Network Working Group
Request for Comments: 4207
Category: Standards Track

J. Lang
Sonos, Inc.
D. Papadimitriou
Alcatel

October 2005

Synchronous Optical Network (SONET)/Synchronous Digital Hierarchy (SDH) Encoding for Link Management Protocol (LMP) Test Messages

Status of This Memo

Copyright Notice

Abstract

This document details the Synchronous Optical Network (SONET)/Synchronous Digital Hierarchy (SDH) technology-specific information needed when sending Link Management Protocol (LMP) test messages.

1. Introduction

For scalability purposes, multiple physical resources that interconnect Label Switching Routers (LSRs) can be combined to form a single traffic engineering (TE) link for the purposes of path computation and signaling. These resources may represent one or more physical links that connect the LSRs, or they may represent a Label Switched Path (LSP) if LSP hierarchy [RFC4206] is used. The management of TE links is not restricted to in-band messaging, but instead can be done using out-of-band techniques.

The Link Management Protocol (LMP) [RFC4204] has been developed as part of the Generalized MPLS (GMPLS) protocol suite to manage TE links. LMP currently consists of four main procedures, of which the first two are mandatory and the last two are optional:

1. Control channel management
2. Link property correlation
3. Link verification
4. Fault management

Control channel management is used to establish and maintain control channel connectivity between adjacent nodes. This is done using a Config message exchange followed by a lightweight keep-alive message exchange. Link property correlation is used to aggregate multiple data links into a single TE Link and to synchronize the link properties. Link verification is used to verify the physical connectivity of the data links and to exchange the Interface_Ids of the data links. Fault management is primarily used to suppress alarms and to localize failures in both opaque and transparent networks. When LMP is used with SONET/SDH, however, the fault management procedures may not be needed as existing SONET/SDH mechanisms can be used.

In this document, the SONET/SDH technology-specific information for LMP is defined. Specifically, the SONET/SDH test procedures used for link verification and link property correlation are detailed. These procedures include the trace correlation transport mechanism (defined for J0, J1, J2) that supports a separation of the transport and control plane identifiers. The latter procedure requires a new trace monitoring function that is discussed in this document. Once the data links have been verified, they can be grouped to form TE links.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

The reader is assumed to be familiar with the terminology in [RFC4204], [G.707], and [T1.105]. The following abbreviations are used in this document:

CRC-N: Cyclic Redundancy Check-N.
DCC: Data communications channel.
LOVC: Lower-order virtual container.
HOVC: Higher-order virtual container.
MS: Multiplex section.
MSOH: Multiplex section overhead.
POH: Path overhead.
RS: Regenerator section.
RSOH: Regenerator section overhead.
SDH: Synchronous digital hierarchy.
SOH: Section overhead.

SONET: Synchronous Optical Network.
STM(-N): Synchronous Transport Module (-N) (SDH).
STS(-N): Synchronous Transport Signal-Level N (SONET).
VC-n: Virtual Container-n (SDH).
VTn: Virtual Tributary-n (SONET).

3. Verifying Link Connectivity

In [RFC4204], a link verification procedure is defined whereby Test messages are transmitted in-band over the data links. This is used for data plane discovery, Interface_Id exchange (Interface_Ids are used in GMPLS signaling, either as port labels [RFC3471] or component link identifiers [RFC4201], depending on the configuration), and physical connectivity verification. Multiple data links can be verified using a single verification procedure; the correlation is done using the Verify_Id that is assigned to the procedure.

As part of the link verification procedure, a BeginVerify message exchange is used to agree upon parameters for the Test procedure. This can be initiated by sending a BeginVerify message over the control channel. This message includes a BEGIN_VERIFY object that contains a number of fields specifying, among other things, the transmission (bit) rate, encoding type, and transport mechanisms for the Test Messages. If the remote node receives a BeginVerify message and is ready to begin the procedure, it sends a BeginVerifyAck message specifying the desired transport mechanism for the Test messages. The remote node also assigns a Verify_Id to the procedure and includes it in the BeginVerifyAck message.

The transmission rate of the data link over which the Test Messages will be transmitted is represented in IEEE floating-point format using a 32-bit number field and expressed in bytes per second. See [RFC3471] for values defined for SONET/SDH.

The encoding type identifies the encoding supported by an interface. The defined encoding is consistent with the LSP Encoding Type as defined in [RFC3471]. For SONET/SDH, this value must equal the value given for "SDH ITU-T G.707/SONET ANSI T1.105".

The transport mechanism is defined using the Verify Transport Mechanism bit mask. The scope of this bit mask is restricted to the link encoding type. Multiple bits may be set when this field is included in the BeginVerify message; however, only one bit may be set when it is included in the BeginVerifyAck message.

In the following subsection, the various options for Verify Transport Mechanism are defined when the encoding is SONET/SDH. The trace correlation transport mechanism (defined for J0, J1, J2) supports a separation of the transport and control plane identifiers.

3.1. Verify Transport Mechanism

This field is 16 bits in length.

In this document, the flags for SONET/SDH encoding are defined. Note that all values are defined in network byte order (i.e., big-endian byte order).

0x0001: Reserved

0x0002 DCCS: Test Message over the Section/RS DCC

Capable of transmitting Test Messages using the DCC Section/RS Overhead bytes with bit-oriented High-Level Data Link Control (HDLC) framing format [RFC1662].

The Test Message is sent as defined in [RFC4204].

0x0004 DCCL: Test Message over the Line/MS DCC

Capable of transmitting Test Messages using the DCC Line/MS Overhead bytes with bit-oriented HDLC framing format [RFC1662].

The Test Message is sent as defined in [RFC4204].

0x0008 J0-trace: J0 Section Trace Correlation

Capable of transmitting SONET/SDH Section/RS trace over J0 Section/RS overhead byte as defined in [T1.105] and [G.707].

The Test Message is not transmitted using the J0 bytes (i.e., over the data link), but is sent over the control channel and correlated for consistency to the received J0 pattern.

In order to get the mapping between the Interface_Id over which the J0 Test Message is sent and the J0 pattern sent in-band, the transmitting node must provide the correlation between this pattern and the J0 Test Message. This correlation is done using the TRACE object as defined in Section 4.

The format of the Test Message is as follows:

<Test Message> ::=<Common Header> <LOCAL_INTERFACE_ID> <VERIFY_ID> <TRACE>

0x0010: Reserved

0x0020: Reserved

0x0040 J1-trace: J1 Path Trace Correlation

Capable of transmitting SONET/SDH STS SPE/HOVC Path trace over J1 Path overhead byte as defined in [T1.105] and [G.707].

The Test Message is not transmitted using the J1 bytes (i.e., over the data link), but is sent over the control channel and correlated for consistency to the received J1 pattern.

In order to get the mapping between the Interface_Id over which the J1 Test Message is sent and the J1 pattern sent in-band, the transmitting node must provide the correlation between this pattern and the J1 Test Message. This correlation is done using the TRACE object as defined in Section 4.

The Test Message format is identical to that defined above in J0-trace.

0x0080 J2-trace: J2 Path Trace Correlation

Capable of transmitting SONET/SDH VT SPE/LOVC Path trace over J2 Path overhead byte as defined in [T1.105] and [G.707].

The Test Message is not transmitted using the J2 bytes (i.e., over the data link), but is sent over the control channel and correlated for consistency to the received J2 pattern.

In order to get the mapping between the Interface_Id over which the J2 Test Message is sent and the J2 pattern sent in-band, the transmitting node must provide the correlation between this pattern and the J2 Test Message. This correlation is done using the TRACE object as defined in Section 4.

The Test Message format is identical to that defined above in J0-trace.

4. Trace Monitoring

The trace monitoring features described in this section allow a node to do trace monitoring by using the SONET/SDH capabilities.

- A node may request its neighbor (the remote node) to monitor a link for a specific pattern in the overhead using the TraceMonitor Message. An example of this overhead is the SONET Section Trace message transmitted in the J0 byte. If the actual trace message does not match the expected trace message, the remote node MUST report the mismatch condition.

- A node may request the value of the current trace message on a given data link using the TraceReq Message.

- A node may request a remote node to send a specific trace message over a data link using the InsertTrace Message.

4.1.1. TraceMonitor Message

The TraceMonitor message (Message Type 21) is sent over the control channel and is used to request the remote node to monitor a data link for a specific trace value. This value is inserted in the <TRACE> object. The format of the TraceMonitor message is as follows:

```
<TraceMonitor Message> ::= <Common Header> <MESSAGE_ID>
                           <LOCAL_INTERFACE_ID> <TRACE>
```

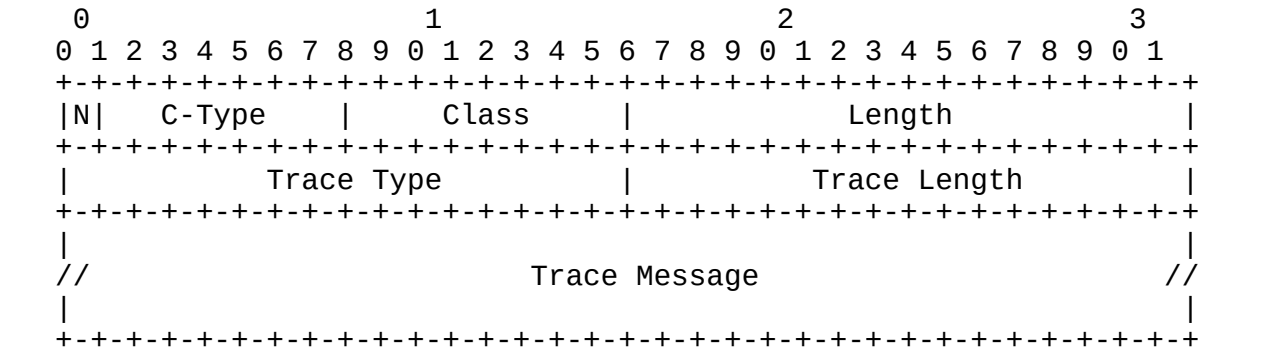
The above transmission order SHOULD be followed.

The remote node MUST respond to a TraceMonitor message with either a TraceMonitorAck or TraceMonitorNack Message.

4.1.1.1. TRACE Object Class

Class = 21

o C-Type = 1, Trace



Trace Type: 16 bits

The type of the trace message. The following values are defined. All other values are reserved.

1 = SONET Section Trace (J0 Byte)
2 = SONET Path Trace (J1 Byte)
3 = SONET Path Trace (J2 Byte)
4 = SDH Section Trace (J0 Byte)
5 = SDH Path Trace (J1 Byte)
6 = SDH Path Trace (J2 Byte)

Trace Length: 16 bits

This is the length in bytes of the trace message (as specified by the Trace Type).

Trace Message:

This is the value of the expected message to be received in-band. The valid length and value combinations are determined by the specific technology: for SONET see [T1.105] and for SDH see [G.707]. The message MUST be padded with zeros to a 32-bit boundary, if necessary. Trace Length does not include padding zeroes.

This object is nonnegotiable.

4.1.2. TraceMonitorAck Message

The TraceMonitorAck message (Message Type 22) is used to acknowledge receipt of the TraceMonitor message and indicate that all of the TRACE Objects in the TraceMonitor message have been received and processed correctly (i.e., no Trace Mismatch).

The format is as follows:

<TraceMonitorAck Message> ::= <Common Header> <MESSAGE_ID_ACK>

The above transmission order SHOULD be followed.

The MESSAGE_ID_ACK object is defined in [RFC4204]. The contents of the MESSAGE_ID_ACK object MUST be obtained from the TraceMonitor message being acknowledged.

4.1.3. TraceMonitorNack Message

The TraceMonitorNack message (Message Type 23) is used to acknowledge receipt of the TraceMonitor message and indicate that the TRACE Object in the TraceMonitor message was not processed correctly. This could be because the trace monitoring requested is not supported or there was an error in the TRACE object value(s).

The format is as follows:

```
<TraceMonitorNack Message> ::= <Common Header> <MESSAGE_ID_ACK>
                               <ERROR_CODE>
```

The above transmission order SHOULD be followed.

The MESSAGE_ID_ACK and ERROR_CODE objects are defined in [RFC4204]. The contents of the MESSAGE_ID_ACK object MUST be obtained from the TraceMonitor message being acknowledged.

If the Trace type is not supported, the ERROR_CODE MUST indicate "Unsupported Trace Type" defined in Section 4.1.3.1.

If the TRACE object was not equal to the value seen in the trace, the TraceMonitorNack message MUST include the ERROR_CODE indicating "Invalid Trace Message". The TraceMismatch message (see Section 4.1.4) SHOULD NOT be sent as a result of the mismatch.

The TraceMonitorNack message uses a new ERROR_CODE C-Type defined in Section 4.1.3.1.

4.1.3.1. ERROR_CODE Class

C-Type = 3, TRACE_ERROR

The following new error code bit-values are defined:

0x01 = Unsupported Trace Type
0x02 = Invalid Trace Message

All other values are Reserved.

Multiple bits may be set to indicate multiple errors.

This Object is nonnegotiable.

4.1.4. TraceMismatch Message

The TraceMismatch message (Message Type 24) is sent over the control channel and is used to report a trace mismatch on a data link for which trace monitoring was requested. The format is as follows:

```
<TraceMismatch message> ::= <Common Header> <MESSAGE_ID>
                            <LOCAL_INTERFACE_ID>
                            [<LOCAL_INTERFACE_ID> ...]
```

The above transmission order SHOULD be followed.

A neighboring node that receives a TraceMismatch message MUST respond with a TraceMismatchAck message.

The LOCAL_INTERFACE_ID object is defined in [RFC4204]. The LOCAL_INTERFACE_ID in this message is the local Interface Id of the data link that has a trace mismatch. A trace mismatch for multiple LOCAL_INTERFACE_IDs may be reported in the same message.

4.1.5. TraceMismatchAck Message

The TraceMismatchAck message (Message Type 25) is used to acknowledge receipt of a TraceMismatch message. The format is as follows:

```
<TraceMismatchAck Message> ::= <Common Header> <MESSAGE_ID_ACK>
```

The MESSAGE_ID_ACK object is defined in [RFC4204]. The contents of the MESSAGE_ID_ACK object MUST be obtained from the TraceMismatch message being acknowledged.

4.1.6. TraceReq Message

The TraceReq message (Message Type 26) is sent over the control channel and is used to request the current trace value of a data link.

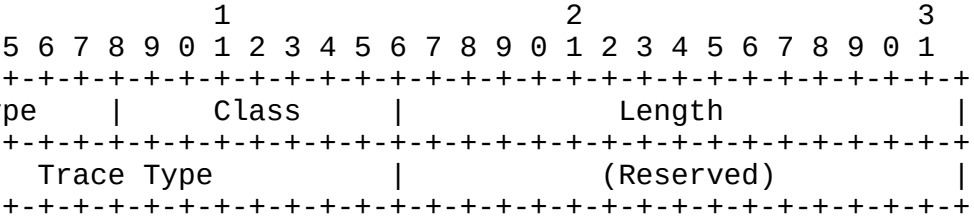
<TraceReq Message> ::= <Common Header> <MESSAGE_ID>
<LOCAL_INTERFACE_ID> <TRACE_REQ>

The above transmission order SHOULD be followed.

The format of the TRACE_REQ object is as follows:

Class = 22

O C-Type = 1, TraceReq



Trace Type: 16 bits

Defined in Section 4.1.1.1.

Reserved: 16 bits

This field MUST be set to zero when sent and ignored when received

4.1.7. TraceReport Message

The TraceReport message (Message Type 27) is sent over the control channel after receiving a TraceReq message.

<TraceReport Message> ::= <Common Header> <MESSAGE_ID_ACK> <TRACE>

The TraceReport message MUST include a TRACE Object (as described in Section 4.1.1.1) for the requested data link.

The MESSAGE_ID_ACK object is defined in [RFC4204]. The contents of the MESSAGE_ID_ACK object MUST be obtained from the TraceReq message being acknowledged.

4.1.8. TraceReqNack Message

The TraceReqNack message (Message Type 28) is sent over the control channel after receiving a TraceReq message.

```
<TraceReqNack Message> ::= <Common Header> <MESSAGE_ID_ACK>
                           <ERROR_CODE>
```

The above transmission order SHOULD be followed.

The MESSAGE_ID_ACK object is defined in [RFC4204]. The contents of the MESSAGE_ID_ACK object MUST be obtained from the TraceReq message being acknowledged.

The TraceReqNack message MUST include an ERROR_CODE Object (as defined in Section 4.1.3.1) for the requested data link.

4.1.9. InsertTrace Message

The InsertTrace message (Message Type 29) is sent over the control channel and is used to request a remote node to send a specific trace message over a data link (this assumes that the remote knows the mapping between the local and remote interface_Ids before fulfilling such request).

The format is as follows:

```
<InsertTrace Message> ::=   <Common Header> <MESSAGE_ID>
                            <LOCAL_INTERFACE_ID> <TRACE>
```

The above transmission order SHOULD be followed.

A node that receives an InsertTrace message MUST respond with either an InsertTraceAck or an InsertTraceNack Message.

Once the InsertTraceAck message is received, the TraceMismatch message (see Section 4.1.4) is used to indicate a trace mismatch has occurred.

The MESSAGE_ID_object is defined in [RFC4204].

4.1.10. InsertTraceAck Message

The InsertTraceAck message (Message Type 30) is used to acknowledge receipt of the InsertTrace message and indicate that the TRACE Object in the InsertTrace message has been received and processed correctly (i.e., no Trace Mismatch). The format is as follows:

<InsertTraceAck Message> ::= <Common Header> <MESSAGE_ID_ACK>

The MESSAGE_ID_ACK object is defined in [RFC4204]. The contents of the MESSAGE_ID_ACK object MUST be obtained from the InsertTrace message being acknowledged.

4.1.11. InsertTraceNack Message

The InsertTraceNack message (Message Type 31) is used to acknowledge receipt of the InsertTrace message and to indicate that the TRACE Object in the InsertTrace message was not processed correctly. This could be because the trace monitoring requested is not supported or there was an error in the value.

The format is as follows:

```
<InsertTraceNack Message> ::= <Common Header> <MESSAGE_ID_ACK>
                              <ERROR_CODE>
```

The above transmission order SHOULD be followed.

The MESSAGE_ID_ACK object is defined in [RFC4204].

The InsertTraceNack message MUST include an ERROR_CODE Object (as defined in Section 4.1.3.1) for the requested data link.

5. Security Considerations

LMP message security uses IPsec as described in [RFC4204]. This document introduces no other new security considerations not covered in [RFC4204].

6. IANA Considerations

LMP [RFC4204] defines the following name spaces and how IANA can make assignments in those namespaces:

- LMP Message Type.
- LMP Object Class.
- LMP Object Class type (C-Type) unique within the Object Class.
- LMP Sub-object Class type (Type) unique within the Object Class.

This memo introduces the following new assignments:

LMP Message Type:

- o TraceMonitor message (Message type = 21)
- o TraceMonitorAck message (Message type = 22)

- o TraceMonitorNack message (Message type = 23)
- o TraceMismatch message (Message type = 24)
- o TraceMismatchAck message (Message type = 25)

- o TraceReq message (Message type = 26)
- o TraceReport message (Message type = 27)
- o TraceReqNack message (Message type = 28)

- o InsertTrace message (Message type = 29)
- o InsertTraceAck message (Message type = 30)
- o InsertTraceNack message (Message type = 31)

LMP Object Class name space and Class type (C-Type):

- o TRACE Class name (21)
  - Type 1 (C-Type = 1)

- o TRACE REQ Class name (22)
  - Type 1 (C-Type = 1)

7. References

7.1. Normative References

[RFC4201] Kompella, K., Rekhter, Y., and L. Berger, "Link Bundling in MPLS Traffic Engineering (TE)", RFC 4201, October 2005.

[G.707] ITU-T Recommendation G.707, "Network node interface for the synchronous digital hierarchy (SDH)," October 2000.

[RFC4204] Lang, J., Ed., "Link Management Protocol (LMP)", RFC 4204, October 2005.

[RFC1662] Simpson, W., "PPP in HDLC-like Framing", STD 51, RFC 1662, July 1994.

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

[RFC3471] Berger, L., "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Functional Description", RFC 3471, January 2003.

[T1.105] T1.105, "Revised Draft T105 SONET Base Standard," January 2001.

7.2. Informative References

[RFC4206] Kompella, K., and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)", RFC 4206, October 2005.

8. Acknowledgements

The authors would like to thank Bernard Sales, Emmanuel Desmet, Gert Grammel, Jim Jones, Stefan Ansorge, John Drake, and James Scott for their many contributions to this document.

We would also like to thank Greg Bernstein and Michiel van Everdingen for their insightful comments and for acting with a strong combination of toughness, professionalism, and courtesy.

Authors' Addresses

Jonathan P. Lang
Sonos, Inc.
223 E. De La Guerra St.
Santa Barbara, CA 93101

EMail: jplang@ieee.org

Dimitri Papadimitriou
Alcatel
Francis Wellesplein 1
B-2018 Antwerpen, Belgium

EMail: dimitri.papadimitriou@alcatel.be

Full Copyright Statement

Intellectual Property

Acknowledgement

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