Network Working Group
Request for Comments: 3612
Category: Informational

A. Farrel
Old Dog Consulting

September 2003

Applicability Statement for Restart Mechanisms for the Label Distribution Protocol (LDP)

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2003). All Rights Reserved.

Abstract

This document provides guidance on when it is advisable to implement some form of Label Distribution Protocol (LDP) restart mechanism and which approach might be more suitable. The issues and extensions described in this document are equally applicable to RFC 3212, "Constraint-Based LSP Setup Using LDP".

1. Introduction

Multiprotocol Label Switching (MPLS) systems are used in core networks where system downtime must be kept to a minimum. Similarly, where MPLS is at the network edges (e.g., in Provider Edge (PE) routers) [RFC2547], system downtime must also be kept to a minimum. Many MPLS Label Switching Routers (LSRs) may, therefore, exploit Fault Tolerant (FT) hardware or software to provide high availability of the core networks.

The details of how FT is achieved for the various components of an FT LSR, including the switching hardware and the TCP stack, are implementation specific. How the software module itself chooses to implement FT for the state created by the LDP is also implementation specific. However, there are several issues in the LDP specification [RFC3036] that make it difficult to implement an FT LSR using the LDP protocols without some extensions to those protocols.

Proposals have been made in [RFC3478] and [RFC3479] to address these issues.

2. Requirements of an LDP FT System

Many MPLS LSRs may exploit FT hardware or software to provide high availability (HA) of core networks. In order to provide HA, an MPLS system needs to be able to survive a variety of faults with minimal disruption to the Data Plane, including the following fault types:

- failure/hot-swap of the switching fabric in an LSR,
- failure/hot-swap of a physical connection between LSRs,
- failure of the TCP or LDP stack in an LSR,
- software upgrade to the TCP or LDP stacks in an LSR.

The first two examples of faults listed above may be confined to the Data Plane. Such faults can be handled by providing redundancy in the Data Plane which is transparent to LDP operating in the Control Plane. However, the failure of the switching fabric or a physical link may have repercussions in the Control Plane since signaling may be disrupted.

The third example may be caused by a variety of events including processor or other hardware failure, and software failure.

Any of the last three examples may impact the Control Plane and will require action in the Control Plane to recover. Such action should be designed to avoid disrupting traffic in the Data Plane. Since many recent router architectures can separate the Control and Data Planes, it is possible that forwarding can continue unaffected by recovery action in the Control Plane.

In other scenarios, the Data and Control Planes may be impacted by a fault, but the needs of HA require the coordinated recovery of the Data and Control Planes to a state that existed before the fault.

The provision of protection paths for MPLS LSP and the protection of links, IP routes or tunnels through the use of protection LSPs is outside the scope of this document. See [RFC3469] for further information.

3. General Considerations

In order for the Data and Control Plane states to be successfully recovered after a fault, procedures are required to ensure that the state held on a pair of LDP peers (at least one of which was affected

directly by the fault) are synchronized. Such procedures must be implemented in the Control Plane software modules on the peers using Control Plane protocols.

The required actions may operate fully after the failure (reactive recovery) or may contain elements that operate before the fault in order to minimize the actions taken after the fault (proactive recovery). It is rare to implement actions that operate solely in advance of the failure and do not require any further processing after the failure (preventive recovery) - this is because of the dynamic nature of signaling protocols and the unpredictability of fault timing.

Reactive recovery actions may include full re-signaling of state and re-synchronization of state between peers and synchronization based on checkpointing.

Proactive recovery actions may include hand-shaking state transitions and checkpointing.

4. Specific Issues with the LDP Protocol

LDP uses TCP to provide reliable connections between LSRs to exchange protocol messages to distribute labels and to set up LSPs. A pair of LSRs that have such a connection are referred to as LDP peers.

TCP enables LDP to assume reliable transfer of protocol messages. This means that some of the messages do not need to be acknowledged (e.g., Label Release).

LDP is defined such that if the TCP connection fails, the LSR should immediately tear down the LSPs associated with the session between the LDP peers, and release any labels and resources assigned to those LSPs.

It is notoriously difficult to provide a Fault Tolerant implementation of TCP. To do so might involve making copies of all data sent and received. This is an issue familiar to implementers of other TCP applications, such as BGP.

During failover affecting the TCP or LDP stacks, therefore, the TCP connection may be lost. Recovery from this position is made worse by the fact that LDP control messages may have been lost during the connection failure. Since these messages are unconfirmed, it is possible that LSP or label state information will be lost.

At the very least, the solution to this problem must include a change to the basic requirements of LDP so that the failure of an LDP session does not require that associated LDP or forwarding state be torn down.

Any changes made to LDP in support of recovery processing must meet the following requirements:

- offer backward-compatibility with LSRs that do not implement the extensions to LDP,

- preserve existing protocol rules described in [RFC3036] for handling unexpected duplicate messages and for processing unexpected messages referring to unknown LSPs/labels.

Ideally, any solution applicable to LDP should be equally applicable to CR-LDP.

5. Summary of the Features of LDP FT

LDP Fault Tolerance extensions are described in [RFC3479]. This approach involves:

- negotiation between LDP peers of the intent to support extensions to LDP that facilitate recovery from failover without loss of LSPs,

- selection of FT survival on a per LSP/label basis or for all labels on a session,

- sequence numbering of LDP messages to facilitate acknowledgement and checkpointing,

- acknowledgement of LDP messages to ensure that a full handshake is performed on those messages either frequently (such as per message) or less frequently as in checkpointing,

- solicitation of up-to-date acknowledgement (checkpointing) of previous LDP messages to ensure the current state is secured, with an additional option that allows an LDP partner to request that state is flushed in both directions if graceful shutdown is required,

- a timer to control how long LDP and forwarding state should be retained after the LDP session failure, but before being discarded if LDP communications are not re-established,

- exchange of checkpointing information on LDP session recovery to establish what state has been retained by recovering LDP peers,

- re-issuing lost messages after failover to ensure that LSP/label state is correctly recovered after reconnection of the LDP session.

The FT procedures in [RFC3479] concentrate on the preservation of label state for labels exchanged between a pair of adjacent LSRs when the TCP connection between those LSRs is lost. There is no intention within these procedures to support end-to-end protection for LSPs.

6. Summary of the Features of LDP Graceful Restart

LDP graceful restart extensions are defined in [RFC3478]. This approach involves:

- negotiation between LDP peers of the intent to support extensions to LDP that facilitate recovery from failover without loss of LSPs,

- a mechanism whereby an LSR that restarts can relearn LDP state by resynchronization with its peers,

- use of the same mechanism to allow LSRs recovering from an LDP session failure to resynchronize LDP state with their peers provided that at least one of the LSRs has retained state across the failure or has itself resynchronized state with its peers,

- a timer to control how long LDP and forwarding state should be retained after the LDP session failure, but before being discarded if LDP communications are not re-established,

- a timer to control the length of the resynchronization period between adjacent peers should be completed.

The procedures in [RFC3478] are applicable to all LSRs, both those with the ability to preserve forwarding state during LDP restart and those without. LSRs that can not preserve their MPLS forwarding state across the LDP restart would impact MPLS traffic during restart. However, by implementing a subset of the mechanisms in [RFC3478] they can minimize the impact if their neighbor(s) are capable of preserving their forwarding state across the restart of their LDP sessions or control planes by implementing the mechanism in [RFC3478].

7. Applicability Considerations

This section considers the applicability of fault tolerance schemes within LDP networks and considers issues that might lead to the choice of one method or another. Many of the points raised below should be viewed as implementation issues rather than specific drawbacks of either solution.

7.1. General Applicability

The procedures described in [RFC3478] and [RFC3479] are intended to cover two distinct scenarios. In Session Failure, the LDP peers at the ends of a session remain active, but the session fails and is restarted. Note that session failure does not imply failure of the data channel even when using an in-band control channel. In Node Failure, the session fails because one of the peers has been restarted (or at least, the LDP component of the node has been restarted). These two scenarios have different implications for the ease of retention of LDP state within an individual LSR, and are described in sections below.

These techniques are only applicable in LDP networks where at least one LSR has the capability to retain LDP signaling state and the associated forwarding state across LDP session failure and recovery. In [RFC3478], the LSRs retaining state do not need to be adjacent to the failed LSR or session.

If traffic is not to be impacted, both LSRs at the ends of an LDP session must at least preserve forwarding state. Preserving LDP state is not a requirement to preserve traffic.

[RFC3479] requires that the LSRs at both ends of the session implement the procedures that it describes. Thus, either traffic is preserved and recovery resynchronizes state, or no traffic is preserved and the LSP fails.

Further, to use the procedures of [RFC3479] to recover state on a session, both LSRs must have a mechanism for maintaining some session state and a way of auditing the forwarding state and the resynhcronized control state.

[RFC3478] is scoped to support preservation of traffic if both LSRs implement the procedures that it describes. Additionally, it functions if only one LSR on the failed session supports retention of forwarding state, and implements the mechanisms in the document. In this case, traffic will be impacted by the session failure, but the forwarding state will be recovered on session recovery. Further, in the event of simultaneous failures, [RFC3478] is capable of

relearning and redistributing state across multiple LSRs by combining its mechanisms with the usual LDP message exchanges of [RFC3036].

7.2. Session Failure

In Session Failure, an LDP session between two peers fails and is restarted. There is no restart of the LSRs at either end of the session and LDP continues to function on those nodes.

In these cases, it is simple for LDP implementations to retain the LDP state associated with the failed session and to associate the state with the new session when it is established. Housekeeping may be applied to determine that the failed session is not returning and to release the old LDP state. Both [RFC3478] and [RFC3479] handle this case.

Applicability of [RFC3478] and [RFC3479] to the Session Failure scenario should be considered with respect to the availability of the data plane.

In some cases the failure of the LDP session may be independent of any failure of the physical (or virtual) link(s) between adjacent peers; for example, it might represent a failure of the TCP/IP stack. In these cases, the data plane is not impacted and both [RFC3478] and [RFC3479] are applicable to preserve or restore LDP state.

LDP signaling may also operate out of band; that is, it may use different links from the data plane. In this case, a failure of the LDP session may be a result of a failure of the control channel, but there is no implied failure of the data plane. For this scenario [RFC3478] and [RFC3479] are both applicable to preserve or restore LDP state.

In the case where the failure of the LDP session also implies the failure of the data plane, it may be an implementation decision whether LDP peers retain forwarding state, and for how long. In such situations, if forwarding state is retained, and if the LDP session is re-established, both [RFC3478] and [RFC3479] are applicable to preserve or restore LDP state.

When the data plane has been disrupted an objective of a recovery implementation might be to restore data traffic as quickly as possible.

7.3. Controlled Session Failure

In some circumstances, the LSRs may know in advance that an LDP session is going fail (e.g., perhaps a link is going to be taken out of service).

[RFC3036] includes provision for controlled shutdown of a session. [RFC3478] and [RFC3479] allow resynchronization of LDP state upon re-establishment of the session.

[RFC3479] offers the facility to both checkpoint all LDP states before the shut-down, and to quiesce the session so that no new state changes are attempted between the checkpoint and the shut-down. This means that on recovery, resynchronization is simple and fast.

[RFC3478] resynchronizes all state on recovery regardless of the nature of the shut-down.

7.4. Node Failure

Node Failure describes events where a whole node is restarted or where the component responsible for LDP signaling is restarted. Such an event will be perceived by the LSR's peers as session failure, but the restarting node sees the restart as full re-initialization.

The basic requirement is that the forwarding state is retained, otherwise the data plane will necessarily be interrupted. If forwarding state is not retained, it may be relearned from the saved control state in [RFC3479]. [RFC3478] does not utilize or expect a saved control state. If a node restarts without preserved forwarding state it informs its neighbors, which immediately delete all label-FEC bindings previously received from the restarted node.

The ways to retain a forwarding and control state are numerous and implementation specific. It is not the purpose of this document to espouse one mechanism or another, nor even to suggest how this might be done. If state has been preserved across the restart, synchronization with peers can be carried out as though recovering from Session Failure as in the previous section. Both [RFC3478] and [RFC3479] support this case.

How much control state is retained is largely an implementation choice, but [RFC3479] requires that at least small amount of per-session control state be retained. [RFC3478] does not require or expect control state to be retained.

It is also possible that the restarting LSR has not preserved any state. In this case, [RFC3479] is of no help. [RFC3478] however,

allows the restarting LSR to relearn state from each adjacent peer through the processes for resynchronizing after Session Failure. Further, in the event of simultaneous failure of multiple adjacent nodes, the nodes at the edge of the failure zone can recover state from their active neighbors and distribute it to the other recovering LSRs without any failed LSR having to have saved state.

7.5. Controlled Node Failure

In some cases (hardware repair, software upgrade, etc.), node failure may be predictable. In these cases all sessions with peers may be shutdown and existing state retention may be enhanced by special actions.

[RFC3479] checkpointing and quiesce may be applied to all sessions so that state is up-to-date.

As above, [RFC3478] does not require that state is retained by the restarting node, but can utilize it if it is.

7.6. Speed of Recovery

Speed of recovery is impacted by the amount of signaling required.

If forwarding state is preserved on both LSRs on the failed session, then the recovery time is constrained by the time to resynchronize the state between the two LSRs.

[RFC3479] may resynchronize very quickly. In a stable network, this resolves to a handshake of a checkpoint. At the most, resynchronization involves this handshake plus an exchange of messages to handle state changes since the checkpoint was taken. Implementations that support only the periodic checkpointing subset of [RFC3479] are more likely to have additional state to resynchronize.

[RFC3478] must resynchronize state for all label mappings that have been retained. At the same time, resources that have been retained by a restarting upstream LSR but are not actually required, because they have been released by the downstream LSR (perhaps because it was in the process of releasing the state), they must be held for the full resynchronization time to ensure that they are not needed.

The impact of recovery time will vary according to the use of the network. Both [RFC3478] and [RFC3479] allow advertisement of new labels while resynchronization is in progress. Issues to consider are re-availability of falsely retained resources and conflict between retained label mappings and newly advertised ones. This may

cause incorrect forwarding of data (since labels are advertised from downstream), an LSR upstream of a failure may continue to forward data for one FEC on an old label while the recovering downstream LSR might re-assign that label to another FEC and advertise it. For this reason, restarting LSRs may choose to not advertise new labels until resynchronization with their peers has completed, or may decide to use special techniques to cover the short period of overlap between resynchronization and new LSP setup.

7.7. Scalability

Scalability is largely the same issue as speed of recovery and is governed by the number of LSPs managed through the failed session(s).

Note that there are limits to how small the resynchronization time in [RFC3478] may be made given the capabilities of the LSRs, the throughput on the link between them, and the number of labels that must be resynchronized.

Impact on normal operation should also be considered.

[RFC3479] requires acknowledgement of all messages. These acknowledgements may be deferred as for checkpointing described in section 4, or may be frequent. Although acknowledgements can be piggy-backed on other state messages, an option for frequent acknowledgement is to send a message solely for the purpose of acknowledging a state change message. Such an implementation would clearly be unwise in a busy network.

[RFC3478] has no impact on normal operations.

7.8. Rate of Change of LDP State

Some networks do not show a high degree of change over time, such as those using targeted LDP sessions; others change the LDP forwarding state frequently, perhaps reacting to changes in routing information on LDP discovery sessions.

Rate of change of LDP state exchanged over an LDP session depends on the application for which the LDP session is being used. LDP sessions used for exchanging <FEC, label> bindings for establishing hop by hop LSPs will typically exchange state reacting to IGP changes. Such exchanges could be frequent. On the other hand, LDP sessions established for exchanging MPLS Layer 2 VPN FECs will typically exhibit a smaller rate of state exchange.

In [RFC3479], two options exist. The first uses a frequent (up to per-message) acknowledgement system which is most likely to be applicable in a more dynamic system where it is desirable to preserve the maximum amount of state over a failure to reduce the level of resynchronization required and to speed the recovery time.

The second option in [RFC3479] uses a less-frequent acknowledgement scheme known as checkpointing. This is particularly suitable to networks where changes are infrequent or bursty.

[RFC3478] resynchronizes all state on recovery regardless of the rate of change of the network before the failure. This consideration is thus not relevant to the choice of [RFC3478].

7.9. Label Distribution Modes

Both [RFC3478] and [RFC3479] are suitable for use with Downstream Unsolicited label distribution.

[RFC3478] describes Downstream-On-Demand as an area for future study and is therefore not applicable for a network in which this label distribution mode is used. It is possible that future examination of this issue will reveal that once a label has been distributed in either distribution mode, it can be redistributed by [RFC3478] upon session recovery.

[RFC3479] is suitable for use in a network that uses Downstream-On-Demand label distribution.

In theory, and according to [RFC3036], even in networks configured to utilize Downstream Unsolicited label distribution, there may be occasions when the use of Downstream-On-Deman distribution is desirable. The use of the Label Request message is not prohibited in a Downstream Unsolicited label distribution LDP network.

Opinion varies as to whether there is a practical requirement for the use of the Label Request message in a Downstream Unsolicited label distribution LDP network. Current deployment experience suggests that there is no requirement.

7.10. Implementation Complexity

Implementation complexity has consequences for the implementer and also for the deployer since complex software is more error prone and harder to manage.

[RFC3479] is a more complex solution than [RFC3478]. In particular, [RFC3478] does not require any modification to the normal signaling and processing of LDP state changing messages.

[RFC3479] implementations may be simplified by implementing only the checkpointing subset of the functionality.

7.11. Implementation Robustness

In addition to the implication for robustness associated with complexity of the solutions, consideration should be given to the effects of state preservation on robustness.

If state has become incorrect for whatever reason, then state preservation may retain incorrect state. In extreme cases, it may be that the incorrect state is the cause of the failure in which case preserving that state would be inappropriate.

When state is preserved, the precise amount that is retained is an implementation issue. The basic requirement is that forwarding state is retained (to preserve the data path) and that that state can be accessed by the LDP software component.

In both solutions, if the forwarding state is incorrect and is retained, it will continue to be incorrect. Both solutions have a mechanism to housekeep and free the unwanted state after resynchronization is complete. [RFC3478] may be better at eradicating incorrect forwarding state, because it replays all message exchanges that caused the state to be populated.

In [RFC3478], no more data than the forwarding state needs to have been saved by the recovering node. All LDP state may be relearned by message exchanges with peers. Whether those exchanges may cause the same incorrect state to arise on the recovering node is an obvious concern.

In [RFC3479], the forwarding state must be supplemented by a small amount of state specific to the protocol extensions. LDP state may be retained directly or reconstructed from the forwarding state. The same issues apply when reconstructing state but are mitigated by the fact that this is likely a different code path. Errors in the retained state specific to the protocol extensions will persist.

7.12. Interoperability and Backward Compatibility

It is important that new additions to LDP interoperate with existing implementations at least in provision of the existing levels of function.

Both [RFC3478] and [RFC3479] do this through rules for handling the absence of the FT optional negotiation object during session initialization.

Additionally, [RFC3478] is able to perform limited recovery (i.e., redistribution of state) even when only one of the participating LSRs supports the procedures. This may offer considerable advantages in interoperation with legacy implementations.

7.13. Interaction With Other Label Distribution Mechanisms

Many LDP LSRs also run other label distribution mechanisms. These include management interfaces for configuration of static label mappings, other distinct instances of LDP, and other label distribution protocols. The last example includes traffic engineering label distribution protocol that are used to construct tunnels through which LDP LSPs are established.

As with re-use of individual labels by LDP within a restarting LDP system, care must be taken to prevent labels that need to be retained by a restarting LDP session or protocol component from being used by another label distribution mechanism. This might compromise data security, amongst other things.

It is a matter for implementations to avoid this issue through the use of techniques, such as a common label management component or segmented label spaces.

7.14. Applicability to CR-LDP

CR-LDP [RFC3212] utilizes Downstream-On-Demand label distribution. [RFC3478] describes Downstream-On-Demand as an area for future study and is therefore not applicable for CR-LDP. [RFC3479] is suitable for use in a network entirely based on CR-LDP or in one that is mixed between LDP and CR-LDP.

8. Security Considerations

This document is informational and introduces no new security concerns.

The security considerations pertaining to the original LDP protocol [RFC3036] remain relevant.

[RFC3478] introduces the possibility of additional denial-of- service attacks. All of these attacks may be countered by use of an authentication scheme between LDP peers, such as the MD5-based scheme outlined in [LDP].

In MPLS, a data mis-delivery security issue can arise if an LSR continues to use labels after expiration of the session that first caused them to be used. Both [RFC3478] and [RFC3479] are open to this issue.

9. Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on the IETF's procedures with respect to rights in standards-track and standards-related documentation can be found in BCP-11. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementors or users of this specification can be obtained from the IETF Secretariat.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this standard. Please address the information to the IETF Executive Director.

10. References

10.1. Normative References

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3036] Andersson, L., Doolan, P., Feldman, N., Fredette, A. and B. Thomas, "LDP Specification", RFC 3036, January 2001.

[RFC3478] Leelanivas, M., Rekhter, Y. and R. Aggarwal, "Graceful Restart Mechanism for LDP", RFC 3478, February 2003.

[RFC3479] Farrel, A., Editor, "Fault Tolerance for the Label Distribution Protocol (LDP)", RFC 3479, February 2003.

10.2. Informative References

[RFC2547] Rosen, E. and Y. Rekhter, "BGP/MPLS VPNs", RFC 2547, March 1999.

[RFC3212] Jamoussi, B., Editor, Andersson, L., Callon, R., Dantu, R., Wu, L., Doolan, P., Worster, T., Feldman, N., Fredette, A., Girish, M., Gray, E., Heinanen, J., Kilty, T. and A. Malis, "Constraint-Based LSP Setup using LDP", RFC 3212, January 2002.

[RFC3469] Sharma, V., Ed., and F. Hellstrand, Ed., "Framework for Multi-Protocol Label Switching (MPLS)-based Recovery", RFC 3469, February 2003.

11. Acknowledgements

The author would like to thank the authors of [RFC3478] and [RFC3479] for their work on fault tolerance of LDP. Many thanks to Yakov Rekhter, Rahul Aggarwal, Manoj Leelanivas and Andrew Malis for their considered input to this applicability statement.

12. Author’s Address

Adrian Farrel
Old Dog Consulting

Phone: +44 (0) 1978 860944
EMail: adrian@olddog.co.uk

13. Full Copyright Statement

Copyright (C) The Internet Society (2003). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assignees.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.