Link Layer Discovery

Protocol and MIB

v1.0

Paul Congdon
5/20/02

<<**Version Change Notes**:
v1.0
1. Updated overview text to include discussion about architectural positioning and protocol objectives
2. Added note to on message number in PDU for discussion
3. Added new TLVs per email discussion
4. Left in vendor specific TLV, but put restrictions on usage
5. Added simple tx and rx state machines
6. No changes to MIB yet

>>

Acknowledgements

This document is heavily leveraged from an Internet-Draft developed for the IETF PTOPO working group. The original draft, titled draft-ietf-ptopomib-pdp-03.txt, and authored by Andy Bierman and Keith McCloghrie has expired and has not been renewed nor forwarded on for RFC status by the IETF working group. The original PTOPO Discovery Protocol is a product of the IETF PTOPOMIB Working Group.

The intention of this document is bring forward relevant text and concepts from the original draft as input into a proposed work item to develop a standard discovery protocol within the IEEE 802.1 working group.

Abstract

This document defines a protocol, and a set of management objects for use with IEEE 802 devices. In particular, it describes a physical topology discovery protocol and managed objects used for managing the protocol. The protocol is not restricted from running on non-802 media, however, a specification of this operation is beyond the scope of this document.

Overview

There is a need for a standardized way of representing the physical network connections pertaining to a given management domain. A standardized discovery mechanism is also required to increase the likelihood of multi-vendor interoperability of such physical topology management information. It is also desirable to discover certain configuration inconsistencies or assumptions that may result in impaired communication or network malfunction at higher layers.

This document specifies a discovery protocol, suitable for use with the Physical Topology MIB [RFC2922].

Terms

Some terms are used throughout this document:

SNMP Agent

This term refers to an SNMP agent co-located with a particular LLDP Agent. Specifically, it refers to the SNMP Agent providing LLDP MIB, Entity MIB, Interfaces MIB, and possibly PTOPO MIB support for a particular chassis.

LLDP Agent

This term refers to a software entity which implements the Link Layer Discovery Protocol for a particular chassis.

NMS

This term refers to a Network Management System capable of utilizing the information gathered by LLDP and the PTOPO MIB.

Link Layer Discovery Protocol

This section defines a discovery protocol, suitable for supporting the data requirements of the PTOPO MIB [RFC2922] and capable of advertising device information to peer devices on the same physical LAN.

The Link Layer Discovery Protocol (LLDP) is a media independent protocol intended to be run on all IEEE 802 devices, allowing a LLDP agent to learn higher layer management reachability and connection endpoint information from adjacent devices.

LLDP runs on all 802 media. Additionally the protocol runs over the data-link layer only, allowing two systems running different network layer protocols to learn about each other.

Architecturally, LLDP runs on top of the uncontrolled port of an 802 MAC client. LLDP may be run over an aggregated MAC client as specified by Std. 802.3, 2000 Edition Clause 43, but must run over the physical MAC client. It may be desirable for stations to prohibit the transmission of LLDP PDUs over the uncontrolled port until the controlled port has been authorized, but this is not a requirement. The spanning tree state of a port does not effect the transmission of LLDP PDUs.

The LLDP protocol is essentially a one-way protocol. Each device configured with an active LLDP Agent sends periodic messages to the Slow Protocols multicast MAC address as specified by Std 802.3, 2000 Edition Annex 43B. The device sends the periodic messages on all physical interfaces enabled for LLDP transmission, and listens for LLDP messages on the same set on interfaces. Each LLDP message contains information identifying the source port as a connection endpoint identifier. It also contains at least one network address which can be used by an NMS to reach a management agent on the device (via the indicated source port). Each LLDP message contains a configurable time-to-live value, which tells the recipient LLDP agent when to discard each element of learned topology information. Additional optional information may be contained in LLDP PDUs to assist in the detection of configuration inconsistencies.

The LLDP protocol is designed to advertise information useful for discovering pertinent information about a remote peer and to populate topology management information databases such as RFC2922. It is not intended to act as a configuration protocol for remote devices, nor as a mechanism to signal control information between peers. During the operation of LLDP it may be possible to discover configuration inconsistencies between devices on the same physical LAN. This protocol does not provide a mechanism to resolve those inconsistencies, rather a means to report discovered information to higher layer management entities. Acting upon discovered information typically requires careful consideration and is clearly out of the scope of this document.

Frame Encapsulation

An LLDP PDU is encapsulated within an 802 frame that corresponds to frame formatted to meet the requirements of an 802 Slow Protocol as defined by Std 802.3, 2000 Edition, Annex 43B. The format is shown in the following figure:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Slow Protocols Multicast DA                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Multicast DA (cont)         |         Station SA            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Station SA (cont)                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Slow Protocols Type       |   Subtype     |    reserved   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       LLDP PDU Message                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[figure 1 – Slow Protocols LLDP Message Format]

The Slow Protocol encapsulation has the following fields:

Slow Protocols Multicast DA
The Slow Protocols Multicast destination address is 01-80-C2-00-00-02. This address is within the range reserved by ISO/IEC 15802-3 (MAC Bridges) for link-constrained protocols and will not be forwarded by conformant MAC bridges.

Station SA
The source MAC address of the sending station

Slow Protocols Type
The Slow Protocols Type field encoding of the Length/Type field is 88-09

Subtype
The Slow Protocols Subtype field is TBD

All reserved fields shall be set to zero.

LLDP Message Format

The basic LLDP protocol data unit consists of a header, followed by a variable number of Type-Length-Value (TLV) attributes. A single LLDP PDU is transmitted in a single 802 media frame.

LLDP Header Format

The LLDP header is a 4 byte header, in network byte order, containing 3 fields, as shown in figure 2:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      Version  |     Flags     |         Time To Live          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[figure 2 -- LLDP Message Format]

The LLDP header contains the following fields:

Version

The LLDP protocol version number, set to 0x01 for this version of the protocol.

Flags

The LLDP flags field provide for future header extensions and keep the header word-aligned for easier processing. No flag definition bits are defined at this time. This field must be set to zero in all version 1 LLDP messages.

Time to Live

The number of seconds the information in this LLDP message should be regarded as valid by the recipient. Agents of the PTOPO MIB must not return MIB information based on expired LLDP messages. The valid range is 0 to 65535 for this field.

<<Message Number>>

<<I have a note to add a message number to the frame to assist in detecting anomalies. I'm not exactly sure how this would work and what the anomaly detection scheme would be. My assumption is that we would simply increment the message number on each transmission, and ignore message that appeared to be old – taking care for the sequence number wrapping case. I guess the message number would be reset every time the protocol was restarted. – comments?>>

TLV Format

Following the LLDP header are a variable number of TLVs, depending on implementation and maximum message size. See figure 3 for TLV field layout.

A 2 byte type field identifies the specific TLV, and a 2 byte length, in octets, indicates the length of the value field contained in the TLV. A TLV shall always start on a 4 octet boundary. Pad octets are placed at the end of the previous TLV in order to align the next TLV. These pad octets are not counted in the length field of the TLV.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
| Value byte 0  |  ... repeated through value byte[Length-1]
+-+-+-+-+-+-+-+-+
```

[Figure 3 - TLV Format]

The header fields are defined as follows:

Type
The integer value identifying the type of information contained in the value field.

Length
The length, in octets, of the value field to follow.

Value
A variable-length octet-string containing the instance-specific information for this TLV.

Standard TLV Definitions

The mandatory LLDP TLVs allow for a LLDP agent to support the PTOPO MIB for connections terminating on the local chassis. Optional TLVs allow for vendor specific extensions.

The following table summarizes the TLVs defined in this document.

Type	TLV Name	Example Usage
1	Chassis ID	{ chasIdIfAlias(2), "acme.rg1-sw.0000c07cf297" }
2	Port ID	{ portIdIfAlias(1), "eth0/0/0" }
3	Mgmt Address	{ ipV4(1), 4, '0x01020304' }
4	PVID	{ ‘2030’)
5	Other PVIDs	{ ‘0’)

```
|  6   | Link Duplex     | { ‘1’ )                              |
+------+-----------------+--------------------------------------+
|  7   | Capabilities    | { ‘0x00001100’)                      |
+------+-----------------+--------------------------------------+
|  8   | Version         | { “F.04.09”)                         |
+------+-----------------+--------------------------------------+
|  9   | Vendor Specific | { VendorID, 'vendor specific’)       |
+------+-----------------+--------------------------------------+
```

[Figure 4 - TLV Summary]

Chassis ID

The Chassis ID is a mandatory TLV which identifies the chassis component of the endpoint identifier associated with the transmitting LLDP agent.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x1          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        Chassis ID Type        |       Chassis ID String . . .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 5 – Chassis ID TLV Format]

The Chassis ID fields are defined as follows:

Chassis ID Type

This field identifies the format and source of the chassis identifier string. It is an enumerator defined by the PtopoChassisIdType object from RFC2922

Chassis ID String

The binary string containing the actual chassis identifier for this device. The source of this field is defined by PtopoChassisId from RFC2922.

Port ID

The Port ID is a mandatory TLV which identifies the port component of the endpoint identifier associated with the transmitting LLDP agent. If the specified port is an IEEE 802.3 Repeater port, then this TLV is optional.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x2          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Port ID Type         |          Port ID String . . .
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 6 – Port ID TLV Format]

The Port ID fields are defined as follows:

Port ID Type

This field identifies the format and source of the port identifier string. It is an enumerator defined by the PtopoPortIdType object from RFC2922

Port ID String

The binary string containing the actual port identifier for the port which this LLDP PDU was transmitted. The source and format of this field is defined by PtopoPortId from RFC2922.

Management Address

The Management Address is a mandatory TLV which identifies a network address associated with the local LLDP agent, which can be used to reach the agent on the port identified in the Port ID TLV. The value field of this TLV has the following record 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x3          |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|        IANA AddressFamily     |       Address Length          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

 0
 0 1 2 3 4 5 6 7 8 9    .....
+-+-+-+-+-+-+-+-+-+-+-+  ... -+-+-+-+-+-+-+-+-+-+
| Addr byte 1   |        ...    | Addr byte N   |
+-+-+-+-+-+-+-+-+-+-+-+ .....-+-+-+-+-+-+-+-+-+-+
```

[Figure 7 -- Management Address TLV Format]

The Management Address fields are defined as follows:

IANA Address Family

The enumerated value for the network address type identified in this TLV. This enumeration is defined in the “Assigned Numbers” RFC [RFC3232] and the ianaAddressFamilyNumbers object.

Address Length

The number of octets contained in the address string to follow.

Address

The binary string containing the network management address for this TLV.

PVID

The PVID TLV (Port VLAN Identifier) is an optional TLV which identifies the VLAN identifier associated with untagged or priority tagged frames received on the port as specified in IEEE 802.1Q-1998. In some cases the sending device may not know or support the PVID as defined in IEEE 802.1Q-1998.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x4          |        Length = 0x4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
|       |         PVID            |           reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 8 – PVID TLV Format]

The PVID TLV fields are defined as follows:

PVID

The Port VLAN Identifier for the port. It defined by the dot1qPvid object from RFC2674. A value of 0 shall be used if the device either does not know the PVID or does not support port based VLANs per the operation of IEEE 802.1Q-1998.

Other PVIDs

The Other PVIDs TLV is an optional TLV which identifies if the port has additional PVIDs defined for the port. Additional PVIDs may only be used when classification methods other the Port Based VLAN classification are used on the port (e.g. Port and Protocol VLAN classification as defined by 802.1v).

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x5          |        Length = 0x4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          OtherPVIDs           |           reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 9 – Other PVIDs TLV Format]

The Other PVIDs TLV fields are defined as follows:

OtherPVIDs

A Boolean value indicating that additional PVIDs have been configured for the port. A value of 0 indicates FALSE

Link Duplex

The Link Duplex TLV is an optional TLV which identifies the duplex setting of the MAC connected to the physical medium. In some 802 networks, it is possible for MAC entities to be connected to the same physical link, but with different duplex settings, resulting in impaired communication.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x6          |        Length = 0x4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Duplex            |           reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 10 – Link Duplex TLV Format]

The Link Duplex TLV fields are defined as follows:

Duplex

The current duplex status of the MAC. For 802.3 MACs this field is defined by the dot3StatsDuplexStatus object from RFC2665. Other

MACs shall confirm to the 802.3 list of choices which include: 1=unknown, 2=halfDuplex, 3=fullDuplex.

Capabilities

The Capabilities TLV is an optional TLV which identifies the capabilities of the device and its primary function. It is intended to improve the discovery of managed services on the device

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Type = 0x7           |        Length = 0x4           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Capabilities          |          reserved             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 11 – Capabilities TLV Format]

The Capabilities TLV fields are defined as follows:

Capabilities

A bit map of capabilities defining the primary function of the device. The capabilities are defined by the sysServices object in RFC 1213.

<< NOTE: This is not really sufficient as it has a single bit for each layer of the OSI model and the bits tend to have less meaning the further up the stack. It doesn’t really provide many hints on where to start managing the device. Something more useful would narrow the scope to something at and below L3 – Consider bits for: port is in an aggregation, Spanning Tree is supported, VLANs are supported, L3 routing is supported, etc… I suggest the following alternative for discussion:

- PortAccessControlEnabled
- PortInAggregation
- PVIDEnabled
- PortAndProtocolPVIDsEnabled
- TaggedVLANsEnabled
- L2Forwarding
- SourceRouteBridging
- SpanningTreeEnabled
- IGMPSnoopingEnabled
- L3Forwarding
- L3MulticastForwarding
- HigherLayerForwarding
- NonForwardingStation

>>

Version

The Version TLV is an optional TLV which uses a display string to identify product version information about the device.

```
 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
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Type = 0x8          |        Length                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                         Version String . . .                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

[Figure 12 – Version TLV Format]

The Version TLV fields are defined as follows:

Version
A string that identifies product version information for the device. The string shall be less than 256 octets.

Vendor-Specific

This TLV is available to allow vendors to support their own extended attributes not suitable for general usage. The information conveyed in the TLV MUST not affect the operation of the LLDP protocol and MUST comply with the following restrictions:

- Information transmitted in the TLV is intended to be a one-way advertisement. It must not solicit a response and must not provide an acknowledgement.
- Information transmitted in the TLV must be independent from information received in a TLV from a peer.

LLDP agents not equipped to interpret the vendor-specific information sent by other LLDP agents MUST ignore it (although it may be reported). LLPD agents which do not receive desired vendor-specific information SHOULD make an attempt to operate without it, although they may do so (and report they are doing so) in a degraded mode.

A summary of the Vendor-Specific TLV format is shown below. The fields are transmitted from left to right.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Type = 0x9            |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Vendor-Id                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           String...
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

Vendor-Id
The high-order octet is 0 and the low-order 3 octets are the SMI Network Management Private Enterprise Code of the Vendor in network byte order, as defined in the "Assigned Numbers" RFC [RFC3232].

String
The String field is one or more octets. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets. Multiple subattributes MAY be encoded within a single Vendor-Specific TLV, although they do not have to be.

Protocol Operation

An active LLDP Agent must perform the following tasks:

- transmit LLDP messages
- process received LLDP messages
- maintain an instance of the LLDP MIB
- maintain an instance of the PTOPO MIB
- maintain appropriate ifEntry and/or entPhysicalEntry instances
- implement ifAlias and/or entPhysicalAlias MIB objects

Protocol Initialization

Upon system reinitialization, the following tasks are performed by the LLDP agent:

Non-volatile configuration for the LLDP MIB is retrieved if applicable, otherwise appropriate default values are assigned to all LLDP configuration variables.

If LLDPAdminStatus is equal to 'disabled(2)', then LLDP initialization is terminated (until such time that the LLDPAdminStatus object is set to 'enabled(1)'), otherwise continue.

Internal (implementation-specific) data structures are initialized such that appropriate local physical topology information and LLDP transmission parameters are set.

Message Encoding

This section does not assume a particular buffering strategy, and such details are omitted.

Header Fields

The version field is set to one (0x01).

The flags field is set to zero (0x00).

The time-to-live field is set to the value obtained by the following formula:

TTL = min(65535, (LLDPMessageTxInterval * LLDPMessageTxHoldMultiplier))

TLVs

Each message must contain one instance of each of the mandatory LLDP TLV elements. Additional TLV data elements may be added as maximum frame size permits.

The mandatory TLVs include: Chassis ID, Port ID (optional for repeaters) and Management Address.

TLVs are always to be aligned on a 4 octet boundary.

Message Transmission

LLDP agents must follow the rules for Slow Protocols transmission as defined by Std 802.3, 2000 Edition, Annex 43B. In addition to these rules, an active LLDP agent must transmit a LLDP message out each appropriate port, once each message interval, as determined by the LLDPMessageTxInterval MIB object, subject to the restriction of transmission rules for Slow Protocols. Messages transmitted on repeater devices may be sent for each repeater backplane, once per message interval. Actual transmission intervals should be jittered to prevent synchronization effects.

Note that the agent must suppress the transmission of multiple LLDP messages during a single message interval, in the event message transmission cannot be restricted to a single port, but rather a group of ports (e.g., a repeater device).

In this case, a single port in the port group should be selected (in an implementation-specific manner) to represent the port group. Note that an agent is encouraged to represent port groups as 'backplanes', in the entPhysicalTable of the Entity MIB, rather than individual ports in either the Entity MIB or Interfaces MIB.

Regarding the transmission of a single LLDP message, for the indicated physical interface contained in the local system:

The LLDP agent checks for the existence of a LLDPSuppressEntry for the port. If an entry exists then this port is skipped, otherwise continue.

The LLDP message is encapsulated as appropriate for the port.

The MAC header is filled in with appropriate SA and DA and EtherType fields as defined above.

The frame is transmitted or passed to a lower layer for transmission.

The LLDPStatsOutPkts counter is incremented for the indicated local port.

Message Forwarding

As indicated by the operation of Slow Protocols, LLDP agents should not forward LLDP messages received on any port. However, some devices, such as repeaters, cannot examine each frame received on an interface or port. Such a device will allow LLDP messages to be retransmitted on one or more local ports, and will transmit its own LLDP messages on those ports as well. These agents are termed 'forwarding' LLDP agents.

LLDP agents located on devices which examine each frame before retransmitting it (e.g., routers and bridges), are expected to process received LLDP messages and not retransmit them on any local port. These agents are termed 'non-forwarding' LLDP agents.

An NMS may find physical topology information about the same physical port, represented by several LLDP agents. This may occur for one of several reasons, including a mixture of forwarding and non-forwarding LLDP agents within a network.

Received Message Processing

An active LLDP agent must process LLDP messages received on each appropriate port, as such messages arrive. Before LLDP specific receive rules are executed, the frame is subject to the receive processing rules of Slow Protocols defined in Std 802.3, 2000 Edition, Annex 43B.

The following sections refer to the reception of a single LLDP message, for the indicated physical interface contained in the local system:

Header Fields

The LLDP message and the chassis/port indices associated with the receiving port are retrieved.

The LLDP version and flags field are checked. The version should equal one (0x01) and the flags should equal zero (0x00). If not, the LLDPStatsInErrors counter for the receiving port is incremented and processing is terminated; otherwise continue.

TLVs

The TLV portion of the message is decoded. If this portion of the LLDP message is not properly encoded, as defined above, then the LLDPStatsInErrors counter for the receiving port is incremented, and processing is terminated; otherwise continue.

The list of TLV elements is examined. The agent must skip and ignore PDU data elements unknown to the agent. If any of the mandatory data elements are missing, then the LLDPStatsInErrors counter for the receiving port is incremented, and processing is terminated; otherwise continue.

The LLDPStatsInGoodPkts counter is incremented for the receiving port.

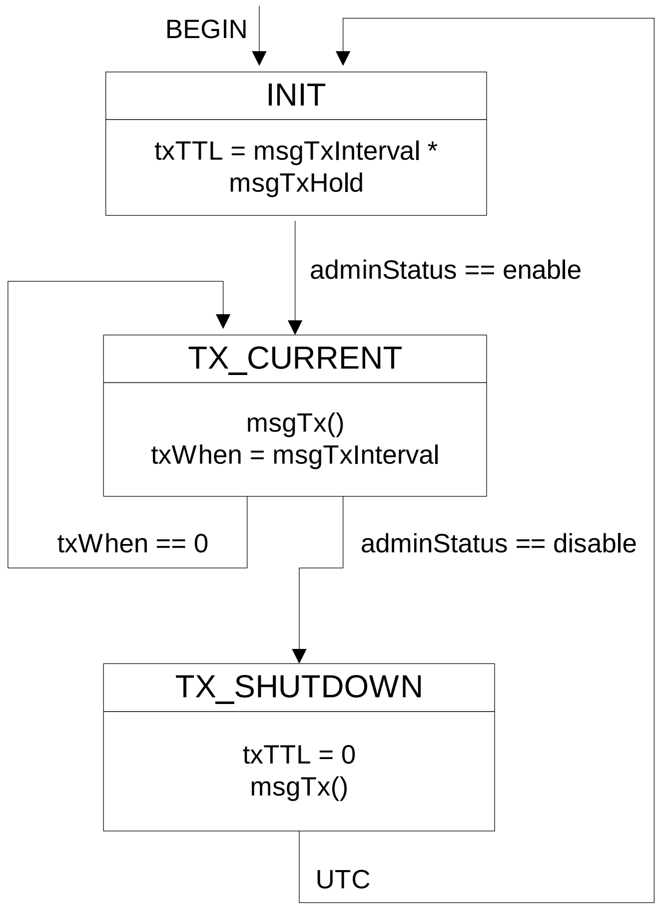
State Machines

The operation of the LLDP protocol can be represented with three simple state machine; a timer state machine, a transmit state machine and a receive state machine.

The timer state machine is trivial and simply decrements a txWhen variable once a second until zero. An example of such a machine is the Port Timers machine in IEEE 802.1X.

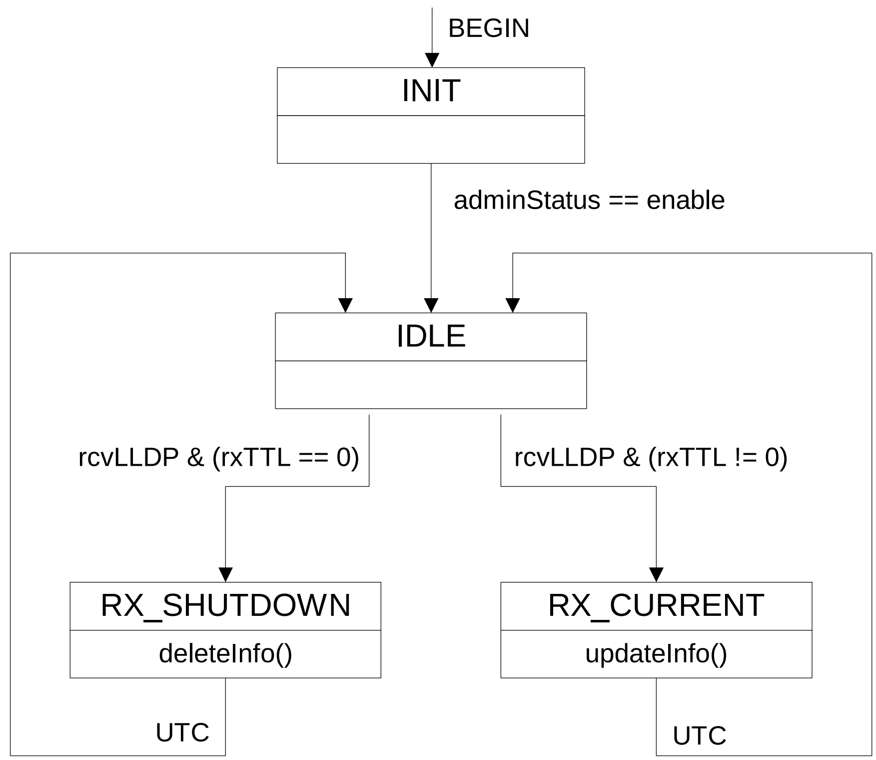
The transmit state machine is responsible for sending the periodic LLDP messages as well as the shutdown message. The following figure represents the transmit state machine.

Transmit State Machine



The receive state machine is assumed to interface with a module that processes the received information. The following figure represents the receive state machine.

Receive State Machine



The functions updateInfo() and deleteInfo() are responsible for processing received information. These functions are responsible for updating the PTOPO MIB and other management objects.

PTOPO MIB Update

If the time-to-live field in the LLDP message header is zero then execute this interface shutdown procedure, described below. Processing of the LLDP message is now complete.

If the time-to-live field is non-zero, then the appropriate ptopoConnEntry is found or created, based on the data elements included in the LLDP message. If the indicated entry is dynamic (i.e., ptopoConnIsStatic is true), then the current sysUpTime value is stored in the ptopoConnLastVerifyTime field for the entry.

If a ptopoConnEntry was added then the ptopoConnTabInserts counter is incremented.

If any ptopoConnEntry was added or deleted, or if information other than the ptopoConnLastVerifyTime changed for any entry due to the processing of this LLDP message, then the ptopoLastChangeTime object is set with the current sysUpTime, and a ptopoConfigChange trap event is generated. (See the PTOPO MIB for information on ptopoConfigChange trap generation.)

Interface Shutdown Procedure

A special procedure exists for the case in which a LLDP agent knows a particular port is about to become non-operational.

Note that the LLDPSuppressTable has precedence over these procedures, and they are only executed if the indicated interface is not specified in the LLDPSuppressTable.

If any entries are deleted as a result of these procedures, the ptopoConnTabDeletes counter is incremented for each deleted entry.

LLDP Shutdown Transmission

In the event an interface, currently configured with LLDP message transmission enabled, either becomes disabled for LLDP activity, or the interface is administratively disabled, a final LLDP message is transmitted with a time to live value of zero (before the interface is disabled).

In the event the LLDPOperStatus is transitioning to the disabled state, then this shutdown procedure should be executed for all local interfaces.

LLDP Shutdown Reception

After reception of a valid LLDP message with a time-to-live value equal to zero, the LLDP Agent must remove all information in the PTOPO MIB learned from the particular LLDP agent, which is associated with the indicated remote connection endpoint.

Link Level Discovery Protocol MIB

This section defines the MIB used to configure LLDP agent behavior.

```
LLDP-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Integer32, Counter32
        FROM SNMPv2-SMI
    RowStatus
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF
    PhysicalIndex
        FROM ENTITY-MIB;

LLDPMIB MODULE-IDENTITY
    LAST-UPDATED "9707300000Z"
    ORGANIZATION "IETF PTOPOMIB Working Group"
    CONTACT-INFO
       "PTOPOMIB WG Discussion:
```

```
        ptopo@3com.com
        Subscription:
        majordomo@3com.com
          msg body: [un]subscribe ptopomib

        Andy Bierman
        Cisco Systems Inc.
        170 West Tasman Drive
        San Jose, CA 95134
        408-527-3711
        abierman@cisco.com

        Keith McCloghrie
        Cisco Systems Inc.
        170 West Tasman Drive
        San Jose, CA 95134
        408-526-5260
        kzm@cisco.com"
    DESCRIPTION
            "The MIB module for managing the Physical Topology Discovery
            Protocol."
    ::= { experimental xx }

LLDPMIBObjects   OBJECT IDENTIFIER ::= { LLDPMIB 1 }

-- MIB groups
LLDPConfig    OBJECT IDENTIFIER ::= { LLDPMIBObjects 1 }
LLDPStats     OBJECT IDENTIFIER ::= { LLDPMIBObjects 2 }


LLDPPortIdType ::=  TEXTUAL-CONVENTION
    STATUS        current
    DESCRIPTION
            "The type of index value used to represent a port component.

            If an object of this type has a value of 'ifIndexType(1)',
            then the associated 'port ID' value represents an ifEntry,
            with the same ifIndex value.

            If an object of this type has a value of
            'entPhysicalIndexType(2)', then the associated 'port ID'
            value represents an entPhysicalEntry, with the same
            entPhysicalIndex value."
    SYNTAX        INTEGER  {
                     ifIndexType(1),
                     entPhysicalIndexType(2)
                  }

--
--  ***********************************************************
--
--                     L L D P    C O N F I G
--
--  ***********************************************************
--
--   The Physical Topology Discovery Protocol Configuration Group
```

```
LLDPAdminStatus OBJECT-TYPE
    SYNTAX      INTEGER  {
                        enabled(1),
                        disabled(2)
                }
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "The administratively desired status of the the local LLDP
            agent.

            If the agent is capable of storing non-volatile
            configuration, then the value of this object must be
            restored after a re-initialization of the management
            system."
    ::= { LLDPConfig 1 }

LLDPOperStatus OBJECT-TYPE
    SYNTAX      INTEGER  {
                        enabled(1),
                        disabled(2)
                }
    MAX-ACCESS  read-only
    STATUS      current
    DESCRIPTION
            "The current operational status of the local LLDP agent."
    ::= { LLDPConfig 2 }

LLDPMessageTxInterval OBJECT-TYPE
    SYNTAX      Integer32 (5..32768)
    UNITS       "seconds"
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "The interval at which LLDP messages are transmitted on
            behalf of this LLDP agent.

            If the agent is capable of storing non-volatile
            configuration, then the value of this object must be
            restored after a re-initialization of the management
            system."
    DEFVAL       { 60 }
    ::= { LLDPConfig 3 }

LLDPMessageTxHoldMultiplier OBJECT-TYPE
    SYNTAX      Integer32 (2..10)
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION
            "The time-to-live value expressed as a multiple of the
            LLDPMessageTxInterval object.  The actual time-to-live value
            used in LLDP messages, transmitted on behalf of this LLDP
            agent, can be expressed by the following formula:
            TTL = min(65535, (LLDPMessageTxInterval *
LLDPMessageTxHoldMultiplier))

            For example, if the value of LLDPMessageTxInterval is '60',
```

```
            and the value of LLDPMessageTxHoldMultiplier is '3', then the
            value '180' is encoded in the TTL field in the LLDP header.

            If the agent is capable of storing non-volatile
            configuration, then the value of this object must be
            restored after a re-initialization of the management
            system."
    DEFVAL        { 3 }
    ::= { LLDPConfig 4 }

--
-- LLDPSuppressTable:
--   Disable LLDP activity on individual local ports

LLDPSuppressTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF LLDPSuppressEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A table controlling LLDP message transmission on individual
            interfaces, ports, or backplanes."
    ::= { LLDPConfig 6 }

LLDPSuppressEntry   OBJECT-TYPE
    SYNTAX      LLDPSuppressEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "LLDP message configuration information for a particular
            port.  The port must be contained in the same chassis as the
            LLDP agent. LLDP messages will not be transmitted or received
            on the indicated port, even if the port is enabled.

            If the agent is capable of storing non-volatile
            configuration, then each active LLDPSuppressEntry must be
            re-created after a re-initialization of the management
            system.  An agent should store enough information about the
            associated entPhysicalEntry (e.g., entPhysicalAlias) or
            ifEntry (e.g. ifAlias), to properly re-create the entry,
            even if the LLDPSuppressChassisId and/or LLDPSuppressPortId
            values change across a system re-initialization."
    INDEX  {
        LLDPSuppressChassisId,
        LLDPSuppressPortIdType,
        LLDPSuppressPortId
    }
    ::= { LLDPSuppressTable 1 }

LLDPSuppressEntry ::= SEQUENCE {
      LLDPSuppressChassisId       PhysicalIndex,
      LLDPSuppressPortIdType      LLDPPortIdType,
      LLDPSuppressPortId          Integer32,
      LLDPSuppressRowStatus       RowStatus
}

LLDPSuppressChassisId  OBJECT-TYPE
    SYNTAX      PhysicalIndex
```

```
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The entPhysicalIndex value used to identify the chassis
            component associated with this entry. The associated
            entPhysicalEntry must be active, and the associated
            entPhysicalClass object must be equal to 'chassis(3)'."
    ::= { LLDPSuppressEntry 1 }

LLDPSuppressPortIdType OBJECT-TYPE
    SYNTAX        LLDPPortIdType
    MAX-ACCESS    not-accessible
    STATUS        current
    DESCRIPTION
            "The type of index value contained in the associated
            LLDPSuppressPortId object."
    ::= { LLDPSuppressEntry 2 }

LLDPSuppressPortId    OBJECT-TYPE
    SYNTAX      Integer32  (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The index value used to identify the port component of this
            entry.  The type of index value depends on the
            LLDPSuppressPortIdType value for this entry.

            If the associated LLDPSuppressPortIdType is equal to
            'ifIndexType(1)', then  this LLDPSuppressPortId represents an
            ifEntry with the same ifIndex value.  The associated ifEntry
            must be active, and represent a physical interface on the
            local chassis.

            If the associated LLDPSuppressPortIdType is equal to
            'entPhysicalIndexType(2)', then this LLDPSuppressPortId
            represents an entPhysicalEntry with the same
            entPhysicalIndex value.  The associated entPhysicalEntry
            must be active, and the associated entPhysicalClass object
            must be equal to 'port(10)' or 'backplane(4)'.

            Note that some devices, such as repeaters, cannot restrict
            frame transmission to a single port, but rather to a group
            of ports.  In such an event, an agent will disable LLDP
            activity on all ports in the port group, if any of the
            individual ports in the group are specified in this table."
    ::= { LLDPSuppressEntry 3 }

LLDPSuppressRowStatus  OBJECT-TYPE
    SYNTAX        RowStatus
    MAX-ACCESS    read-create
    STATUS        current
    DESCRIPTION
            "The status of this entry."
    ::= { LLDPSuppressEntry 4 }

--
--  ***********************************************************
```

```
--
--                    L L D P    S T A T S
--
--  **********************************************************
--
-- LLDP Stats Group

LLDPStatsTable   OBJECT-TYPE
    SYNTAX      SEQUENCE OF LLDPStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "A table containing LLDP statistics for individual ports.

            Entries are not required to exist in this table while the
            LLDPAdminStatus or LLDPOperStatus objects are equal to
            'disabled(2)'.
            Entries are not required to exist in this table if a
            corresponding entry (with identical index values) exists in
            the LLDPSuppressTable."
    ::= { LLDPStats 1 }

LLDPStatsEntry   OBJECT-TYPE
    SYNTAX      LLDPStatsEntry
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "LLDP message statistics for a particular port.  The port
            must be contained in the same chassis as the LLDP agent."
    INDEX  {
             LLDPStatsChassisId,
             LLDPStatsPortIdType,
             LLDPStatsPortId
    }
    ::= { LLDPStatsTable 1 }

LLDPStatsEntry ::= SEQUENCE {
      LLDPStatsChassisId       PhysicalIndex,
      LLDPStatsPortIdType      LLDPPortIdType,
      LLDPStatsPortId          Integer32,
      LLDPStatsInGoodPkts      Counter32,
      LLDPStatsInErrors        Counter32,
      LLDPStatsOutPkts         Counter32
}

LLDPStatsChassisId  OBJECT-TYPE
    SYNTAX      PhysicalIndex
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The entPhysicalIndex value used to identify the chassis
            component associated with this entry. The associated
            entPhysicalEntry must be active, and the associated
            entPhysicalClass object must be equal to 'chassis(3)'."
    ::= { LLDPStatsEntry 1 }

LLDPStatsPortIdType OBJECT-TYPE
```

```
    SYNTAX        LLDPPortIdType
    MAX-ACCESS    not-accessible
    STATUS        current
    DESCRIPTION
            "The type of index value contained in the associated
            LLDPStatsPortId object."
    ::= { LLDPStatsEntry 2 }

LLDPStatsPortId    OBJECT-TYPE
    SYNTAX      Integer32  (1..2147483647)
    MAX-ACCESS  not-accessible
    STATUS      current
    DESCRIPTION
            "The index value used to identify the port component of this
            entry.  The type of index value depends on the
            LLDPStatsPortType value for this entry.

            If the associated LLDPStatsPortIdType is equal to
            'ifIndexType(1)', then  this LLDPStatsPortId represents an
            ifEntry with the same ifIndex value.  The associated ifEntry
            must be active, and represent a physical interface on the
            local chassis.

            If the associated LLDPStatsPortIdType is equal to
            'entPhysicalIndexType(2)', then this LLDPStatsPortId
            represents an entPhysicalEntry with the same
            entPhysicalIndex value.  The associated entPhysicalEntry
            must be active, and the associated entPhysicalClass object
            must be equal to 'port(10)' or 'backplane(4)'."
    ::= { LLDPStatsEntry 3 }

LLDPStatsInGoodPkts  OBJECT-TYPE
    SYNTAX        Counter32
    MAX-ACCESS    read-only
    STATUS        current
    DESCRIPTION
            "The number of valid LLDP messages received by this LLDP agent
            on the indicated port, while this LLDP agent is enabled."
    ::= { LLDPStatsEntry 4 }

LLDPStatsInErrors  OBJECT-TYPE
    SYNTAX        Counter32
    MAX-ACCESS    read-only
    STATUS        current
    DESCRIPTION
            "The number of invalid LLDP messages received by this LLDP
            agent on the indicated port, while this LLDP agent is
            enabled. A LLDP message may be invalid for several reasons,
            including:
               - invalid MAC header; length or DA fields
               - invalid LLDP header; version or flags fields
               - invalid LLDP VarBindList ASN.1/BER encoding
               - invalid or missing LLDP VarBindList data elements"
    ::= { LLDPStatsEntry 5 }

LLDPStatsOutPkts  OBJECT-TYPE
    SYNTAX        Counter32
```

```
    MAX-ACCESS    read-only
    STATUS        current
    DESCRIPTION
            "The number of LLDP messages transmitted by this LLDP agent on
            the indicated port."
    ::= { LLDPStatsEntry 6 }


-- conformance information
LLDPConformance OBJECT IDENTIFIER ::= { LLDPMIB 2 }

LLDPCompliances OBJECT IDENTIFIER ::= { LLDPConformance 1 }
LLDPGroups      OBJECT IDENTIFIER ::= { LLDPConformance 2 }

-- compliance statements

LLDPCompliance MODULE-COMPLIANCE
   STATUS  current
    DESCRIPTION
            "The compliance statement for SNMP entities which implement
            the LLDP MIB."
    MODULE  -- this module
        MANDATORY-GROUPS { LLDPConfigGroup, LLDPStatsGroup }

    ::= { LLDPCompliances 1 }

-- MIB groupings

LLDPConfigGroup    OBJECT-GROUP
    OBJECTS {
        LLDPAdminStatus,
        LLDPOperStatus,
        LLDPMessageTxInterval,
        LLDPMessageTxHoldMultiplier,
        LLDPSuppressRowStatus
    }
    STATUS  current
    DESCRIPTION
            "The collection of objects which are used to configure the
            Link Layer Discovery Protocol implementation behavior.

            This group is mandatory for agents which implement the Link Layer
            Discovery Protocol."
    ::= { LLDPGroups 1 }

LLDPStatsGroup    OBJECT-GROUP
    OBJECTS {
        LLDPStatsInGoodPkts,
        LLDPStatsInErrors,
        LLDPStatsOutPkts
    }
    STATUS  current
    DESCRIPTION
            "The collection of objects which are used to represent Link Layer
            Discovery Protocol statistics.

            This group is mandatory for agents which implement the Link Layer
```

```
            Discovery Protocol."
    ::= { LLDPGroups 2 }

END
```

Persistent Identifiers

The PTOPO MIB [RFC2922] utilizes non-volatile identifiers to distinguish individual chassis and port components. These identifiers are associated with external objects in order to relate topology information to the existing managed objects.

In particular, an object from the Entity MIB or Interfaces MIB can be used as the 'reference-point' for a connection component identifier.

Relationship to the Physical Topology MIB

The Physical Topology MIB [RFC2922] allows a LLDP Agent to expose learned physical topology information, using a standard MIB. LLDP is intended to fully support the PTOPO MIB.

Relationship to Entity MIB

The Entity MIB [RFC2037] allows the physical component inventory and hierarchy to be identified. The chassis identifier strings passed in LLDP messages identify entPhysicalTable entries, and implementation of the entPhysicalTable as specified in the Version 1 of the Entity MIB [RFC2037], and implementation of the entPhysicalAlias object from Version 2 of the Entity MIB [ENTITY-MIB], are required for SNMP agents which also implement the LLDP MIB.

Relationship to Interfaces MIB

The Interfaces MIB provides a standard mechanism for managing network interfaces. The port identifier strings passed in LLDP messages identify ifTable (or entPhysicalTable) entries, and implementation of the ifTable and ifXTable [RFC2233] are required for SNMP agents which also implement the LLDP MIB, for the ports which are represented in the Interfaces MIB.

Security Considerations

This protocol and associated MIB can expose the existence of physical components, MAC layer addresses, and network layer addresses, pertaining to devices within a given network. A network administrator may wish to restrict access to this management information, using SNMP access control mechanisms, and restrict LLDP message processing to a particular set of ports, by configuring entries in the LLDPSuppressTable.

References

[RFC2737]
McCloghrie, K., and A. Bierman, "Entity MIB (Version 2)", RFC 2737, Cisco Systems, December 1999.

[RFC2922]
Bierman, A., and K. Jones, "Physical Topology MIB", RFC 2922, Cisco Systems, Bay Networks, November 1998.

[RFC2037]
McCloghrie, K., and A. Bierman, "Entity MIB using SMIv2", RFC 2037, Cisco Systems, October 1996.

[RFC2233]
McCloghrie, K., and F. Kastenholtz, "The Interfaces Group MIB using SMIv2", RFC 2233, Cisco Systems, FTP Software, November 1997.

[RFC3232]
Reynolds, J. “Assigned Numbers: RFC 1700 is Replaced by an On-line Database”, RFC 3232, RFC Editor, January 2002