Network Working Group
Internet Draft
Expiration Date: September 2001

Jonathan P. Lang (Calient Networks)
Krishna Mitra (Calient Networks)
John Drake (Calient Networks)
Kireeti Kompella (Juniper Networks)
Yakov Rekhter (Juniper Networks)
Lou Berger (Movaz Networks)
Debanjan Saha (Tellium)
Debashis Basak (Accelight Networks)
Hal Sandick (Nortel Networks)
Alex Zinin (Cisco Systems)
Bala Rajagopalan (Tellium)

Link Management Protocol (LMP)

draft-ietf-mpls-lmp-02.txt

Status of this Memo

Abstract

Future networks will consist of photonic switches, optical crossconnects, and routers that may be configured with control channels, links, and bundled links. This draft specifies a link management protocol (LMP) that runs between neighboring nodes and is used to manage traffic engineering (TE) links. Specifically, LMP will be used to maintain control channel connectivity, verify the physical connectivity of the data-bearing channels, correlate the link property information, and manage link failures. A unique feature of the fault management technique is that it is able to localize failures in both opaque and transparent networks, independent of the encoding scheme used for the data.

Table of Contents

Changes from previous version:

- Added LMP length field to the common header.
- Decoupled control channel and TE Link dependence. Removed control channel switchover procedure.
- Modified the FSMs to align with current procedures.
- Modified ConfigAck & ConfigNack to echo the NodeId received in the Config message.
- Modified the Test messages to include VerifyId (provided by remote node) to differentiate test messages from different TE-links or LMP peers when testing in parallel.
- Added Link Mux Capability to LinkSummary.
- Added flags to LinkSummary indicating status of the ports or component links.
- Added Channel Activate messages to Fault Management procedure.
- General Text clarification including:
  - difference between port and component link
  - use of control channels

1. Introduction

Future networks will consist of photonic switches (PXCs), optical crossconnects (OXCs), routers, switches, DWDM systems, and add-drop multiplexors (ADMs) that use the Generalized MPLS (GMPLS) control plane to dynamically provision resources and to provide network survivability using protection and restoration techniques. A pair of nodes (e.g., two PXCs) may be connected by thousands of fibers, and each fiber may be used to transmit multiple wavelengths if DWDM is used. Furthermore, multiple fibers and/or multiple wavelengths may be combined into a single traffic-engineering (TE) link for routing purposes. To enable communication between nodes for routing, signaling, and link management, a control channel must be established between the node pair. This draft specifies a link management protocol (LMP) that runs between neighboring nodes and is used to manage TE links.

In this draft, we will follow the naming convention of [LAMBDA] and use OXC to refer to all categories of optical crossconnects, irrespective of the internal switching fabric. We distinguish between crossconnects that require opto-electronic conversion, called digital crossconnects (DXCs), and those that are all-optical, called photonic switches or photonic crossconnects (PXCs) - referred to as pure crossconnects in [LAMBDA], because the transparent nature of PXCs introduces new restrictions for monitoring and managing the data links (see [PERF-MON] for proposed extensions to MPLS for performance monitoring in photonic networks). LMP can be used for any type of node, enhancing the functionality of traditional DXCs and routers, while enabling PXCs and DWDMs to intelligently interoperate in heterogeneous optical networks.

In GMPLS, the control channel between two adjacent nodes is no longer required to use the same physical medium as the data-bearing

links between those nodes. For example, a control channel could use a separate wavelength or fiber, an Ethernet link, or an IP tunnel through a separate management network. A consequence of allowing the control channel(s) between two nodes to be physically diverse from the associated data links is that the health of a control channel does not necessarily correlate to the health of the data links, and vice-versa. Therefore, a clean separation between the fate of the control channel and data-bearing links must be made. Furthermore, new mechanisms must be developed to manage the data-bearing links, both in terms of link provisioning and fault localization.

For the purposes of this document, a data-bearing link may be either a "port" or a "component link" depending on its multiplexing capability; component links are multiplex capable, whereas ports are not multiplex capable. This distinction is important since the management of such links (including, for example, resource allocation, label assignment, and their physical verification) is different based on their multiplexing capability. For example, a SONET crossconnect with OC-192 interfaces may be able to demultiplex the OC-192 stream into four OC-48 streams. If multiple interfaces are grouped together into a single TE link using link bundling [BUNDLE], then the link resources must be identified using three levels: TE link Id, component interface Id, and timeslot label. Resource allocation happens at the lowest level (timeslots), but physical connectivity happens at the component link level. As another example, consider the case where a PXC transparently switches OC-192 lightpaths. If multiple interfaces are once again grouped together into a single TE link, then link bundling [BUNDLE] is not required and only two levels of identification are required: TE link Id and port Id. Both resource allocation and physical connectivity happen at the lowest level (i.e. port level). LMP is designed to support aggregation of one or more data-bearing links into a TE link (either ports into TE links, or component links into TE links).

2. LMP Overview

LMP runs between a pair of nodes and includes a core set of functions; two additional tools are defined in this draft to extend the functionality of LMP and are optional. The core function set includes control channel management and link property correlation. Control channel management is used to establish and maintain control channel connectivity between neighboring nodes. This is done using lightweight Hello messages that act as a fast keep-alive mechanism between the nodes. Link property correlation consists of a LinkSummary message exchange that is used to synchronize the link properties (e.g., local/remote Interface ID mappings) between the adjacent nodes.

LMP requires that a pair of nodes have at least one active bi-directional control channel between them. This control channel may

be implemented using two uni-directional control channels that are coupled together using the LMP Hello messages. All LMP messages are IP encoded [except, in some cases, the Test Message which may be limited by the transport mechanism for in-band messaging]; the link level encoding of the control channel is outside the scope of this document.

In LMP, multiple control channels may be active simultaneously between a pair of nodes. Each control channel MUST individually negotiate the control channel parameters, and each active control channel MUST exchange LMP hello packets to maintain LMP connectivity. If a group of control channels share a common node pair and support the same LMP capabilities, then LMP control messages MAY be transmitted over any of the active control channels of that group without coordination between the local and remote nodes. LMP also allows secondary (or backup) control channels to be defined. For example, data-bearing may be used as backup control channels provided control channel traffic has preemptive priority over the data traffic on the link. Secondary control channels only become active control channels when the switchover is complete and they inherit the configuration properties of the primary control channel that is being switched over to it.

The link property correlation function of LMP is designed to aggregate multiple ports or component links into a TE link, and to synchronize the properties of the TE link. As part of the link property correlation function, a LinkSummary message exchange is defined. The LinkSummary message includes the local and remote TE Link Id, a list of all ports or component links that comprise the TE link, and various link properties. In addition, TLVs may be included further describing the TE link. A LinkSummaryAck or LinkSummaryNack message MUST be sent in response to the receipt of a LinkSummary message indicating agreement or disagreement of the link properties.

In this draft, two additional tools are defined that extend the functionality of LMP: link connectivity verification and fault management. These tools are particularly useful when the control channel is transmitted out-of-band from the data-bearing links. Link connectivity verification is used to verify the physical connectivity between the nodes and exchange the Interface Ids (either Port Ids or Component Interface Ids, depending on the configuration); these Ids are used in GMPLS signaling. The procedure uses in-band Test messages that are sent over the data-bearing links and TestStatus messages that are transmitted over the control channel. The fault management scheme uses ChannelActive and ChannelFail message exchanges between a pair of nodes to localize failures in both opaque and transparent networks, independent of the encoding scheme used for the data. As a result, both local span and

end-to-end path protection/restoration procedures can be initiated.

For the LMP Test procedure, the free (unallocated) data-bearing links MUST be opaque (i.e., able to be terminated); however, once a data link is allocated, it may become transparent. The LMP Test procedure is coordinated using a BeginVerify message exchange over the control channel. To support various degrees of transparency (e.g., examining overhead bytes, terminating the payload, etc.), and hence, different mechanisms to transport the Test messages, a Verify Transport Mechanism is included in the BeginVerify and BeginVerifyAck messages. Note that there is no requirement that all of the data-bearing links must be terminated simultaneously, but at a minimum, they must be able to be terminated one at a time. There is also no requirement that the control channel and TE link share the same physical medium; however, the control channel MUST terminate on the same two nodes that the TE link spans. Since the BeginVerify message exchange coordinates the Test procedure, it also naturally coordinates the transition of the data links between opaque and transparent modes.

The LMP fault management procedure is based on two message exchanges: ChannelActive and ChannelFail. The ChannelActive message is used to indicate that one or more data-bearing channels are now carrying user data. This is particularly useful for detecting unidirectional channel failures in the transparent case. Receipt of a ChannelActive message MUST be acknowledged with a ChannelActiveAck message, the data-bearing channels MUST move to the ACTIVE state (if not already there), and fault monitoring SHOULD be verified for the corresponding data channels. The ChannelFail message is used to indicate that one or more active data channels or an entire TE link have failed. Receipt of a ChannelFail message MUST be acknowledged with either a ChannelFailNack or ChannelFailAck message, depending on if the channel failure is CLEAR or not in the adjacent node.

The organization of the remainder of this document is as follows. In Section 3, we discuss the role of the control channel and the messages used to establish and maintain link connectivity. In Section 4, the link property correlation function using the LinkSummary message is described. The link verification procedure is discussed in Section 5. In Section 6, we show how LMP will be used to isolate link and channel failures within the optical network. Several finite state machines (FSMs) are given in Section 7 and the message formats are defined in Section 8.

3. Control channel management

To initiate an LMP session between two nodes, a bi-directional control channel MUST be established. The control channel can be used to exchange MPLS control-plane information such as link provisioning and fault isolation information (implemented using a messaging protocol such as LMP, proposed in this draft), path

management and label distribution information (implemented using a signaling protocol such as RSVP-TE [RSVP-TE] or CR-LDP [CR-LDP]), and network topology and state distribution information (implemented

using traffic engineering extensions of protocols such as OSPF [OSPF-TE] and IS-IS [ISIS-TE]). For the purposes of LMP, we do not specify the exact implementation of the control channel; it could be, for example, a separate wavelength or fiber, an Ethernet link, an IP tunnel through a separate management network, or the overhead bytes of a data-bearing link. Rather, we assign a node-wide unique 32-bit non-zero integer control channel identifier (CCId) to each direction of the control channel. One possible way to assign a CCId is to use the IP address or ifindex of the interface. Furthermore, we define the control channel messages (which have control channel identifiers in them) to be IP encoded. This allows the control channel implementation to encompass both in-band and out-of-band mechanisms; including the case where the control channel messages are transmitted separately from the associated data link(s). Furthermore, since the messages are sent directly over IP, the link level encoding is not part of LMP.

The control channel can be either explicitly configured or automatically selected, however, for the purpose of this document we will assume the control channel is explicitly configured. Note that for in-band signaling, a control channel could be allocated to a data-bearing link; however, this is not true when the control channel is transmitted separately from the data links.

Control channels exist independently of TE links and multiple control channels may be active simultaneously between a pair of nodes. The control channels may also be used for transmitting and receiving signaling and routing messages. Each LMP control channel MUST individually negotiate the control channel parameters, and each active control channel MUST exchange LMP Hello packets to maintain LMP connectivity. If a group of control channels share a common node pair and support the same LMP capabilities, then LMP control channel messages (except Config, ConfigAck, ConfigNack, and Hello) may be transmitted over any of the active control channels without coordination between the local and remote nodes.

For LMP, it is essential that at least one control channel is always available. In the event of a control channel failure, it may be possible to use an alternate active control channel without coordination as mentioned above. Since control channels are electrically terminated at each node, lower layers (e.g., SONET/SDH) may also be used to detect control channel failures.

3.1. Parameter Negotiation

For LMP, a generic parameter negotiation exchange is defined using Config, ConfigAck, and ConfigNack messages. The contents of these messages are built using TLV triplets. Config TLVs can be either negotiable or non-negotiable (identified by the N flag in the TLV

header). Negotiable TLVs can be used to let the devices agree on certain values. Non-negotiable TLVs are used for announcement of specific values that do not need, or do not allow, negotiation.

To initiate the configuration procedure, a node MUST transmit Config messages to the remote node. It is possible that both the local and remote nodes initiate the configuration procedure at effectively the same time. To avoid ambiguities, the node with the higher Node Id wins the contention; the node with the lower Node Id SHOULD stop transmitting the Config message and respond to the Config messages it receives.

The Config message MUST be periodically transmitted until (1) it receives a ConfigAck or ConfigNack message, (2) a timeout expires and no ConfigAck or ConfigNack message has been received, or (3) it receives a Config message from the remote node and has lost the contention (e.g., the Node Id of the remote node is higher than the Node Id of the local node). Both the retransmission interval and the timeout period are local configuration parameters.

The Config message MUST include the LMP Capability TLV and the HelloConfig TLV.

The ConfigAck message is used to acknowledge receipt of the Config message and express agreement on ALL of the configured parameters (both negotiable and non-negotiable). The ConfigNack message is used to acknowledge receipt of the Config message, indicate which (if any) non-negotiable parameters are unacceptable, and propose alternate values for the negotiable parameters.

3.2. Hello protocol

Once a control channel is configured between two neighboring nodes, a Hello protocol will be used to establish and maintain control channel connectivity between the nodes and to detect control channel failures. The Hello protocol of LMP is intended to be a lightweight keep-alive mechanism that will react to control channel failures rapidly so that IGP Hellos are not lost and the associated link-state adjacencies are not removed unnecessarily. Furthermore, the RSVP Hello of [RSVP-TE] is not needed since the LMP Hellos will detect link layer failures.

The Hello protocol consists of two phases: a negotiation phase and a keep-alive phase. Negotiation MUST only be done when the control channel is in the CONFIG state, and is used to exchange the CCIds and agree upon the parameters used in the keep-alive phase. The keep-alive phase consists of a fast lightweight Hello message exchange.

3.2.1. Hello Parameter Negotiation

Before sending Hello messages as part of the keep-alive phase, the HelloInterval and HelloDeadInterval parameters MUST be agreed upon.

These parameters are exchanged as a HelloConfig TLV object in the Config message. The HelloInterval indicates how frequently LMP

Hello messages will be sent, and is measured in milliseconds (ms). For example, if the value were 100, then the transmitting node would send the Hello message at least every 100ms. The HelloDeadInterval indicates how long a device should wait to receive a Hello message before declaring a control channel dead, and is measured in milliseconds (ms). The HelloDeadInterval MUST be greater than the HelloInterval, and SHOULD be at least 3 times the value of HelloInterval.

When a node has either sent or received a ConfigAck message, it may begin sending Hello messages. Once it has both sent and received a Hello message, the control channel moves to the UP state. If, however, a node receives a ConfigNack message instead of a ConfigAck message, the node MUST not send Hello messages.

In the event that multiple control channels are run over the same physical control channel interface, the parameter negotiation phase is run multiple times. The various LMP parameter negotiation messages associated with their corresponding control channels are tagged with their node-wide unique identifiers (CCIds).

3.2.2. Fast Keep-alive

Each Hello message contains two sequence numbers: the first sequence number (TxSeqNum) is the sequence number for this Hello message and the second sequence number (RcvSeqNum) is the sequence number of the last Hello message received over this control channel from the adjacent node. Each node increments its sequence number when it sees its current sequence number reflected in Hellos received from its peer. The sequence numbers start at 1 and wrap around back to 2; 0 is used in the RcvSeqNum to indicate that a Hello has not yet been seen and 1 is used in the TxSeqNum to indicate a control channel boot/reboot.

Under normal operation, the difference between the RcvSeqNum in a Hello message that is received and the local TxSeqNum that is generated will be at most 1. There are two cases where this difference can be more than 1: when a control channel reboots and when switching over to a backup control channel.

Having sequence numbers in the Hello messages allows each node to verify that its peer is receiving its Hello messages. This provides a two-fold service. First, the remote node will detect that a control channel has rebooted if TxSeqNum=1. If this occurs, the remote node will indicate its knowledge of the reboot by setting RcvSeqNum=1 in the Hello messages that it sends and SHOULD wait to receive a Hello message with TxSeqNum=2 before transmitting any messages other than Hello messages. Second, by including the RcvSeqNum in Hello packets, the local node will know which Hello

packets the remote node has received.

The following example illustrates how the sequence numbers operate:

1) After completing the configuration stage, Node A sends a Hello message with {TxSeqNum=1;RcvSeqNum=0}.
2) When Node A receives a Hello with {TxSeqNum=1;RcvSeqNum=1}, it sends Hellos with {TxSeqNum=2;RcvSeqNum=1}.
3) After some time, the control channel on Node B reboots.
4) Node A is sending Hellos with {TxSeqNum=45;RcvSeqNum=44} and receives a Hello from Node B with {TxSeqNum=1;RcvSeqNum=0}, indicating that Node B has rebooted. Node A sends Hello messages with {TxSeqNum=45;RcvSeqNum=1}.
4) When Node A receives a Hello with {TxSeqNum=2;RcvSeqNum=45}, it sends Hellos with {TxSeqNum=46;RcvSeqNum=2}.

3.2.3. Control Channel Availability

As mentioned above, LMP requires that a bi-directional control channel is available, and LMP includes mechanisms to ensure that a control channel is available. Control channels may need to be switched as a result of the associated physical control channel interface or link failure, or for administration purposes (e.g., routine fiber maintenance). During these times, peer connectivity must be maintained to ensure that unnecessary rerouting of user traffic is avoided and false failures are not reported.

If multiple active control channels share a common node pair and support the same LMP capabilities, then any of the active control channels may be used without coordination between the local and remote nodes.

3.2.4. Taking a Control Channel Down Administratively

To ensure that bringing a control channel DOWN for administration purposes is done gracefully, a ControlChannelDown flag is available in the Common Header of LMP packets. When data links (ports or component links) are still in use between a pair of nodes, a control channel SHOULD only be taken down administratively when there are other active control channels that can be used to manage the data links.

When a node receives LMP packets with ControlChannelDown = 1, it must first verify that it is able to bring the control channel down on its end. Once the verification is done, it must set the ControlChannelDown flag to 1 on all of the LMP packets that it sends. When the node that initiated the ControlChannelDown procedure receives LMP packets with ControlChannelDown = 1, it may then stop sending Hello packets.

3.2.5. Degraded State

A consequence of allowing the control channels and data links to be transmitted along a separate medium is that the TE link may be in a

state where no active control channels are available, but the data links (ports or component links) are still in use. For many applications, it is unacceptable to tear down a link that is carrying user traffic simply because the control channel is no longer available; however, the traffic that is using the data links may no longer be guaranteed the same level of service. Hence the TE link is in a Degraded state.

When a TE link is in the Degraded state, routing and signaling SHOULD be notified so that new connections are not accepted and resources are no longer advertised for the TE link.

4. Link Property Correlation

As part of LMP, a link property correlation exchange is defined using the LinkSummary, LinkSummaryAck, and LinkSummaryNack messages. The contents of these messages are built using TLV triplets. LinkSummary TLVs can be either negotiable or non-negotiable (identified by the N flag in the TLV header). Negotiable TLVs can be used to let both sides agree on certain link parameters. Non-negotiable TLVs are used for announcment of specific values that do not need, or do not allow, negotiation.

The LinkSummary message is used to aggregate multiple data links (either ports or component links) into a TE link; exchange, correlate, or change TE link parameters; and exchange, correlate, or change Interface Ids (either Port Ids or Component Interface Ids).

The LinkSummary message can be exchanged at any time a link is UP and not in the Verification process. The LinkSummary mesasge MUST be periodically transmitted until (1) the node receives a LinkSummaryAck or LinkSummaryNack message or (2) a timeout expires and no LinkSummaryAck or LinkSummaryNack message has been received. Both the retransmission interval and the timeout period are local configuration parameters.

If the LinkSummary message is received from a remote node and the Interface Id mappings match those that are stored locally, then the two nodes have agreement on the Verification process (see Section 5). If the verification procedure is not used, the LinkSummary message can be used to verify agreement on manual configuration.

Furthermore, any protection definitions that are included in the LinkSummary message MUST be accepted or rejected by the local node. The LinkSummaryAck message is used to signal agreement on the Interface Id mappings and link property definitions. Otherwise, a LinkSummaryNack message MUST be transmitted, indicating which Interface mappings are not correct and/or which link properties are not accepted. If a LinkSummaryNack message indicates that the

Interface Id mappings are not correct and the link verification procedure is enabled, the link verification process SHOULD be repeated for all mismatched free data links; if an allocated data

link has a mapping mismatch, it SHOULD be flagged and verified when it becomes free.

It is possible that the LinkSummary message could grow quite large due to the number of Data Link TLVs. Since the LinkSummary message is IP encoded, normal IP fragmentation should be used if the resulting PDU exceeds the MTU.

5. Verifying Link Connectivity

In this section, we describe an optional mechanism that may be used to verify the physical connectivity of the data-bearing links (either ports or component links). The use of this procedure is negotiated as part of the configuration exchange using the Verification Procedure flag of the LMP Capability TLV. The procedure SHOULD be done when establishing a TE link, and subsequently, on a periodic basis for all unallocated (free) data links of the TE link.

A unique characteristic of all-optical PXCs is that the data-bearing links are not terminated at the PXC, but instead passes through transparently. This characteristic of PXCs poses a challenge for validating the connectivity of the data links since shining unmodulated light through a link may not result in received light at the next PXC. This is because there may be terminating (or opaque) elements, such as DWDM equipment, in between the PXCs. Therefore, to ensure proper verification of data link connectivity, we require that until the links are allocated, they must be opaque. To support various degrees of opaqueness (e.g., examining overhead bytes, terminating the payload, etc.), and hence different mechanisms to transport the Test messages, a Verify Transport Mechanism is included in the BeginVerify and BeginVerifyAck messages. There is no requirement that all data links be terminated simultaneously, but at a minimum, the data links must be able to be terminated one at a time. Furthermore, we assume that the nodal architecture is designed so that messages can be sent and received over any data link. Note that this requirement is trivial for DXCs (and OEO devices in general) since each data link is received electronically before being forwarded to the next DXC, but that in PXCs this is an additional requirement.

To interconnect two nodes, a TE link is added between them, and at a minimum, there MUST be at least one active control channel between the nodes. A TE link MUST include at least one data link.

Once a control channel has been established between the two nodes, data link connectivity can be verified by exchanging Ping-type Test messages over each of the data links specified in the bundled link. It should be noted that all LMP messages except for the Test message

are exchanged over the control channel and that Hello messages continue to be exchanged over the control channel during the data link verification process. The Test message is sent over the data

link that is being verified. Data links are tested in the transmit direction as they are uni-directional, and as such, it may be possible for both nodes to exchange the Test messages simultaneously.

To initiate the link verification process, the local node MUST send a BeginVerify message over the control channel. The BeginVerify message contains fields for the local and remote TE Link Ids. When non-zero, these fields limit the scope of the data links being verified to the corresponding TE link; if the fields are zero, the data links can span multiple TE links and/or they may comprise a TE link that is yet to be configured. The BeginVerify message contains the number of data links that are to be verified; the interval (called VerifyInterval) at which the Test messages will be sent; the encoding scheme, the transport mechanisms that are supported, and data rate for Test messages; when the data links correspond to fibers, the wavelength over which the Test messages will be transmitted is also included.

The BeginVerify message MUST be periodically transmitted until (1) the node receives either a BeginVerifyAck or BeginVerifyNack message to accept or reject the verify process or (2) a timeout expires and no BeginVerifyAck or BeginVerifyNack message has been received. Both the retransmission interval and the timeout period are local configuration parameters.

If the remote node receives a BeginVerify message and it is ready to process Test messages, it MUST send a BeginVerifyAck message back to the local node specifying the desired transport mechanism for the TEST messages. The remote node includes a 32-bit node unique VerifyID in the BeginVerifyAck message. The VerifyID is then used in all corresponding Test messages to differentiate them from different LMP peers and/or parallel Test procedures. When the local node receives a BeginVerifyAck message from the remote node, it may begin testing the data links by transmitting periodic Test messages over each data link. The Test message includes the Verify Id and the local Interface Id for the associated data link. The remote node MUST send either a TestStatusSuccess or a TestStatusFailure message in response for each data link. A TestStatusAck message MUST be sent to confirm receipt of the TestStatusSuccess and TestStatusFailure messages.

The local (transmitting) node sends a given Test message periodically (at least once every VerifyInterval ms) on the corresponding data link until (1) it receives a correlating TestStatusSuccess or TestStatusFailure message on the control channel from the remote (receiving) node or (2) all active control channels between the two nodes have failed. The remote node will send a given TestStatus message periodically over the control

channel until it receives either a correlating TestStatusAck message or an EndVerify message is received over the control channel.

It is also permissible for the sender to terminate the Test procedure without receiving a TestStatusSuccess or TestStatusFailure message by sending an EndVerify message. Message correlation is done using message identifiers and the Verify Id; this enables verification of data links, belonging to different link bundles or LMP sessions, in parallel.

When the Test message is detected at a node, the received Interface Id (used in GMPLS as either a Port Id or Component Interface Id depending on the configuration) is recorded and mapped to the local Interface Id for that channel. The receipt of a TestStatusSuccess message indicates that the Test message was detected at the remote node and the physical connectivity of the data link has been verified. The TestStatusSuccess message includes the local Interface Id and the remote Interface Id (received in the Test message), along with the VerifyId received in the Test message. When the TestStatusSuccess message is received, the local node SHOULD mark the data link as UP, send a TestStatusAck message to the remote node, and begin testing the next data link. If, however, the Test message is not detected at the remote node within an observation period (specified by the VerifyDeadInterval), the remote node will send a TestStatusFailure message over the control channel indicating that the verification of the physical connectivity of the data link has failed. When the local node receives a TestStatusFailure message, it will mark the data link as FAILED, send a TestStatusAck message to the remote node, and begin testing the next data link. When all the data links on the list have been tested, the local node SHOULD send an EndVerify message to indicate that testing has been completed on this link. The EndVerify message will be periodically transmitted until an EndVerifyAck message has been received.

Both the local and remote nodes SHOULD maintain the complete list of Interface Id mappings for correlation purposes.

5.1. Example of Link Connectivity

Figure 1 shows an example of the link verification scenario that is executed when a link between PXC A and PXC B is added. In this example, the TE link consists of three free ports (each transmitted along a separate fiber) and is associated with a bi-directional control channel (indicated by a "c"). The verification process is as follows: PXC A sends a BeginVerify message over the control channel ôcö to PXC B indicating it will begin verifying the ports. PXC B receives the BeginVerify message and returns the BeginVerifyAck message over the control channel to PXC A. When PXC A receives the BeginVerifyAck message, it begins transmitting periodic Test messages over the first port (Interface Id=1). When PXC B receives the Test messages, it maps the received Interface Id to its own

local Interface Id = 10 and transmits a TestStatusSuccess message over the control channel back to PXC A. The TestStatusSuccess message will include both the local and received Interface Ids for

the port. PXC A will send a TestStatusAck message over the control channel back to PXC B indicating it received the TestStatusSuccess message. The process is repeated until all of the ports are verified. At this point, PXC A will send an EndVerify message over the control channel to PXC B to indicate that testing is complete and PXC B will respond by sending an EndVerifyAck message over the control channel back to PXC A.

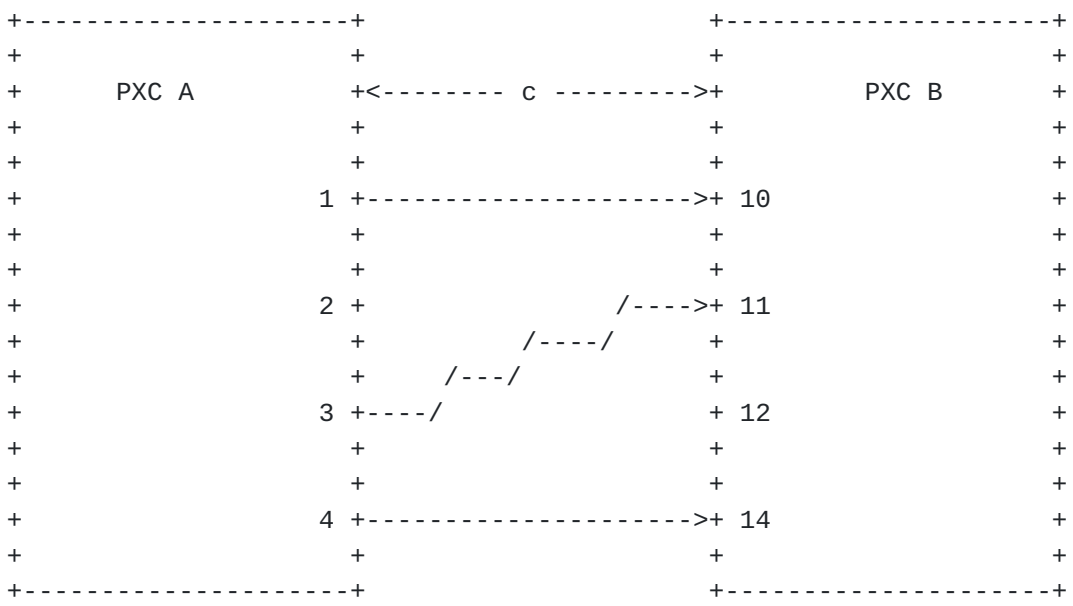


Figure 2: Example of link connectivity between PXC A and PXC B.

6. Fault Management

In this section, we describe an optional LMP mechanism that is used to manage failures by rapidly locating link or channel failures. The use of this procedure is negotiated as part of the configuration exchange using the Fault Management Procedure flag of the LMP Capability TLV. As before, we assume each link has a bi-directional control channel that is always available for inter-node communication and that the control channel spans a single hop between two neighboring nodes. The case where a control channel is no longer available between two nodes is beyond the scope of this draft. The mechanism used to rapidly isolate link failures is designed to work for unidirectional LSPs, and can be easily extended to work for bi-directional LSPs; however, for the purposes of this document, we only discuss the operation when the LSPs are unidirectional.

Recall that a TE link connecting two nodes may consist of a number of data links (ports or component links). If one or more data links fail between two nodes, a mechanism must be used to rapidly locate the failure so that appropriate protection/restoration mechanisms

can be initiated. An important implication of using PXCs is that traditional methods that are used to monitor the health of allocated data links in OEO nodes (e.g., DXCs) may no longer be appropriate,

since PXCs are transparent to the bit-rate, format, and wavelength. Instead, fault detection is delegated to the physical layer (i.e., loss of light or optical monitoring of the data) instead of layer 2 or layer 3.

6.1. Fault Detection

As mentioned earlier, fault detection must be handled at the layer closest to the failure; for optical networks, this is the physical (optical) layer. One measure of fault detection at the physical layer is simply detecting loss of light (LOL). Other techniques for monitoring optical signals are still being developed and will not be further considered in this document. However, it should be clear that the mechanism used to locate the failure is independent of the mechanism used to detect the failure, but simply relies on the fact that a failure is detected.

6.2. Fault Localization Mechanism

If data links fail between two PXCs, the power monitoring system in all of the downstream nodes may detect LOL and indicate a failure. To correlate multiple failures between a pair of nodes, a monitoring window can be used in each node to determine if a single data link has failed or if multiple data links (possibly an entire TE link) have failed.

As part of the fault localization, a downstream node that detects data link failures will send a ChannelFail message to its upstream neighbor (bundling together the notification of all of the failed data links). An upstream node that receives the ChannelFail message will correlate the failure to see if there is a failure on the corresponding LSP(s). If the failure has also been detected on the input port(s) of the upstream node, the node will return a ChannelFailAck message to the downstream node (bundling together the notification of all the data links), indicating that it too has detected a failure. If, however, the fault is CLEAR in the upstream node (e.g., there is no LOL on the corresponding input channels), then the upstream node will have localized the failure and will return a ChannelFailNack message to the downstream node. Once the failure has been localized, the signaling protocols can be used to initiate span or path protection/restoration procedures.

If all of the data links of a TE link have failed, then the upstream node MAY be notified of the TE link failure without specifying that each data link of the TE link has failed. To do this, the Interface Id of the ChannelFail subobject MUST be 0.

6.3. Channel Activiation Indication

The ChannelActive message is the counterpart to the ChannelFail message described in Section 6.2 and is used to notify the downstream neighboring node that the data link is in the Active

state. This is particularly useful in networks with transparent nodes where the status of data links may need to be triggered using control channel messages. For example, if a data link is pre-provisioned and the physical link fails after verification and before inserting user traffic, the pair of nodes need a mechanism to indicate the data link is active or they may not be able to detect the failure.

The ChannelActive message is used to indicate that a channel or group of channels are now active. The ChannelActiveAck message MUST be transmitted upon receipt of a ChannelActive message. When a ChannelActive message is received, the corresponding data link(s) MUST be put into the Active state. If upon putting them into the Active state, a failure is detected, the ChannelFail message MUST be transmitted as described in Section 6.2.

6.4. Examples of Fault Localization

In Fig. 2, a sample network is shown where four PXCs are connected in a linear array configuration. The control channels are bi-directional and are labeled with a "c". All LSPs are uni-directional going left to right.

In the first example [see Fig. 2(A)], there is a failure on a single data link between PXC2 and PXC3. Both PXC3 and PXC4 will detect the failure and each node will send a ChannelFail message to the corresponding upstream node (PXC3 will send a message to PXC2 and PXC4 will send a message to PXC3). When PXC3 receives the ChannelFail message from PXC4, it will correlate the failure and return a ChannelFailAck message back to PXC4. Upon receipt of the ChannelFailAck message, PXC4 will move the associated ports into a standby state. When PXC2 receives the ChannelFail message from PXC3, it will correlate the failure, verify that it is CLEAR, localize the failure to the data link between PXC2 and PXC3, and send a ChannelFailNack message back to PXC3.

In the second example [see Fig. 2(B)], there is a failure on three data links between PXC3 and PXC4. In this example, PXC4 has correlated the failures and will send a bundled ChannelFail message for the three failures to PXC3. PXC3 will correlate the failures, localize them to the channels between PXC3 and PXC4, and return a bundled ChannelFailNack message back to PXC4.

In the last example [see Fig. 2(C)], there is a failure on the tributary link of the ingress node (PXC1) to the network. Each downstream node will detect the failure on the corresponding input ports and send a ChannelFail message to the upstream neighboring node. When PXC2 receives the message from PXC3, it will correlate the ChannelFail message and return a ChannelFailAck message to PXC3

(PXC3 and PXC4 will also act accordingly). Since PXC1 is the ingress node to the optical network, it will correlate the failure and

localize the failure to the data link between itself and the network element outside the optical network.

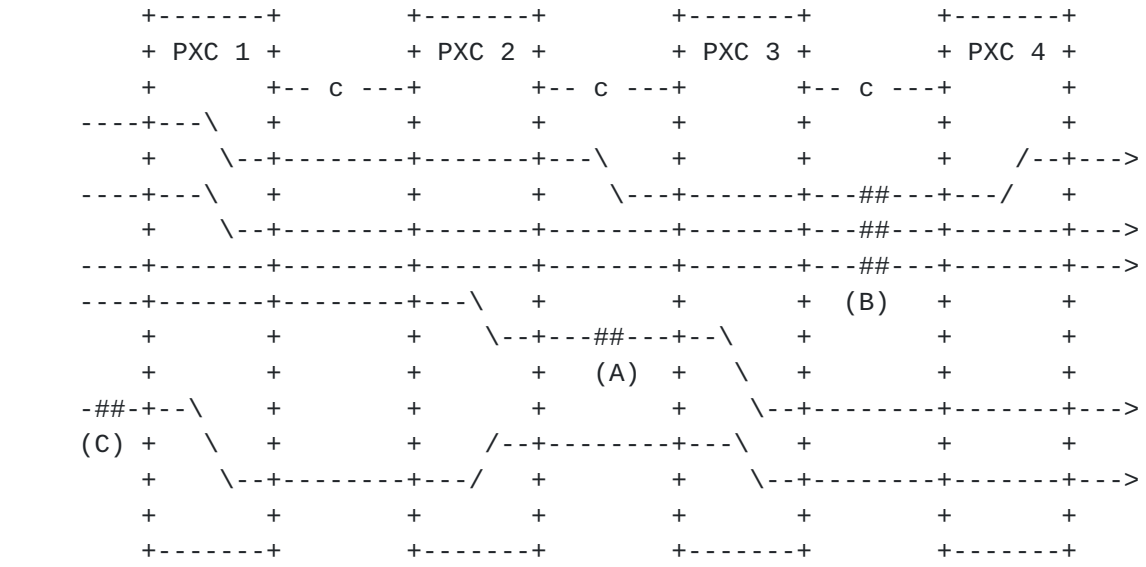


Figure 3: We show three types of data link failures (indicated by ## in the figure): (A) a single data link fails between two PXCs, (B) three data links fail between two PXCs, and (C) a single data link fails on the tributary input of PXC 1. The control channel connecting two PXCs is indicated with a "c".

7. LMP Authentication

LMP authentication is optional (included in the Common Header) and, if used, MUST be supported by both sides of the control channel. The method used to authenticate LMP packets is based on the authentication technique used in [OSPF]. This uses cryptographic authentication using MD5.

As a part of the LMP authentication mechanism, a flag is included in the LMP common header indicating the presence of authentication information. Authentication information itself is appended to the LMP packet. It is not considered to be a part of the LMP packet, but is transferred in the same IP packet.

When the Authentication flag is set in the LMP packet header, an authentication data block is attached to the packet. This block has a standard authentication header and a data portion. The contents of the data portion depend on the authentication type. Currently, only MD5 is supported for LMP.

8. LMP Finite State Machines

8.1. Control Channel FSM

The control channel FSM defines the states and logics of operation of an LMP control channel. The description of FSM state transitions and associated actions is given in Section 3.

8.1.1. Control Channel States

A control channel can be in one of the states described below. Every state corresponds to a certain condition of the control channel and is usually associated with a specific type of LMP message that is periodically transmitted to the far end.

Down: This is the initial control channel state. In this state, no attempt is being made to bring the control channel up and no LMP messages are sent. The control channel parameters should be set to the initial values.

ConfigSnd: The control channel is in the parameter negotiation state. In this state the node periodically sends a Config message, and is expecting the other side to reply with either a ConfigAck or ConfigNack message. The FSM does not transition into the Active state until the remote side positively acknowledges the parameters.

ConfRcv: The control channel is in the parameter negotiation state. In this state, the node is waiting for acceptable configuration parameters from the remote side. Once such parameters are received and acknowledged, the FSM can transition to the Active state.

Active: In this state the node periodically sends a Hello message and is waiting to receive a valid Hello message. Once a valid Hello message is received, it can transition to the UP state.

Up: The CC is in an operational state. The node receives valid Hello messages and sends Hello messages.

GoingDown: A CC may go into this state because of two reasons: administrative action, and a ControlChannelDown bit received in an LMP message. While a CC is in this state, the node sets the ControlChannelDown bit in all the messages it sends.

8.1.2. Control Channel Events

Operation of the LMP control channel is described in terms of FSM states and events. Control channel Events are generated by the underlying protocols and software modules, as well as by the packet processing routines and FSMs of associated TE links. Every event

has its number and a symbolic name. Description of possible control channel events is given below.

1 : evBringUp: This is an externally triggered event indicating that the control channel negotiation should begin. This event, for example, may be triggered by a provisioner command or by the successful completion of a control channel bootstrap procedure. Depending on the configuration, this will trigger either
    1a) the sending of a Config message,
    1b) a period of waiting to receive a Config message from the remote node.

2 : evCCDn: This event is generated when there is indication that the control channel is no longer available.

3 : evConfDone: This event indicates a ConfigAck message has been received, acknowledging the Config parameters.

4 : evConfErr: This event indicates a ConfigNack message has been received, rejecting the Config parameters.

5 : evNewConfOK: New Config message was received from neighbor and positively Acknowledged.

6 : evNewConfErr: New Config message was received from neighbor and rejected with a ConfigNack message.

7 : evContenWin: New Config message was received from neighbor at the same time a Config message was sent to the neighbor. The Local node wins the contention. As a result, the received Config message is ignored.

8 : evContenLost: New Config message was received from neighbor at the same time a Config message was sent to the neighbor. The Local node looses the contention. As a result, the node must (positively or negatively) respond to the Config message.

9 : evAdminDown: The administrator has requested that the control channel is brought down administratively.

10: evDownOk: A packet with the LinkDown flag has been received and the local node was the initiator of the link down procedure.

11: evDownErr: A timer has expired indicating that the other node did not start setting the LinkDown flag in its messages.

12: evNbrGoesDn: A packet with LinkDown flag is received from the

neighbor.

13: evHelloRcvd: A Hello packet with expected SeqNum has been received.

14: evHoldTimer: The HelloDeadInterval timer has expired indicating that no Hello packet has been received. This moves the control channel back into the Negotiation state, and depending on the local configuration, this will trigger either

14a) the sending of periodic Config messages,

14b) a period of waiting to receive Config messages from the remote node.

15: evSeqNumErr: A Hello with unexpected SeqNum received and discarded.

16: evReconfig: Control channel parameters have been reconfigured and require renegotiation.

17: evConfRet: A retransmission timer has expired and a Config message is resent.

18: evHelloRet: The HelloInterval timer has expired and a Hello packet is sent.

8.1.3 Control Channel FSM Description

Figure 4 illustrates operation of the control channel FSM in a form of FSM state transition diagram.

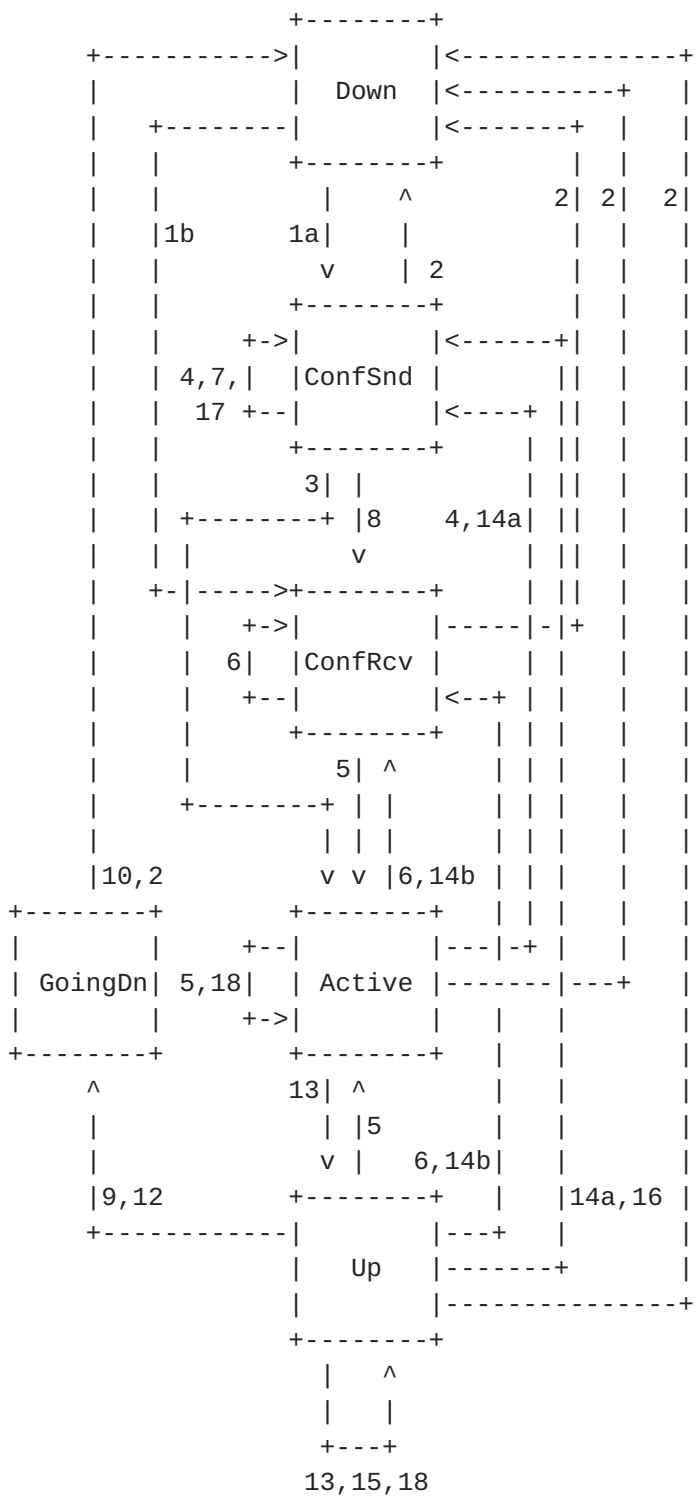


Figure 4: Control Channel FSM

Event evCCDn always forces the FSM to the Down State. Events evHoldTimer evReconfig always force the FSM to the Negotiation state (either ConfigSnd or ConfigRcv).

8.2 TE Link FSM

The TE Link FSM defines the states and logics of operation of an LMP TE Link.

8.2.1 TE Link States

An LMP TE link can be in one of the states described below. Every state corresponds to a certain condition of the TE link and is usually associated with a specific type of LMP message that is periodically transmitted to the far end via the associated control channel or in-band via the data links.

Down: There are no control channels available and no data links are allocated to the TE link.

LinkVrf: In this state, the link verification procedure is performed for the data links of the TE link. LinkVrf is a composite state that consists of two substates described below.

VrfBegin: This state is valid only for the side initiating the verification process. In this state, the node periodically sends a BeginVerify message and expects an BeginVerifyAck or BeginVerifyNack message. The BeginVerify messages include information about the data links in the BegVerify state.

VrfProcess: In this state, two FSMs are performing the link verification procedure. The initiator periodically sends Test messages over the data links in the Testing state and waits for TestStatus messages to be received over a control channel. The passive side listens for incoming link test messages on the data links in the PasvTst state.

VrfResult: In this state, the passive side periodically retransmits the TestStatus messages for the data links verified during the link verification procedure and waits for acknowledgement. Once all messages have been acknowledged, the passive side can go out of VrfResult state. The initiator waits for the incoming TestStatus message and goes out of it after receiving and acknowledging TestStatus messages for all data links. Note that the initiator must be prepared to receive and acknowledge the TestStatus messages even after it has transitioned out of the VrfResult state.

Summary: In this state, the new TE link configuration is

announced by periodically sending the LinkSummary
messages over the control channel.

Up: This is the normal operational state of the TE link. At least one primary CC is required to be operational between the nodes sharing the TE link.

Degraded: In this state, all primary CCs are down, but the TE link still includes some allocated data links.

STDBY: A ChannelFail message has been received indicating a failure has been detected on the far-end node of the TE link. The failure is locally correlated to determine if the failure can be localized to the TE link or if the failure is further upstream along the path.

8.2.2 TE Link Events

Operation of the LMP TE link is described in terms of FSM states and events. TE Link events are generated by the packet processing routines and by the FSMs of the associated primary control channel(s) and the data links. Every event has its number and a symbolic name. Description of possible control channel events is given below.

1 : evCCUp: First primary CC goes Up
2 : evCCDown: Last primary CC goes Down
3 : evVerDone: Verification done; EndVerifyAck message received.
4 : evVerify: An external event indicates that the Link verification procedure should begin.
5 : evRecnfReq: TE link has been reconfigured and the new configuration needs to be announced/agreed upon.
6 : evSummaryAck: LinkSummaryAck message has been received acknowledging the TE link configuration.
7 : evLastCompDn: The last allocated data link has been freed.
8 : evStartVer: BeginVerifyAck message has been received indicating the remote node is ready to start link verification.
9 : evTELinkOk: An external event has indicated that the TE link is available.
10: evBeginRet: Retransmission timer expires and no BeginVerifyAck or BeginVerifyNack message has been received. BeginVerify message is resent.
11: evSummaryRet: Retransmission timer expires and no LinkSummaryAck or LinkSummaryNack message has been received. LinkSummary message is resent.
12: evChannFail: ChannelFail message is received for TE link. The failure is locally correlated and either a ChannelFailAck or a ChannelFailNack message is transmitted.

```
   13: evNodeReBoot:   The neighboring node has rebooted.
   14: evSummaryNack1: LinkSummaryNack message has been received
                       indicating negotiable parameters not accepted.
```

15: evSummaryNack2 LinkSummaryNack message received indicating misconfiguration of non-negotiable parameters. Free ports that are misconfigured are moved to Down state. Allocated ports that are misconfigured are flagged.

16: evSummaryNack3: LinkSummaryNack message has been received indicating misconfiguration of non-negotiable parameters for all ports.

8.2.3 TE Link FSM Description

Figure 5 illustrates operation of the LMP TE Link FSM in a form of FSM state transition diagram.

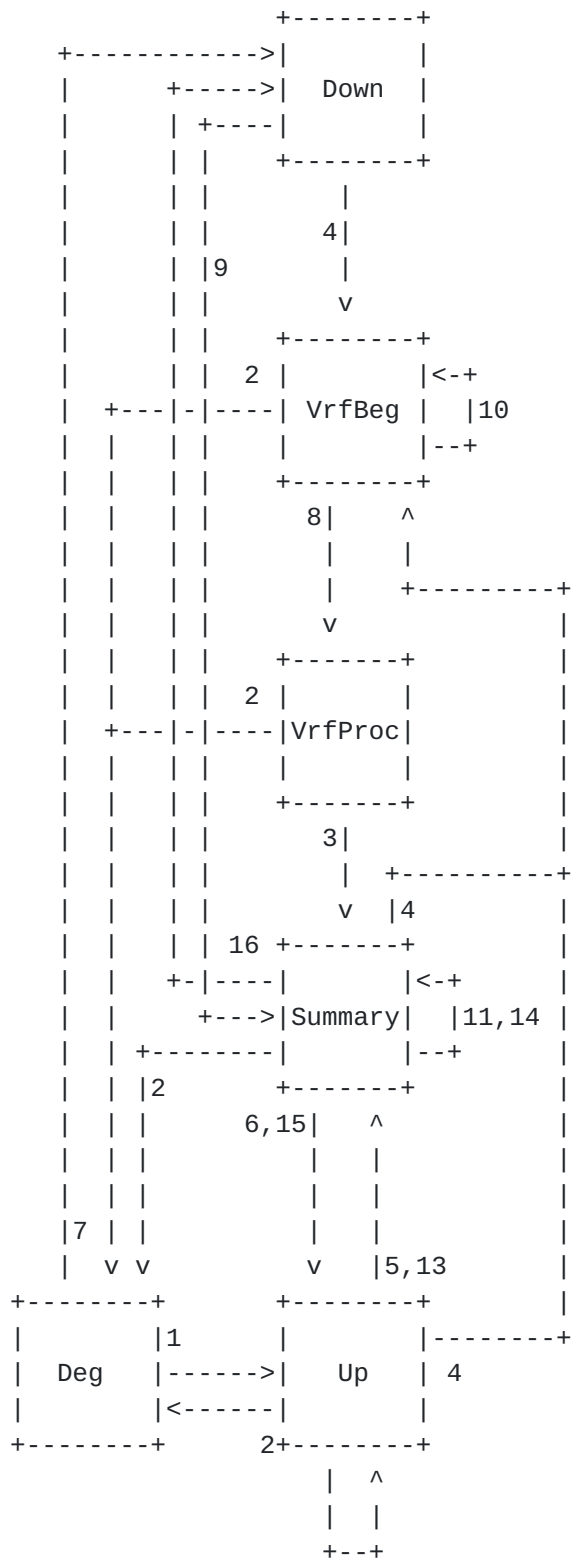


Figure 5: LMP TE Link FSM

8.3 Data Link FSM

The data link FSM defines the states and logics of operation of a port or component link within an LMP TE link. Operation of a data link is described in terms of FSM states and events. Data-bearing links can either be in the active (transmitting) state, where Test messages are transmitted from them, or the passive (receiving) state, where Test messages are received through them. For clarity, we define separate FSMs for the active/passive data-bearing links; however, we define a single set of data link states and events.

8.3.1 Data Link States

Any data link can be in one of the states described below. Every state corresponds to a certain condition of the TE link.

Down: The data link has not been put in the resource pool.

Test: The data link is being tested. An LMP Test message is periodically sent through the link.

PasvTest: The data link is being checked for incoming test messages.

Retest: The data link is being re-validated. An LMP Test message is periodically sent through the link.

PasvRetest: The data link is being checked for incoming test.messages as part of link re-validation.

Up/Free: The link has been successfully tested and is now put in the pool of resources. The link has not yet been allocated to data traffic.

Up/Allocated: The link has been allocated for data traffic.

Degraded: The link was in the Up/Allocated state when the last CC associated with data link's TE Link has gone down. The link is put in the Degraded state, since it is still being used for data LSP.

TstRecv: A Test message has been detected on the data link and a TestStatusSuccess message has been sent to the transmitter over the control channel.

8.3.2 Data Link Events

Data bearing link events are generated by the packet processing

routines and by the FSMs of the associated control channel and the

TE link. Every event has its number and a symbolic name. Description of possible data link events is given below:

1 :evCCUp: CC has gone up.

2 :evCCDown: LMP neighbor connectivity is lost. This indicates the last LMP control channel has failed between neighboring nodes.

3 :evStartTst: This is an external event that triggers the sending of Test messages over the data bearing link.

4 :evStartPsv: This is an external event that triggers the listening for Test messages over the data bearing link.

5 :evTestOK: Link verification was successful and the link can be used for path establishment.

(a) This event indicates the Link Verification procedure (see Section 5) was successful for this data link and a TestStatusSuccess message was received over the control channel.

(b) This event indicates the link is ready for path establishment, but the Link Verification procedure was not used. For in-band signaling of the control channel, the control channel establishment may be sufficient to verify the link.

6 :evTestRcv: Test message was received over the data port and a TestStatusSuccess message is transmitted over the control channel.

7 :evTestFail: Link verification returned negative results. This could be because (a) a ChannelStatusFailure message was received, or (b) an EndVerifyAck message was received without receiving a ChannelStatusSuccess or ChannelStatusFailure message for the data link.

8 :evPsvTestFail:Link verification returned negative results. This indicates that a Test message was not detected and either (a) the VerifyDeadInterval has expired or (b) an EndVerifyAck messages has been received and the VerifyDeadInterval has not yet expired.

9 :evLnkAlloc: The data link has been allocated.

10:evLnkDealloc: The data link has been deallocated.

11:evTestRet: A retransmission timer has expired and the Test message is resent.

11:evVerifyAbrt: The other side did not confirm it is ready to perform link verification.

12:evSummaryFail:The LinkSummary did not match for this data port.

7.3.3 Active Data Link FSM Description

Figure 6 illustrates operation of the LMP active data link FSM in a form of FSM state transition diagram.

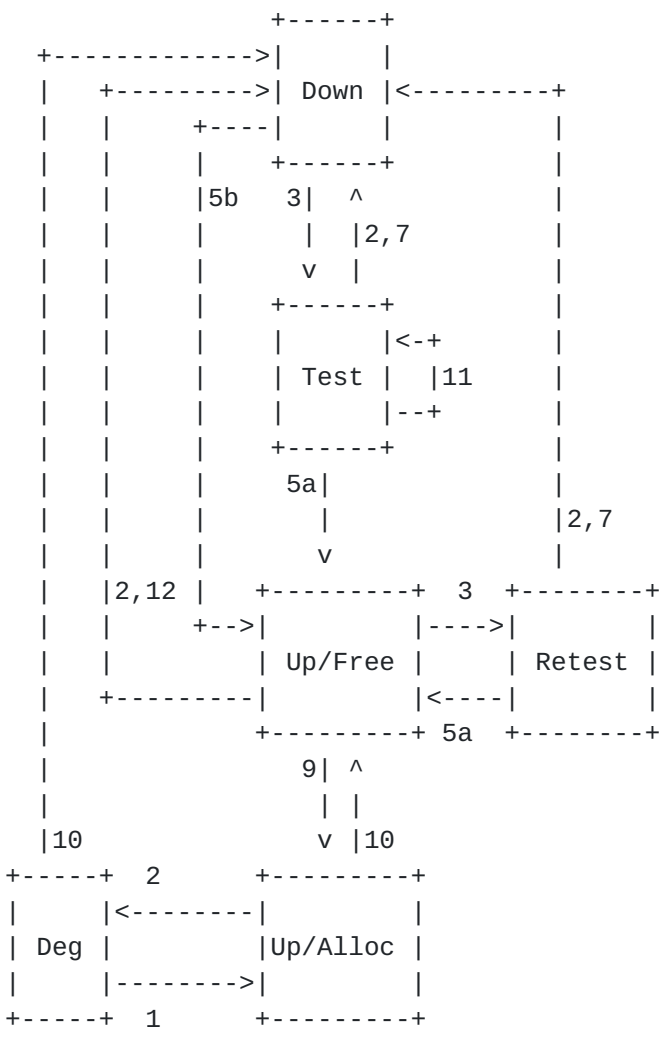


Figure 6: Active LMP Data Link FSM

8.3.3 Passive Data Link FSM Description

Figure 7 illustrates operation of the LMP passive data link FSM in a form of FSM state transition diagram.

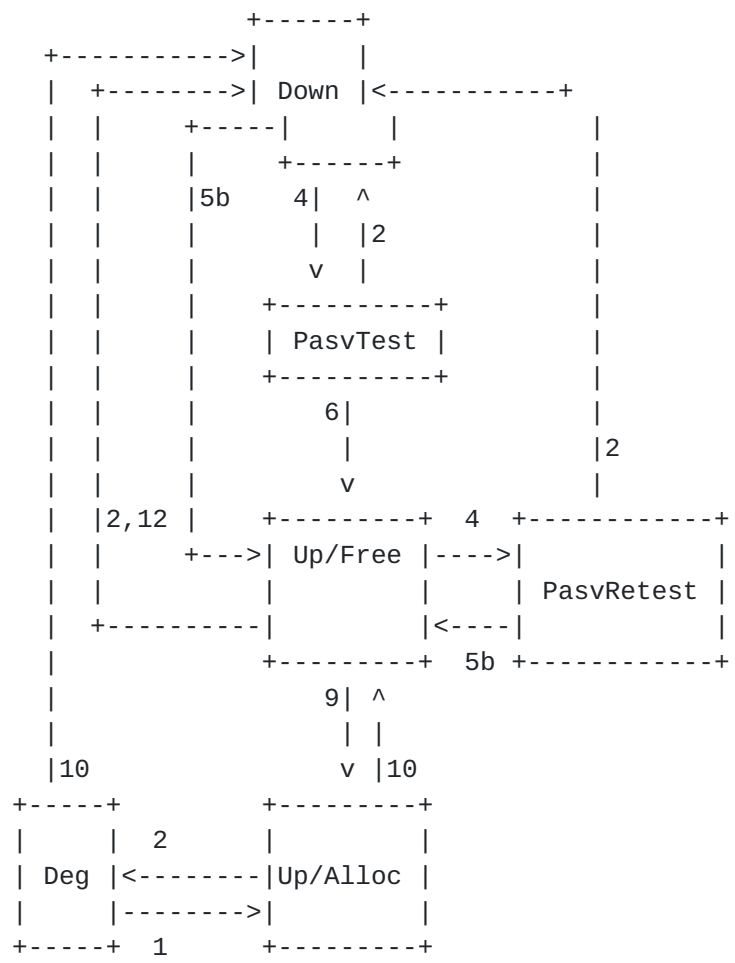


Figure 7: Passive LMP Data Link FSM

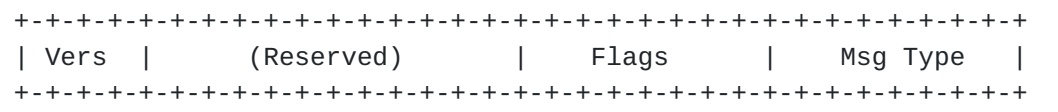
9. LMP Message Formats

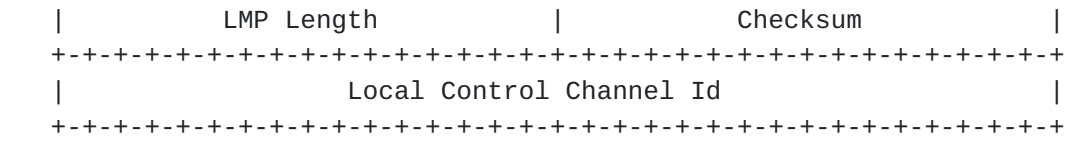
All LMP messages are IP encoded (except, in some cases, the Test message are limited by the transport mechanism for in-band messaging) with protocol Id = 140 (value not yet assigned by IANA).

9.1. Common Header

In addition to the standard IP header, all LMP messages (except, in some cases, the Test messages are limited by the transport mechanism for in-band messaging) have the following common header:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```


| Vers | (Reserved) | Flags | Msg Type |



Vers: 4 bits

Protocol version number. This is version 1.

Flags: 8 bits. The following values are defined. All other values are reserved.

0x01: ControlChannelDown

0x02: Node Reboot

This bit is set to indicate the node has rebooted. This flag may be reset to 0 when a Hello message is received with RcvSeqNum equal to the local TxSeqNum.

0x04: DWDM Node

If this bit is set, the node is identifying itself as a DWDM system. This is used when running LMP-DWDM extensions as defined in [LMP-DWDM].

0x08: Authenticatino

When set, this bit indicates that an authentication block is attached at the end of the LMP message. See Sections 7 and 8.3 for more details.

Msg Type: 8 bits. The following values are defined. All other values are reserved.

1 = Config

2 = ConfigAck

3 = ConfigNack

4 = Hello

5 = BeginVerify

6 = BeginVerifyAck

7 = BeginVerifyNack

8 = EndVerify

```
        9  = EndVerifyAck
```

10 = Test

11 = TestStatusSuccess

12 = TestStatusFailure

13 = TestStatusAck

14 = LinkSummary

15 = LinkSummaryAck

16 = LinkSummaryNack

17 = ChannelFail

18 = ChannelFailAck

19 = ChannelFailNack

20 = ChannelActive

21 = ChannelActiveAck

All of the messages are sent over the control channel EXCEPT the Test message which is sent over the data link that is being tested.

LMP Length: 16 bits

The total length of this LMP message in bytes, including the common header and any variable-length objects that follow.

Checksum: 16 bits

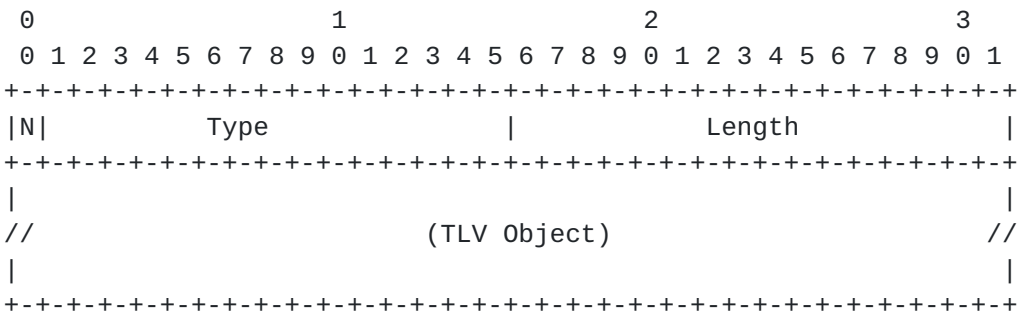
The standard IP checksum of the entire contents of the LMP message, starting with the LMP message header. This checksum is calculated as the 16-bit one's complement of the one's complement sum of all the 16-bit words in the packet. If the packet's length is not an integral number of 16-bit words, the packet is padded with a byte of zero before calculating the checksum.

Local Control Channel Id: 32 bits

The Local Control Channel Id (CCId) identifies the control channel of the sender associated with the message and is node-wide unique. This value MAY be ignored upon receipt of the Test message.

9.2 LMP TLV Format

Many LMP messages are TLV based. The format the LMP TLV is as follows:



N: 1 bit

The N flag indicates if the object is a negotiable parameter (N=1) or a non-negotiable parameter (N=0).

Type: 15 bits

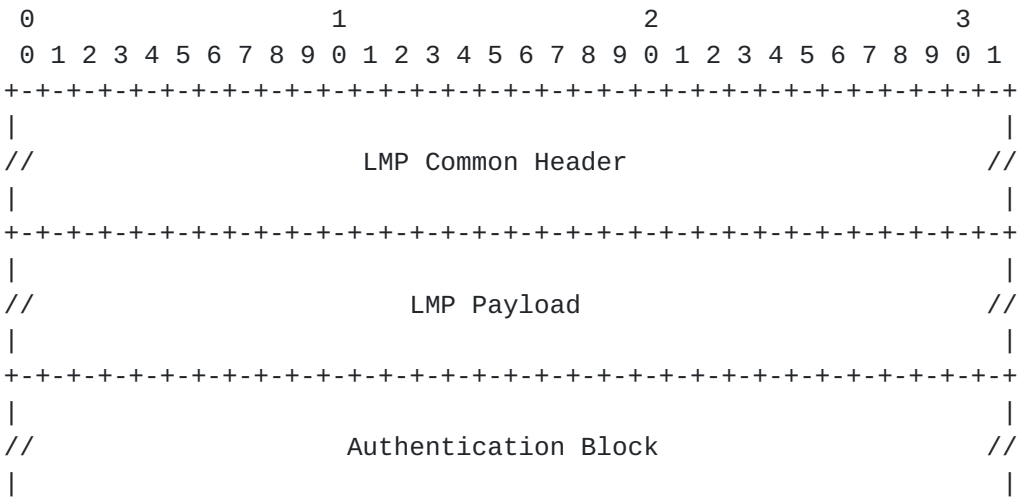
The Type field indicates the TLV type.

Length: 16 bits

The Length field indicates the length of the TLV object in bytes.

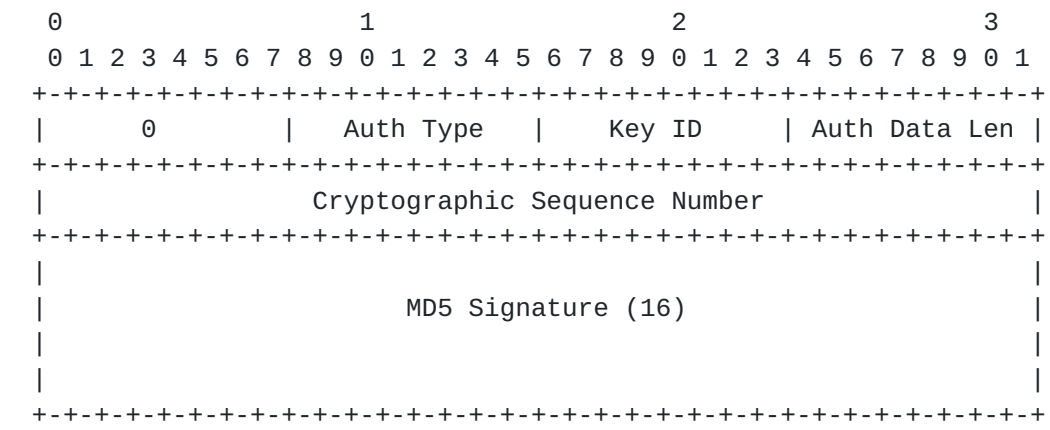
9.3 Authentication

When authentication is used for LMP, the authentication itself is appended to the LMP packet. It is not considered to be a part of the LMP packet, but is transmitted in the same IP packet as shown below:



```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The authentication block looks as follows:



Auth Type: 8 bits

This defines the type of authentication used for LMP messages. The following authentication types are defined, all other are reserved for future use:

0 No authentication
1 Cryptographic authentication

Key ID: 8 bits

This field is defined only for cryptographic authentication.

Auth Data Length: 8 bits

This field contains the length of the data portion of the authentication block.

LMP authentication is performed on a per control channel basis. The packet authentication procedure is very similar to the one used in OSPF, including multiple key support, key management, etc. The details specific to LMP are defined below.

Sending authenticated packets

When a packet needs to be sent over a control channel and an authentication method is configured for it, the Authentication flag in the LMP header is set to 1, the LMP Length field is set to the length of the LMP packet only, not including the authentication block.

1) The Checksum field in the LMP packet is set to zero (this will make the receiving side drop the packet if authentication is not supported).
2) The LMP authentication header is filled out properly. The message

digest is calculated over the LMP packet together with the LMP authentication header. The input to the message digest

calculation consists of the LMP packet, the LMP authentication header, and the secret key. When using MD5 as the authentication algorithm, the message digest calculation proceeds as follows:

(a) The authentication header is appended to the LMP packet.
(b) The 16 byte MD5 key is appended after the LMP authentication header.
(c) Trailing pad and length fields are added, as specified in [MD5].
(d) The MD5 authentication algorithm is run over the concatenation of the LMP packet, authentication header, secret key, pad and length fields, producing a 16 byte message digest (see [MD5]).
(e) The MD5 digest is written over the secret key (i.e., appended to the original authentication header).

The authentication block is added to the IP packet right after the LMP packet, so IP packet length includes the length of both LMP packet and LMP authentication blocks.

Receiving authenticated packets

When an LMP packet with the Authentication flag set has been received on a control channel that is configured for authentication, it must be authenticated. The value of the Authentication field MUST match the authentication type configured for the control channel (if any).

If an LMP protocol packet is accepted as authentic, processing of the packet continues. Packets that fail authentication are discarded. Note that the checksum field in the LMP packet header is not checked when the packet is authenticated.

(1) Locate the receiving control channel's configured key having Key ID equal to that specified in the received LMP authentication block. If the key is not found, or if the key is not valid for reception (i.e., current time does not fall into the key's active time frame), the LMP packet is discarded.
(2) If the cryptographic sequence number found in the LMP authentication header is less than the cryptographic sequence number recorded in the control channel data structure, the LMP packet is discarded.
(3) Verify the message digest in the data portion of the authentication block in the following steps:
    (a) The received digest is set aside.
    (b) A new digest is calculated, as specified in the previous section.
    (c) The calculated and received digests are compared. If they do not match, the LMP packet is discarded. If they do

match, the LMP protocol packet is accepted as authentic, and the "cryptographic sequence number" in the control channel's

data structure is set to the sequence number found in the packet's LMP header.

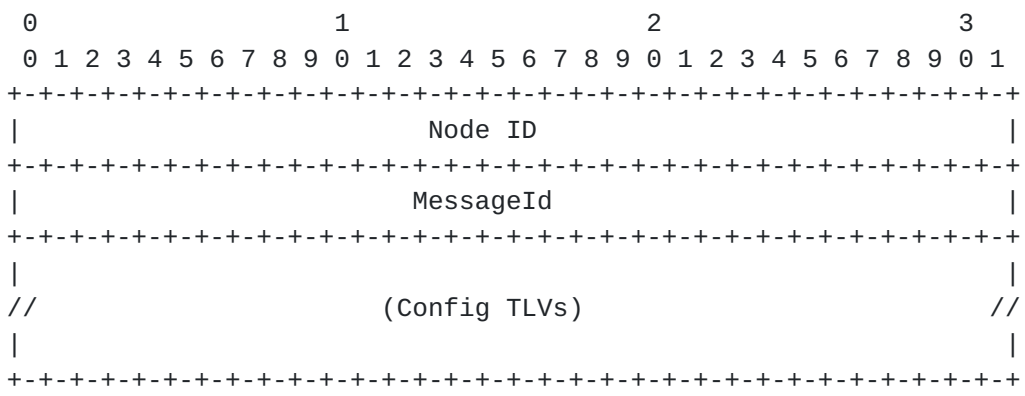
9.4 Parameter Negotiation

9.4.1 Config Message (MsgType = 1)

The Config message is used in the negotiation phase of LMP. The contents of the Config message are built using TLV triplets. TLVs can be either negotiable or non-negotiable (identified by the N flag in the TLV header). Negotiable TLVs can be used to let the devices agree on certain values. Non-negotiable TLVs are used for announcement of specific values that do not need or do not allow negotiation. The format of the Config message is as follows:

<Config Message> ::= <Common Header> <Config>

The Config Object has the following format:



Node ID: 32 bits.

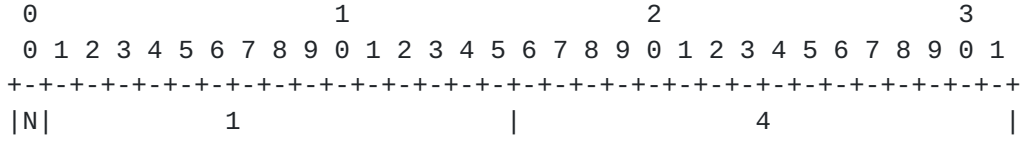
This is the Node ID for the node.

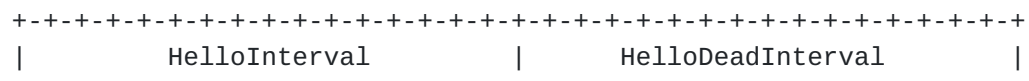
MessageId: 32 bits.

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgment.

9.4.1.1 HelloConfig TLV

The HelloConfig TLV is TLV Type=1 and is defined as follows:




HelloInterval
HelloDeadInterval

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Length field of HelloConfig is always set to 4.

N: 1 bit

The N flag indicates if the parameter is negotiable (N=1) or non-negotiable (N=0).

HelloInterval: 16 bits.

Indicates how frequently the Hello packets will be sent and is measured in milliseconds (ms).

HelloDeadInterval: 16 bits.

If no Hello packets are received within the HelloDeadInterval, the control channel is assumed to have failed and is measured in milliseconds (ms).

9.4.1.2 LMP Capability TLV

The LMP Capability TLV is TLV Type=2 and is defined as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|N|           2                 |               4               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Capability Flags                     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Length field of LMP Capability TLV is always set to 4.

N: 1 bit

The N flag indicates if the parameter is negotiable (N=1) or non-negotiable (N=0).

Capability Flags: 32 bits

The Capability Flags indicate which extended LMP procedures will be supported. A value of 0 indicates that only the base LMP procedures are supported. More than one bit may be set to indicate multiple extended LMP procedures are supported.

The following flags are defined:

0x01 Link Verification Procedure

0x02 Fault Management Procedure

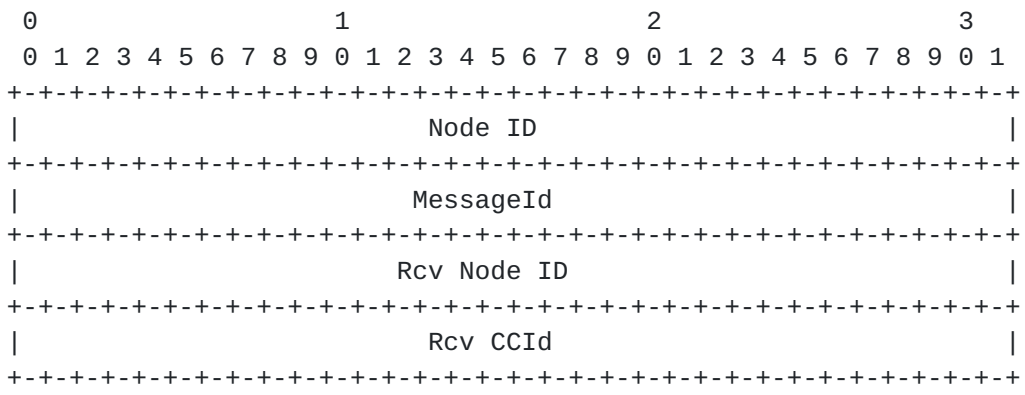
0x04 LMP-DWDM Procedure. See [LMP-DWDM].

9.4.2 ConfigAck Message (MsgType = 2)

The ConfigAck message is used to indicate the receipt of the Config message and indicate agreement on all parameters.

<ConfigAck Message> ::= <Common Header> <ConfigAck>

The ConfigAck Object has the following format:



Node ID: 32 bits.

This is the Node ID for the node sending the ConfigAck message.

MessageId: 32 bits.

This is copied from the Config message being acknowledged.

Rcv Node ID: 32 bits.

This is copied from the Config message being acknowledged.

Rcv CCId: 32 bits

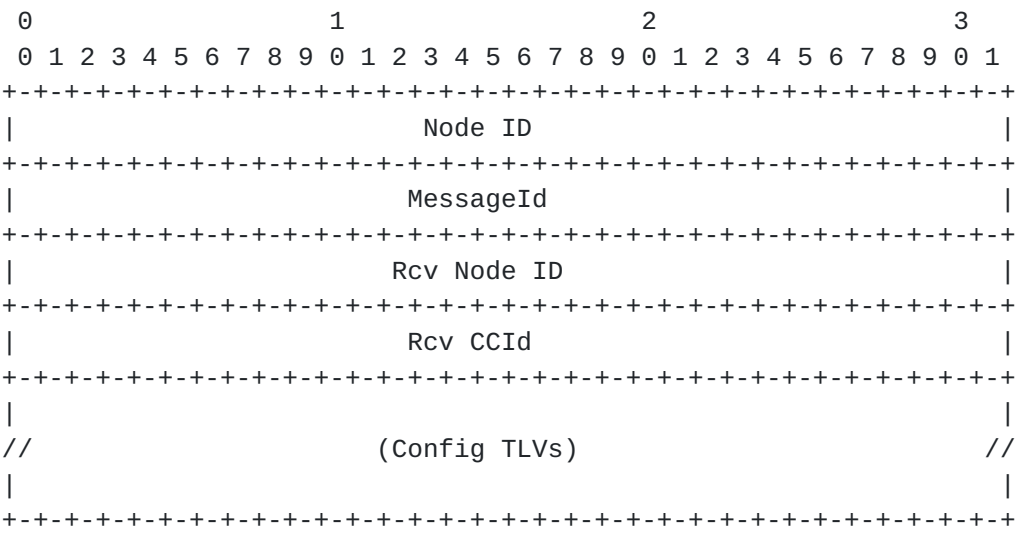
This is the Control Channel Id copied from the Common Header of the Config message being acknowledged.

9.4.3 ConfigNack Message (MsgType = 3)

The ConfigNack message is used to indicate disagreement on non-negotiable parameters or propose other values for negotiable parameters. Parameters where agreement was reached MUST NOT be included in the ConfigNack Object. The format of the ConfigNack message is as follows:

<ConfigNack Message> ::= <Common Header> <ConfigNack>

The ConfigNack Object has the following format:



Node ID: 32 bits.

This is the Node ID for the node.

MessageId: 32 bits.

This is copied from the Config message being negatively acknowledged.

Rcv Node ID: 32 bits.

This is copied from the Config message being acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the Config message being negatively acknowledged.

The Config TLVs MUST include acceptable values for all negotiable parameters. If the ConfigNack includes Config TLVs for non-negotiable parameters, they MUST be copied from the Config TLVs received in the Config message.

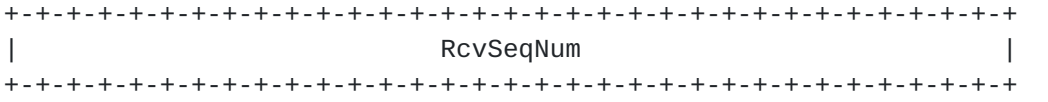
9.5 Hello Message (MsgType = 4)

The format of the Hello message is as follows:

<Hello Message> ::= <Common Header> <Hello>.

The Hello object format is shown below:

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                           TxSeqNum                            |
```



TxSeqNum: 32 bits

This is the current sequence number for this Hello message. This sequence number will be incremented when either (a) the sequence number is reflected in the RcvSeqNum of a Hello packet that is received over the control channel, or (b) the Hello packet is transmitted over a backup control channel.

TxSeqNum=0 is not allowed.

TxSeqNum=1 is reserved to indicate that a node has booted or rebooted.

RcvSeqNum: 32 bits

This is the sequence number of the last Hello message received over the control channel. RcvSeqNum=0 is reserved to indicate that a Hello message has not yet been received.

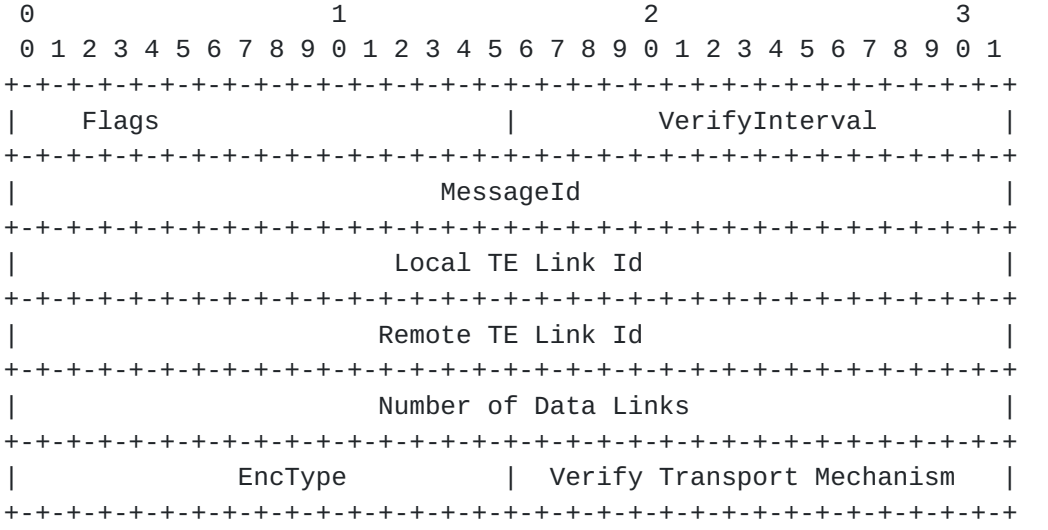
9.6 Link Verification

9.6.1 BeginVerify Message (MsgType = 5)

The BeginVerify message is sent over the control channel and is used to initiate the link verification process. The format is as follows:

<BeginVerify Message> ::= <Common Header> <BeginVerify>

The BeginVerify object has the following format:



```
   |                          BitRate                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
|                          Wavelength                           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Flags: 16 bits

The following flags are defined:

0x01 TE Link type
: If this bit is set, the TE Link Id is numbered; otherwise the TE Link Id is unnumbered.

0x02 Verify all Links
: If this bit is set, the verification process checks all unallocated links; else it only verifies new ports or component links that have been added to this TE link.

0x04 Data Link Type
: If set, the data links to be verified are ports, otherwise they are component links

VerifyInterval: 16 bits

This is the interval between successive Test messages and is measured in milliseconds (ms).

MessageId: 32 bits

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgment in the BeginVerifyAck and BeginVerifyNack messages.

Local TE Link Id: 32 bits

This identifies the TE LinkId of the local node, which may be numbered or unnumbered (see Flags), for the ports or component links that are being verified. If this value is set to 0, the port or component links to be verified are not yet locally assigned to a TE link.

Remote TE Link Id: 32 bits

This identifies the TE Link Id of the remote node, which may be numbered or unnumbered (see Flags), for the ports or component links that are being verified. If this value is set to 0, the local node has no knowledge of the remote TE Link Id. It is expected that for unnumbered TE LinkÆs this will be set to 0.

Number of Data Links: 32 bits

This is the number of data links that will be verified.

EncType: 16 bits

This is the encoding type of the data link and is required for the purpose of testing where the data links are not required to be the same encoding type as the control channel. The defined EncType values are consistent with the Link Encoding Type values of [OSPF-GEN] and [ISIS-GEN].

Verify Transport Mechanism: 16 bits

This defines the transport mechanism for the Test Messages. The scope of this bit mask is restricted to each link encoding type. The local node will set the bits corresponding to the various mechanisms it can support for transmitting LMP test messages. The receiver chooses the appropriate mechanism in the BeginVerifyAck message.

For SONET/SDH Encoding Type, the following flags are defined:
0x01 Capable of communicating using JO overhead bytes.
Test Message is transmitted using the J0 bytes.
0x02 Capable of communicating using Section DCC bytes.
Test Message is transmitted using the DCC Section Overhead bytes with an HDLC framing format.
0x04 Capable of communicating using Line DCC bytes.
Test Message is transmitted using the DCC Line Overhead bytes with an HDLC framing format.
0x08 Capable of communicating using POS.
Test Message is transmitted using Packet over SONET framing using the encoding type specified in the EncType field.

For GigE Encoding Type, the following flags are defined: TBD

For 10GigE Encoding Type, the following flags are defined: TBD

BitRate: 32 bits

This is the bit rate of the data link over which the Test messages will be transmitted and is expressed in bytes per second.

Wavelength: 32 bits

When a data link is assigned to a port or component link that is capable of transmitting multiple wavelengths (e.g., a fiber or waveband-capable port), it is essential to know which wavelength the test messages will be transmitted over. This value corresponds to the wavelength at which the Test messages will be transmitted over and is measured in nanometers (nm). If each data link corresponds to a separate wavelength and there is no ambiguity as to the wavelength over which the

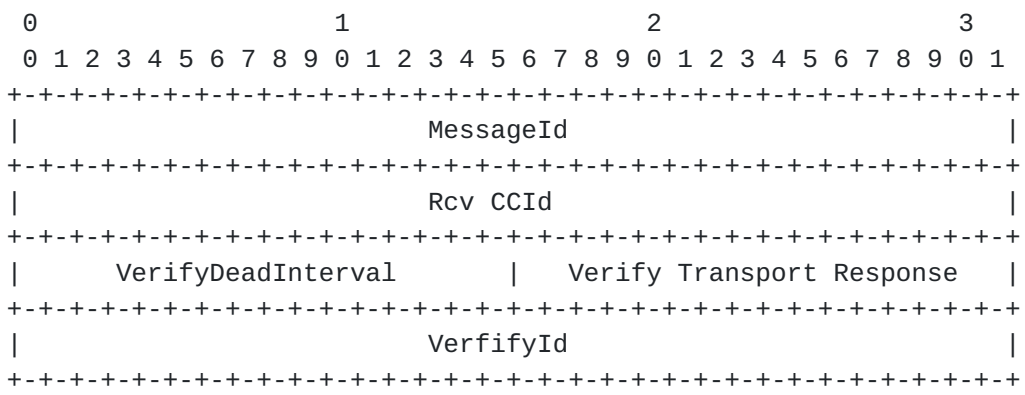
message will be sent, than this value SHOULD be set to 0.

9.6.2 BeginVerifyAck Message (MsgType = 6)

When a BeginVerify message is received and Test messages are ready to be processed, a BeginVerifyAck message MUST be transmitted.

<BeginVerifyAck Message> ::= <Common Header> <BeginVerifyAck>

The BeginVerifyAck object has the following format:



MessageId: 32 bits

This is copied from the BeginVerify message being acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the BeginVerify message being negatively acknowledged.

VerifyDeadInterval: 16 bits

If a Test message is not detected within the VerifyDeadInterval, then a node will send the TestStatusFailure message for that data link.

Verification Transport Response: 16 bits

It is illegal to set more than one bit in the verification transport response. The recipient of the BeginVerify message (and the future recipient of the TEST messages) chooses the transport mechanism from the various types that are offered by the transmitter of the Test messages.

VerifyId: 32 bits

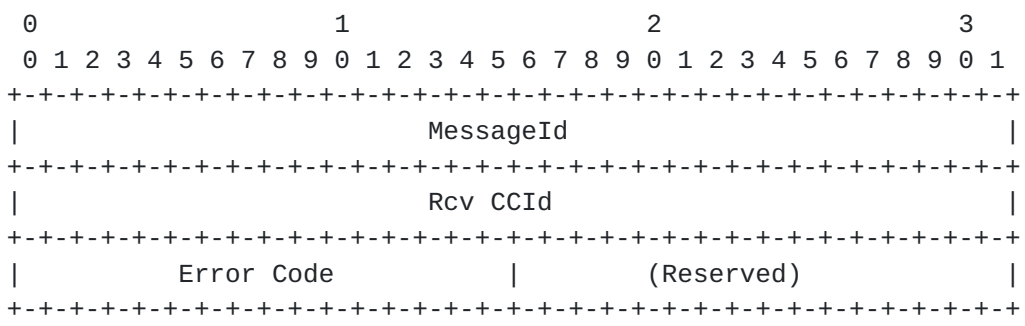
This is used to differentiate Test messages from different TE links and/or LMP peers. The recipient of the BeginVerify message assigns this value and it MUST node unique. This is a node-unique value that is assigned by the recipient of the BeginVerify message.

9.6.3 BeginVerifyNack Message (MsgType = 7)

If a BeginVerify message is received and a node is unwilling or unable to begin the Verification procedure, a BeginVerifyNack message MUST be transmitted.

<BeginVerifyNack Message> ::= <Common Header> <BeginVerifyNack>

The BeginVerifyNack object has the following format:



MessageId: 32 bits

This is copied from the BeginVerify message being negatively acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the BeginVerify message being negatively acknowledged.

Error Code: 16 bits

The following values are defined:
1 = Unwilling to verify at this time
2 = TE Link Id configuration error
3 = Unsupported verification transport mechanism

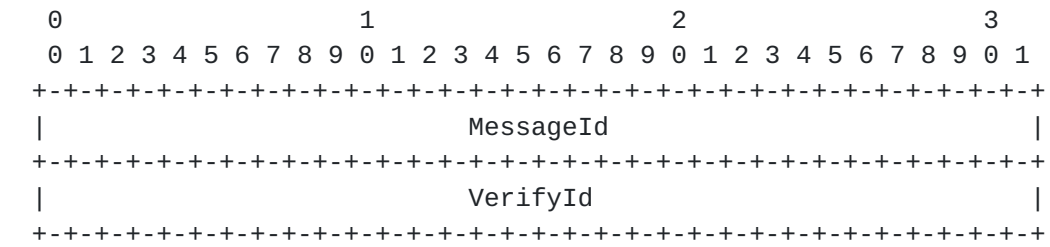
If a BeginVerifyNack message is received with Error Code 1, the node that originated the BeginVerify SHOULD schedule a BeginVerify retransmission after Rf seconds, where Rf is a locally defined parameter.

9.6.4 EndVerify Message (MsgType = 8)

The EndVerify message is sent over the control channel and is used to terminate the link verification process. The EndVerify message may be sent at any time a node desires to end the Verify procedure. The format is as follows:

<EndVerify Message> ::= <Common Header> <EndVerify>

The EndVerify object has the following format:



MessageId: 32 bits

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgement in the EndVerifyAck message.

VerifyId: 32 bits

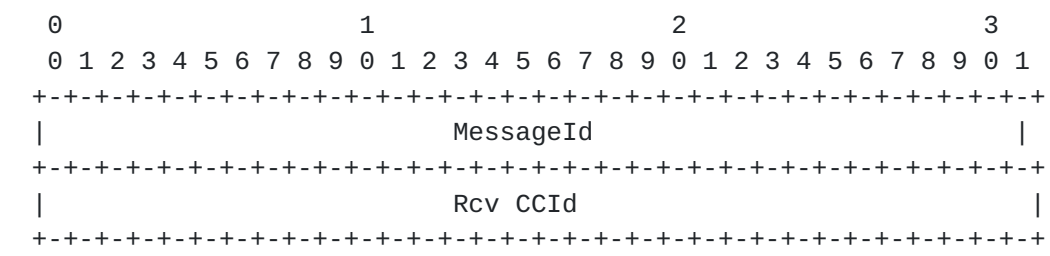
This is the VerifyId corresponding to the link verification process that is being terminated.

9.6.5 EndVerifyAck Message (MsgType =9)

The EndVerifyAck message is sent over the control channel and is used to acknowledge the termination of the link verification process. The format is as follows:

<EndVerifyAck Message> ::= <Common Header> <EndVerifyAck>

The EndVerifyAck object has the following format:



MessageId: 32 bits

This is copied from the EndVerify message being acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the EndVerify message being acknowledged.

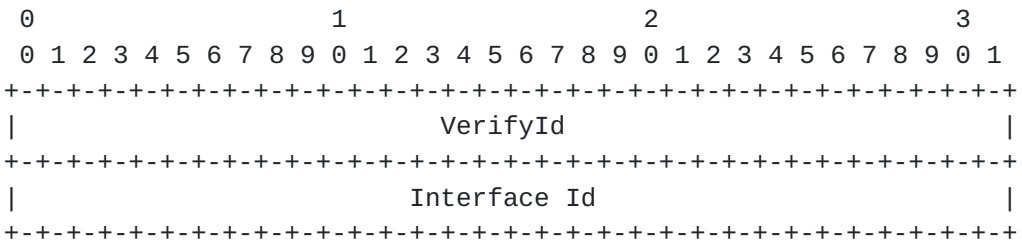
9.6.6 Test Message

The Test message is transmitted over the data link and is used to

verify its physical connectivity. Unless explicitly stated below, this is transmitted as an IP packet with payload format as follows:

<Test Message> ::= <Common Header> <Test>

The Test object has the following format:



VerifyId: 32 bits

The VerifyId identifies the link verification procedure with which the data link verification is associated.

Interface Id: 32 bits

The Interface Id identifies the data link (either port or component link) over which this message is sent. A valid Interface Id MUST be nonzero.

Note that this message is sent over a data link and NOT over the control channel.

9.6.7 TestStatusSuccess Message (MsgType = 10)

The TestStatusSuccess message is transmitted over the control channel and is used to transmit the mapping between the local Interface Id and the Interface Id that was received in the Test message.

<TestStatus Message> ::= <Common Header> <TestStatusSuccess>

The TestStatusSuccess object has the following format:

MessageId: 32 bits

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgement in the TestStatusAck message.

Received Interface Id: 32 bits

This is the value of the Interface Id that was received in the Test message. A valid Interface Id MUST be nonzero.

Local Interface Id: 32 bits

This is the local value of the Interface Id. A valid Interface Id MUST be nonzero.

VerifyId: 32 bits

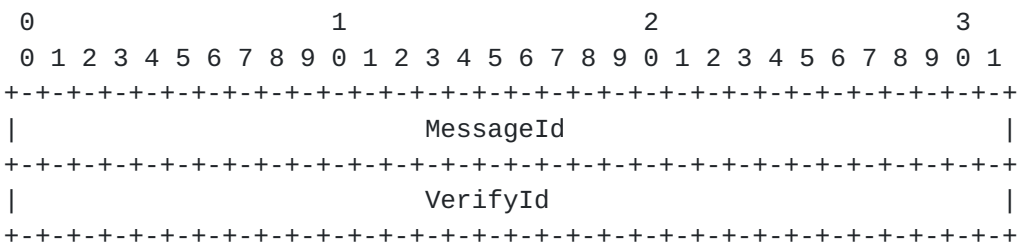
The VerifyId identifies the link verification procedure with which the data link is associated.

9.6.8 TestStatusFailure Message (MsgType = 11)

The TestStatusFailure message is transmitted over the control channel and is used to indicate that the Test message was not received.

<TestStatus Message> ::= <Common Header> <TestStatusFailure>

The TestStatusFailure object has the following format:



MessageId: 32 bits

When combined with the CCId and MsgType, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgement in the TestStatusAck message.

VerifyId: 32 bits

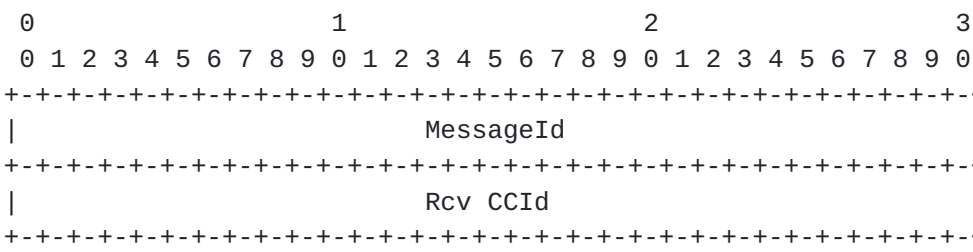
The VerifyId identifies the link verification procedure for which the timer has expired and no TEST messages have been received.

9.6.9 TestStatusAck Message (MsgType = 12)

The TestStatusAck message is used to acknowledge receipt of the TestStatusSuccess or TestStatusFailure messages.

<TestStatusAck Message> ::= <Common Header> <TestStatusAck>

The TestStatusAck object has the following format:



MessageId: 32 bits

This is copied from the TestStatusSuccess or TestStatusFailure message being acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the TestStatusSuccess or TestStatusFailure message being acknowledged.

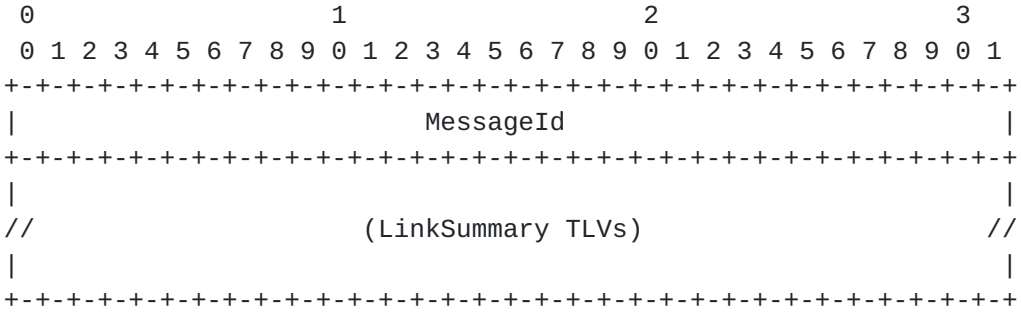
9.7 Link Summary Messages

9.7.1 LinkSummary Message (MsgType = 13)

The LinkSummary message is used to synchronize the Interface Ids and correlate the properties of the TE link. The format of the LinkSummary message is as follows:

<LinkSummary Message> ::= <Common Header> <LinkSummary>

The LinkSummary Object has the following format:

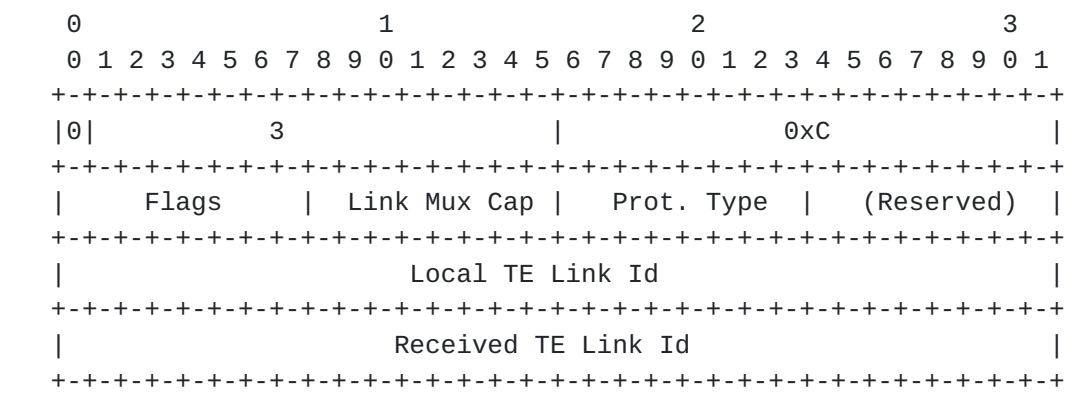


MessageId: 32 bits

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgement in the LinkSummaryAck and LinkSummaryNack messages.

9.7.1.1 TE Link TLV

The TE Link TLV is TLV Type=3 and is defined as follows:



The TE Link TLV is non-negotiable.

Flags: 8 bits

The following flags are defined:
0x01 TE Link Id Type
If this bit is set, the TE Link Id is numbered; otherwise the TE Link Id is unnumbered.

Link Mux Cap: 8 bits

This is used to identify the associated multiplexing/demultiplexing capability of the TE link. See [LSP-HIER].

Protection Type: 8 bits

The Protection Type Flags indicate the link protection, if any, that is used. Multiple bits may be set when multiple link protection types are available. The following flags are defined:

0x01 Extra Traffic

Indicates that the TE link is protecting one or more (primary) link(s). Any LSPs using a link of this type will be lost if the primary links being protected fail.

0x02 Unprotected

Indicates that the link is unprotected.

0x04 Shared (M:N)

Indicates that the link is protected using a M:N shared protection scheme.

0x08 Dedicated 1:1

Indicates that the link is protected using a 1:1 dedicated link protection scheme,

0x10 Dedicated 1+1

Indicates that the link is protected using a 1+1 dedicated link protection scheme.

Local TE Link Id: 32 bits

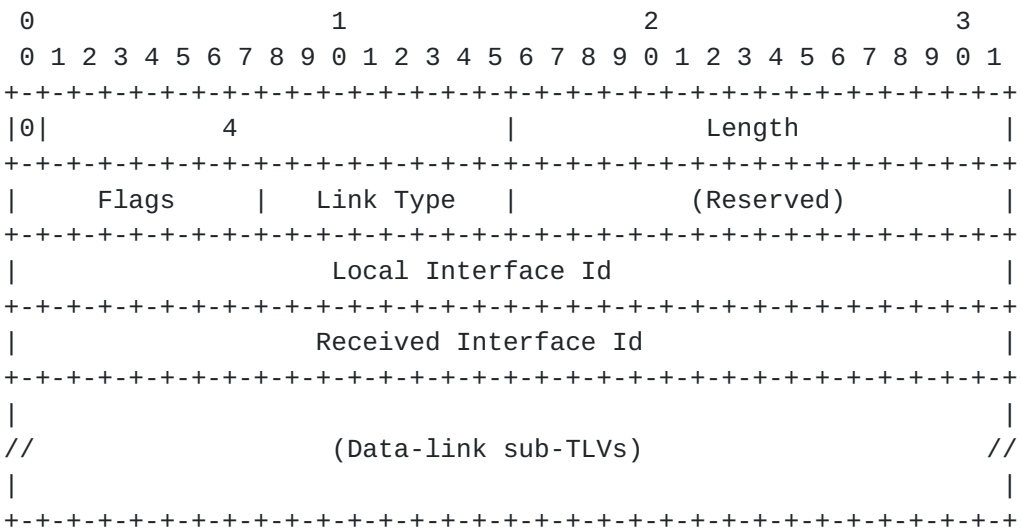
This identifies the TE link of the local node, which may be numbered or unnumbered (see Flags).

Remote TE Link Id: 32 bits

This identifies the TE link of the remote node, which may be numbered or unnumbered (see Flags). If the local node has no knowledge of the remote TE Link Id, this value MUST be set to 0.

9.7.1.2 Data-link TLV

The Data Link TLV is TLV Type=4 and is defined as follows:



The Data Link TLV is non-negotiable.

Length: 16 bits

The Length of the Primary Data Link TLV including all data-link sub-TLVs.

Flags: 8 bits

The following flags are defined. All other values are reserved.

0x01 Interface Type: If set, the data link is a port, otherwise it is a component link.
0x02 Allocated Link: If set, the data link is currently allocated for user traffic.

Link Type: 8 bits

This is used to identify the encoding type of the data link. See [OSPF-GEN] or [ISIS-TE].

Local Interface Id: 32 bits

This is the local value of the Interface Id (for the port or component link) or CCId (for control channel).

Received Interface Id: 32 bits

This is the value of the corresponding Interface Id. If this is a port or component link, then this is the value that was received in the Test message. If this is the primary control channel, then this is the value that is received in all of the Verify messages.

9.7.1.3 Data Link Sub-TLV

The data link sub-TLV is used to provide characteristics of the data-bearing links. Currently, there are no data link sub-TLVs defined.

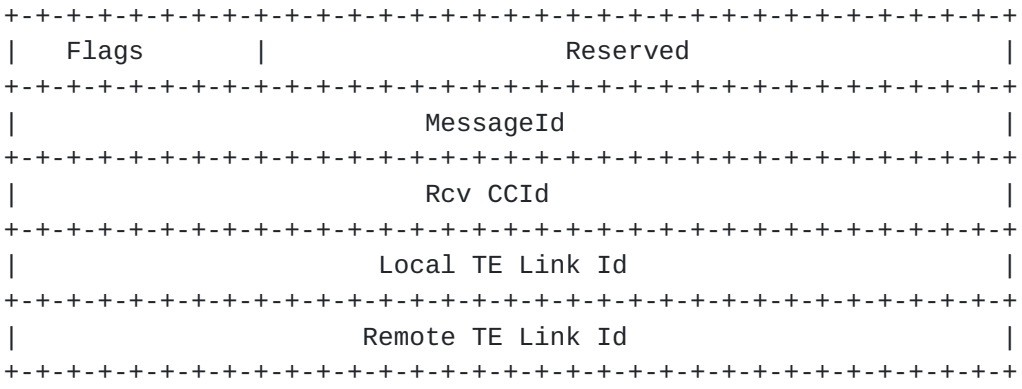
9.7.2 LinkSummaryAck Message (MsgType = 14)

The LinkSummaryAck message is used to indicate agreement on the Interface Id synchronization and acceptance/agreement on all the link parameters. It is on the reception of this message that the local node makes the TE Link Id associations.

<LinkSummaryAck Message> ::= <Common Header> <LinkSummaryAck>

The LinkSummaryAck object has the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
```



Flags: 8 bits

The following flags are defined:
0x01 TE Link Id type
If this bit is set, the TE Link Id is numbered;
otherwise the TE Link Id is unnumbered.

MessageId: 32 bits

This is copied from the LinkSummary message being acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the LinkSummary message being acknowledged.

Local TE Link Id: 32 bits

This identifies the TE Link Id of the local node, which may be numbered or unnumbered (see Flags).

Remote TE Link Id: 32 bits

This identifies the TE Link Id of the remote node, which may be numbered or unnumbered (see Flags).

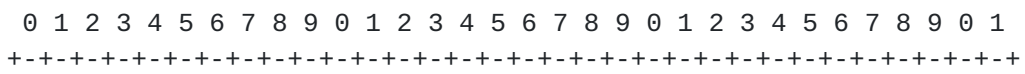
9.7.3 LinkSummaryNack Message (MsgType = 15)

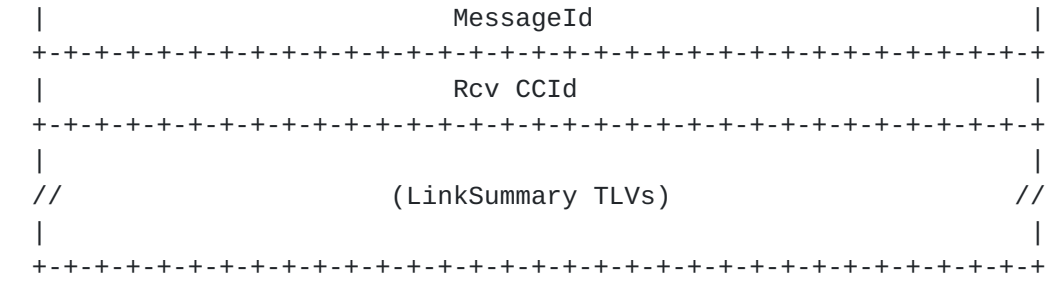
The LinkSummaryNack message is used to indicate disagreement on non-negotiated parameters or propose other values for negotiable parameters. Parameters where agreement was reached MUST NOT be included in the LinkSummaryNack Object.

<LinkSummaryNack Message> ::= <Common Header> <LinkSummaryNack>

The LinkSummaryNack object has the following format:

0 1 2 3


0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1



MessageId: 32 bits

This is copied from the LinkSummary message being negatively acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the LinkSummary message being negatively acknowledged.

The LinkSummary TLVs MUST include acceptable values for all negotiable parameters. If the LinkSummaryNack includes LinkSummary TLVs for non-negotiable parameters, they MUST be copied from the LinkSummary TLVs received in the LinkSummary message.

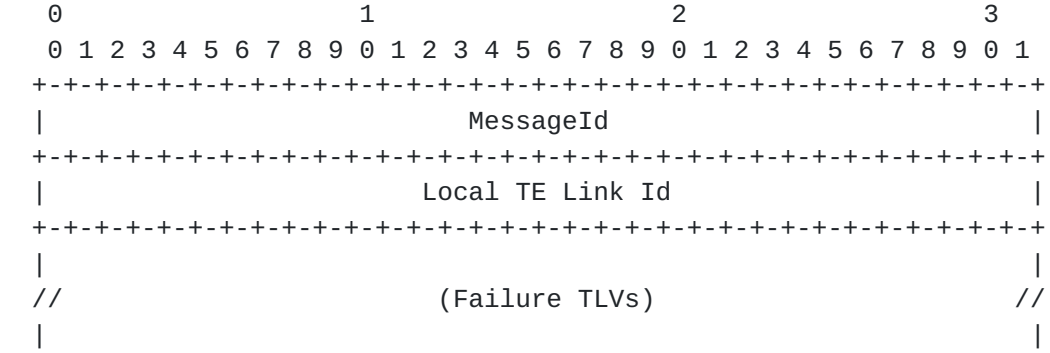
9.8 Fault Management Messages

9.8.1 ChannelFail Message (MsgType = 16)

The ChannelFail message is sent over the control channel and is used to notify a neighboring node that a data link (port or component link) failure has been detected. A neighboring node that receives a ChannelFail message MUST respond with either a ChannelFailAck or a ChannelFailNack message indicating that a failure has also been detected in the corresponding data link in the neighboring node. The format is as follows:

<ChannelFail Message> ::= <Common Header> <ChannelFail>

The format of the ChannelFail object is as follows:



```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

MessageId: 32 bits

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgement in the ChannelFailAck and ChannelFailNack messages.

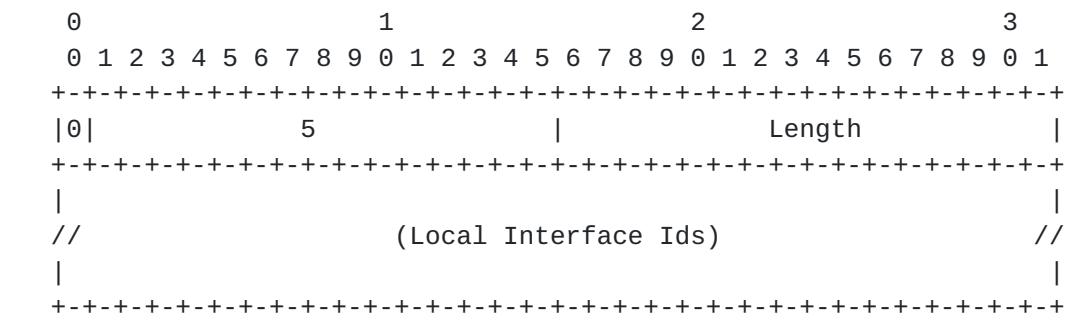
Local TE Link Id: 32 bits

This is the local TE Link Id for the failed TE link.

If no Failure TLVs are included, the ChannelFail message indicates the entire TE Link has failed.

9.8.1.2 Failed Channel TLV

The Failed Channel TLV is TLV Type=5. This TLV contains one or more Failed Channels of a TE link and has the following format:



The Failed Channel TLV is non-negotiable.

Length: 16 bits

The Length has a minimum value of 0x08 and MUST be a multiple of 4.

Local TE Link Id: 32 bits

This is the local TE Link Id within which the data link has failed.

Local Interface Id: 32 bits

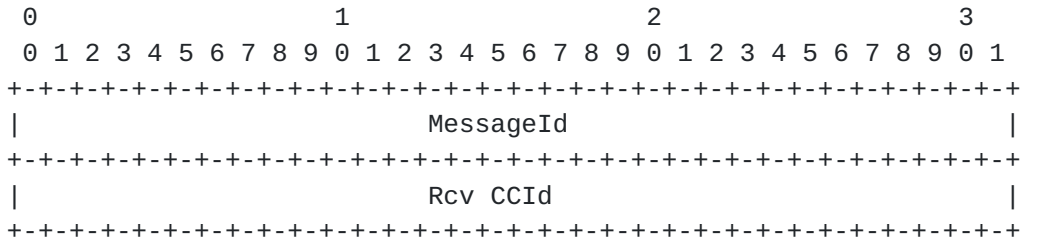
This is the local Interface Id (either Port Id or Component Interface Id) of the data link that has failed. This is within the scope of the TE Link Id. Multiple Local Interface Ids may be placed into a single Failed Channel TLV if they belong to the same TE Link.

9.8.2 ChannelFailAck Message (MsgType = 17)

The ChannelFailAck message is used to indicate that all of the reported failures in the ChannelFail message also have failures on the corresponding input channels. The format is as follows:

<ChannelFailureAck Message> ::= <Common Header> <ChannelFailureAck>

The ChannelFailureAck object has the following format:



MessageId: 32 bits

This is copied from the ChannelFail message being acknowledged.

Rcv CCId: 32 bits

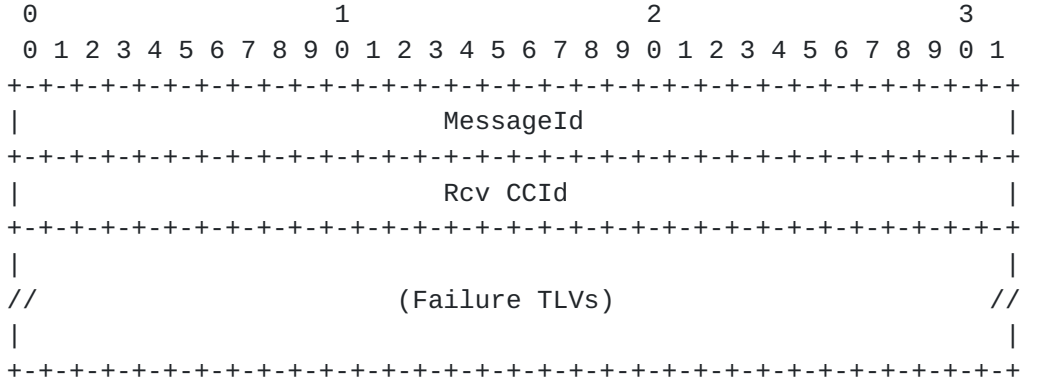
This is the Control Channel Id copied from the Common Header of the ChannelFail message being acknowledged.

9.8.3 ChannelFailNack Message (MsgType = 18)

The ChannelFailNack message is used to indicate that the reported failures are CLEAR in the upstream node, and hence, the failure has been isolated between the two nodes.

<ChannelFailNack Message> ::= <Common Header> <ChannelFailNack>

The ChannelFailNack object has the following format:



MessageId: 32 bits

This is copied from the ChannelFail message being negatively acknowledged.

Rcv CCId: 32 bits

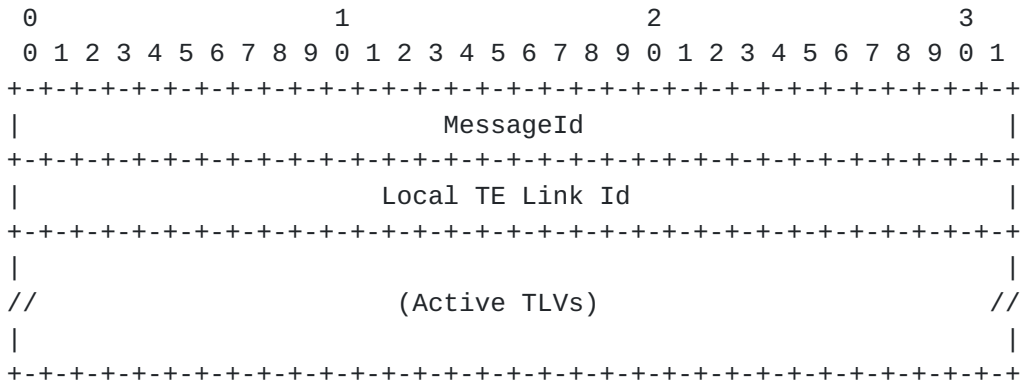
This is the Control Channel Id copied from the Common Header of the ChannelFail message being negatively acknowledged.

9.8.4 ChannelActive Message (MsgType = 19)

The ChannelActive message is sent over the control channel and is used to notify a neighboring node that a data link (port or component link) is now carrying user data traffic. A ChannelActiveAck message MUST be sent to acknowledge receipt of the ChannelActive message. The format is as follows:

<ChannelActive Message> ::= <Common Header> <ChannelActive>

The format of the ChannelActive object is as follows:



MessageId: 32 bits

When combined with the CCId, the MessageId field uniquely identifies a message. This value is incremented and only decreases when the value wraps. This is used for message acknowledgement in the ChannelActiveAck message.

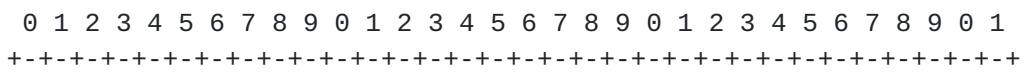
Local TE Link Id: 32 bits

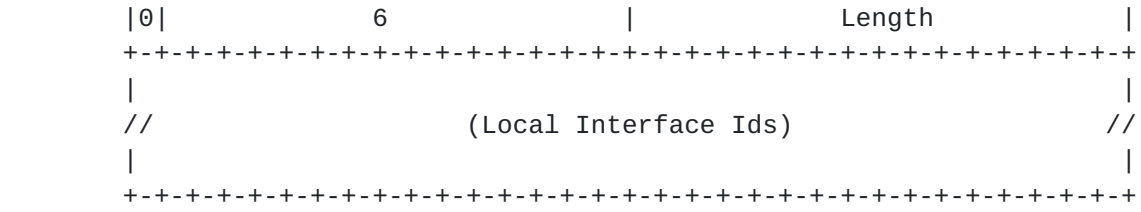
This is the local TE Link Id within which the data link has become active.

There MUST be at least one Active TLV.

9.8.4.1 Active Channel TLV

The Active Channel TLV is TLV Type=6. This TLV contains one or more Active Channels of a TE link and has the following format:


0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1



The Active Channel TLV is non-negotiable.

Length: 16 bits

The Length has a minimum value of 0x08 and MUST be a multiple of 4.

Local Interface Id: 32 bits

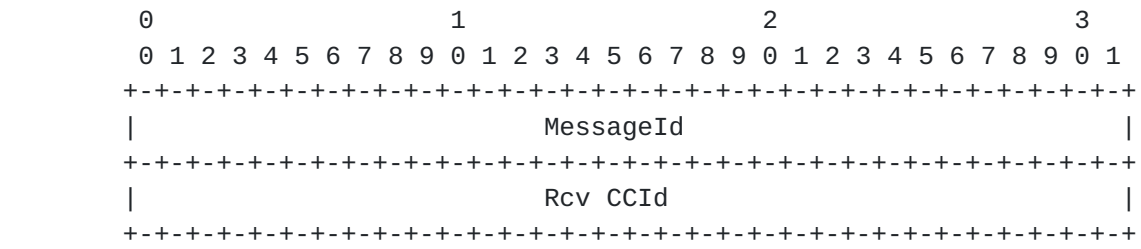
This is the local Interface Id (either Port Id or Component Interface Id) of the data link that has become active. This is within the scope of the TE Link Id. Multiple Local Interface Ids may be placed into a single Active Channel TLV if they belong to the same TE Link.

9.8.5 ChannelActiveAck Message (MsgType = 20)

The ChannelActiveAck message is used to acknowledge receipt of the ChannelActive message. The format is as follows:

<ChannelActiveAck Message> ::= <Common Header> <ChannelActiveAck>

The ChannelActiveAck object has the following format:



MessageId: 32 bits

This is copied from the ChannelActive message being acknowledged.

Rcv CCId: 32 bits

This is the Control Channel Id copied from the Common Header of the ChannelActive message being acknowledged.

10. Security Considerations

LMP exchanges may be authenticated using the Cryptographic authentication option. MD5 is currently the only message digest algorithm specified.

11. References

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

[LAMBDA] Awduche, D. O., Rekhter, Y., Drake, J., Coltun, R., "Multi-Protocol Lambda Switching: Combining MPLS Traffic Engineering Control with Optical Crossconnects," Internet Draft, draft-awduche-mpls-te-optical-02.txt, (work in progress), July 2000.

[PERF-MON] Ceuppens, L., Blumenthal, D., Drake, J., Chrostowski, J., Edwards, W. L., "Performance Monitoring in Photonic Networks," Internet Draft, draft-ceuppens-mpls-optical-00.txt, (work in progress), March 2000.

[BUNDLE] Kompella, K., Rekhter, Y., Berger, L., ôLink Bundling in MPLS Traffic Engineering,ö Internet Draft, draft-kompella-mpls-bundle-04.txt, (work in progress), November 2000.

[RSVP-TE] Awduche, D. O., Berger, L., Gan, D.-H., Li, T., Srinivasan, V., Swallow, G., "Extensions to RSVP for LSP Tunnels," Internet Draft, draft-ietf-mpls-rsvp-lsp-tunnel-07.txt, (work in progress), August 2000.

[CR-LDP] Jamoussi, B., et al, "Constraint-Based LSP Setup using LDP," Internet Draft, draft-ietf-mpls-cr-ldp-03.txt, (work in progress), September 1999.

[OSPF-TE] Katz, D., Yeung, D., "Traffic Engineering Extensions to OSPF," Internet Draft, draft-katz-yeung-ospf-traffic-03.txt, (work in progress), August 2000.

[ISIS-TE] Li, T., Smit, H., "IS-IS extensions for Traffic Engineering," Internet Draft,draft-ietf-isis-traffic-02.txt, (work in progress), September 2000.

[OSPF] Moy, J., "OSPF Version 2," RFC 2328, April 1998.

[LMP-DWDM] Fredette, A., Snyder, E., Shantigram, J., et al, ôLink Management Protocol (LMP) for WDM Transmission Systems,ö Internet Draft, draft-fredette-lmp-wdm-00.txt, (work in progress), December 2000.

[MD5] Rivest, R., "The MD5 Message-Digest Algorithm," RFC 1321, April 1992.

[OSPF-GEN] Kompella, K., Rekhter, Y., Banerjee, A., et al, "OSPF Extensions in Support of Generalized MPLS," Internet Draft, draft-kompella-ospf-extensions-00.txt, (work in progress), July 2000.

[ISIS-GEN] Kompella, K., Rekhter, Y., Banerjee, A., et al, "IS-IS Extensions in Support of Generalized MPLS," Internet

Draft, draft-kompella-isis-extensions-00.txt, (work in progress), July 2000.

[LSP-HIER] Kompella, K. and Rekhter, Y., ôLSP Hierarchy with MPLS TE,ö Internet Draft, draft-ietf-mpls-lsp-hierarchy-01.txt, (work in progress), September 2000.

12. Acknowledgments

The authors would like to thank Ayan Banerjee, George Swallow, Andre Fredette, and Adrian Farrel for their insightful comments and suggestions. We would also like to thank John Yu, Suresh Katukan, and Greg Bernstein for their helpful suggestions for the in-band control channel applicability.

13. Author's Addresses

Jonathan P. Lang
Calient Networks
25 Castilian Drive
Goleta, CA 93117
Email: jplang@calient.net

Krishna Mitra
Calient Networks
5853 Rue Ferrari
San Jose, CA 95138
email: krishna@calient.net

John Drake
Calient Networks
5853 Rue Ferrari
San Jose, CA 95138
email: jdrake@calient.net

Kireeti Kompella
Juniper Networks, Inc.
385 Ravendale Drive
Mountain View, CA 94043
email: kireeti@juniper.net

Yakov Rekhter
Juniper Networks, Inc.
385 Ravendale Drive
Mountain View, CA 94043
email: yakov@juniper.net

Lou Berger
Movaz Networks
email: lberger@movaz.com

Debanjan Saha
Tellium Optical Systems
2 Crescent Place
Oceanport, NJ 07757-0901
email:dsaha@tellium.com

Debashis Basak
Accelight Networks
70 Abele Road, Suite 1201
Bridgeville, PA 15017-3470
email: dbasak@accelight.com

Hal Sandick
Nortel Networks
email: hsandick@nortelnetworks.com

Alex Zinin
Cisco Systems
150 W. Tasman Dr.
San Jose, CA 95134
email: azinin@cisco.com

Bala Rajagopalan
Tellium Optical Systems
2 Crescent Place
Oceanport, NJ 07757-0901
email: braja@tellium.com