Fast Source Routed Connection Setup - Proposal for a RSVP Implementation

Florin Josef Lataretu and Corneliu Toma

***Abstract* — Modern networks are expecting fast and efficient setup procedures for their connections. Strict time limits are requested for signaling in the normal operational mode and also for the recovery/restoration mode. As a possible solution, I proposed a method for fast source routed connection setup that takes advantage of the existing distributed processing potential in order to minimize the dependency of setup time on the path length. Two alternatives for the parallel setup are presented: sequential synchronization in the intermediate nodes and final synchronization in the egress node. The expected time reduction is considerable. The proposal is in general terms, using abstract messages instead of a specific signaling protocol. In the meantime, RSVP is established as the state of the art signaling protocol. The research community is still looking for ways to increase its speed and efficiency. In this context, I'm proposing an implementation of the mentioned method based on the current RSVP standard and I'm comparing it with a recent similar proposal.**

***Keywords* — component, GMPLS signaling, RSVP, Source routed connection setup.**

I. Introduction

AUTOMATICALLY Switched Optical Networks (ASON) use connections as a central concept that supports the user to transport its data between an ingress and an egress node [1]. Thus a connection path through the network is based on a sequence of nodes starting with a begin node (ingress, "A" node) and terminating at the end node (egress, "Z" node). One or more intermediate nodes ("I" node) are traversed. In the traditional source routed connection establishment, the setup procedure is strictly sequential. A setup request message is generated at the starting A node and transmitted from there to a first intermediate node in a sequence of nodes - well known at the A node - that will be traversed by the connection. The A node processes the setup request (Path message in RSVP implementation) in the time PpA and after processing, forwards the message to the first intermediate node I1. The transmission of this message takes some duration TpI1. The first intermediate node processes the setup request message received from the begin node (PpI1) and then forwards the setup request message to the next intermediate node in the sequence (TpI2), and so on. This process continues until the final intermediate node in the sequence transmits the connection request message to the end node in the sequence.

Florin Josef Lataretu (corresponding author) is with the Optics Division, Alcatel-Lucent Deutschland AG, Nuernberg; Germany (e-mail: florin.lataretu@alcatel-lucent.com).

Corneliu Toma is with the "Politehnica" University of Timisoara, Timisoara, Romania.

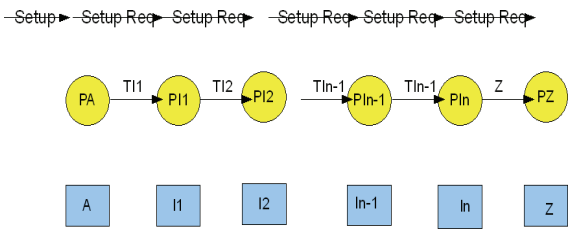

Fig. 1. Sequential Setup Request.

The total time to perform a setup request is:

$$SetupTP_{seq} = PpA + \sum_{i=1}^{N}(TpIi + PpIi) + TpZ + PpZ \quad (1)$$

where i=1...N are the intermediate nodes to the nodes A, Z.

The same sequential procedure applies for the Setup Response, which is also traversing hop by hop the intermediate nodes up to the begin node consuming for the transmission a certain time (*TrIi*). On each hop a specific processing time (*PrIi*) is needed. The total time to perform a setup response is:

$$SetupTR_{seq} = PrA + \sum_{i=1}^{N}(TrIi + PrIi) + TrZ + PrZ\ . \quad (2)$$

Notice that depending on the implementation some of the time consuming activities (e.g. different validation checks, resource reservations or allocations which require the interaction with a different controller) may be performed either on the Setup Request or on the Setup Response. In the following the generic term SetupTime is used for both the Request and the Response part of the procedure and time calculation. However, it should be noticed that for the total calculation the time consumed as well as the time saved with the alternatives is to be accumulated. As the size of communication networks continues to grow, the connection paths provisioned through a network tend to traverse an increasing number of nodes, also significantly increasing the time required to establish a connection path through the network. So, the problem is that some critical time restrictions cannot be fulfilled if the path exceeds a certain length. Therefore, a faster method of performing source routed connection path provisioning is desirable.

II. Basic Idea of the Parallel Setup Method

In particular because of the tied requirements on restoration times, I proposed the method of parallel setup, which was registered 2006 as patent appliance [2]. In a parallel connection setup, the A node sends respective connection request messages to each of the intermediate

nodes elected to form a connection path. In the context of source routing it is assumed that the sequence of nodes to be traversed (A, Ii, Z) is known. In the RSVP protocol this sequence is actually included in the Path message as part of the Explicit Route Object (ERO). Two alternatives are presented: Sequential synchronization on intermediate nodes and final synchronization in the end node.

A. *Sequential Synchronization*

In the sequential synchronization, each intermediate node operates as a partial synchronization point for a previous intermediate node in the sequence of nodes scheduled to form a connection path. Each intermediate node transmits a Forward Acknowledge ($FwdAck_I$) message to a next intermediate node (Ii) - compare Fig. 2. The end node decides on completion as soon as the corresponding FwdAck message was received from its previous node I_N. In general the setup message reaches the node Z (after *T'z*) before the $FwdAck_Z$ from its upstream node. Therefore also node Z can perform its processing already before it received the $FwdAck_Z$. Processing time *PIi* is in general greater than the transmission time *T'Ii*, since it also covers some resource allocations, usually related to additional communication with different controllers in the node. In particular, in overload situations it is rather the processing time which is affected. So, the usual temporal sequence of events is *PpA*, $FwdAck_2$, $FwdAck_i$,…, $FwdAck_Z$. The SetupTime for the sequential synchronization (assuming for simplicity equal transmission times for the Forward Acknowledgement) is:

$$SetupT_{seqSync} = PpA + N * FwdAckI_1 . \quad (3)$$

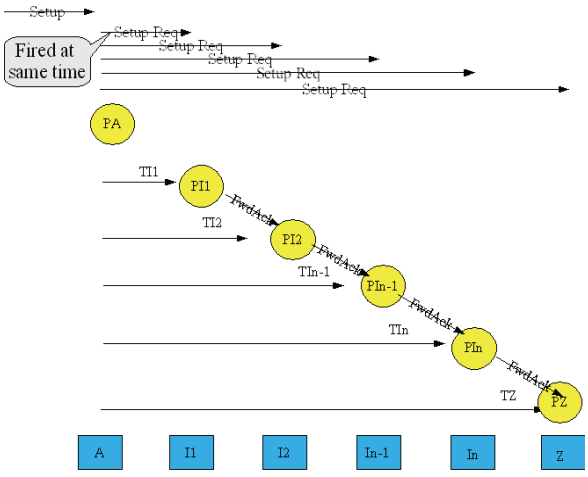

Fig. 2. Parallel Setup with sequential synchronization.

The direct transmission time *T'* is slightly increasing towards Z, therefore *T'i*+1 > *T'i*. However it remains comparable with *FwdAckI,* the transmission time of the ForwardAcknowledge. So, the difference between the traditional and the parallel setup with sequential synchronization is:

$$SetupT_{seq} - SetupT_{seqSync} \geq \sum_{i=1}^{N} PpI_i + PpZ . \quad (4)$$

In other words the benefit of the sequential synchronization is the result of the enforced parallelization in the intermediate and final nodes, saving the processing time in these nodes. The estimated result was validated by simulation.

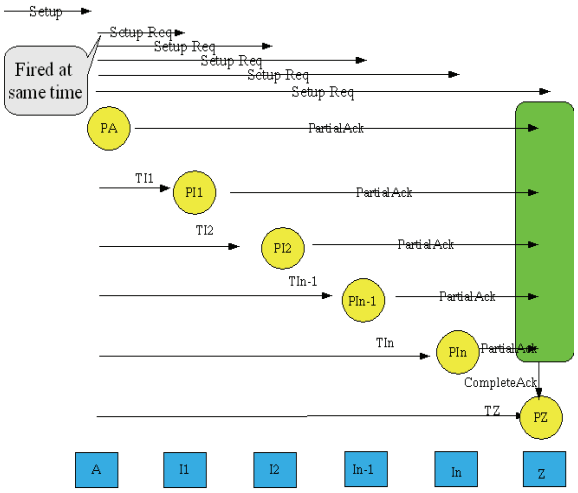

Fig. 3. Parallel Setup with final synchronization.

B. *Final Synchronization*

This option may be used if there is no need for intermediate synchronization points. Again the begin node A sends Setup Requests to all intermediate nodes as well to the end node. The intermediate nodes send Partial Acknowledge (*PartAck*) as soon as they complete their processing. The end node operates as a final synchronization point for each of the intermediate nodes in the sequence of nodes elected to be traversed by the connection – comp. Fig. 3. It decides on completion as soon as all Partial Acknowledge messages have been received. The setup time is at worst the sum of the maximum of the transmission time, the maximum processing time and the maximum transmission time for the *PartialAck* message. Again, compared with the sequential setup the parallel setup with final synchronization saves N processing times *Pi.* In the best case in a completely meshed topology each node is directly accessible, so all transmission times for Setup Request and *PartialAck* are the same. So the best case setup time for final synchronization is:

$$SetupT_{FinSynchBestCase} = PZ + T'I_Z . \quad (5)$$

The reduction of the Setup Time compared to the traditional sequential Setup is in the following range:

$$N * PI_1 \leq ReductionSetup < N * PI_i + N * TI_i . \quad (6)$$

The estimated result was validated by simulation.

III. RSVP Implementation of the Parallel Setup

Whereas it is obvious that the abstract Setup Request translates in the RSVP context to the *Path* message and that the Setup Response translates to the *Resv* message, there are no specific RSVP messages for the abstract *FwdAck* and *PartialAck*. The first approach would be to implement them as RSVP Notify message, which is different from the common RSVP messages in that it is "targeted" to another node than the immediate up- or downstream neighbor. This would fit for instance for the

PartialAck, which is directly addressed to the final node. However this solution would violate RFC3473 [4] which states that a notify message should be sent to a node only if it was explicitly requested by a corresponding notify request.

A. Sequential Synchronisation

Looking closer to the specific activities behind the sequential synchronization, it turns out that the Setup Req messages, which are fired simultaneously to all nodes implied in the path are essentially a kind of preliminary – let us call them "*prePath"*- messages, suitable for some preparing operations like validation checks. Instead, the abstract FwdAck message could be directly implemented as regular Path traveling the LSP hop-by-hop in the traditional way. The *prePath* message is to be sent to the end node and to all intermediate nodes, except the first intermediate node, which is informed by a regular Path. For the *prePath* message, it would not be necessary to set the complete Explicit Routing Object (ERO), since these messages end in the intermediate nodes. However, the presence of the ERO may provide additional information about the expected resource allocation.

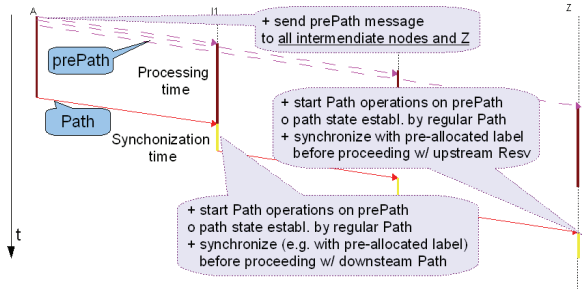

Fig. 4. Sequential Synchronization: Modified behavior.

The *prePath* message has the same content as the traditional Path, except that it is addressed directly to an intermediate node via a corresponding IP destination address. In order to avoid confusions with the regular Path, the MessageID is omitted, which in turn prevents the A node from receiving a bulk of acknowledges from every intermediate node. The RSVP_HOP can be set to the A node, thus giving an implicit indication (by comparing with the ERO object) for the special treatment. The explicit indication of the special semantic of the prePath is possible by introducing a new bit: "PREliminary" to the ADMIN_STATUS object, in addition to the existing R, T, A, D bits specified by RFC3473 [4]. This would be the natural way, however if there are objections because the reserved part of this object, the session name as part of the Session_Attribute may be used instead. The procedure of a node receiving the *prePath* message is as follows: The RSVP Path state is established as normal. In addition preliminary checks may be performed on the related resources and on a positive result they may be reserved (e.g. time slots, database records). The actual allocation as well as related database commitment may occur after the regular Path message is received from the previous intermediate node. If no regular Path is received in the following refreshing time, the Path state and the related pre-reserved resources are freed. This behavior is actually already given by the traditional RSVP soft state behavior. The association of the regular Path with the prePath message is by using the same TunnelID in the Session object. Notice the path setup shows now the characteristic of a two phased transaction: preparation phase followed by the commit or a roll-back phase. In general most of the initiated Path Setups will be probably successful since they have been planned and initiated on the A node based on the initial availability of the related resources. However, if some conditions changed in the mean time the roll-back is implicitly supported by the regular RSVP procedure with the soft state approach. In addition, the A node will initiate the Path Teardown in case of an outstanding Resv message. Similar considerations can be made for the setup Response, which may use a special preliminary Resv – let's call it "preResv" message, possibly triggering some preliminary label operations. PreResv is directly IP addressed to all intermediate nodes and to the A node, whereas the abstract *FwdAck* message is implemented as a regular Resv message, traveling the LSP hop-by-hop in the traditional way.

B. Final Synchronisation

The final synchronization is in particular interesting for such cases where the intermediate nodes are restricting some of the setup conditions (e.g. suggested label set, resource affinity) potentially leading to a negative result (PathError), which can be concluded more faster. Also for the positive case, the setup time is reduced since the Z node can proceed with the Resv rigth after receiving all *PartialAcks,* even before the regular Path was received. The *PartialAkcs* can be implemented as Path with some modifications: It is IP addressed directly to the egress node, without the Message_ID.

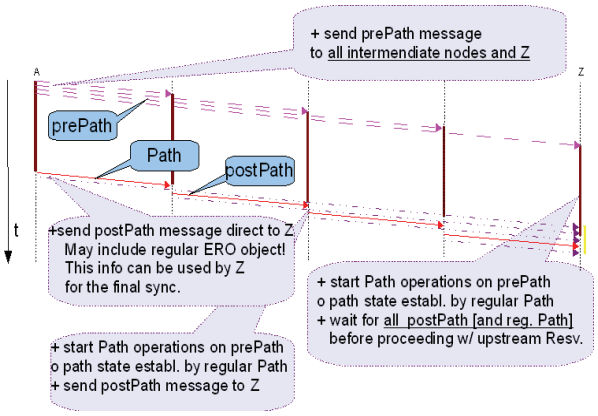

Fig. 5. Final Synchronization: Modified behavior.

Its special semantics could be indicated indirectly e.g. by comparing RSVP_HOP, ERO, and Record Route Object (RRO). The preferred option is the explicit one by setting a new ADMIN_STATUS.PostBit. Alternatively the session name could be used for this purpose. The modified behavior is illustrated in Fig. 5. The A node sends the prePath message to Z and to all intermediate nodes. The intermediate nodes can proceed with the postPath message as soon as they have decided on a

positive result of the preliminary Path operations. The Z node should wait for all postPath messages before proceeding with the upstream Resv message.

C. Comparison with a similar proposal

In order to solve the performance deficiency, a recent IEEE article [3] proposed a novel segmented signaling (SSP) based on the concept of intermediate destinations. A special case of this protocols is called "parallel reservation protocol" (PRP), where the destination node sends the RESV_INFO message (with the actual wavelength reservation) to all the intermediate nodes along the route, whereas in the SSP the destination node sends the RESV_INFO message only to some specific intermediate node, determined by dividing the route in smaller segments on which RSVP runs in parallel. Here are the specific differences between these proposals: The proposal in [3] is dedicated to WDM Networks whereas my proposal may apply to any kind of GMPLS signaling which provides source routing. Its optimization addresses segments while my optimization addresses any intermediate nodes for the parallelization. [3] suggests the modification of the RSVP protocol by defining a new control message (RESV_INFO, RESV_SUCCESS, FAIL_INFO) while my implementation proposal is based on the existing control messages. [3] indicates parallelization only for the Resv message, while my proposal may also include the Path. In addition, my proposal offers two different variants for sequential and final synchronization. One of the applications which may take a benefit of the RSVP implementation suggested here is the setup of the protecting LSP on restoration (comp. "make-before-break" in RFC3209 [4]). A specific of this path setup is the evaluation of the setup and holding priority, which is necessary in order to decide on the preemption of traffic. Also the operations related to the resource affinity (exclude, include-any/all) may be processed in parallel on each of the intermediate nodes. The preemption is in general an operation which can benefit from the preliminary verification of sufficient bandwidth at a particular priority along the entire path. As is indicated in the RFC3209 [4] "*If a Path message is allowed to progress when there are insufficient resources, then there is a danger that lower priority reservations downstream of this point will unnecessarily be preempted in a futile attempt to service this request"*. Another suitable application is the path setup over multiple segments (ENNI domains). The specific of this path setup is the presence of some intermediate border nodes which are accessing potentially opaque domains/segments. They are good candidates for sequential synchronization. The final synchronization may be performed in the corresponding intermediate egress nodes.

IV. Conclusions

This document is proposing the RSVP implementation of a previously published method for fast source routed connection setup. The method is based on the parallelization of the signaling messages and their processing using the inherently available distributed processing power of the network nodes. Instead of the sequential setup, two alternatives of the parallel setup with sequential or with final synchronization are discussed. Their benefit is a reduced setup time, minimizing the dependency on the path length. The proposed RSVP implementation is evolutionary and backward compatible, so it can be applied also to networks containing nodes handling traditional and enhanced RSVP setup messages. Compared with a similar proposal [3], it is more generic and avoids the introduction of new RSVP messages. The preferred option for extending the semantics of the RSVP message is by introduction of a new ADMIN_STATUS bit. However, also an implicit indication of the new semantics would be possible. The ingress node may decide on any of these alternative procedures, since they do not require a divergent behavior in any of the implied nodes. In general, the *Final Synchronization* can be adopted if there are no dependencies from the previous hop (e.g. Path for unidirectional setup). Instead, the *Sequential Synchronization* may be adopted when there are dependencies from the previous hop e.g. Resv.Label). One drawback is some increased complexity of the RSVP state machine. However, it should be noticed that this proposal is introducing a kind of a two phased transaction (preparation phase followed by the commit or a roll-back phase), which is in general a useful pattern. Another drawback is some decreased scalability because of the additional number of pre/postPath messages. However, this deficiency may be attenuated if the end nodes and intermediate nodes are well meshed. The benefit is a considerably reduced setup time for the positive case as well as for the negative case since resource conflicts can be detected faster.

References

[1] ITU-T Rec. G.8080/Y.1304, "Architecture for the Automatically Switched Networks (ASON)", ITU-T Standardization Organization, 2001.

[2] F. Lataretu, "Method for Fast Source Routed Connection Setup", United States Patent Application, Pub. No.: US 2006/0034288 A1.

[3] C. V. Saradhi, G. S. Kumar, Z. Luying, and G. Mohan, "A Fast and Efficient Segmented Signalling Protocol for GMPLS/WDM Optical Networks," *IEEE Intern. Conf. on Communications*, pp. 5422 – 5426, 2008.

[4] RFC's on RSVP: RFC2205, RFC3209, RFC3473 (http://www.faqs.org/rfcs).

[5] D. Saha, "An Efficient Wavelength Reservation Protocol for Lightpath Establishment in All-Optical Networks (AON's)," *Proc. IEEE Globecom*, vol. 2, pp. 1264-1268, Nov. 2000.

[6] D. Saha, "A Comparative Study of Distributed Protocols for Wavelength Reservation in WDM Optical Networks," *Optical Networks Magazine*, vol. 3, Jan/Feb. 2002.

[7] F. Feng, et. al., "Performance Study of Distributed Wavelength Reservation Protocols with Both Single and Multi-Fiber WDM Networks," *Photonic network communications*, vol. 6, no. 2, pp. 95-103, Oct. 2004.

[8] K. Lu, et. al., "Intermediate-Node Initiated Reservation (IIR): A New Signaling Scheme for Wavelength Routed Networks," *IEEE Journal on Selected Areas in Communications*, vol. 21, no. 8, pp. 1285-1294, Oct. 2003.

[9] H. Zang, et. al., "A Review of Routing and Wavelength Assignment Approaches for Wavelength Routed Optical WDM Networks," *Optical Networks Magazine*, vol. 1, pp. 4760, Jan. 2000.