Extending Link Layer Discovery Protocol for Larger Databases

Paul Bottorff

Aruba a Hewlett Packard Enterprise Company

Paul.Bottorff@hpe.com

Abstract

The Link Layer Discovery Protocol (LLDP, IEEE Std 802.1AB) is widely used throughout the network industry in enterprise, data center, and carrier networks for discovering links, stations, port characteristics, network topologies, and device attributes. However, the current LLDP database size limitations prevents LLDP from supporting more demanding network discovery applications such as router discovery. Currently, LLDP limits the size of the information advertised from a port to a single frame of information. Here we propose a protocol for extending LLDP to support multi-frame databases allowing a port to advertise over 100 Kilobytes of information. The technique proposed for extending LLDP is backward compatible with existing LLDP implementations to allow easy deployment in existing networks.

Introduction

The LLDP protocol executes over network links to build a distributed database with information from all the neighbor ports attached to each link. Each LLDP Protocol Data Unit (LLDPDU) used to advertise information is encoded within a single frame as a series of Type Length Values (TLVs). The TLVs exchanged by LLDP contain information advertised from the originating port. The LLDP TLV's formats and usage are defined by multiple organizations which fall into three catagories: 1)the IEEE 802.1 committee, 2)other standards organizations, 3)any other organization using an IEEE 802 Organizationally Unique Identifier (OUI).

Currently the IEEE 802.1 and 802.3 committees define 30 TLVs which LLDP can advertise from a port. These include the basic set defined in the LLDP standard IEEE Std 802.1AB-2016, additional TLVs defined in IEEE Std 802.1Q-2018, and TLVs defined in IEEE Std 802.3-2018. Some of these TLVs are variable in size and some can have multiple copies in the databases. If we size a database containing a single copy of the maximum sized TLV for all current 802.1 and 802.3 TLVs, the size of the LLDPDU would be about 2533 octets which is larger than fits in the maximum LLDPDU for a common Ethernet (i.e. 1500 bytes for a basic Ethernet without VLAN tagging). These numbers do not consider TLVs with multiple copies, TLVs defined by other standards organizations or TLVs defined by other organizations. Current LLDP implementations work around these limitations by not including all the TLVs and rarely use maximum sized TLVs, however, applications for LLDP continue to grow. The information exchanged will eventually requiring LLDP extensions or force users to support multiple discovery protocols.

A major application for an extended discovery protocol is discovering router to router interfaces along with their network addressing and characteristics. This application is driven by the rise of new routing protocols currently under development at the Internet Engineering Task Force (IETF) such as Link State Vector Routing (lsvr). Though traditional link state routing protocols such as Open Shortest Path First (OSPF) and Intermediate System to Intermediate System (IS-IS) have built in discovery protocols, other routing protocols such as Boarder Gateway Protocol (BGP) do not. Given the broad deployment of LLDP an extended LLDP with sufficient database capacity would allow LLDP to support the discovery for these new routing protocols consolidation network operation to a single discovery protocol.

Another application for an extended LLDP protocol is support for Time Sensitive Networks (TSN) where the network imposes a small maximum frame size to limit latency for time sensitive traffic. For these TSN networks the amount of information LLDP can exchange can be highly constrained since current LLDP can only advertise a single frame of information from a port even if the single frame is restricted to a small size, such as 64 bytes. The ability to advertise LLDP information using multiple frames would allow these LLDP applications to use LLDP without highly constrained LLDP database sizes.

New Discovery Protocols

Currently we have multiple new discovery and configuration protocols at the IEEE and IETF. These are targeted at specific applications for data center virtualization, stream registration, and router discovery.

At the IETF work on new link state protocols for data centers such as Link State Vector Routing (lsvr) has resulted in new discovery protocol proposals: 1)the Layer 3 Discovery and Liveness protocol ([3], https://datatracker.ietf.org/doc/draft-ietf-lsvr-l3dl/); 2)the BGP Neighbor Discovery protocol ([4], https://datatracker.ietf.org/doc/draft-xu-idr-neighbor-autodiscovery/).

To support virtualization in data centers the IEEE has standardized the Virtual Station Interface Discovery and Configuration Protocol (VDP, Clause 41 of IEEE Std 802.1Q-2018, IEEE Std 802.1Qcy-2019) which is used to exchange information about a virtual interface within a server with an edge bridge or router. The VDP protocol is specifically targeted at data center server virtualization and allows discovery, configuration, state, and liveliness of virtualized interfaces of virtual machines residing within a server. The VDP protocol may scale to any number of Virtual Station Interfaces (VSIs).

To support the requirements of industrial automation TSNs the IEEE is standardizing the Link-local Registration Protocol (LRP, IEEE P802.1CS) and the Resouce Allocation Protocol (RAP, IEEE P802.1Qdd). The P802.1CS protocol provides the ability to replicate a database optimized for 1 Megabyte of data over a data link. The Resource Allocation Protocol runs as a distributed application over the Link-local Registration Protocol to support large stream reservation databases for industrial applications.

These new discovery protocols from the IEEE and IETF are very capable at their target applications, however are more complex than required for simple discovery applications. LLDP on the other hand is a general purpose discovery protocol which can be used for a wide variety of applications and is already very broadly deployed.

Objectives for an Extended LLDP Protocol

The IEEE has approved a project to extended LLDP (P802.1ABdh, https://1.ieee802.org/tsn/802-1abdh/) which intends to:

- Extend LLDP to support transmission and reception of a set of Link Layer Discovery Protocol (LLDP) Type Length Values (TLVs) that exceed the space available in a single frame;
- Maintains existing functionality while communicating with a peer LLDP that supports updated functionality;
- restrict the size of the LLDP Data Unit (LLDPDU) and extensions in order to meet timing constraints in the network

For the protocol proposed in this paper we consider some clarifications and additional objectives which are motivated by ongoing work in other committees and by general protocol considerations.

The IEEE objectives leave open the question of how big a database should be supported. The discovery application on the forefront with the largest amount of information exchange appears to be router discovery. If we look at the IETF l3dl protocol the IETF has settled on a database size of around 64K octets as sufficient. Looking a bit more at the IETF they could support 64K bytes of addressing for each protocol stack supported and so in a dual stack application they could support 128K octets of TLVs. It appears the IETF limits are simply based on numbers large enough to not raise objects, therefore these are probably an upper bound on the requirements.

Given the disposition of the IETF we can assume a reasonable objective for router discovery would be around 100K octet database.

The IEEE project objectives provide for backward compatibility with existing LLDP implementations. The objective here is that a new P802.1ABdh implementation is able to exchange a single frame of information with an existing IEEE Std 802.1AB implementation without any changes in the existing LLDP implementation. Of course an existing 802.1AB implementation will not be able to exchange the extended database.

The requirement for supporting small maximum frame sizes has many unexplored variables. First, it is important to determine how large the required database will be for this application since small frame sizes will constrain the maximum database size even with multiple frame support. In addition, we need to determine if the TLV sizes can be constrained when using small database sizes or if TLVs need to be fragmented over frames. We don't have any clear figures on these requirements at the present and so propose providing the extensions which follow from multi-frame support without TLV fragmentation. The constrains which result include:

- TLVs may not span multiple frames
- the chassis+port+ttl+extension TLVs must fit in a single frame
- the number of additional frames allowed in the database will be constrained along with frame size

For small max frame sizes the extended LLDP database can be small though larger than a single frame database. For instance if the maximum frame size is chosen to be 64 bytes (minimum Ethernet frame) and we assume the chassis and port are identified by MAC addresses then the maximum database size would be on the order of 400 bytes.

The implementation needs to operate over both shared and point-to-point media. This requirement also includes handling duplicate addresses. The most common case for duplicate L2 switch addresses is a result of switches which share a single port MAC Address on all switch ports. Though this is not standards practice it is common industry practice. Another problem can occur when an address is duplicated on different stations, which is a possible misconfiguration.

The current LLDP protocol is a multicast protocol, however for support of larger databases it is desirable to use a unicast protocol along with data pacing to prevent overflow. This will avoid the historic problems seen in protocols such as IS-IS and OSPF which use multicast to build distributed databases.

Another historic problem with multicast protocols is the required periodic transmission result in background traffic which can overloading station control processors. To prevent these overloads it is desirable to only transmit/update when there are changes in the database. Also it is desirable to only transmit the frames which require updating.

For router discovery applications two additional requirements are being considered by the IETF. These are authentication of the database and liveness. The LLDP protocol makes periodic updates and supports a time to live, however this is at a very slow speed and so is a poor liveliness indication. The LLDP protocol is not well suited to frequent keep alive exchanges, however rather than use LLDP for liveness the IEEE Connectivity Fault Management protocol (CFM, IEEE Std 802.1Q-2018 cl 18-22) already has the ability to implement very capable keep alives.

Authentication for LLDP is a topic which needs focused study by security experts. As such it is more rational for IEEE to launch a separate security project where the topic can be handled in depth rather than try to add authentication to the exiting P802.1ABdh project.

Summarizing these objectives for an extended LLDP protocol:

- Support advertisement of multi-frame databases from each link neighbor of over 100K bytes when using

maximum Ethernet sized frames;
- Support exchange of a single-frame database from each link neighbor with any previous LLDP version
- Support multi-frame databases for small frames where the chassis+port+ttl+extension TLVs fit in a single frame and where the size of the advertised database can be constrained by the frame size
- Support both shared and pt-pt media
- Operation on links where MAC addresses are duplicated
- Minimize multicast use, periodic transmissions, and database updates
- Support pacing of data by receivers

Current LLDP Operation

LLDP is a layer 2 protocol that allows a station to advertise capabilities and state at each MAC attached to a link. Each LLDP agent uses the information advertised by other LLDP agents from each MAC attached to the link to build a database composed of remote information advertised by the other agents along with its local information.

An LLDP (IEEE Std 802.1AB-2016) agent transmits information associated with a local MAC periodically (figure 1, A). The transmission is typically to a multicast address (unicast is supported for Wi-Fi networks) which is selected from those multicasts that do not pass through 802 bridges. Transmission begins when the local MAC is enabled and continues periodically (figure 1, A and B). Rapid updates are performed whenever some information in the local database changes (figure 1, C).

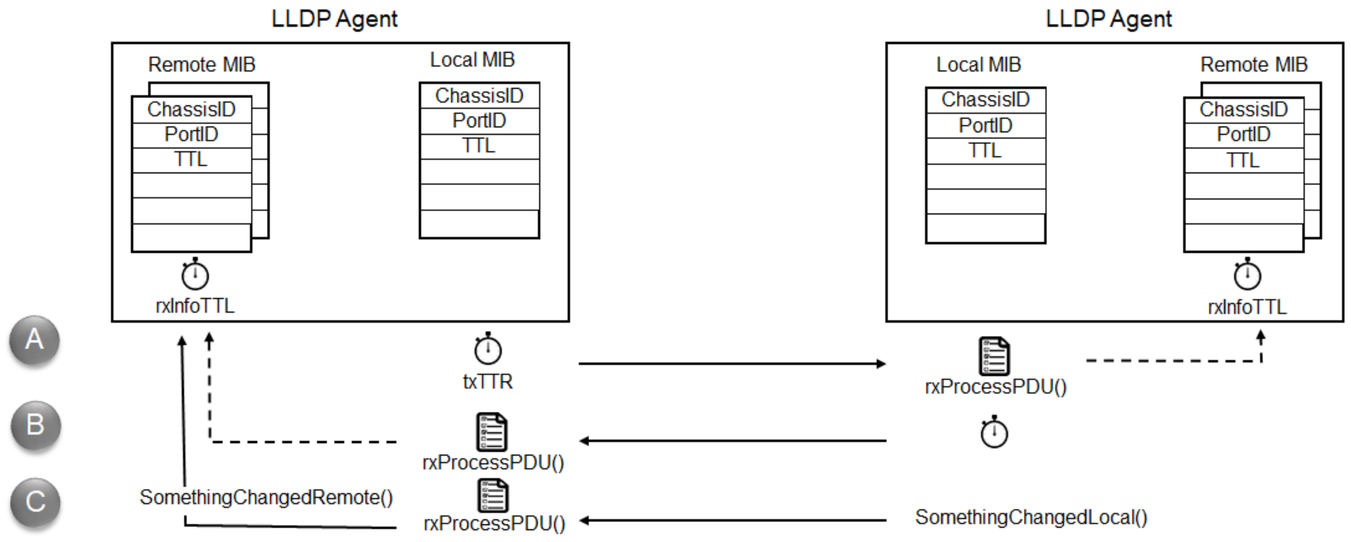


Figure 1: IEEE Std 802.1AB-2016 Link Layer Discovery Protocol Overview

Extending LLDP

The basic idea used to extend the 802.1AB-2016 protocol to multiple PDU databases is to encode a description of the extension PDUs within the initial frames. We call the initial frame, which is the single frame exchanged by the 802.1AB-2016 protocol, the LLDP Foundation PDU or F-PDU. The F-PDU is an 802.1AB-2016 LLDPDU which includes a description of the extension PDUs (X-PDUs). The description of all X-PDUs is carried in and LLDP TLV within the F-PDU called the Manifest TLV. The Manifest TLV contains a unique identifier for each X-PDU that changes whenever any revision is made an X-PDU. In this way any change in the database is reflected by a change in the F-PDU.

The Manifest TLV included in the F-PDU is used by the receivers that support the extended LLDP (XLLDP) to request transmission of the X-PDUs specified by the list of descriptors within the Manifest TLV.

If an 802.1AB-2016 implementation receives an F-PDU with a Manifest TLV it simply ignores the Manifest TLV since it is an unrecognized TLV type. An 802.1AB-2016 implementation therefore operates without the X-PDUs, however maintains its existing single LLDPDU operation.

The XLLDP is a unicast request response protocol where the XLLDP receiver requests each X-PDU. The XLLDP only makes requests for X-PDUs that are required to update its database, therefore X-PDU requests are only made to build the initial database and when an X-PDU is changed or added to the database. The XLLDP uses the X-PDU descriptions listed in the Manifest TLV to determine if it needs to update an X-PDU by comparing the Manifest TLV list with the current database.

The unicast addresses used by XLLDP to make requests and responses are those used as the source MAC address in the frame which encodes the F-PDU. Additional addressing is provided on each request by specifying the destination chassisID+portID rather than the source chassisID+portID used in a conventional LLDPDU. Response X-PDUs use the source chassisID+portID as a normal LLDPDU transmission.

Each request PDU specifies a list of X-PDUs for the source to transmit. The requester can use the requests to pace transmission by controlling the number and spacing between requests and can request re-transmission if desired. The XLLDP allows only a single outstanding request to each remote LLDP agent.

New PDU Formats

Three new types of PDUs are used by the Extended Link Layer Discovery Protocol (XLLDP). These are the Foundation PDU (F-PDU), the Extension PDU (X-PDU) and the Extension Request PDU (XREQ-PDU).

Foundation PDU (F-PDU)

Figure 2 shows the format for the F-PDU. The F-PDU is an LLDPDU as specified by 802.1AB-2016 and is identified by the LLDP Ethertype as specified in 802.1AB-2016. An F-PDU is a particular type of LLDPDU which contains a Manifest TLV describing a list of X-PDUs. The Manifest TLV is located following the TTL TLV within the F-PDU. An 802.1AB-2016 implementation which receives an F-PDU will simply ignore the Manifest TLV processing the remainder of TLVs as normal.

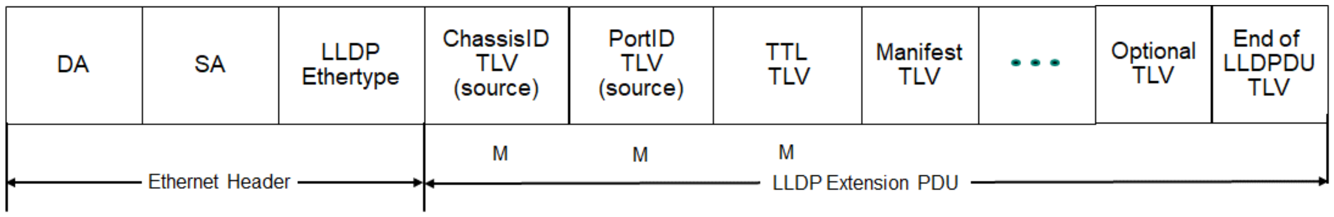


Figure 2: Foundation PDU (F-PDU) Format

The Manifest TLV format shown in figure 3 is identified by a LLDP TLV type. The Manifest TLV contains a 7 bit field with the number of X-PDUs described by the Manifest TLV followed by an array of n 6 octet Extension PDU descriptors. The number of X-PDUs is indicated by a separate field form the TLV length to allow implementations to use a fixed size Manifest TLV with a variable number of Extension PDU descriptions. If the number of Extension PDUs is less than the TLV string length, then the first n Extension PDU descriptors are valid.

Each X-PDU desciptor in the Manifest TLV provides a unique identifier for the X-PDU. The identifier is composed of 3 fields which are a 7 bit X-PDU identifier, an 8 bit X-PDU revision, and a 32 bit X-PDU check. The X-PDU number along with the X-PDU revision provide a 16 bit identifier which uniquely identifies most X-PDU. Encoding the X-PDU number and revision within the Manifest TLV allows insertion and deletion of X-PDUs without updating the other X-PDUs. The check is used to cover boundary cases where the X-PDU number and revision fail to uniquely identify the X-PDU. The check bits are calculated using all octets of the X-PDU in an MD5 hash calculation and then taking the low order 32 bits of the result. An example where the check would be used is the case where an X-PDU number is re-used and the X-PDU revision is restarted rather than updated, then the X-PDU check along with the new random X-PDU revision provide uniquess for the new entry within an error of about 1 in a trillion. It is expected this condition would only happen when the XLLDP agent looses its state for instance in in a power up reset.

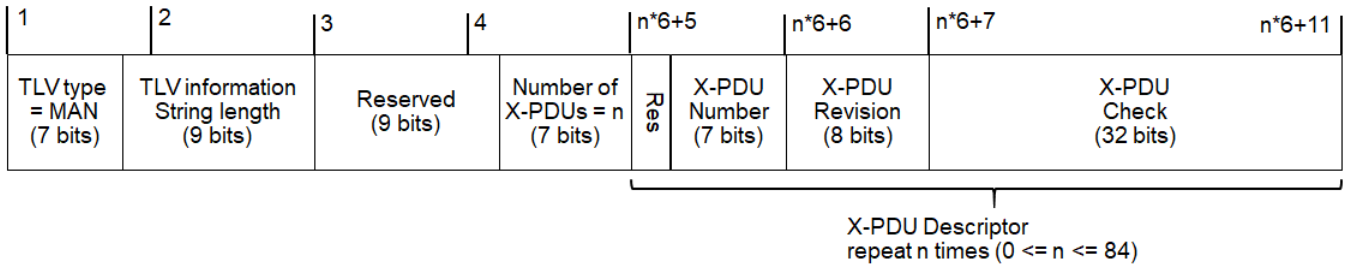


Figure 3: Manifest TLV Format

Extension PDU (X-PDU)

Figure 4 shows the format of an Extension PDU (X-PDU). An X-PDU is identified by an XLLDP Ethertype plus an XLLDP subtype. This Ethertype is different form the LLDP Ethertype to assure the XLLDPDUs are not mistakenly used by an 802.1AB-2016 implementation. The subtype is used to identify the XLLDPDU as an X-PDU. There is also a XLLDP revision number which is intended for future protocol revisions.

Each X-PDU contains a ChassisID TLV and PortID TLV which identify the source for the X-PDU. There is no time to live TLV in an X-PDU since the TTL is set from the F-PDU which established the database. Following the ChassisID and PortID TLVs is an Extension PDU Identifier TLV (XID TLV) used to identify this X-PDU. The Extension Identifier TLV can be followed by any number of optional TLVs.

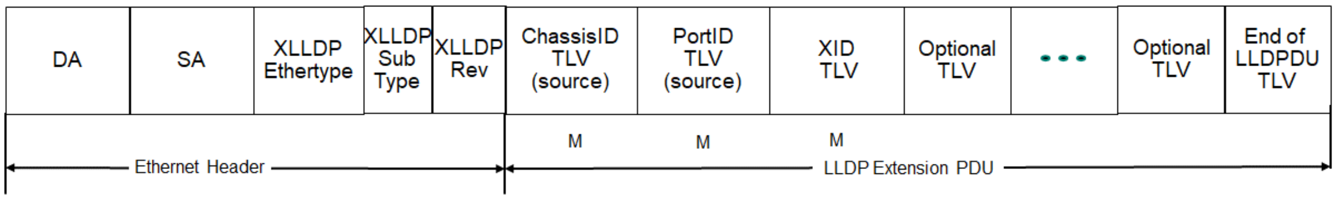


Figure 4: Extension PDU (X-PDU) Format

The X-PDU Identifier TLV (figure 5) uniquely identifies this X-PDU within the database addressed by the ChassisID+PortID using an X-PDU number and an X-PDU revision. The 7 bit X-PDU number identifies this X-PDU from the other X-PDUs local to this XLLDP agent. The 8 bit X-PDU revision indicates the revision since database initialization. The X-PDU revision starts at a randomly selected number on reset and is incremented modulo 256 whenever any change is made to the PDU. At reset the X-PDU revision number starts at a new randomly selected number. The 32 bit check, used in the Manifest TLV to verify the unique identification of the X-PDU, is not carried in the X-PDU and therefore is calculated when a new X-PDU is received by the XLLDP agent.

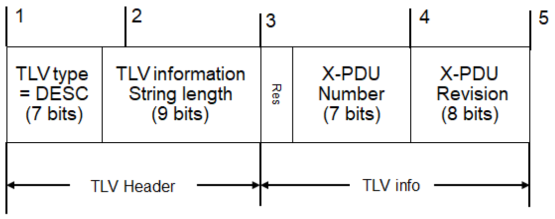


Figure 5: Extension PDU Identifier TLV (XID TLV) Format

Extension Request PDU (XREQ-PDU)

Figure 6 shows the format of the Extension Request PDU (XREQ-PDU). An XREQ-PDU is identified by the XLLDP Ethertype plus an XLLDP subtype. This Ethertype is different form the LLDP Ethertype to assure the XREQ-PDUs will not be used mistakenly by an 802.1AB-2016 implementation. The subtype is used to identify the XLLDP PDU as an XREQ-PDU. There is also a XLLDP revision number which is intended for future protocol revisions.

Each XREQ-PDU contains a ChassisID TLV and PortID TLV which identify the destination for the XREQ-PDU rather than the source like a normal F-PDU or X-PDU. There is no time to live TLV in an XREQ-PDU. Immediately following the destination ChassisID and PortID TLVs is an XREQ-TLV which provides a list of the wanted X-PDUs. No other TLVs except the End of LLDPDU TLV are present in the XREQ-PDU.

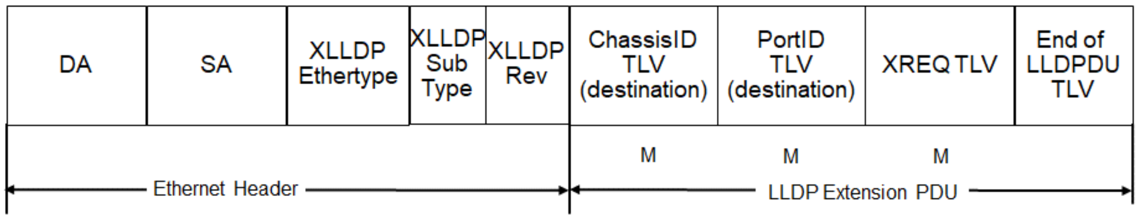


Figure 6: Extension Request PDU Format

Each XREQ-PDU contains an XREQ-TLV (figure 7). The XREQ-TLV contains a list of X-PDU descriptors which the sender of the XREQ-PDU needs updated. Each descriptor list element provides the X-PDU number, revision and check for the needed X-PDU. The number of X-PDUs descriptors encoded in the XREQ-TLV is determined by the length of the TLV. Each XREQ-TLV also includes a 16 bit request number which the sender increments modulo 64K for each new request issued. On reset the request number begins at a randomly selected starting point.

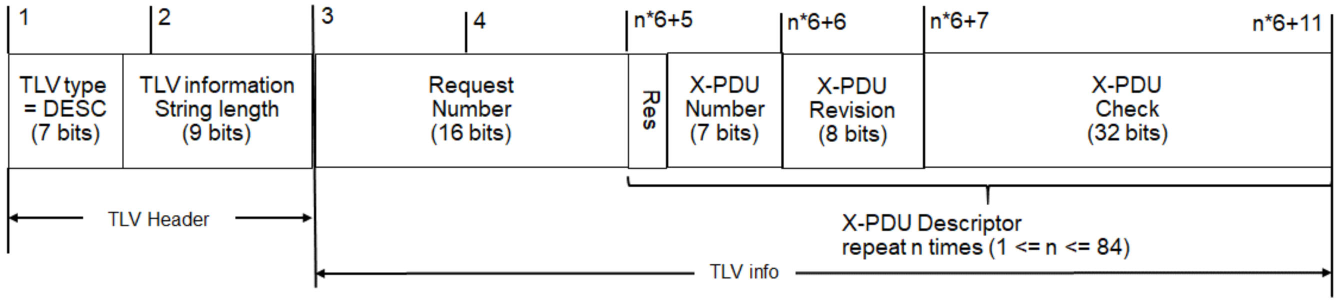


Figure 7: Extension PDUs Request TLV (XREQ TLV) Format

Extended Link Layer Discovery Protocol (XLLDP)

Figure 8 illustrates the XLLDP operation. Any LLDP exchange including an XLLDP exchange begins with the

LLDP protocol as specified in IEEE Std 802.1AB-2016. LLDP is used to establish all databases and to exchange the F-PDUs which describe the X-PDUs extending each database to multiple PDUs (figure 8, G). The XLLDP never creates new databases, however is used to transfer the X-PDUs which extend the databases established by LLDP.

XLLDP does not run periodically, but rather is triggered by the receipt of an F-PDU containing a Manifest TLV with descriptors for X-PDUs or X-PDU updates not currently in the database. Once missing or revised X-PDUs are identified by comparing the Manifest TLV X-PDU descriptions with the current database, XLLDP is responsible for obtaining any required X-PDUs and updating the database. Once the database is up to date XLLDP will not run again until it is determines through the Manifest TLV that the database needs to update some X-PDUs. Since the Manifest TLV is transmitted in the F-PDU by LLDP, any change to an X-PDU will be result in a change to the Manifest TLV and therefore a change in the F-PDU (figure 8. A). LLDP will therefore immediately update the F-PDU whenever any X-PDU is changed in turn resulting in XLLDP running to update the required X-PDUs (figure 8, B).

Once the required X-PDUs are determined by inspection of the changes to the Manifest TLV within the F-PDU, XLLDP will transfer the list of required X-PDUs identified from the Manifest TLV. XLLDP executes until it has retrieved all the needed X-PDUs or determined they are unavailable.

To transfer the X-PDUs XLLDP makes explicit requests for each needed X-PDU to the source XLLDP agent(figure 8, C). These requests are encoded in XLLDP XREQ-PDUs (figures 6, 7). Each XREQ-PDU can request multiple X-PDUs. Multiple XREQ-PDUs can be used by XLLDP to transfer the needed X-PDUs (figure 8, C&E). XLLDP only allows a single pending XREQ-PDU for each source XLLDP agent at a time, however may further limit the number of outstanding XREQ-PDUs to pace its receiver. XLLDP controls pacing of frames by the number of outstanding X-PDU requests and the time interval between these requests. XLLDP times out each XREQ-PDU. If no response is received after the timeout XLLDP can retry the request. After retrying XLLDP will assume the X-PDU is unavailable, log an error, and continue to recover all the X-PDUs that are available at this time.

Each XREQ-PDU is transmitted in a unicast MAC frame addressed to the MAC address provided as the source address of the frame carrying the F-PDU. The XREQ TLV carries a list of the descriptors from the Manifest TLV for the requested X-PDUs.

The receiver of an XREQ-PDU uses the ChassisID and PortID of the XREQ-PDU to determine what local database contains the requested X-PDUs (figure 8, C&D). If the receiver does not have the database matching the ChassisID and PortID of the XREQ-PDU it discards the XREQ-PDU. If the receiver has the database it locates the X-PDUs that match the XREQ TLV descriptors and transmits the X-PDUs (figure 2) within a frame addressed to the source MAC or the XREQ-PDU making the request. If the receiver has the database, however does not have the matching X-PDUs it sets somethingChangedLocal to initiate an update of the F-PDU.

To suppress duplicate responses the sender keeps track of the last request number from the most recent XREQ TLV. If a new XREQ TLV with a request number matching the last response is received the sender can discard the XREQ PDU and not send any of the requested X-PDUs.

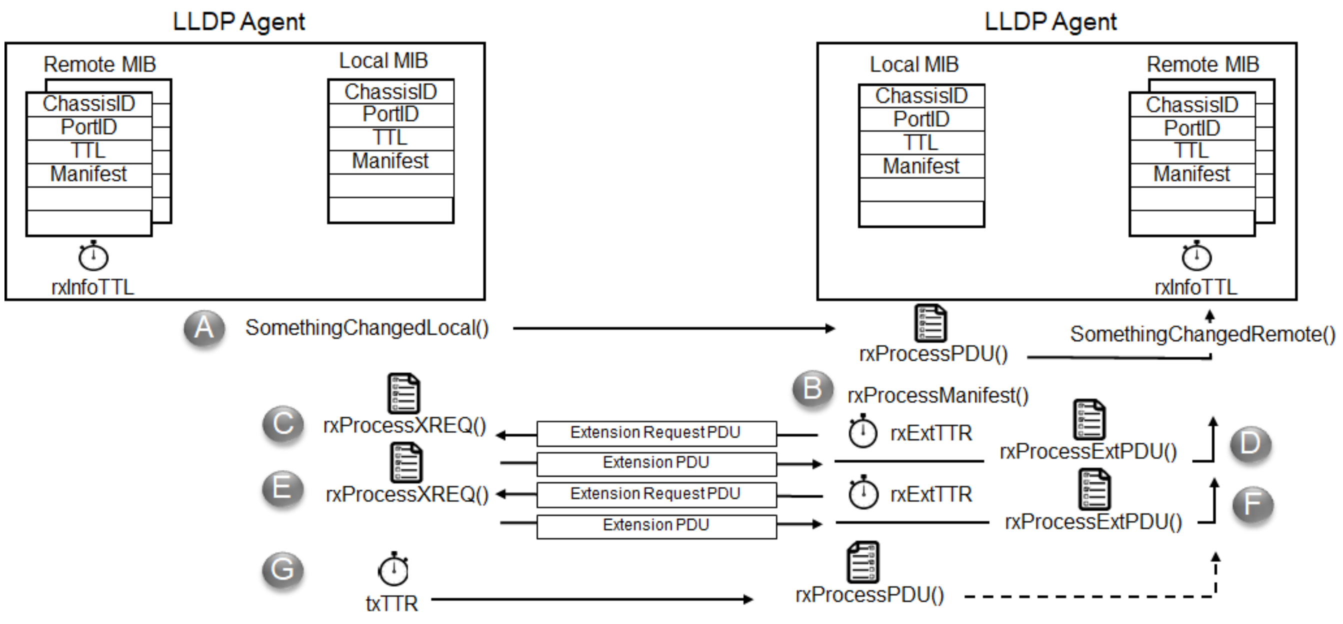


Figure 8: Extended LLDP (XLLDP) Operation

XLLDP Operation Over Links with Duplicate MAC Addresses

In most cases 802 links don’t have any duplicate MAC addresses, however duplicates are possible under some circumstances. LLDP is designed to operate correctly even when duplicate MAC addresses are present on the link and like LLDP the XLLDP is also resilient when operating on links with duplicate MACs.

To describe the operation of XLLDP when duplicate addresses are present on the link we consider a classic CSMA/CD Ethernet example (figure 9) with three cases of duplicate addresses:

- the switch sourcing an X-PDU can have multiple MACs connected to the link each with the same MAC address (figure 9, switch A);
- the switch requesting an X-PDU can have multiple MACs connected to the link, each with the same MAC address (figure 9, switch B);
- some switch, which is not a party to an X-PDU exchange, may have MACs connected to the link with duplicates of the MAC addresses of switch A or switch B (figure 9, switch C).

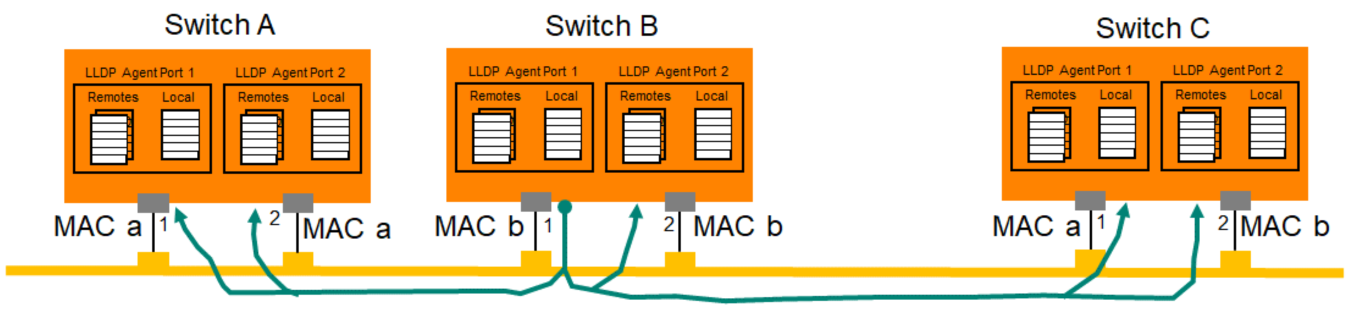
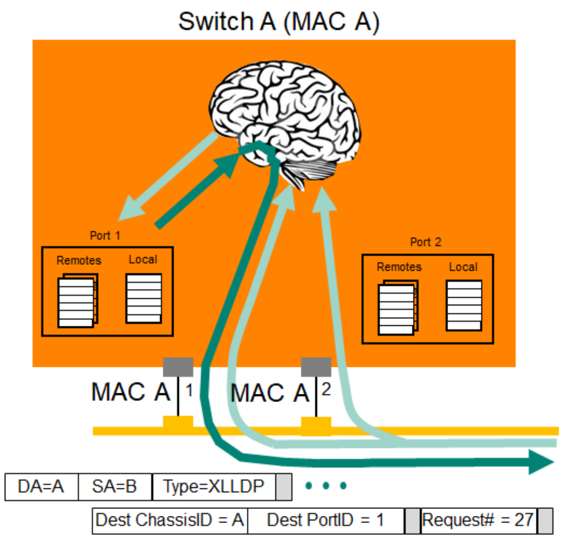


Figure 9: Example CSMA/CD Ethernet with Duplicate MAC Addresses

Figure 9 illustrates the example Ethernet connecting three switches. The switches each have 2 MACs attached to the Ethernet. The switch A uses the same MAC address “a” on both ports 1 and 2 connected to the Ethernet link. The switch B uses the same MAC address “b” on both of its ports 1 and 2 connected to the Ethernet link. The switch C has two ports 1 and 2 connected to the Ethernet with MAC address duplicates of the Switch A and B addresses. Any frame transmitted on this Ethernet link is delivered to all MACs attached to the Ethernet. Within a switch incoming frames are steered to the Higher Layer Entities, including LLDP and XLLDP, as described in IEEE Std 802.1Q-2018 subclause 8.5.3. In our example frames addressed to “a” and “b” will be directed to the Higher Layer Entities of switches A, B and C where they will be further decoded based on the LLDP and XLLDP Ethertypes. An XLLDP transmission from switch B to destination address “a” will be delivered to the XLLDP agent on switch A ports 1 and 2 and to the XLLDP agent on switch C port 1. An XLLDP transmission from switch A to destination address “b” will be delivered to the XLLDP agent on switch B ports 1 and 2 and to the XLLDP agent on switch C on port 2.

For example an XLLDP agent operating in switch B port 1 of the figure 9 network sends an XREQ-PDU for an X-PDU of the switch A port 1 database. Switch B sends the XREQ-PDU in a frame to MAC destination address “a” since this was the source address used by switch A LLDP to deliver the F-PDU which caused the request. The XREQ-PDU sent from switch B will then be received by the XLLDP agents for switch A ports 1 and 2 and switch C port 1. On switch C the XLLDP agent will look at the destination ChassisID and PortID contained in the XREQ-PDU and determine these are not for any of its ports and therefore will discard the XREQ-PDU. On switch A (figure 10) a copy of the XREQ-PDU will be delivered to the XLLDP agent for both ports 1 and 2. Since the XLLDP agent will begin by checking the destination ChassisID and PortID the XLLDP agents will determine the XREQ-PDU is intended for the port 1 database. The XLLDP agent at port 1 checks the XREQ TLV request number against the last response it sent. If a response has already been sent for this XREQ XLLDP deletes the request. Otherwise, the XLLDP agent transmits the requested X-PDUs and records the XREQ TLV request number for future reference.

In our example the XLLDP agent operating in switch A port 1 will send the X-PDUs requested by the XREQ-PDU. These will be sent to the source address of the XREQ-PDU which in our example is MAC address “b” and therefore the X-PDUs will be received at the XLLDP agent for switch B ports 1 and 2 and switch C port 2. The XLLDP agent for switch B looks at the source ChassisID and PortID in the X-PDU and updates the remote databases which match the X-PDU ChassisID and PortID (figure 11). The X-PDU is also delivered to switch C on port 2. If switch C already has a remote database established by LLDP with the ChassisID and Port ID of the X-PDU, then the XLLDP agent updates that remote database.



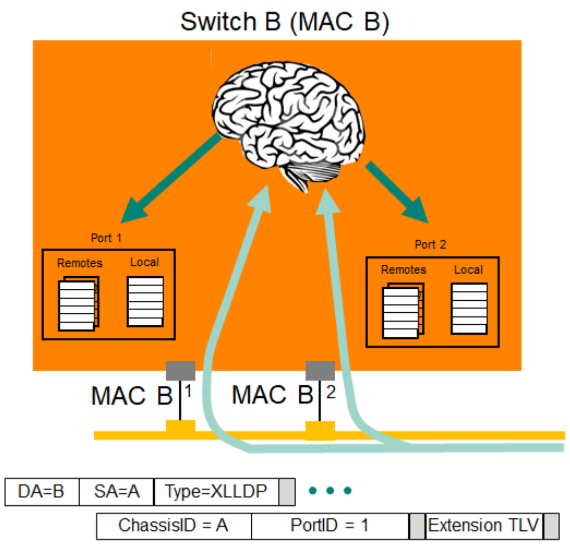


Figure 10: XLLDP Example Switch A

Figure 11: XLLDP Example Switch B

Summary

The LLDP protocol can be extended to allow advertisement of local databases of over 100 Kbytes while retaining backward compatibility with existing single frame implementation. The extension of LLDP allows applications to more demanding discovery applications such as discovery of router neighbors and addressing. The proposed extension method uses an Extension Link Layer Discovery Protocol (XLLDP) to exchange Extended LLDPDUs (X-PDUs). XLLDP is based on a unicast rather than multicast protocol which enables pacing the transmissions of the databases by the receivers and elimination of periodic transmissions of X-PDUs.

References

[1] IEEE Std 802.1AB-2016, Station and Media Access Control Connectivity Discovery, 2016, IEEE

[2] IEEE Std 802.1Qcy-2019, Bridges and Bridgged Networks – Amendment: Virtual Station Interface (VSI) Discovery and Configuration Protocol (VDP) Extension to Support Network Virtualization Overlays Over Layer 3 (NVO3), 2019, IEEE

[3] draft-xu-idr-neighbor-autodiscovery-12, X. Xu, K. Talaulikar, K. Bi, J. Tantsura, N. Triantafillis, DRAFT BGP Neighbor Discovery, Nov 26, 2019, IETF

[4] draft-ietf-lsvr-l3dl-03, R. Bush, R. Austein, K. Patel, DRAFT Layer 3 Discovery and Liveness, Nov 9, 2019, IETF

[5] draft-congdon-lsvr-lldp-tlvs-00, P. Congdon, P. Bottorff, DRAFT IETF Organizationally Specific TLVs for IEEE Std. 802.1AB (LLDP), Oct 22, 2019