

Modeling Route Change in Soft-State Signaling Protocols Using SDL: A Case of RSVP

Constantin Werner, Xiaoming Fu, and Dieter Hogrefe

Telematics Group, University of Göttingen,
Lotzestrasse 16-18, D-37083 Göttingen, Germany
{werner, fu, hogrefe}@cs.uni-goettingen.de

Abstract. Soft state signaling protocols install and maintain states in network nodes, expiring without receiving refreshes. These states require proper reparation when the flow path changes, especially in case of link or node failures. As the specifications usually do not describe in detail how to handle these failures, we present insights by developing SDL models for RSVP on this issue.

1 Introduction

For the last decade, a group of protocols have been designed using soft state for state maintenance. In contrast to hard state, a soft state itself expires if no periodical refreshes are received. Soft state protocols are expected to have less protocol complexity in state maintenance operations especially with extreme network situations. However, as far as we know, rigorous investigations have rarely been performed on modeling these behaviors, especially for multi-hop soft state signaling protocols such as the Resource Reservation Protocol (RSVP) [1]. RSVP was the first soft state signaling protocol for Quality of Service (QoS) resource reservation developed by the Internet Engineering Task Force (IETF). RSVP specifications provide necessary message formats and processing rules for establishing and maintaining a state along a flow path. However, in common with most of the follow-up soft state signaling protocols, the RSVP specification does not describe in detail how a link failure is detected and circumvented.

This paper presents a formal model based on the Specification and Description Language (SDL) [7] of soft state signaling protocols. We investigate the RSVP protocol as a case study and particularly with respect to route changes. The model is built on a simplified IP layer model for RSVP message routing. Different from existing modeling approaches, our model allows an easy change of the analyzed network scenario without the need of any re-specification of the SDL router blocks. There is no centralized entity that responsible for routing, avoiding the necessity of re-specification for any new network topology. We show how the RSVP state recovery is verified and validated. We believe this modeling approach will be useful for the validation, modeling, and analysis of soft state protocols in general.

The remainder of this paper is organized as follows: in the following subsections, we summarize existing studies on soft state protocols, and give a short introduction to RSVP and the formal process. In section 2, we describe the network layer model used for IP routing and route re-establishment in case of link failures. In section 3, we show the formal analysis of the RSVP soft state maintenance in the normal case and in the case of link failures. In section 4, we discuss formal description techniques versus textual description used by the IETF. We outline where ambiguities arise through unclear descriptions in the RSVP standard. Finally, we discuss the conclusions and give an outlook of the further work for formal modeling soft state protocols in SDL.

1.1 Studies on Soft State Protocols

System designers argue soft state is “better” than hard state, and using soft state the handling of network condition changes is “easy” [9, 10]. However, these claims have been more based on intuitive, high-level thoughts and explanations, instead of formal, exhaustive modeling and analysis. In contrast to the original expectations, soft state protocols developed so far are still far from being simple, especially when coupled with channel reliability, multicast sessions or traffic control models. Soft state protocols developed so far can be categorized into two types: end-to-end protocols and hop-by-hop protocols.

The former only involves certain types of state in an end-to-end way, without bothering any other nodes in between; examples of this type include RTCP and SIP. On the other hand, hop-by-hop protocols (such as RSVP and NSIS [12]) involve state in one or more router(s) in between, in addition to state in the communicating ends. The latter is more representative and more comprehensively demonstrates the soft state operations, so we choose this as the example for general discussions of soft state. Given the particular importance of soft state protocols, there have recently been a few efforts on their modeling and analysis: Raman and McCanne [10] presented a model for the soft state notion based on Jackson queuing networks, and a performance study of hard state and soft state signaling protocols was performed by Ji et al. [8]. However, more detailed formal modeling and validation is still missing. A general formal soft state protocol analysis has been presented in [6] but a concrete analysis of an existing soft state protocol is missing as well.

1.2 Overview of RSVP

RSVP aims to provide end-to-end quality of service (QoS) signaling for application data streams. Hosts use RSVP to request a specific QoS from the network for a particular application flows. Routers use RSVP to deliver QoS requests to all routers along the data path. RSVP can also maintain and refresh states for a requested QoS application flow. RSVP carries QoS signaling messages through the network, visiting each node along the data path. If the reservation succeeds, the RSVP module sets parameters in a packet classifier and packet scheduler to obtain the desired QoS. The design of RSVP distinguished itself in a number

of fundamental ways, particularly: soft state management, two-pass signaling message exchanges, receiver-based resource reservation and separation of QoS signaling from routing [11]. Because the flow of delivery paths might change during the life of an application flow, RSVP takes a soft state approach in its design, creating and removing the protocol states (Path and Resv states) in routers and hosts. RSVP sends periodic refresh messages (Path and Resv) to maintain its states and to recover from occasional message loss. In the absence of refresh messages, the RSVP states automatically time out and are deleted. RSVP is not a routing protocol, but rather is designed to interoperate with current and future unicast and multicast routing protocols. While routing protocols are responsible for choosing the routes to use to forward packets, RSVP consults local routing tables to obtain routes and is responsible only for reservation setup along a data path.

1.3 Introducing Formal Process into the IETF Protocol Development

Traditionally, IETF protocols in the Request for Comments (RFC) documents are specified in a textual, informal format. A formal description using SDL of such a protocol can help to clearly and unambiguously specify the functional operation, because it allows easier detection of protocol anomalies or design errors such as deadlock or livelock situations. Previous studies such as [2, 4, 5] presented analysis and validation of several IETF protocols using formal description techniques. However, their analyses were limited to a single or only very few fixed use cases and applied only to protocols operating in an end-to-end fashion or using hard state in principle. None of them investigated any soft state signaling protocol, nor considered randomly chosen link failures. We argue it is important to guarantee the proper protocol operations in dynamic environments, especially that soft state signaling protocols are error-free and also precisely presented for

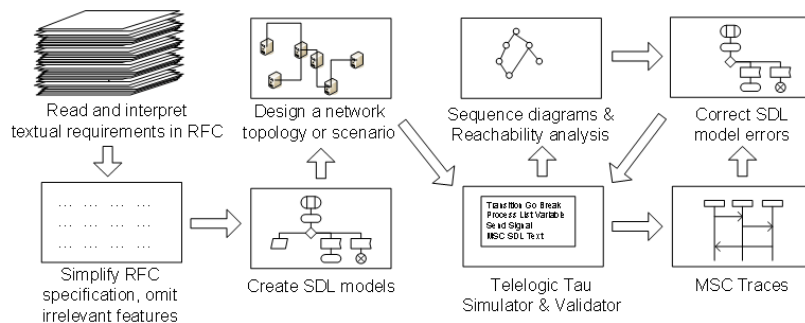


Fig. 1. The formal process. After the interpretation of an RFC, the SDL models are specified and a network scenario is created. An integrated SDL development tool, *Telelogic Tau 4.6*, is used for the formal verification and validation of the created model

the correctness of implementations. In this paper, we show a modeling approach that proves that despite the interactions between the possibly dynamic chain of intermediate hops and random link failures, the correctness and robustness in soft state protocols can still be proven by way of formal description and validation.

Figure 1 shows the formal process flow which we used in our modeling, starting with reading and interpreting the RFC. RSVP features which do not relate to route change detection and recovery were chosen to be omitted to reduce complexity. A network topology was created which is assumed to be sufficiently complex to show the route re-establishment functionality of RSVP. Due to the decentralized IP network layer architecture, additional network topologies can be created and analyzed without the need to re-specify the SDL models.

2 SDL Modeling of Message Routing in IP Networks

To the best of our knowledge, formal models focused on IP based networks developed so far either model end-to-end protocols (which are formally specified for their special purpose), or simply three entities are assumed: a sender, a recipient, and a general transport block as a centralized entity for routing, forwarding and packet loss modeling (see [2, 4, 5]). These approaches have the disadvantage that for each new network topology the central routing entity or intermediate nodes have to be re-specified and adapted to the new network configuration. Additionally, link failures are hard to emulate. Multi-hop protocols with route failures cannot be modeled using a centralized entity or fixed formalized nodes for message transport.

We propose a formal decentralized network layer architecture, which automatically learns its neighboring entities and reachable destinations by itself. However, modeling of IP based communication protocols in dynamic network topologies can suffer from some SDL shortcomings and limitations of Telelogic Tau. SDL does not offer a dynamic number of channels connected to a block, therefore, our router models have a fixed amount of three channels (network links) available. Furthermore, SDL does not provide native support of IP addresses, so we use SDL process IDs (*Pid*) for addressing of nodes in the topology instead. We believe that this is no drawback of our model if small network topologies without the need for special routing are required. The signal *myPID* is used to announce the destination node's address (process IDs) to other nodes. In reality, this is defined by the user or user's application.

Our routing algorithm is inspired by *Distance Vector Routing* protocols like the *Routing Information Protocol* (RIP) [3]. To reduce complexity, the periodically broadcasted distance vector updates are replaced by signals that trigger distance vector updates between neighboring routers. This feature is especially useful in not being confused by minor relevant network layer messages if the upper layer's soft state protocol messages are to be analyzed and validated. Furthermore, this is required for formal analysis using the Telelogic Tau Validator. The Validator does not include signals from the environment if any transitions

are still scheduled. While the routing tables are updating, the system converges to a stable state. Some more enhancements and simplification have been undertaken to bypass known Distance Vector Routing problems (such as the *count-to-infinity* problem [13]). We do not discuss them here in detail, since this is not the scope of this paper. Note that our model of RSVP is intentionally not bound to any specific IP routing protocol, so the use of a modified routing protocol here does not violate any RSVP requirement.

The IP routing layer is modeled as a block consisting of a *forwarding* and a *routing* block. The basic operational principle is the following: The *forwarder* receives *datagrams*, which is an SDL structure consisting of the variables *Source* (sender of this packet), *Destination* (destination for this packet), *Phop* (previous hop), and a payload *msg* from the upper layer – RSVP messages in this case. If the *forwarder* receives a datagram, it queries the *routing* block for the address of the next hop and forwards the *datagram* to this hop. Routing table updates

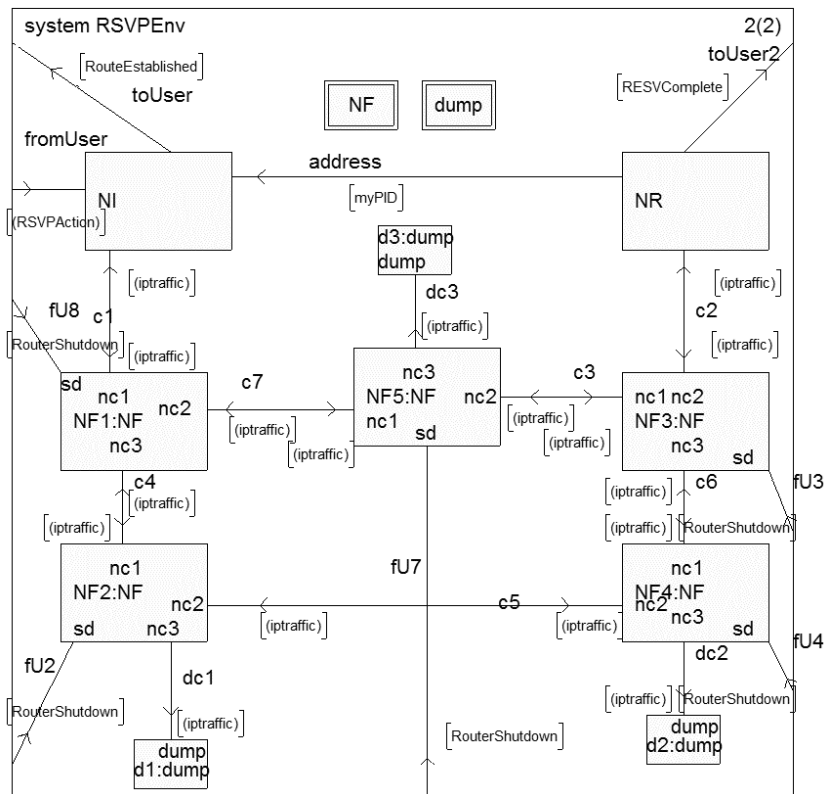


Fig. 2. The network scenario model generated in *Telelogic Tau 4.6*. The message flow used in this scenario is from NI down to NR via several NFs and vice versa. Note that the shortest route between NI and NR is via NF1, NF5, and NF3. After a possible shutdown of NF5, an alternative route is established via NF1, NF2, NF4, and NF3

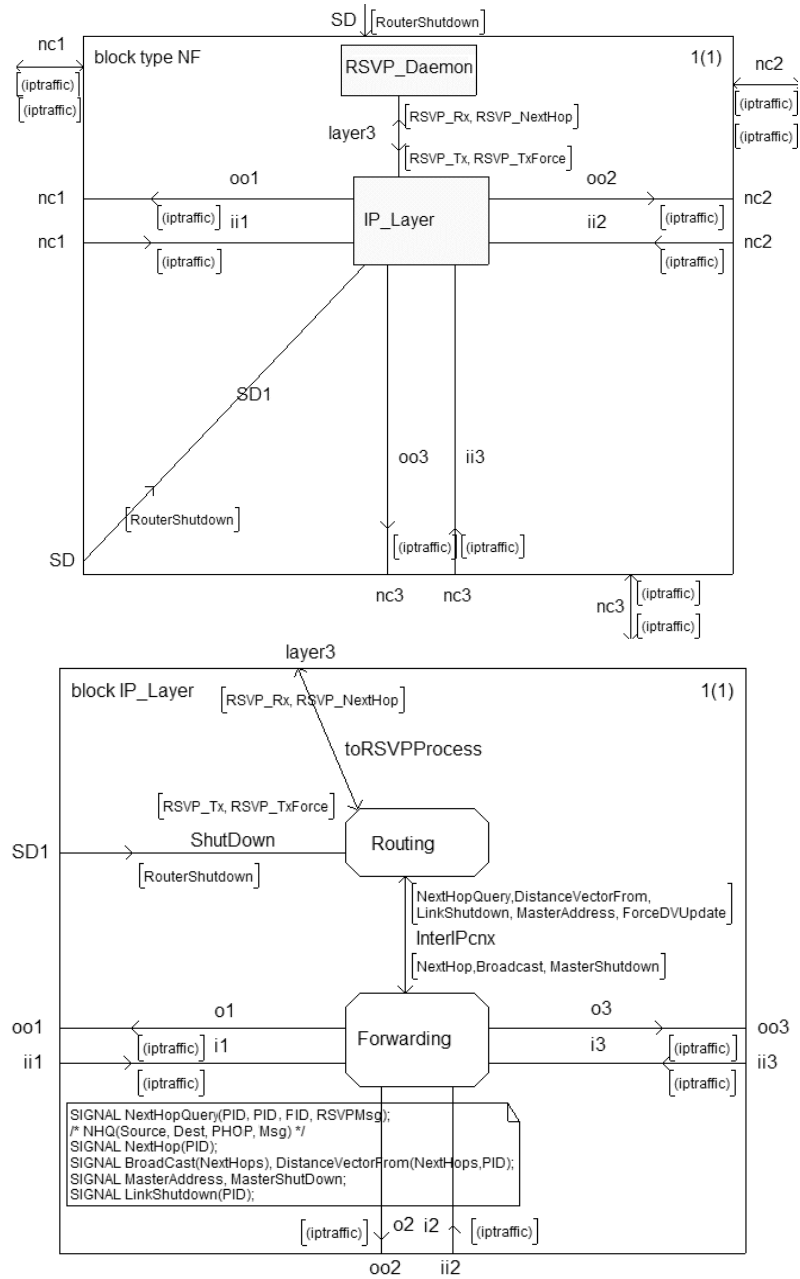


Fig. 3. The internal network structure block of all NF nodes. The NF block consists of a layered block structure which is the IP layer with routing functionality and a higher layer which is the RSVP daemon here. The *forwarding* block is responsible for the receiving and forwarding datagrams. The *routing* block selects the next hop for a received datagram and maintains its local routing table. RSVP messages are sent to the RSVP daemon block by the *routing* block if received

are received by a special signal *DistanceVector* containing the routing table of the neighbor’s routing layer. This information is used by the *routing* block to update its local routing table.

Figure 2 shows the SDL system for the described network topology. The investigated scenario consists of one *NI* (network initiator), multiple *NFs* (network forwarder with routing functionality) and one *NR* (network receiver). The NI is the entity, which generates RSVP messages and tries to establish a reservation state along the path from NI via multiple NFs down to the receiving NR. Every single hop on the path establishes a requested RSVP state. All NF nodes have three connectors available for creating a network scenario. Unconnected signal channels have to be connected to *dump* blocks that silently consume all signals they receive.

All routing layers feature an external *Shutdown* signal from the environment, which allows the user to shutdown any or multiple instances freely. If a routing layer is triggered by such a *Shutdown* signal, it announces shutdown by sending a *LinkFailure* signal to all its neighbors. Note that this is one modification to Distance Vector routing protocols, which detect a node failure by the absence of the failed node’s routing table updates. Because periodic routing table update messages add avoidable communication complexity to the scenario, the *LinkFailure* signal has been introduced.

All neighboring hops are now trying to update their routing tables with new routing information and request table updates from their neighbors as well. The routing layer, once being shutdown, is no longer operational from now on and only consumes each signal or message silently which it receives. The whole entity cannot operate any more. This allows the analysis if the soft state timing is able to maintain its state even if refresh messages are lost at the non-operational hop until the new route is established. See fig. 3 for an overview of the IP routing and RSVP block.

3 Formal Analysis of RSVP State Maintenance with Link Failure

We analyzed with our models and Telelogic Tau how RSVP can restore a valid path after a link failure. When the simulation was started, the system announces via a special signal that it is ready for operation and all routing tables are built up to allow a complete routing between all nodes. The NI accepts three different signal triggers from the environment: *RSVPStart*, *RSVPTeardown*, and *RSVPStop*. *RSVPStart* begins creating a path state and resource reservation along the path down to the NR. The NI periodically sends new path messages to keep the RSVP soft state alive. The *RSVPTeardown* signal triggers the NI to stop sending refresh messages and to send a *PathTeardown* message towards the NR. All nodes in-between delete the associated states from this reservation and forward the teardown message to the next hop (*Explicit Teardown*). The *RSVPStop* signal just stops the NI from sending new state refresh messages

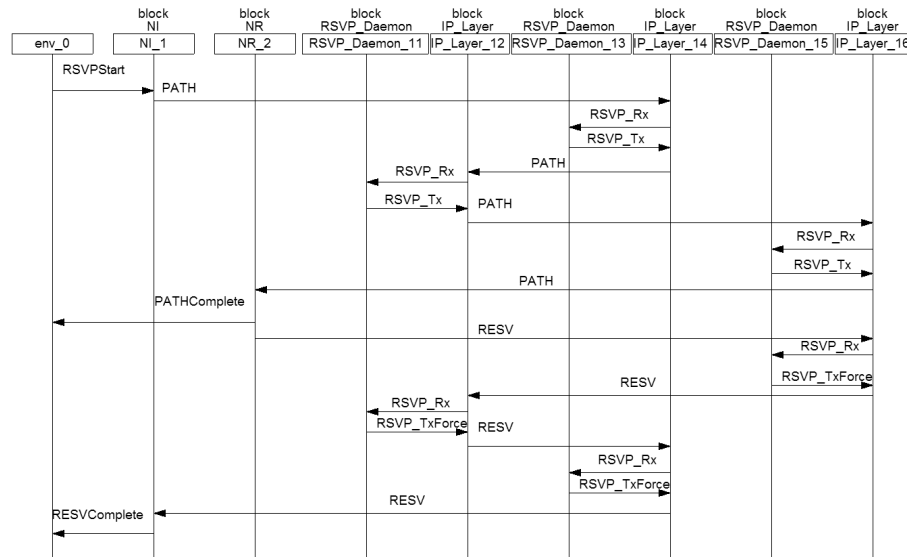


Fig. 4. Message flow of a RSVP resource reservation. The *Path* message is sent downstream from the NI hop-by-hop to the NR. The corresponding *Resv* message is sent upstream from the NR hop-by-hop to the NI. Message parameters and process states are not shown for clarity

towards the NR. This leads to a state timeout at all hops and the states are deleted after the state lifetime expiry.

In fig. 4 a default RSVP *Path* and *Resv* message refresh flow is shown. The message exchange in this MSC is shown from the NI via N1, N5, and N3 down to the NR. This is the shortest path. Notice the internal message exchange between the IP Layer and the RSVP instances. The RSVP daemon is notified of the reception of RSVP messages by the *RSVP_Rx* signal and itself sends a RSVP message using the *RSVP_Tx* or *RSVP_TxForce* signal. While the *RSVP_Tx* signal allows the IP layer to select the next hop, the *RSVP_TxForce* explicitly addresses the next hop. The Telelogic Tau SDL simulator is being used to trace the correct establishment of *RSVP_Established* states in all RSVP intermediate hops and the NR.

Next, a router shutdown is triggered. In this scenario, NF5 is selected as the failure hop. By doing this, an alternative route has to be established by the IP routing layer. After NF5 has been shutdown, it announces its imminent death by sending a *LinkFailure* signal to all its neighbors. Because of this signal, they try to update their local routing tables with their neighbors as well. See the following fig. 5 for an MSC which shows the message exchange in case of the NF5 shutdown. While the routers try to update their information, a newly created RSVP message is lost while being routed, visible at the signal marked with the dotted arrow.

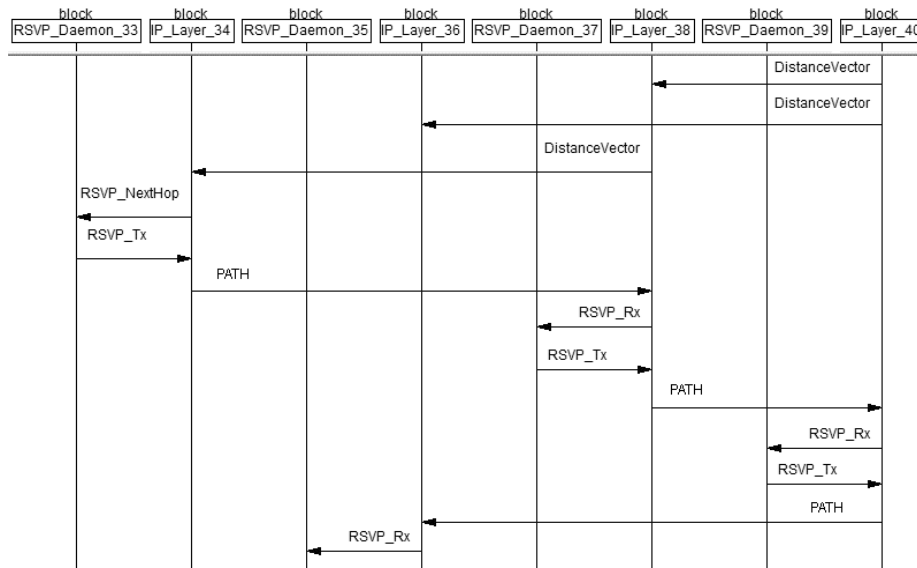


Fig. 7. A cut-out of the new route establishment using *Local repair*. This MSC is another time continuation of the MSC shown in fig. 5. Some routing table update messages and inactive instances are skipped. The new RSVP message (*Path*) is delivered via the alternative route NF1, NF2, NF4, NF3 down to the NR. Note that a *Path* message is sent triggered by the routing table update by receiving the *DistanceVector* signal. The RSVP process gets notified of the route change by the *RSVP_NextHop* signal

route. The SDL simulator confirms that all new hops are able to establish a correct *RSVP_Established* state. The RSVP soft state operation continues with correct behavior. This is caused by the detection of the route change if the previous hop of the new RSVP message differs from the one which has been recorded on previous RSVP messages. The same detection applies on changes of the next hop which is decided by the IP routing layer. This operation has been validated using the built-in Validator of Telelogic Tau using exhaustive state space exploration.

Note that our RSVP model does not include RSVP features like multicast, and the admission and policy modules, since these are not particularly interesting for route re-establishment. RSVP multicast adds a high level of complexity to the protocol design and multicast support (actually one of its succeeding IETF efforts, NSIS, has decided to remove multicast from basic signaling support), thus it is not considered here. Therefore, all multicast related operations like merging and styles processing are not considered. *Local repair* has been implemented, which improves route recovery by immediately sending *Path* and *Resv* messages towards the previous and next hop if a route change is detected. Figure 7 shows an MSC with local repair action triggered by routing table updates.

4 Formal Description Techniques Versus Textual Description

The RSVP standard is not very clear in its route change detection specification and handling. The *Local Repair* section of the RSVP document describes how route changes are detected by evaluating the previous hop field and being informed by the routing layer of the outgoing interface selection. Unfortunately, a precise formalism of how route changes are detected and handled is not present.

As an example, it is stated in the *Local Repair* section of the RSVP specification: “[...] To provide fast adaptation to routing changes without the overhead of short refresh periods, the local routing protocol module can notify the RSVP process of route changes for particular destinations. The RSVP process should use this information to trigger a quick refresh of state for these destinations, using the new route. When routing detects a change of the set of outgoing interfaces for destination G , RSVP should update the path state, wait for a short period W , and then send Path refreshes for all sessions. [...] When a Path message arrives with a Previous Hop address that differs from the one stored in the path state, RSVP should send immediate Resv refreshes to that PHOP.”

From this description it is unclear for the reader whether the routing layer is responsible for the route change detection (and just notifies to the RSVP process) or the RSVP process has to query the routing module for each RSVP packet transmission. Additionally, the description about the case of *previous hop detection* does not explicitly state how RSVP handles this case. That is, detailed interactions are missing in the specification. How does the routing protocol know that there is a routing change? Does it save the next hop node for all outgoing RSVP messages and processes or does it notify the RSVP process of the outgoing interface or next hop which was taken for the most recent RSVP message and it is up to the RSVP process to detect that the next hop has changed? When does an action have to be taken? The same thing applies for the previous hop detection.

Besides, how the RSVP process is able to instruct the routing layer to send a message hop-by-hop using the previous hop is not specified. The RSVP specification gives information saying in the *RSVP messages* section: “[...] These Path messages store ‘path state’ in each node along the way. This path state includes at least the unicast IP address of the previous hop node, which is used to route the Resv messages hop-by-hop in the reverse direction. [...]”. We have to make the assumption that RSVP looks up the previous hop information in the path state according to a matching between the flow identifier (however, this can be some other metric) and the path state, and then sets the IP header of the Resv message as the previous hop’s IP address. However, due to the ambiguity of the specification, even ignoring the looking up metric, one can also interpret the Resv message to be encapsulated with a routing header where destination addresses are the flow sender’s address (inner IP header destination field) and previous hop’s IP address (direct IP header destination field). We have developed a concrete communication model which allows the RSVP process to explicitly

specify the outgoing hop. See fig. 3 for an overview of this model. Our communication model allows the RSVP process to receive the RSVP message received by a previous hop, and to transmit a message implicitly by specifying the destination but the next hop decision is up to the routing layer. The RSVP process is able to get a notification of the next hop address which has been selected by the routing layer when routing table updates have occurred. Additionally, the RSVP process is able to explicitly specify the next hop which is required to send RESV messages upstream on the same path.

We have shown a precise, formalized way how the RSVP process can detect and handle such extreme situations. We modeled the previous hop and next hop change detection in the RSVP process so that there is a clearly defined interaction needed between the RSVP process and the routing layer. The RSVP process gets a notification of the next hop address via the *RSVP_Nexthop* signal shown in fig. 3 chosen by the routing layer. It can compare this with the stored next hop address from previous RSVP messages. The RSVP process detects previous hop changes by evaluating the previous hop field in the RSVP message. The *RSVP_Nexthop* signal may be send asynchronously, which means that the signal can be transmitted at any time from the routing layer to the RSVP process.

In our model and particularly with this investigated scenario, this approach was able to build up the reservation state in the alternative route and to maintain the state in hops which were already involved in the prior resource reservation.

5 Conclusions and Future Work

It is often argued that soft state protocols are “better” but these claims have been more based on intuitive, high-level thoughts and explanations, instead of formal, exhaustive modeling and analysis. This calls for a detailed formal modeling and validation. A general formal soft state protocol analysis has recently been done but a concrete analysis of an existing soft state protocol is still missing.

In this paper, we presented an SDL model of a concrete, existing soft state protocol, namely the RSVP resource reservation protocol developed by the IETF. Because this protocol relies on IP routing, an IP routing layer was also implemented in SDL which performs the message routing. We focused on the modeling of a typical soft state operation caused by a link failure and the RSVP route re-establishment through an alternative route. The studied scenarios consisted of one initiator, one responder and a dynamic amount of intermediate nodes. In this paper, we only considered one single scenario where the shortest route between the initiator and responder is broken by a link failure and an alternative route is established. We showed the message exchange in normal operation, with local repair and the route and state recovery after a link failure has occurred in MSC diagrams. RSVP operations especially under route change situations were simulated and validated using the tool Telelogic Tau 4.6.

The RSVP standard lacks a detailed and formalized description of how route changes are detected and handled. Although a description can be found about this in the RFC document, it is unclear which process is responsible and how it

is achieved in detail. Our model proposes a way to detect a route change and handling within the RSVP process. We have shown that a formal description and analysis of an existing soft state protocol and its state maintenance operations under various network conditions is possible using formal description techniques.

The RSVP model used in this paper was simplified. Some features of RSVP were left out: multicast support, detailed presentations for reservation flows and sessions, admission control and policy control. Though most of these RSVP features are not relevant for this operation and link failure analysis, the actual flow and session presentations, and admission control are important towards a more realistic modeling of RSVP. These features are not irrelevant in route change modeling because some new reservations may be rejected on alternative paths which we did not consider here. In future versions of this model, we want to integrate these modules into our SDL model of RSVP. Additionally, we want to analyze multiple sessions between multiple initiators and responders. Currently, we support multiple intermediate hops but only with a single initiator and responder, which create a resource reservation along the path hop-by-hop.

References

1. Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S.: Resource ReSerVation Protocol (RSVP) – Version 1 functional specification. RFC 2205, IETF (1997)
2. Monkewich, O., Sales, I., Probert, R.: OSPF Efficient LSA Refreshment Function in SDL. In: Reed, R., Reed, J. (eds.): SDL 2001, Lecture Notes in Computer Science, Vol. 2078, Springer-Verlag, Berlin Heidelberg New York (2001) 300–315
3. Malkin, G.: RIP Version 2 – Carrying Additional Information. RFC 1723, IETF (1994)
4. Chan, K. Y., v. Bochmann, G.: Modeling IETF Session Initiation Protocol and its services in SDL. In: Reed, R., (ed.): SDL 2003, Lecture Notes in Computer Science, Vol. 2708, Springer-Verlag, Berlin Heidelberg (2003) 352–373
5. Cavalli, A., Grepert, C., Maag, S., and Tortajada, V.: A Validation Model for the DSR protocol. ICDCS 2004 (2004)
6. Fu, X., and Hogrefe, D.: Modeling Soft State Protocols with SDL. To appear in: Proceedings of IFIP International Conference on Networking, Waterloo, Canada (2005)
7. Ellsberger, J., Hogrefe, D., and Sarma, A.: SDL – Object Oriented Language for Communication Systems. Prentice Hall (1997)
8. Ji, P., Ge, Z., Kurose, j., and Towsley, D.: A comparison of hard-state and soft-state signaling protocols. In: Proc. of SIGCOMM 2003, Karlsruhe, Germany (2003)
9. Sharma, P., Estrin, D., Floyd, S., and Jacobson, V.: Scalable timers for soft state protocols. In: INFOCOM'97, Kobe, Japan (1997)
10. Raman, S. and McCanne, S.: A model, analysis, and protocol framework for soft state-based communication. In: Proc. of SIGCOMM 1999, Cambridge, MA (1999)
11. Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D.: RSVP: a New Resource Reservation Protocol. IEEE Network (1993).
12. Hancock, R., Karagiannis, G., Loughney, J., and v. d. Bosch, S.: Next Steps in Signaling: Framework. Internet draft, work in progress, IETF (2004)
13. Tanenbaum, A.S.: Computer Networks. 4th Edition, Prentice Hall (2002)