for traffic engineering, a RSVP-TE protocol transmits and receives periodic refresh messages for maintaining the connection state of a flow-based path. Such a soft state characteristic gives a heavy processing overhead to routers for maintaining connection states in case of being a large number of paths established. In this paper, we propose a lightweight implementation approach of a RSVP-TE protocol reducing processing overhead on periodic messages by adapting the Hello mechanism. We evaluate the processing overhead of the implemented lightweight RSVP-TE protocol compared with the refresh reduction scheme. Our evaluation results show that the proposed RSVP-TE protocol can manage a large number of LSP states without increasing overhead of refresh messages.

Keywords: MPLS networks; RSVP-TE protocol; Lightweight implementation; Hello protocol; Refresh reduction

1. Introduction

The RSVP-TE protocol is developed by extending the RSVP protocol for providing traffic engineering capability in MPLS networks [2,10,12]. The RSVP-TE protocol supports the instantiation of explicitly routed label switched paths (LSPs) [1]. For providing such a functionality, the RSVP-TE protocol performs establishment, release, and management of quality-guaranteed LSPs through transmitting the explicitly routed path information, label information, and traffic flow information. Like as the original RSVP protocol [4,14], the RSVP-TE manages the connection state of the established LSP by a soft-state mechanism.

Due to such the soft-state mechanism, the general implementation of the RSVP-TE protocol has been slowed mainly by concerns about its scalability [9,11]. The load generated by the management of thousands of RSVP sessions has been judged too heavy for a single core router. The requirements posed by the RSVP protocol in terms of link bandwidth, memory usage, and processing capability are essentially proportional to the number of sessions maintained by the router. Like as the RSVP protocol, the RSVP-TE for MPLS networks periodically transmits refresh messages to manage the LSPs connection status by the soft-state mechanism. Hence, the scalability problem of the RSVP-TE protocol still remains even in MPLS networks.

Many researcher tried to solve the scalability dealing with a large number of flows [3,5,7,13]. Especially, the IETF RSVP Working Group published RFC2961 entitled with "RSVP Refresh Overhead Reduction Extensions" [3]. This RFC exploits the inherent flexibility of the protocol and extends the RSVP protocol reducing the dependency

* Corresponding author. Tel.: +82625301796; fax: +82625301809.
E-mail address: hyeoung@chonnam.ac.kr (J. Park).

0140-3664/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2006.12.003

of the total overhead of the RSVP signaling on the number of RSVP sessions. Such schemes are focused on reducing the link bandwidth and/or router's memory usage used by sending and receiving refresh messages. In the contrary, to reduce CPU load for processing protocol messages, the hardware-accelerated implementation approach is proposed in [13]. The proposed accelerator still has processing load for maintaining the flow's connection state.

Such schemes, even the hardware-accelerated approach, does not still alleviate the processing load of protocol messages which gives an effect on scalability. In this paper, we analyze how the reduction mechanism of refresh messages is effective to the RSVP-TE protocol. This paper suggests a lightweight implementation of RSVP-TE protocol that has less overhead of message processing, by adapting the Hello mechanism to manage lots of LSPs. And then, we evaluate performance of the implemented RSVP-TE protocol compared with the legacy refresh and the refresh reduction scheme in terms of the processing overhead of periodic messages. From the results of evaluation, the proposed lightweight implementation of RSVP-TE protocol can manage a large number of LSP states with increasing a little overhead of refresh messages.

This paper is organized as follows. Section 2 describes the RSVP-TE signaling protocol and the refresh reduction mechanism. In Section 3, we propose a lightweight implementation approach of the RSVP-TE protocol in MPLS networks. In Section 4, we evaluate the processing overhead of the implemented RSVP-TE protocol and conclude in Section 5.

2. Backgrounds

In this section, we describe the RSVP-TE signaling protocol and the refresh reduction scheme for high reliability and scalability.

2.1. RSVP-TE protocol

RSVP (Resource ReserVation Protocol) [4,14] is designed as a protocol that can reserve the network resource to offer the real time applied service in internet. To reserve the resource that is required in service, RSVP protocol transmits and receives Path and Resv message, and then it sets up the flow to transmit the created traffic from service. Moreover, it executes soft state mechanism that periodically transmits and receives same Path/Resv message, to manage the status of established flow. Here, the periodical message is called the *Refresh message*.

Most of current internet routers is implemented with UNIX-based operating system. On routers with UNIX OS, it is known that the UNIX socket performance has been significantly affected by the number of messages rather than the size of messages [6,8]. In other words, the processing overhead of UNIX socket input/output does not matter the size of a message but the number of messages. Hence, the RSVP protocol with the soft state mechanism creates many refresh messages with many flows on the established tunnels. Due to this characteristics, the RSVP protocol is applied in a small-scale network.

For supporting MPLS functionality, the RSVP-TE protocol provides LSP establishment and release capabilities by adding some objects to set explicitly routed path and to transmit an individual MPLS label [1]. However, for the RSVP-TE protocol preserves the soft state mechanism like as the existing RSVP, LSPs connection management mechanism still has a disadvantage of processing overhead of periodic protocol messages. Also, the RSVP-TE protocol applied in MPLS network transmits the refresh message containing the established LSPs status information through the IP routing path not through the established LSP.

2.2. Refresh reduction techniques

The *Bundle message* defined in RFC 2961 [3] packs a number of RSVP messages sent to the same RSVP neighbor within a single larger RSVP message. To this purpose, a new RSVP bundle message is defined: this message has its own header and a body which is made up with a sequence of RSVP messages. Bundling is performed on a per-hop base. The receiving router unpacks the sub-messages and processes them. The advantages brought by the adoption of such mechanism are given by the reduced usage of bandwidth, as a number of IP and data link headers are replaced by a single one, and by the fewer network interruptions for the router's OS to deal with.

The goal of message bundling is to reduce the total number of RSVP messages that routers have to process. That is, the total number of messages that routers send and receive is reduced to $\frac{N}{B}$, where N is the number of the original refresh (PATH and RESV) message and B is the bundling factor which several refresh messages are bundled into one message. B is calculated as $\frac{\mathrm{MTU}}{M}$, where MTU is the maximum transmission unit (MTU) size of data link layer and M is the average size of refresh messages. This solution is particularly useful to relieve the processing burden on routers that are running a UNIX operating system, since the socket interface has always been a processing bottleneck.

The *Summary Refresh (Srefresh)* extension [3] is based on the creation and the transmission of the identifier associated with a refresh message. The periodical refresh of the flow's connection state is then simply performed by sending the identifiers of the original messages. Thus the amount of refresh traffic is dramatically reduced. PATH and RESV trigger messages are sent with a new MESSAGE-ID object. This object has a 32-bits sized field that identifies the refresh message. The value of the identifier field has a scope that is local to a pair of nodes.

Once a node has created a new state by sending a trigger message together with its identifier, it can subsequently use the MESSAGE-ID object to refresh its related connection state. Multiple states can be refreshed with a new summary

refresh message made up by a sequence of identifiers. The use of small-sized identifiers greatly reduces the amount of the bandwidth wasted for transmitting signaling messages and the usage of CPU for maintaining the soft state. In the summary refresh scheme, the amount of link bandwidth for signaling messages can be reduced to $\frac{I}{M}$, where I is the size of identifiers and M is the average size of signaling messages. Moreover, in the summary refresh scheme with bundling, the total number of messages that routers send and receive is reduced to $\frac{N}{B \times S}$, where S is the summarizing factor as $\frac{M}{I}$.

In above two schemes, scalability problem of the RSVP-TE signaling protocol is focused on link bandwidth and/or router's memory usage. However, CPU processing overhead of maintain the flow's connection state is one of the most important factors [9,11]. Therefore, we focus on CPU processing overhead in this paper.

3. A lightweight implementation of RSVP-TE protocol

In this section, we propose a lightweight implementation approach of RSVP-TE protocol by adapting the Hello mechanism in order to reduce CPU processing overhead of maintaining the LSPs state in MPLS networks. Also, we describe the implemented architecture and procedures for protocol messages.

3.1. Adaptation of the hello mechanism

First, we investigate the nature of a soft-state mechanism operated in MPLS networks. The purpose of the soft-state mechanism in RSVP-TE protocol is to manage the connection state of established LSPs. However, the refresh message containing the LSPs connection state transmits through an IP routing path, not through the established LSP. That is, the RSVP-TE protocol applied in MPLS network transmits the refresh message containing the established LSPs status information by the IP routing path within a fixed refresh period. Therefore, such discrepancy is inevitable in MPLS networks deploying the RSVP-TE protocol.

The RSVP Hello extension [4] enables RSVP nodes to detect when a neighboring node is not reachable. The mechanism can provide node-to-node failure detection or link failure detection. The Hello extension is composed of a Hello message including a HELLO REQUEST object and a HELLO ACK object. As previously described, the Hello mechanism is used for node failure and/or link failure only between an RSVP node and its neighbor node. In order to manage an end-to-end RSVP-TE flow between the ingress and the egress LSR of the established LSP, we extend the Hello mechanism that the RSVP node can advertise its neighbor's failure condition to all ingress and egress LSRs by applying RSVP-TE error reporting mechanism. And then, all ingress and egress LSRs are going to begin disconnecting the corresponding LSPs through the failure node or link.

3.2. An implemented architecture of RSVP-TE protocol

Our implemented architecture of a lightweight RSVP-TE protocol for MPLS traffic engineering is shown in Fig. 1. The implemented RSVP-TE protocol engine consists of five components. Among these blocks, Path/Resv state block (PSB/RSB) and Timer Management block (TMB) are a core of the basic RSVP protocol. The protocol engine saves the LSP information to PSB/RSB when a new LSP is established. TM block executes the soft-state mechanism sending and receiving periodic refresh messages.

Another three blocks are implemented for supporting scalability and reliability. These are MsgId, ReTrB, and NbrSt blocks. MsgId block saves the Message-ID values that were already determined when the RSVP-TE triggering message is sent. And the Message-ID value is directly linked with each PSB/RSB element and used in the summary refresh scheme. ReTrB block performs retransmitting protocol messages which the corresponding ACK message is not received, for reliability. NbrSt block maintains states of neighbor RSVP nodes by the Hello mechanism in the protocol engine. If a Hello message cannot be successfully delivered from an adjacent router, an RSVP error message is created and passed to the ingress and egress LSRs of all relevant LSPs.

3.3. Procedures for processing protocol messages

The implemented RSVP-TE protocol executes the summary refresh scheme and the retransmission mechanism for reducing refresh messages and transmitting protocol messages with reliability.

The summary refresh scheme utilizes bandwidth usage by replacing Path/Resv refresh message with Srefresh message with many MESSAGE-ID objects. After completion of an LSP establishment in MPLS networks, LSR/LER already saved the identifier value used in MESSAGE-ID object. For refreshing the established LSP, the summary refresh scheme creates Srefresh messages by sequentially appending the identifier values extracted from RSB/PSB table. Fig. 2 shows the procedure that creates and transmits a Srefresh message. And, after receiving a Srefresh

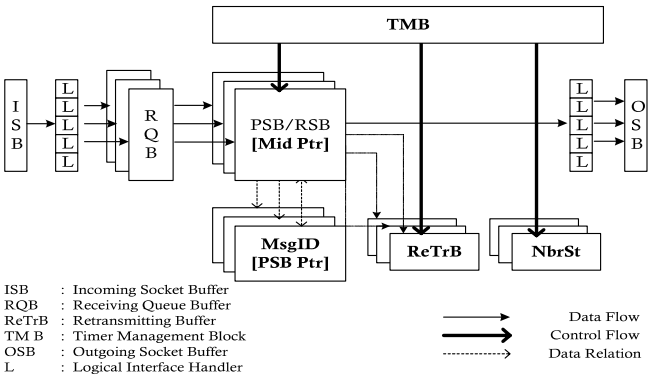

Fig. 1. An implemented architecture of RSVP-TE protocol.

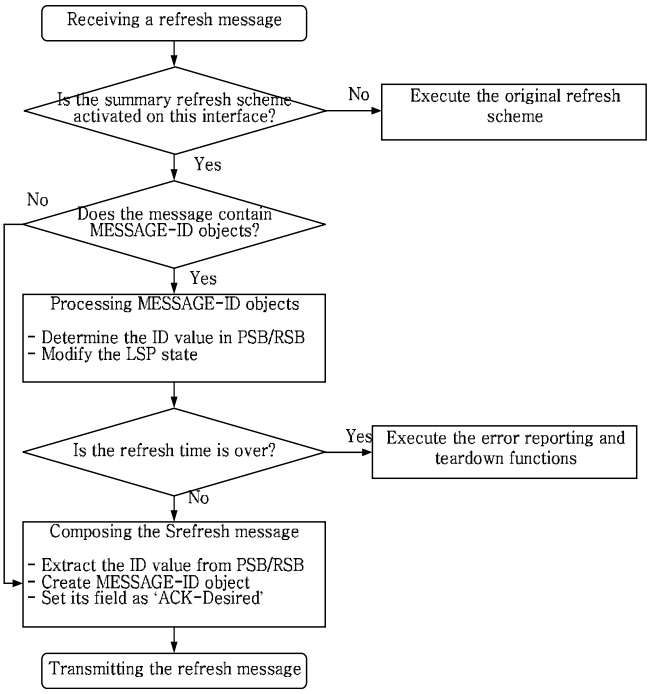

Fig. 2. Flow for processing the Srefresh message.

message, the procedure verifies the identifier value and modifies the state value in RSB/PSB table for the corresponding LSP state.

If there is no corresponding ACK to the sent message that includes 'ACK-Desired' flag of a MESSAGE-ID object, then the protocol engine retransmits the same message which is already sent. To achieve such a procedure, a transmitting router creates a ACK-Desired table, and checks whether the ACK of the message is replied or not. If no reply is received, then the protocol engine retransmits the same message within the configured time.

4. Performance evaluations

In this section, we describe simulation environment including some assumptions and simulation topology for evaluation of our implemented RSVP-TE protocol. And then, we analyze CPU processing overhead affected by signaling messages of the RSVP-TE protocol for maintaining LSP states in MPLS networks.

4.1. Simulation environments

For evaluating processing overhead affected by signaling messages of the RSVP-TE protocol, we trace the trend of CPU load as the number of LSPs increases. Signaling messages are generated by the RSVP-TE protocol for managing LSP states in MPLS networks. Such signaling messages gives a load to router's CPU for sending, receiving, and processing them.

In this evaluation, CPU's processing overhead is measured on sending and receiving the refresh messages and the summary refresh messages and processing them only after setting up LSPs. As the global values configured in the RSVP-TE protocol, refresh interval value is 30 s and refresh interval multiplier is 3 times. That is, LSP is assumed as disconnected after no refresh message is received in 90 s. The hello interval is configured 30 s in the implemented RSVP-TE with the Hello mechanism.

Fig. 3 shows our simulation topology to evaluate processing overhead of signaling messages in the RSVP-TE protocol. For simulation, there are six LERs and four LSRs in MPLS networks where they are connected by 100 Mbp Ethernet. All LER/LSR routers are implemented from Linux-based PC running only the RSVP-TE protocol. In our simulation, LSPs are set up through the target LSR, i.e., from LSR A, LSR B, and LSR D to LSR C, LSR E, and LSR F, respectively. We measure processing overhead in the target LSR as increasing a number of LSPs.

4.2. Effects of the summary refresh scheme

Fig. 4 shows the processing overhead for refresh messages in the RSVP-TE with legacy refresh scheme per CPUs with different processing power. LSPs connection timeout has occurred when LSPs are setting up over 24,000 in the LSR with 333 MHz CPU and over 36,000 in the LSR with 500 MHz CPU, respectively. In the LSR with 1400 MHz CPU, LSPs are still setting up more than 42,000 LSPs. In timeout case, the refresh message is not delivered to a neighbor LSR within 90 s due to increased processing time of refresh messages. As the result, Fig. 4 shows that the more CPU processing power has the LSR, the less CPU overhead for processing refresh messages occurs.

On the other hand, Fig. 5 shows the processing overhead of Srefresh messages in the RSVP-TE protocol with the summary refresh scheme per CPUs with different processing power. Similar to processing overhead for refresh messages in the RSVP-TE with the legacy refresh mechanism, the LSR's processing overhead is lessened as CPU processing power increases.

As the results from Figs. 4 and 5, the number of setup LSPs increases in the RSVP-TE with the summary refresh

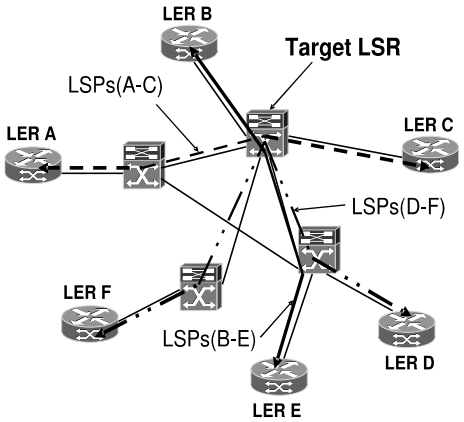

Fig. 3. Simulation topology of MPLS networks.

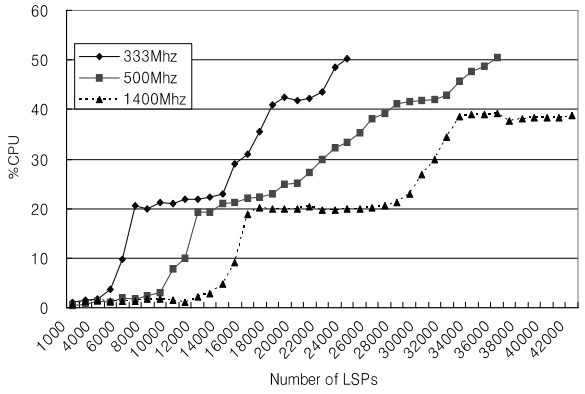

Fig. 4. CPU processing load in the legacy refresh scheme as increasing the number of setup LSPs.

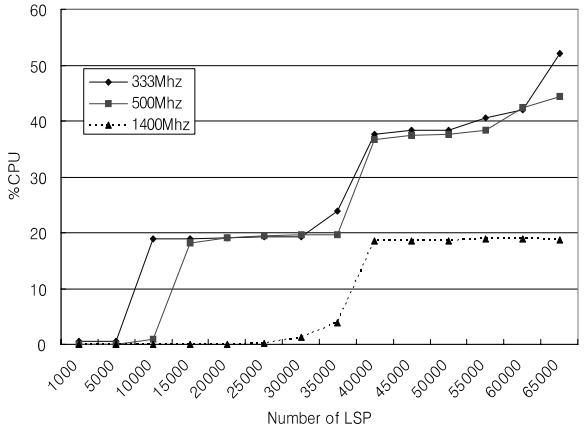

Fig. 5. CPU processing load in the summary refresh scheme as increasing the number of setup LSPs.

scheme, compared to the number of LSPs in the legacy refresh scheme. The RSVP-TE with the summary refresh mechanism reduces the processing overhead by reducing the number of sending and receiving messages. Even though the RSVP-TE protocol uses the summary refresh mechanism, the scalability problem still remains for large number of LSPs. This is the reason that the RSVP-TE protocol runs on LSR with another necessary protocols such as routing protocols.

4.3. Effects of bundling messages

The implemented RSVP-TE signaling protocol with the refresh reduction mechanism can be provided together with the message bundling technique that compacts multiple Message-IDs into one message.

Fig. 6 shows processing overhead of the RSVP-TE with the message bundling technique on 500 MHz CPU-based LSR router, where the number of encapsulated Message-IDs into one message is 10, 50, 100, 150, 200, 250, 300, and 350. As shown in Fig. 6, the number of encapsulated Message-IDs has little impact on processing overhead of refresh messages. Therefore, the processing overhead in the RSVP-TE protocol is greatly affected by the number

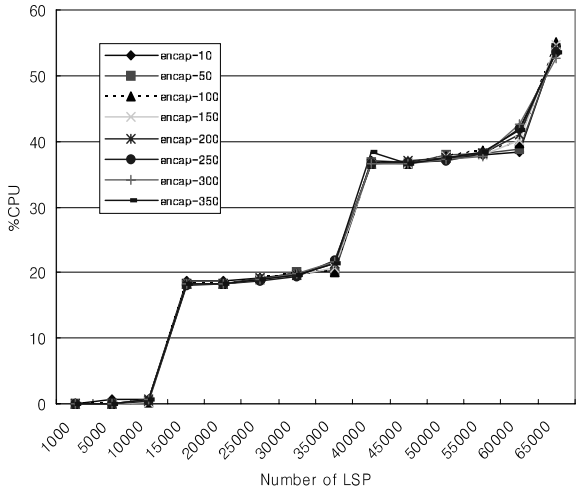

Fig. 6. CPU processing load in the summary refresh scheme with the bundling technique as increasing the number of setup LSPs.

of setup LSPs. That is the reason why the amount of refresh messages that should be processed is not reduced for managing LSP states.

4.4. Overhead of the adopted hello mechanism

Fig. 7 shows the processing overhead of Hello messages in the RSVP-TE with the adopted Hello mechanism proposed in this paper. The result comes from Linux-based router with CPU speed of 1400 MHz. The processing overhead has little relation with the number of setup LSPs in the adopted Hello mechanism differently from the refresh reduction scheme.

In addition to low CPU overhead, the RSVP-TE with the Hello mechanism supports detection of link/node failure and the message retransmission mechanism that supports a reliable message delivery. Therefore, the Hello mechanism would be better in terms of CPU processing overhead for implementing a lightweight RSVP-TE maintaining a large number of LSPs in MPLS network.

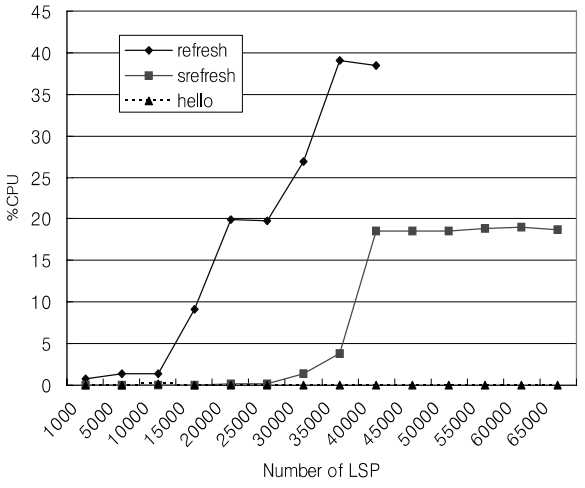

Fig. 7. CPU processing load in the lightweight implementation of RSVP-TE protocol as increasing the number of setup LSPs.

5. Conclusions

Traffic engineering techniques have been developed to guarantee the quality of services provided to the users on the Internet. Such a traffic engineering is applicable in MPLS networks providing Internet services. Although the RSVP-TE protocol for traffic engineering is consider as the signaling protocol that sets up the LSP with QoS in MPLS networks, it has drawback on scalability due to the soft-state mechanism. This paper evaluates the effect of the RSVP-TE protocol with the refresh scheme. From the results of evaluation, the refresh scheme increases the processing overhead to a router on periodic protocol messages. Also, the evaluation results show that even the RSVP-TE protocol implemented with the refresh reduction scheme does not decrease the processing overhead. This paper proposes the lightweight implementation of the RSVP-TE protocol by adopting the Hello mechanism. The implemented RSVP-TE can manage a large number of LSP states without increasing overhead of refresh messages.

References

[1] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, G. Swallow, RSVP-TE: extensions to RSVP for LSP tunnels, IETF RFC3209, 2001.
[2] D. Awduche, A. Chiu, A. Elwalid, L. Widjaja, X. Xiao, Overview and principles of internet traffic engineering, IETF RFC3722, 2002.
[3] L. Berger, D. Gan, G. Swallow, P. Pan, F. Tommasi, S. Molendini, RSVP refresh overhead reduction extensions, IETF RFC2961,2001.
[4] R. Braden, L. Zhang, S. Berson, S. Herzog, S. Jamin, Resource ReSerVation Protocol (RSVP) – Version 1 functional specification, IETF RFC2205, 1997.
[5] X. Fu, C. Kappler, Toward RSVP lite: light-weight RSVP for generic signaling, Proceedings of AINA (2003).
[6] M. McKusick, K. Bostic, M. Karels, J.S. Quarterman, The Design and Implementation of the 4.4BSD UNIX Operating System, Addison-Wesley Publishing Company, Reading, MA, 1996.
[7] M. Menth, A Scalable Protocol Architecture for End-to-End Signaling and Resource Reservation in IP Networks, Technical Reports, Dept. of Computer Science, University of Wurzburg, 2001.
[8] P. Pan, H. Schulzrinne, PF IPOPTION: A Kernel Extension for IP Option Packet Processing, Technical Memorandum 10009669-02TM, Bell Labs, 2000.
[9] P. Pan, H. Schulzrinne, An evaluation on RSVP transport mechanism, draft-pan-nsis-rsvp-transport-01.txt, IETF Internet Draft (2003).
[10] E. Rosen, A. Viswanathan, R. Callon, Multiprotocol label switching architecture, IETF RFC3031, 2000.
[11] F. Tommasi, S. Molendini, S. Zacchino, Measurements of the performance of the RSVP protocol, Proceedings of the Workshop on Architectures for Quality of Service in the Internet (2003) 24–25.
[12] A. Viswanathan, N. Feldman, Z. Wang, R. Callon, Evolution of multiprotocol label switching, IEEE Communications Magazine 5 (1998).
[13] H. Wang, R. Karri, M. Veeraraghavan, T. Li, A Hardware-Accelerated Implementation of the RSVP-TE Signaling Protocol, Proceedings of ICC, 2004.
[14] L. Zhang, S. Deering, D. Estrin, S. Shenker, D. Zappala, RSVP: a new resource ReSerVation protocol, IEEE Network 7 (1993) 8–18.

Young Woo Lee got both, his BEng (1984) and MEng degree (1990) in the Department of Electronics Engineering from Soongsil University and received his Ph.D. (2005) in the Department of Computer Engineering from Chungnam National University, Korea. He has developed some types of Network Management Systems (PcComm-NMS, ATM-NMS, WiBro-NMS and etc.) since he has been joining KT (Korea Telecom) R&D Group as a member of researcher in 1990. He is currently working on Wireless Network Research Division of KT Network Technology Laboratory as a principal researcher. His research interests include; wireless networks and protocols, mobility management, modeling and designing management systems, TE and QoS techniques and 4G communications paradigms.

Suncheul Kim received his B.S. in Statistics and his M.S. in computer science from Chungbuk National University, Korea, in 1995 and 2000, respectively. From 2000, he has been working with Broadband Converged Network Research Division in ETRI. His research interests are Internet Routing and Signaling Protocols, Router and Switch Architectures, and Wireless/Mobile/Ad-hoc Networks.

Jaehyung Park received his B.S. in computer science from Yonsei University, Korea, in 1991, and his M.S. and Ph.D. in computer science from Korea Advanced Institute of Science and Technology (KAIST), Korea, in 1993 and 1997, respectively. From 1997 to 1998, he was with the Center for Artificial Intelligence Research (CAIR) in KAIST. From 1998 to 2002, he was with the Network Laboratory in ETRI. Since 2002, he has been with the faculty of Chonnam National University, Korea, where he is currently an associate professor with the School of Electronics and Computer Engineering. His research interests are Internet Routing and Protocols, Router and Switch Architectures, Parallel Architecture and Algorithm, and Wireless/Mobile/Ad-hoc Networks.

Sang Ha Kim received Sang Ha Kim the B.S. degree in chemistry from Seoul National University, Seoul, Korea, in 1980, and he received the M.S. and Ph.D. degrees in quantum scattering and computer science from the University of Houston, Houston, TX, in 1984 and 1989, respectively.

He was with the Supercomputing Center, SERI, Korean Institute of Science and Technology (KIST) as a senior researcher between 1990 and 1991. He has joined Chungnam National University, Daejeon, Korea, since 1992, where he is a Professor. His current research interests include wireless networks, ad hoc networks, sensor networks, QoS, optical networks, and network analysis. Dr. Kim is Member of ACM, IEEE Communications Society, and IEEE Computer Society.