Achieving Complete Control Plane Recovery in Optical User Network Interface

Wenhua Jiao, Bo Lei, Peng Wan, Jie Mei, Fang Liu
Lucent Technologies, Bell Labs Research China
Beijing, China
wjiao@lucent.com, leibo.bupt@gmail.com, wanpeng@zju.edu.cn, fliu@lucent.com

Abstract—In this paper, a new control plane recovery mechanism for optical User Network Interface is proposed against ingress nodal failure. Current standards of IETF and OIF only provide mechanisms to recover transit and egress nodal failure, while ingress nodal recovery hasn’t been covered yet. Introducing an upstream RecoveryPath message into nodal fault recovery, the proposed mechanism can be applied to ingress nodal recovery and thus reinforce control plane recovery. Detailed signaling flow is presented and simulation results show that the recovery time is cut down by the proposed mechanism. The proposed mechanism provides better interoperability between IETF and OIF standards.

Keywords- Nodal Fault, Control Plane Recovery, UNI

I. INTRODUCTION

OIF has defined a User Network Interface (UNI) implementation agreement based on the ITU-T ASON architectural model. UNI RSVP signaling is employed to invoke services the transport network offers to the clients, and enables dynamic interconnections between client devices in the IP layer and SONET/SDH devices in the optical transport network. OIF has accomplished the work on the UNI 1.0R2 [1][2] signaling agreements and currently focuses on UNI 2.0 [3].

The control channel can be implemented via in-band Data Communication Channel (DCC) or out-of-band LAN [1]. In this paper, only out-of-band control channel is considered. The control plane recovery process includes fault detection and fault recovery. Fault detection is used to locate the failures in links and nodes. Fault recovery is the procedure to recover those faults. If RSVP is adopted as UNI signaling scheme, control channel maintenance (in the absence of Link Management Protocol [1]) is done via RSVP Hellos. The missing of Hello messages leads to the assumption that the control channel is down, and a rule is made that the failure in a signaling channel or control protocol entity will not result in the deletion of previously established connections.

This paper highlights the control plane recovery scenarios for optical UNI and focuses on the nodal fault recovery. A new mechanism is proposed to recover the ingress node of a UNI session. The rest of the paper is organized as follows: Section II reviews the control plane recovery including fault detection, channel faults and nodal faults. Section III describes the new signaling mechanism to achieve better interoperability between UNI and GMPLS. Section IV compares the recovery time of the proposed mechanism with existing ones. Section V concludes the paper.

II. OVERVIEW OF CONTROL PLANE RECOVERY

A UNI Switched Connection (SC) can be divided into two segments: 1) Connection segment between client and neighbor ingress core node; 2) Connection segment between egress core node and its neighbor client node, as in Fig.1. The UNI RSVP messages should contain values to setup both two single-hop connection segments, one between the initiating UNI-C and UNI-N, and the other between the UNI-N and the terminating UNI-C. Different from the GMPLS peer model, these two segments are two separate RSVP sessions.

Figure 1. UNI Connection Segment.

A UNI failure can be either control plane failure or data plane failure. For data plane failure, SONET/SDH has provided adequate schemes for failure detection and recovery. If such failure can be recovered, UNI won't be impacted. Otherwise, forced deletion from network side will be triggered to release the affected UNI connections.

The recovery process for control plane failure also includes fault detection and fault recovery. Nodal fault indicates the node has lost its control state (e.g., after a reboot) but its data forwarding state remains. The control channel fault indicates control communication was lost between two nodes.

As the control channel failure will not result in the loss of RSVP states, this paper pays more attention to the nodal fault recovery, especially to the reconstruction of Path State Block (PSB) and Resv State Block (RSB) in ingress nodes.
For nodal failure, a recovery procedure has been defined in [4], in which the upstream RSVP neighbor sends a new PATH message to the rebooting node and the latter recreates its control state based on the message content. The limitation of [4] is that it can’t recover the RSVP state in its ingress node (source UNI-C or destination UNI-N) of a session after it rebooted. As a result, most vendors haven’t adopted this feature, while a few others implemented the feature by storing all RSVP state in local persistent database. In case of nodal failure and network element reboot, it obtains the previous recorded RSVP states from its local database. But such implementations are not standardized and do not compatible to [4]. In [5], a new message called RecoveryPath message is introduced in to help solve the problem of [4]. The message type of the RecoveryPath is the same as PATH message. RecoveryPath message is transmitted upstream instead of downstream for PATH messages. The restarting node indicates its ability to process RecoveryPath messages by including a new object called the Capability object with the RecoveryPath Desired bit set in its hello messages sent to the downstream RSVP neighbor. It also extends to [6] by defining a RecoveryPath Summary Refresh procedure with the RecoveryPath Srefresh Capable (S) bit in the Capability object to indicating this ability.

Fig.2 shows the mechanism proposed by [5], which is used to fully recover the connection state after the ingress nodal fault. But it needs more messages exchanged as the number of connections increases. What’s more, it is not compatible to current OIF standard [2] because there are many object formats (such as Capability object) and additional refresh procedures need to be updated.

![Ingress Nodal Faults Recovery Mechanism proposed by [5]](image)

The proposed UNI ingress nodal recovery mechanism is shown in Fig.4. The following procedures are executed when a node reboots.

1. Procedures for the rebooting ingress node (Source UNI-C or Destination UNI-N)

After rebooting its control plane, the ingress node begins to check whether it is able to preserve its data forwarding state. If no forwarding state from prior to the reboot is preserved, the node sets the Recovery Time to 0 in the Hello message and sends to the downstream node (Source UNI-N or Destination UNI-C). If the forwarding state is preserved, the node initiates the state recovery process after setting the Recovery Time to a configured value.

After receiving RecoveryPath message, the node first checks if it has a corresponding RSVP state. If the RSVP state is not found and the message is a RecoveryPath message, the node searches its data forwarding table for an entry whose outgoing interface matches the interface ID in RSVP hop object of RecoveryPath and whose outgoing label is equal to the
upstream label (bi-directional connection is supposed) in the RecoveryPath.

If the entry is not found in its forwarding table, the node sends a PathTear message to expedite the cleanup of unrecovered RSVP and associated forwarding state downstream of the rebooted node. Otherwise, the appropriate RSVP states (both PSB and RSB state) are created according to the information carried in RecoveryPath. For RSB, the label information is obtained from the Recovery_label object as that obtained from a RESV message. In this way, the RSVP state of the rebooted node is recovered. Then the control plane entry is associated with the connection specified in the message, with its related data forwarding tables updated. And the recovery procedures are completed.

We notice that the message ID stored in RSB may be different from the last RESV message from the downstream node. Since Srefresh procedure is suppressed during the recovery period [4], such a difference won’t impact the signaling procedure. After the expiration of recovery period and reboot of Srefresh procedure (Summary the PSB’s message ID), this ID will be automatically re-synchronized [6].

2. Procedures for the downstream node of a rebooting node (Source UNI-N or Destination UNI-C)

When the rebooting of a neighbor node is detected, the downstream node should send a RecoveryPath message to it. Normal RESV and Summary Refresh process are suppressed until RecoveryPath message is sent.

When the Recovery period expires, the downstream node sends a Srefresh message with RESV message ID received before the failure.

On receiving a Srefresh message, the rebooted node responds with an ACK message carrying MESSAGE_ID_NACK objects for all recovered connections, and then the downstream node sends explicit RESV messages to update the message IDs in the rebooted node. In this way the message ID of RSB gets synchronized.

The main advantages of this mechanism over that in [5] are listed as follows:

- Compatibility in signaling procedure
- The proposed mechanism is compatible with current OIF and IETF standards in signaling procedure.

Compared with [5], in our proposed mechanism new RSB is created automatically and explicit PATH refreshment is not necessary when the rebooted node receives a RecoveryPath message with Recovery Label, because the message carries enough information for PSB and RSB state recovery. The explicit PATH and RSB’s message ID are also refreshed automatically after the recovery period according to [6]. Thus, in the proposed method, as in Fig.4, fewer messages are exchanged than the one in Fig.3.

- Compatibility in signaling message format

The proposed mechanism is compatible with current OIF and IETF standards in signaling message formats.

RecoveryPath Capability Objects in Hello message and RecoveryPath Srefresh message are defined in [5] and not needed in the proposed mechanism. In [2], a Restart_Cap object was defined to indicate whether a node supports recovery and it is not use in the proposed mechanism, because in UNI session, every node may act as ingress node and egress node simultaneously for different sessions and the recovery ability must be supported.

B. Fault Recover Time Analysis

It is very complicated to estimate the recovery time of UNI node because there are parallel signal processes when multiple connections are requested in the same interval. Consider a simple situation first, in which only one connection needs to be recovered after a upstream node failure. Fig.5 compares the time consumed to recover a single connection utilizing the proposed mechanism with that proposed by [5]. The ACK message is ignored since it can be piggybacked in other messages [6].

We can get the recovery period of [5] by summing up all the pieces of time:

\[ t_{recovery} = t_{create_RecoveryPath} + t_{send_RecoveryPath} + t_{process_Path} + t_{check_RSVPstate} + t_{search_ForwardTable} + t_{create_RSVP} + t_{send_PathRefresh} + t_{process_Path} + t_{send_ResvRefresh} + t_{process_Resv} + t_{create_RSVP} \]

(1)

In which \( t_{create_RecoveryPath} \) is the time consumed by creating a
RecoveryPath message that sends to the rebooted node. This process consists of searching corresponding PSB states and generating new messages. And $t_{\text{process\_Path}}$ is the time for message decoding and memory processing of a PATH message. $t_{\text{check\_RSVP\_state}}$ is the time spent on database enquiry. $t_{\text{search\_ForwardTable}}$ includes data plane processing, in which a control to data plane communication (by whatever method) is needed.

Assuming all kinds of signaling transmission intervals are the same value, and processing period of PATH message equals to that of RSVP, e.g.:

1. $t_{\text{send\_RecoveryPath}} = t_{\text{send\_Path\_Refresh}} = t_{\text{send\_Resv\_Refresh}} = t_{\text{send\_Msg}}$
2. $t_{\text{process\_Path}} = t_{\text{process\_Resv}} = t_{\text{process\_Msg}}$

Then,

$$t_{\text{recover}} = t_{\text{create\_RecoveryPath}} + t_{\text{check\_RSVP\_state}} + t_{\text{search\_ForwardTable}}$$
$$+ 3 t_{\text{send\_Msg}} + 3 t_{\text{process\_Msg}} + t_{\text{create\_PSB}} + t_{\text{create\_RSB}}$$

(2)

According to Fig.5, recovery period in proposed mechanism is:

$$t_{\text{recover}} = t_{\text{create\_RecoveryPath}} + t_{\text{check\_RSVP\_state}} + t_{\text{search\_ForwardTable}}$$
$$+ t_{\text{send\_Msg}} + t_{\text{process\_Msg}} + t_{\text{create\_PSB}} + t_{\text{create\_RSB}}$$

(3)

Obviously, the proposed mechanism speeds up the recovery process by reducing the time cost in sending and processing signaling messages.

The UNI connection setting up experiment was done repeatedly in our test bed, and data sets and values of the elemental time were obtained with the traffic capturing method mentioned in [8]. The results below are averages.

<table>
<thead>
<tr>
<th>TABLE 1. MEASURED TIME INTERVALS DURING NODAL RECOVERY</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNI-C</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>$t_{\text{create_RecoveryPath}}$</td>
</tr>
<tr>
<td>$t_{\text{send_Msg}}$</td>
</tr>
<tr>
<td>$t_{\text{process_Msg}}$</td>
</tr>
<tr>
<td>$t_{\text{check_RSVP_state}}$</td>
</tr>
</tbody>
</table>

Results above are valid where only one connection is recovered at a time, but this rarely happens in practice. In our test bed, tens of connections request to be recovered after reboot, and analysis of multi-connection recovery time is much more complicated.

It is not advisable to simply add up every single connection recover time to get the total recovery time, as the message processing and state recovery are parallel processes.

Here, we categorize the elemental processes into two groups:

One group is for parallel processes, which are run by individual threads that can be processed simultaneously. These include checking RSVP states, message processing, creating PSB and creating RSB.

The other is for serial processes, which will wait in message queues or processing queues until the required resources are released by the former process. Creating and sending of a RecoveryPath message belong to this group.

Here, a reasonable assumption is made: parallel processes run in a shared period and the serial ones are processed one by one. Then we obtain the total recovery time for the multi-connection recovery scene.

Suppose there are $n$ connections to be recovered, for mechanism in [5], we have

$$t_{\text{recover}} = \sum n \cdot t_{\text{create\_RecoveryPath}} + \sum t_{\text{check\_RSVP\_state}} + \sum t_{\text{search\_ForwardTable}}$$
$$+ 3 \sum t_{\text{send\_Msg}} + 3 \sum t_{\text{process\_Msg}} + \sum t_{\text{create\_PSB}} + \sum t_{\text{create\_RSB}}$$

(4)

For mechanism proposed in this paper, we have

$$t_{\text{recover}} = \sum n \cdot t_{\text{create\_RecoveryPath}} + \sum t_{\text{check\_RSVP\_state}} + \sum t_{\text{search\_ForwardTable}}$$
$$+ \sum t_{\text{send\_Msg}} + \sum t_{\text{process\_Msg}} + \sum t_{\text{create\_PSB}} + \sum t_{\text{create\_RSB}}$$

(5)

IV. COMPARISON OF NODAL FAULTS RECOVERY TIME

In this section, we compare nodal faults recovery time of the proposed mechanism in Fig.4 with the mechanism proposed by [5] (in Fig.3) in BLRC’s (Bell Labs Research China) UNI test bed, in which the UNI-Cs are IP routers and the UNI-Ns are UNI-N proxies, and the UNI-N function is on the outside of transport NE (Network Element) with an internal signaling interface to the NE. Configuration of data plane and control plane is shown in Fig.6. It is assumed that UNI connection is initiated by R1 and the destination can be either R2 or R3. After successfully setting up multiple connections, R1 or E4 is rebooted to simulate nodal faults. The recovery time of both mechanisms is obtained by capturing the signaling messages exchanged through Ethernet. And the experiment was done repeatedly to get the average recovery time for both mechanisms.

Theoretical results and experimental results for both mechanisms are shown in Fig.7 and Fig.8. In Fig.7, the rebooted node is UNI-C R1 and in Fig.8, the rebooted node is UNI-N E4. The theoretical results are calculated by (4) and (5),...
with the parameters defined in Tab.1. The experimental results are obtained by capturing the RSVP packets through Ethernet.

Figure 6. BLRC’s Optical UNI Testbed.

As shown in Fig.7, the theoretical results are consistent well with actual results when the connection number is less than 20. As the number exceeds 30, the experimental recover time curve has a rapid ascent. This is because with the number of connections increasing, the CPU load becomes heavy and the resource contention occurs more often in UNI-C.

From the captured RSVP packets we can see that the proposed mechanism has less signaling message exchanged. In Fig.7, the proposed recovery mechanism is more efficient than the one provided by [5] especially when the number of to-be-recovered connections increases. For example, the given case in which 20 connections exist while nodal failure happens, the recovery time is about 1.4 seconds in the proposed scheme, while 1.8 seconds in scheme of [5].

The recovery time is also measured on the reboot UNI-N node E4 shown in Fig.8. Compared with that of UNI-C, the recovery time in UNI-N is much longer. For example, considering the mechanism in [5], the UNI-N recovery time is 10.5 seconds and 7.0 seconds for UNI-C recovery when 50 connections need to be recovered. This is because NE’s control plane bears more workload (I-NNI signaling, routing, database, network management interfaces, etc.) besides UNI related signaling.

A satisfactory result is observed that the mechanism proposed in this paper outperforms that in [5] when applied to UNI-N. Given 50 connections to be recovered, the UNI-N/UNI-C needs 5.8/4.9 seconds to recover them while that for the mechanism in [5] is 10.3/7.1 seconds. This implies that the performance in the proposed mechanism doesn’t drop as fast as the mechanism in [5] in case of heavy loads on UNI-N.

Figure 7. Recovery time vs. the number of connections in UNI-C.

Figure 8. Recovery time vs. the number of connections in UNI-N.

V. CONCLUSION

In optical UNI, nodal failure recovery procedures are proposed for the situation in which the control node lost its control state but its data forwarding state. Nodal fault recovery in UNI 1.0 R2 is according to RFC3473, which, unfortunately, hasn't covered the mechanism to recover the ingress node failure. In this paper, we solve the problem by a new signaling mechanism to recover the ingress node failure.

The proposed mechanism has been implemented in our test bed. All UNI nodes including both ingress and egress nodes can use the mechanism for nodal fault recovery. It is shown that, the proposed nodal fault recovery mechanism is more efficient especially for recovering more connections and maintains compatibility with existing OIF and IETF standards.

References