Workgroup: Network Working Group
Internet-Draft: draft-ietf-lsvr-l3dl-12
Published: 11 February 2024
Intended Status: Experimental
Expires: 14 August 2024
Authors: R. Bush
Arrcus & Internet Initiative Japan
R. Housley
Vigil Security
R. Austein
Arrcus
K. Patel
Arrcus

Layer-3 Discovery and Liveness

Abstract

In Massive Data Centers, BGP-SPF and similar routing protocols are used to build topology and reachability databases. These protocols need to discover IP Layer-3 attributes of links, such as neighbor IP addressing, logical link IP encapsulation abilities, and link liveness. This Layer-3 Discovery and Liveness protocol collects these data, which may then be disseminated using BGP-SPF and similar protocols.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 August 2024.

Copyright Notice

Table of Contents

1. Introduction

The Massive Data Center (MDC) environment presents unusual problems of scale, e.g. O(10,000) forwarding devices, while its homogeneity presents opportunities for simple approaches. Approaches such as "Jupiter Rising: A study of non-blocking switching networks" [PAYWALLED] use a central controller to deal with scaling, while BGP-SPF [I-D.ietf-lsvr-bgp-spf] provides massive scale-out without centralization using a tried and tested scalable distributed control plane, offering a scalable routing solution in "Clos Networks" and similar environments. But BGP-SPF and similar higher level device-spanning protocols, e.g. [I-D.malhotra-bess-evpn-lsoe], need logical link state and addressing data from the network to build the routing topology. They also need prompt but prudent reaction to (logical) link failure.

Layer-3 Discovery and Liveness (L3DL) provides brutally simple mechanisms for devices to

- Discover each other's unique endpoint identification,
- Discover mutually supported layer-3 encapsulations, e.g. IP/MPLS,
- Discover Layer-3 IP and/or MPLS addressing of interfaces of the encapsulations,
- Present these data, using a very restricted profile of a BGP-LS [RFC7752] API, to BGP-SPF which computes the topology and builds routing and forwarding tables,
- Enable Layer-3 link liveness such as BFD,
- Provide Layer-2 keep-alive messages for session continuity,
- Provide for authenticity verification of protocol messages, and finally.
- Communicate the parameters needed to exchange inter-device Upper Layer Protocol Configuration for upper-layer protocols such as BGP.

In this document, the use case for L3DL is for point to point links in a datacenter Clos in order to exchange the data needed for BGP-SPF [I-D.ietf-lsvr-bgp-spf] bootstrap and continuity. Once layer-2 connectivity has been leveraged to get layer-3 addressability and forwarding capabilities, parameters for routing protocols such as BGP can be communicated, and normal layer-3 forwarding and routing can take over.

L3DL might be found to be more widely applicable to a range of routing and similar protocols which need layer-3 discovery and characterisation.

2. Terminology

Even though it concentrates on the inter-device layer, this document relies heavily on routing terminology. The following attempts to clarify the use of some possibly confusing terms:

ASN: Autonomous System Number [RFC4271], a BGP identifier for an originator of Layer-3 routes, particularly BGP announcements.

BGP-LS: A mechanism by which link-state and TE information can be collected from networks and shared with external components using the BGP routing protocol. See [RFC7752].

BGP-SPF
: A hybrid protocol using BGP transport but a Dijkstra Shortest Path First decision process. See [I-D.ietf-lsvr-bgp-spf].

Clos: A hierarchic subset of a crossbar switch topology commonly used in data centers.

Datagram: The L3DL content of a single Layer-2 frame, sans Ethernet framing. A full L3DL PDU may be packaged in multiple Datagrams.

Encapsulation: Address Family Indicator and Subsequent Address Family Indicator (AFI/SAFI). I.e. classes of layer-2.5 and 3 addresses such as IPv4, IPv6, MPLS, etc.

Frame: A Layer-2 Ethernet packet.

Link or Logical Link: A logical connection between two logical ports on two devices. E.g. two VLANs between the same two ports are two links.

LLEI: Logical Link Endpoint Identifier, the unique identifier of one end of a logical link, see Section 9.

MAC Address: 48-bit Layer-2 addresses are assumed since they are used by all widely deployed Layer-2 network technologies of interest, especially Ethernet. See [IEEE.802_2001].

MDC: Massive Data Center, commonly composed of thousands of Top of Rack Switches (TORs).

MTU: Maximum Transmission Unit, the size in octets of the largest packet that can be sent on a medium, see [RFC1122] 1.3.3.

PDU: Protocol Data Unit, an L3DL application layer message. A PDU's content may need to be broken into multiple Datagrams to make it through MTU or other restrictions.

RouterID: An 32-bit identifier unique in the current routing domain, see [RFC6286].

Session: An established, via OPEN PDUs, session between two L3DL capable link end-points,

SPF: Shortest Path First, an algorithm for finding the shortest paths between nodes in a graph; AKA Dijkstra's algorithm.

System Identifier: An eight octet ISO System Identifier a la [RFC1629] System ID

TOR:
Top Of Rack switch, aggregates the servers in a rack and connects to aggregation layers of the Clos tree, AKA the Clos spine.

ZTP: Zero Touch Provisioning gives devices initial addresses, credentials, etc. on boot/restart.

3. Background

L3DL is primarily designed for a Clos type datacenter scale and topology, but can accommodate richer topologies which contain potential cycles.

While L3DL is designed for the MDC, there are no inherent reasons it could not run on a WAN. The authentication and authorization needed to run safely on a WAN need to be considered, and the appropriate level of security options chosen.

Familiarity with the BGP4 Protocol [RFC4271] is assumed. Familiarity with BGP-SPF, [I-D.ietf-lsvr-bgp-spf], might be useful.

L3DL assumes a new IEEE assigned EtherType (TBD).

The number of addresses of one Encapsulation type on an interface link may be quite large given a TOR with tens of servers, each server having a few hundred micro-services, resulting in an inordinate number of addresses. And highly automated micro-service migration can cause serious address prefix disaggregation, resulting in interfaces with thousands of disaggregated prefixes.

Therefore the L3DL protocol is session oriented and uses incremental announcement and withdrawal with session restart, a la BGP ([RFC4271]).

4. Top Level Overview

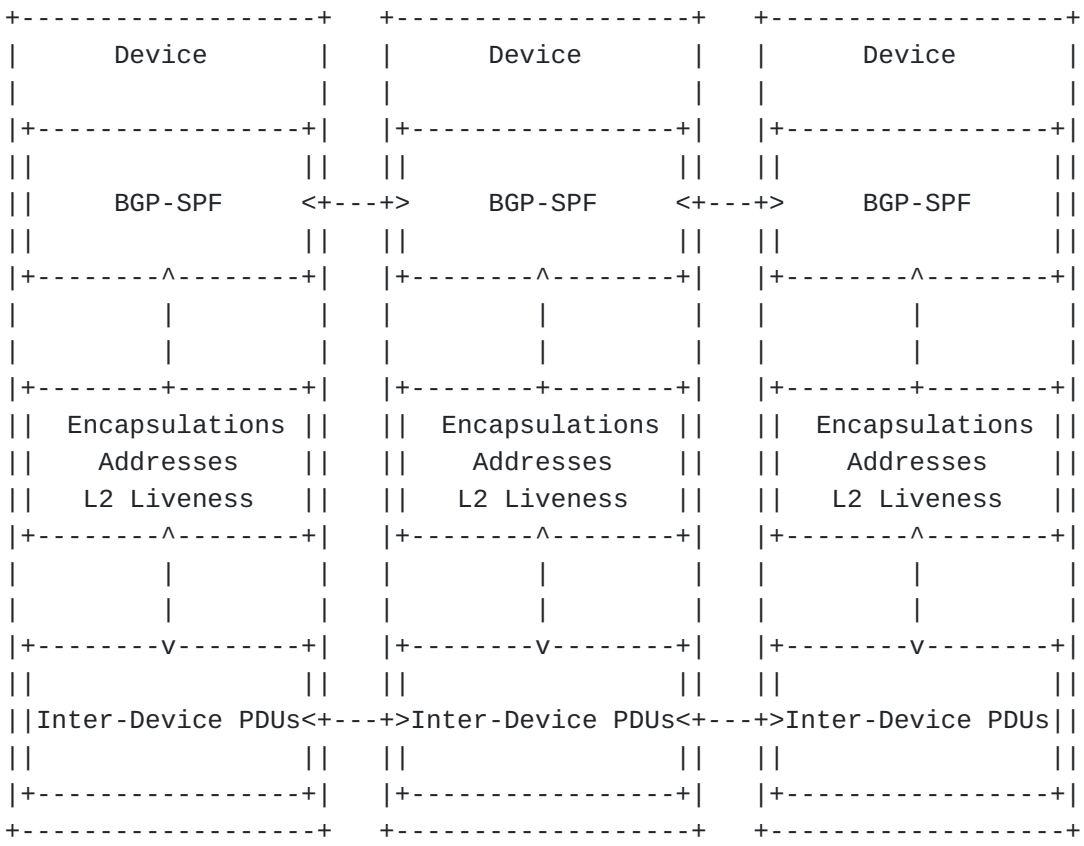
*Devices discover each other on logical links

*Logical Link Endpoint Identifiers (LLEIs) are exchanged

*Layer-2 Liveness checks may be started

*Encapsulation data are exchanged and IP-Level Liveness checks enabled

*A BGP-like upper layer protocol is assumed to use the identifiers and encapsulation data to discover and build a topology database



There are two protocols, the inter-device (left-right in the diagram) per-link layer-3 discovery and the API to the upper level BGP-like routing protocol (up-down in the above diagram):

- Inter-device PDUs are used to exchange device and logical link identities and layer-2.5 (MPLS) and 3 identifiers (not payloads), e.g. device IDs, port identities, VLAN IDs, Encapsulations, and IP addresses.

- A Link Layer to BGP API presents these data up the stack to a BGP protocol or an other device-spanning upper layer protocol, presenting them using the BGP-LS BGP-like data format.

The upper layer BGP family routing protocols cross all the devices, though they are not part of these L3DL protocols.

To simplify this document, Layer-2 framing is not shown. L3DL is about layer-3.

5. Inter-Link Protocol Overview

Two devices discover each other and their respective identities by sending multicast HELLO PDUs (Section 10). To assure discovery of new devices coming up on a multi-link topology, devices on such a

topology, and only on a multi-link topology, send periodic HELLOs forever, see Section 19.1.

Once a new device is recognized, both devices attempt to negotiate and establish a session by sending unicast OPEN PDUs (Section 11) to the source MAC addresses (plus VIDs if VLANs) of the received HELLOs. Once a session is established through the OPEN exchange, the Encapsulations (Section 13) configured on an end point may be announced and modified. Note that these are only the encapsulation and addresses configured on the announcing interface; though a device's loopback and overlay interface(s) may also be announced. When two devices on a link have compatible Encapsulations and addresses, i.e. the same AFI/SAFI and the same subnet, the link is announced via the BGP-LS API.

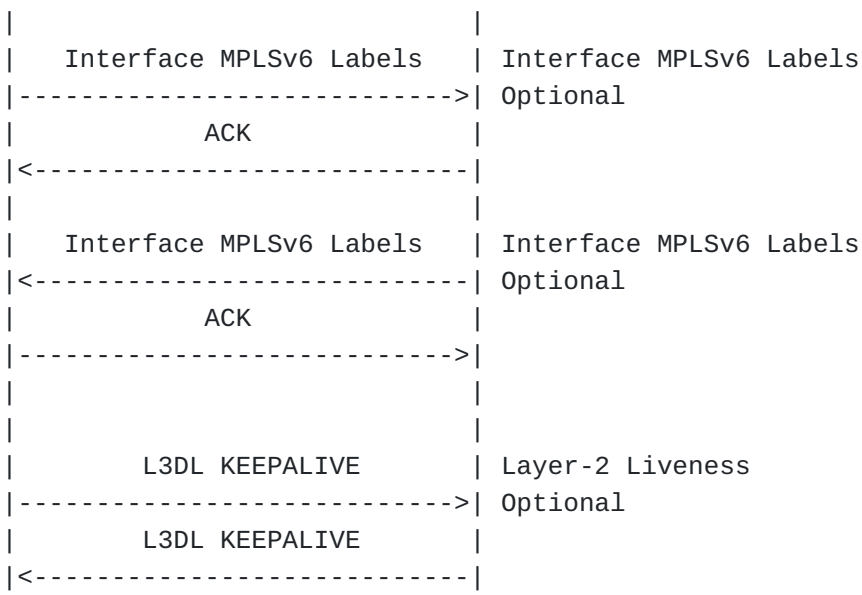
5.1. L3DL Ladder Diagram

The HELLO, Section 10, is a priming message sent on all configured logical links. It is a small L3DL PDU encapsulated in an Ethernet multicast frame with the simple goal of discovering the identities of logical link endpoint(s) reachable from a Logical Link Endpoint, Section 9.

The HELLO and OPEN, Section 11, PDUs, which are used to discover and exchange detailed Logical Link Endpoint Identifiers, LLEIs, and the ACK/ERROR PDU, are mandatory; other PDUs are optional; though at least one encapsulation SHOULD be agreed at some point.

The following is a ladder-style diagram of the L3DL protocol exchanges:


HELLO
Logical Link Peer discovery
HELLO
Mandatory
OPEN
MACs, IDs, etc.
ACK
OPEN
Mandatory
ACK
Interface IPv4 Addresses
Interface IPv4 Addresses
Optional
ACK
Interface IPv4 Addresses
ACK
Interface IPv6 Addresses
Interface IPv6 Addresses
Optional
ACK
Interface IPv6 Addresses
ACK
Interface MPLSv4 Labels
Interface MPLSv4 Labels
Optional
ACK
Interface MPLSv4 Labels
Interface MPLSv4 Labels
Optional
ACK


Interface MPLSv6 Labels
Interface MPLSv6 Labels
Optional
ACK
Interface MPLSv6 Labels
Interface MPLSv6 Labels
Optional
ACK
L3DL KEEPALIVE
Layer-2 Liveness
Optional
L3DL KEEPALIVE

6. Transport Layer

L3DL PDUs are carried by a simple transport layer which allows long PDUs to occupy many Ethernet frames. The L3DL content of a single Ethernet frame, exclusive of Ethernet framing data, is referred to as a Datagram.

The L3DL Transport Layer encapsulates each Datagram using a common transport header.

If a PDU does not fit in a single datagram, it is broken into multiple Datagrams and reassembled by the receiver a la [RFC0791] Section 2.3 Fragmentation.

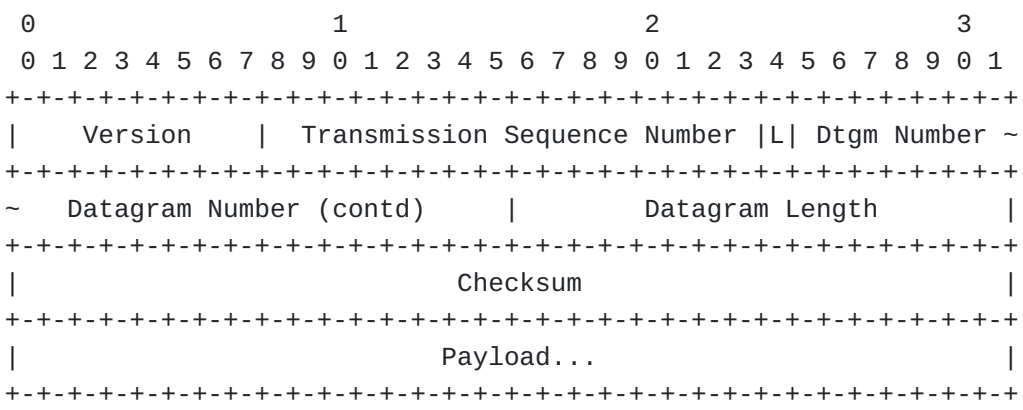
This is not classic 'fragmentation', but rather decomposition at the origin to allow PDU payloads larger than the frame allows. There are no intermediate devices capable of further fragmentation or reassembly.

A PDU might need a large number of frames to be sent. As fragments are not ACK paced (as PDUs are), to avoid overwhelming bursts, the sender should pace fragments of a large PDU.

L3DL is carrying a relatively small amount of data on relatively high bandwidth links, and at a time when the link is not active with other data as it does not yet have layer-3 connectivity. So congestion is not considered a sufficiently significant risk to warrant additional complexity.

Should a PDU need to be retransmitted, it MUST BE sent as the identical Datagram set as the original transmission. The Transmission Sequence Number informs the receiver that it is the same PDU.

The L3DL OPEN PDU contains an algorithm identifier, a key, and a L3DL certificate, which can be used to verify signatures on subsequent PDUs. This document describes two methods of key generation and signing for use by L3DL, Trust On First Use (TOFU) and a PKI-based mechanism to provide authentication as well as session integrity. See Section 21.



The fields of the L3DL Transport Header are as follows:

Version: Eight-bit Version number of the protocol, currently 0. Values other than 0 MUST BE treated as an error. The protocol version needs to be in one and only one place, so it is in the datagram as opposed to, for example, the PDU header.

Transmission Sequence Number: A 16-bit strictly increasing unsigned integer identifying this PDU, possibly across retransmissions, that wraps from 2^16-1 to 0. The initial value is arbitrary. See [RFC1982] on DNS Serial Number Arithmetic for too much detail on comparing and incrementing a wrapping sequence number.

L: A bit that set to one if this Datagram is the last Datagram of the PDU. For a PDU which fits in only one Datagram, it is set to one. Note that this is the inverse of the marking technique used by [RFC0791].

Datagram Number: A monotonically increasing 23-bit value which starts at zero for each PDU. This is used to reassemble frames into PDUs a la [RFC0791] Section 2.3. Note that this limits an L3DL PDU to 2^24 frames.

Datagram Length: Total number of octets in the Datagram including all payloads and fields. Note that this limits a datagram to 2^16 octets; though Ethernet framing is likely to impose a smaller limit.

Checksum: A 32 bit hash over the Datagram to detect bit flips, see Section 7.

If a Datagram fails checksum verification, the datagram is invalid and SHOULD be silently discarded. The sender will retransmit the PDU, and the receiver can assemble it.

Payload: The PDU being transported or a fragment thereof.

To avoid the need for a receiver to reassemble two PDUs at the same time, a sender MUST NOT send a subsequent PDU when a PDU is already in flight and not yet acknowledged; assuming it is an ACKed PDU Type.

7. The Checksum

There is a reason conservative folk use a checksum in UDP. And as many operators stretch to jumbo frames (over 1,500 octets) longer checksums are the prudent approach.

For the purpose of computing a checksum, the checksum field itself is assumed to be zero.

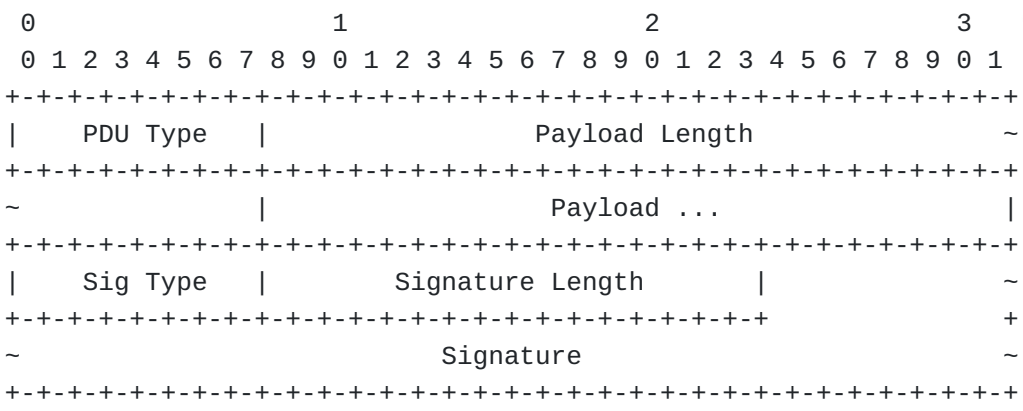
The following code describes a suggested algorithm. This specification avoids mandatory to implement, algorithm agility, etc. What matters is that the same algorithm is used consistently in any deployment.

Sum up 32-bit unsigned ints in a 64-bit long, then take the high-order section, shift it right filling on the left with zeros, rotate, add it in, repeat until the high order 32 bits are all zero.

```
<CODE BEGINS>
#include <stddef.h>
#include <stdint.h>

/* The F table from Skipjack, and it would work for the S-Box. */
static const uint8_t sbox[256] = {
0xa3,0xd7,0x09,0x83,0xf8,0x48,0xf6,0xf4,0xb3,0x21,0x15,0x78,
0x99,0xb1,0xaf,0xf9,0xe7,0x2d,0x4d,0x8a,0xce,0x4c,0xca,0x2e,
0x52,0x95,0xd9,0x1e,0x4e,0x38,0x44,0x28,0x0a,0xdf,0x02,0xa0,
0x17,0xf1,0x60,0x68,0x12,0xb7,0x7a,0xc3,0xe9,0xfa,0x3d,0x53,
0x96,0x84,0x6b,0xba,0xf2,0x63,0x9a,0x19,0x7c,0xae,0xe5,0xf5,
0xf7,0x16,0x6a,0xa2,0x39,0xb6,0x7b,0x0f,0xc1,0x93,0x81,0x1b,
0xee,0xb4,0x1a,0xea,0xd0,0x91,0x2f,0xb8,0x55,0xb9,0xda,0x85,
0x3f,0x41,0xbf,0xe0,0x5a,0x58,0x80,0x5f,0x66,0x0b,0xd8,0x90,
0x35,0xd5,0xc0,0xa7,0x33,0x06,0x65,0x69,0x45,0x00,0x94,0x56,
0x6d,0x98,0x9b,0x76,0x97,0xfc,0xb2,0xc2,0xb0,0xfe,0xdb,0x20,
0xe1,0xeb,0xd6,0xe4,0xdd,0x47,0x4a,0x1d,0x42,0xed,0x9e,0x6e,
0x49,0x3c,0xcd,0x43,0x27,0xd2,0x07,0xd4,0xde,0xc7,0x67,0x18,
0x89,0xcb,0x30,0x1f,0x8d,0xc6,0x8f,0xaa,0xc8,0x74,0xdc,0xc9,
0x5d,0x5c,0x31,0xa4,0x70,0x88,0x61,0x2c,0x9f,0x0d,0x2b,0x87,
0x50,0x82,0x54,0x64,0x26,0x7d,0x03,0x40,0x34,0x4b,0x1c,0x73,
0xd1,0xc4,0xfd,0x3b,0xcc,0xfb,0x7f,0xab,0xe6,0x3e,0x5b,0xa5,
0xad,0x04,0x23,0x9c,0x14,0x51,0x22,0xf0,0x29,0x79,0x71,0x7e,
0xff,0x8c,0x0e,0xe2,0x0c,0xef,0xbc,0x72,0x75,0x6f,0x37,0xa1,
0xec,0xd3,0x8e,0x62,0x8b,0x86,0x10,0xe8,0x08,0x77,0x11,0xbe,
0x92,0x4f,0x24,0xc5,0x32,0x36,0x9d,0xcf,0xf3,0xa6,0xbb,0xac,
0x5e,0x6c,0xa9,0x13,0x57,0x25,0xb5,0xe3,0xbd,0xa8,0x3a,0x01,
0x05,0x59,0x2a,0x46
};

/* non-normative example C code, constant time even */

uint32_t sbox_checksum_32(const uint8_t *b, const size_t n)
{
  uint32_t sum[4] = {0, 0, 0, 0};
  uint64_t result = 0;
  for (size_t i = 0; i < n; i++)
    sum[i & 3] += sbox[*b++];
  for (int i = 0; i < sizeof(sum)/sizeof(*sum); i++)
    result = (result << 8) + sum[i];
  result = (result >> 32) + (result & 0xFFFFFFFFU);
  result = (result >> 32) + (result & 0xFFFFFFFFU);
  return (uint32_t) result;
}

<CODE ENDS>
```

8. TLV PDUs

The basic L3DL application layer PDU is a typical TLV (Type Length Value) PDU. It includes a signature to provide optional integrity and authentication. It may be broken into multiple Datagrams, see Section 6.



The fields of the basic L3DL header are as follows:

PDU Type: An integer differentiating PDU payload types. See Section 24.1.

Payload Length: Total number of octets in the Payload field.

Payload: The application layer content of the L3DL PDU.

Sig Type: The type of the Signature, see Section 24.3. Type 0, a null signature, is defined in this document.

Sig Type 0 indicates a null Signature. For a trivial PDU such as KEEPALIVE, the underlying Datagram checksum may be sufficient for integrity, though it lacks authenticity.

Other Sig Types may be defined in other documents, cf. [I-D.ymbk-lsvr-l3dl-signing].

Signature Length: The length of the Signature, possibly including padding, in octets. If Sig Type is 0, Signature Length MUST BE 0.

Signature: The result of running the signature algorithm specified in Sig Type over all octets of the PDU except for the Signature itself.

9. Logical Link Endpoint Identifier

L3DL discovers neighbors on logical links and establishes sessions between the two ends of all consenting discovered logical links. A

logical link is described by a pair of Logical Link Endpoint Identifiers, LLEIs.

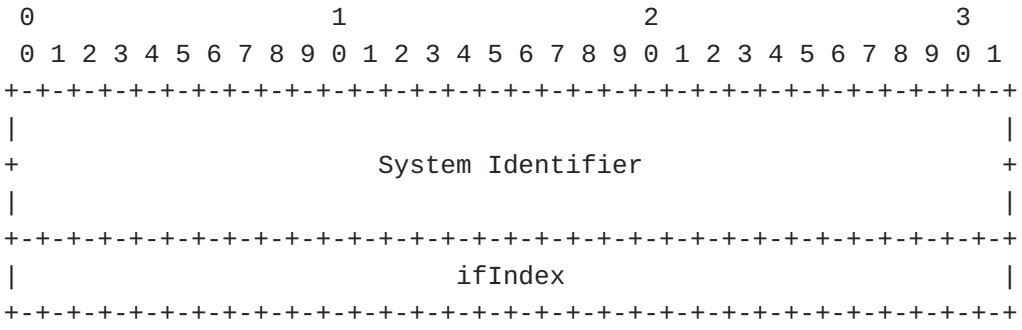
An LLEI is a variable length descriptor which could be an ASN, a classic RouterID, a catenation of the two, an eight octet ISO System Identifier [RFC1629], or any other identifier unique to a single logical link endpoint in the topology.

An L3DL deployment will choose and define an LLEI which suits its needs, simple or complex. Examples of two extremes follow:

A simplistic view of a link between two devices is two ports, identified by unique MAC addresses, carrying a layer-3 protocol conversation. In this case, the MAC addresses might suffice for the LLEIs.

Unfortunately, things can get more complex. Multiple VLANs can run between those two MAC addresses. In practice, many real devices use the same MAC address on multiple ports and/or sub-interfaces.

Therefore, in the general circumstance, a fully described LLEI might be as follows:



System Identifier, a la [RFC1629], is an eight octet identifier unique in the entire operational space. Routers and switches usually have internal MAC Addresses which can be padded with high order zeros and used if no System ID exists on the device. If no unique identifier is burned into a device, the local L3DL configuration SHOULD create and assign a unique one, likely by configuration.

ifIndex is the SNMP identifier of the (sub-)interface, see [RFC1213]. This uniquely identifies the port.

For a layer-3 tagged sub-interface or a VLAN/SVI interface, IfIndex is that of the logical sub-interface, so no further disambiguation is needed.

L3DL PDUs learned over VLAN-ports may be interpreted by upper layer-3 routing protocols as being learned on the corresponding layer-3 SVI interface for the VLAN.

LLEIs are big-endian.

10. HELLO

The HELLO PDU is unique in that it is encapsulated in a multicast Ethernet frame. It solicits response(s) from other LLEI(s) on the link. See Section 19.1 for why multicast is used. The destination multicast MAC Addressees to be used MUST be one of the following, See Clause 9.2.2 of [IEEE802-2014]:

01-80-C2-00-00-0E: Nearest Bridge = Propagation constrained to a single physical link; stopped by all types of bridges (including MPRs (media converters)). This SHOULD be used when the link is known to be a simple point to point link.

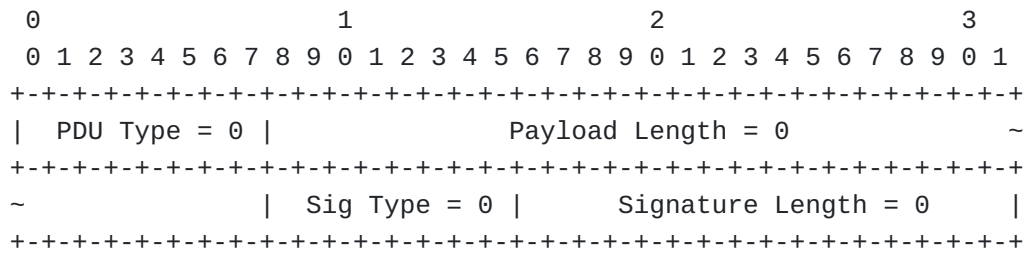
To Be Assigned: When a switch receives a frame with a multicast destination MAC it does not recognize, it forwards to all ports. This destination MAC SHOULD be sent when the interface is known to be connected to a switch. See Section 25. This SHOULD be used when the link may be a multi-point link.

All other L3DL PDUs are encapsulated in unicast frames, as the peer's destination MAC address is known after the HELLO exchange.

When an interface is turned up on a device, it SHOULD issue a HELLO if it is to participate in L3DL sessions.

If a constrained Nearest Bridge destination address has been configured for a point-to-point interface, see above, then the HELLO SHOULD NOT be repeated once a session has been created by an exchange of OPENs.

If the configured destination address is one that is propagated by switches, the HELLO SHOULD be repeated at a configured interval, with a default of 60 seconds. This allows discovery by new devices which come up on the layer-2 mesh. In this multi-link scenario, the operator should be aware of the trade-off between timer tuning and network noise and adjust the inter-HELLO timer accordingly.



If more than one device responds, one adjacency is formed for each unique source LLEI response. L3DL treats each adjacency as a separate logical link.

When a HELLO is received from a source MAC address (plus VID if VLAN) with which there is no established L3DL session, the receiver SHOULD respond by sending an OPEN PDU to the source MAC address (plus VID). The two devices establish an L3DL session by exchanging OPEN PDUs.

To ameliorate possible load spikes during bootstrap or event recovery, there SHOULD be a jittered delay between receipt of a HELLO and issue of the OPEN. The default delay range SHOULD be zero to five seconds, and MUST be configurable.

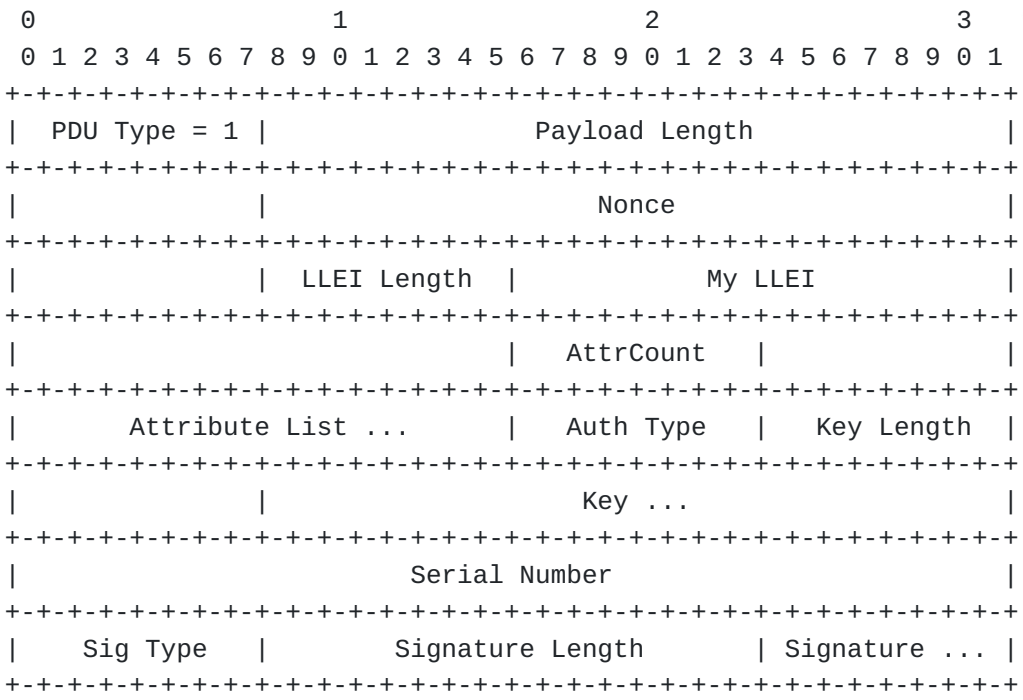
If a HELLO is received from a MAC address with which there is an established session, the HELLO should be dropped.

The Payload Length is zero as there is no payload.

HELLO PDUs can not be signed as keying material has yet to be exchanged. Hence the signature MUST always be the null type.

11. OPEN

Each device has learned the other's MAC Address from the HELLO exchange, see Section 10. Therefore the OPEN and all subsequent PDUs MUST BE unicast, as opposed to the HELLO's multicast frame.



The Payload Length is the number of octets in all fields of the PDU from the Nonce through the Serial Number, not including the three final signature fields.

The Nonce enables detection of a duplicate OPEN PDU. It SHOULD be either a random number or a high resolution timestamp. It is needed to prevent session closure due to a repeated OPEN caused by a race or a dropped or delayed ACK.

My LLEI is the sender's LLEI, see Section 9.

AttrCount is the number of attributes in the Attribute List. Attributes are single octets the semantics of which are operator-defined.

A node may have zero or more operator-defined attributes, e.g.: spine, leaf, backbone, route reflector, arabica, ...

Attribute syntax and semantics are local to an operator or datacenter; hence there is no global registry. Nodes exchange their attributes only in the OPEN PDU.

Auth Type is the Signature algorithm suite, see Section 8.

Key Length is a 16-bit field denoting the length in octets of the Key itself, not including the Auth Type or the Key Length. If the Auth Type is zero, then the Key Length MUST also be zero, and there MUST BE no Key data.

The Key is specific to the operational environment. A failure to authenticate is a failure to start the L3DL session, an ERROR PDU MUST BE sent (Error Code 3), and HELLOs MUST be restarted.

Although delay and jitter in responding with an OPEN were specified above, beware of load created by long strings of authentication failures and retries. A configurable failure count limit (default 8) SHOULD result in giving up on the connection attempt.

The Serial Number is a monotonically increasing 32-bit value representing the sender's state at the time of sending the last PDU. It may be an integer, a timestamp, etc. If incrementing the Serial Number would cause it to be zero, it should be incremented again.

On session restart (new OPEN), a receiver MAY send the last received Serial Number to tell the sender to only send data with a Serial Number greater (in the [RFC1982] sense), or send a Serial Number of zero to request all data.

The Serial Number supports session resumption in anticipation of peers having a very large amount of state they would prefer not to re-exchange because of some glitch. The Serial Number is not expected to wrap for a considerable time, e.g. days or weeks. But to address the rare case it does, [RFC1982] on DNS Serial Number

Arithmetic should be used as it is in the Transmission Sequence Number.

This allows a sender of an OPEN to tell the receiver that the sender would like to resume a session and that the receiver only needs to send data starting with the PDU with the lowest Serial Number greater (in the [RFC1982] sense) than the one sent in the OPEN. If the sender is not trying to resume a dropped session, the Serial Number MUST be zero.

If the receiver of an OPEN PDU with a non-zero Serial Number can not resume from the requested point, it should return an ACK with an Error Code of 2, Session could not be continued. The sender of the failing OPEN PDU SHOULD then send an OPEN PDU with a Serial Number of zero.

The Signature fields are described in Section 8 and in an asymmetric key environment serve as a proof of possession of the signing auth data by the sender.

Once two logical link endpoints know each other, and have ACKed each other's OPEN PDUs, Layer-2 KEEPALIVEs (see Section 16) MAY be started to ensure Layer-2 liveness and keep the session semantics alive. The timing and acceptable drop of KEEPALIVE PDUs are discussed in Section 16.

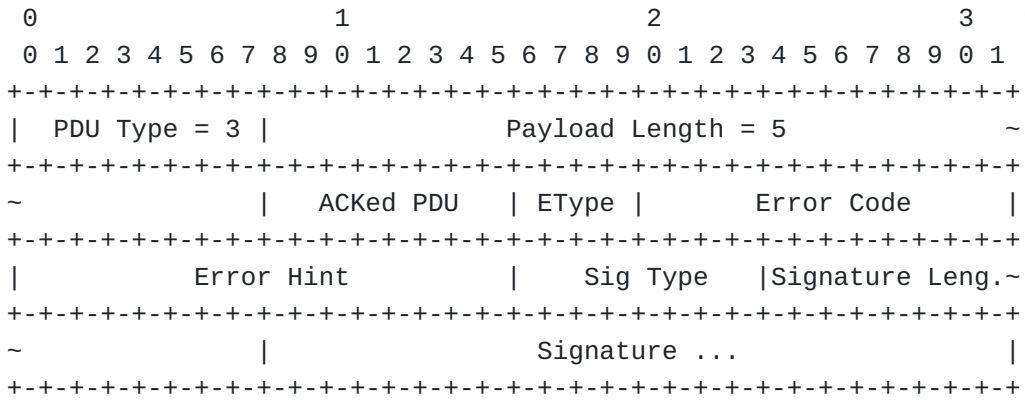
If a sender of OPEN does not receive an ACK of the OPEN PDU, then they MUST resend the same OPEN PDU, with the same Nonce. Resending an unacknowledged OPEN PDU, like other ACKed PDUs, SHOULD use exponential back-off, see [RFC1122].

If a properly authenticated OPEN arrives at L3DL speaker A with a new Nonce from an LLEI, speaker B, with which A believes it already has an L3DL session (OPENs have already been exchanged), and the Serial Number in the OPEN PDU is non-zero, speaker A SHOULD establish a new sending session by sending an OPEN with the Serial Number being the same as that of A's last sent and ACKed PDU. A MUST resume sending encapsulations etc. subsequent to the requested Sequence Number. And B MUST retain all previously discovered encapsulation and other data received from A.

If a properly authenticated OPEN arrives with a new Nonce from an LLEI with which the receiving logical link endpoint believes it already has an L3DL session (OPENs have already been exchanged), and the Serial Number in the OPEN is zero, then the receiver MUST assume that the sending LLEI or entire device has been reset. All Previously discovered encapsulation data MUST NOT be kept and MUST BE withdrawn via the BGP-LS API and the recipient MUST respond with a new OPEN.

12. ACK

The ACK PDU acknowledges receipt of a PDU and reports any error condition which might have been raised.



The ACK acknowledges receipt of an OPEN, Encapsulation, VENDOR PDU, etc.

The ACKed PDU is the PDU Type of the PDU being acknowledged, e.g., OPEN, one of the Encapsulations, etc.

If there was an error processing the received PDU, then the EType is non-zero. If the EType is zero, Error Code and Error Hint MUST also be zero.

A non-zero EType is the receiver's way of telling the PDU's sender that the receiver had problems processing the PDU. The Error Code and Error Hint will tell the sender more detail about the error.

The decimal value of EType gives a strong hint how the receiver sending the ACK believes things should proceed:

0 - No Error, Error Code and Error Hint MUST be zero

1 - Warning, something not too serious happened, continue

2 - Session should not be continued, try to restart

3 - Restart is hopeless, call the operator

4-15 - Reserved

The Error Codes, noting protocol failures, are listed in Section 24.5. Someone stuck in the 1990s might think the catenation of EType and Error Code as an echo of 0x1zzz, 0x2zzz, etc. They might be right; or not.

The Error Hint, an arbitrary 16 bits, is any additional data the sender of the error PDU thinks will help the recipient or the debugger with the particular error.

The Signature fields are described in Section 8.

12.1. Retransmission

If a PDU sender expects an ACK, e.g. for an OPEN, an Encapsulation, a VENDOR PDU, etc., and does not receive the ACK for a configurable time (default one second), and the interface is live at layer-2, the sender resends the PDU using exponential back-off, see [RFC1122]. This cycle MAY be repeated a configurable number of times (default three) before it is considered a failure. The session MAY BE considered closed in this case of this ACK failure.

If the link is broken at layer-2, retransmission MAY BE retried when the link is restored.

13. The Encapsulations

Once the devices know each other's LLEIs, know each other's upper layer (L2.5 and L3) identities, have means to ensure link state, etc., the L3DL session is considered established, and the devices SHOULD exchange L3 interface encapsulations, L3 addresses, and L2.5 labels.

The Encapsulation types the peers exchange may be IPv4 (Section 13.3), IPv6 (Section 13.4), MPLS IPv4 (Section 13.6), MPLS IPv6 (Section 13.7), and/or possibly others not defined here.

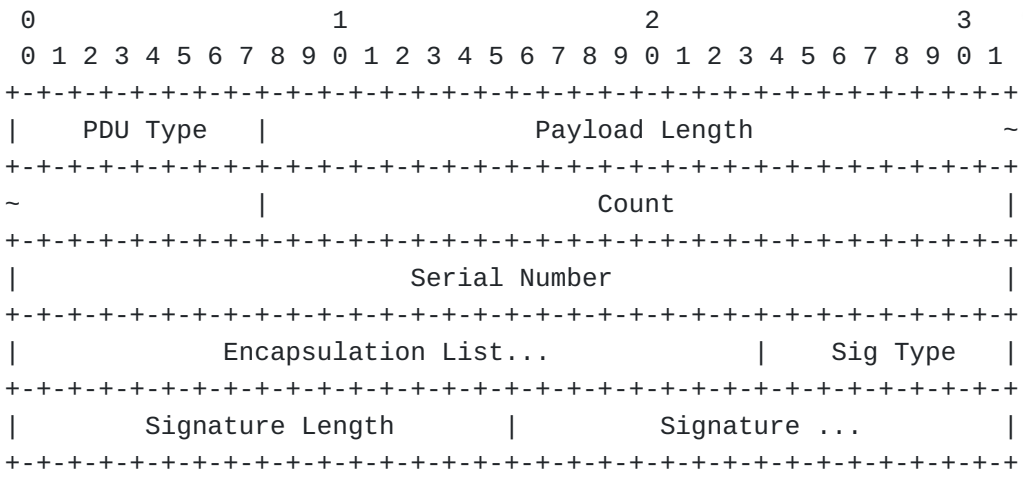
The sender of an Encapsulation PDU MUST NOT assume that the peer is capable of the same Encapsulation Type. An ACK (Section 12) merely acknowledges receipt. Only if both peers have sent the same Encapsulation Type is it safe for Layer-3 protocols to assume that they are compatible for that type.

A receiver of an encapsulation might recognize an addressing conflict, such as both ends of the link trying to use the same address. In this case, the receiver SHOULD respond with an error (Error Code 2) ACK. As there may be other usable addresses or encapsulations, this error might log and continue, letting an upper layer topology builder deal with what works.

Further, to consider a logical link of a type to formally be established so that it may be pushed up to upper layer protocols, the addressing for the type must be compatible, e.g. on the same IP subnet.

13.1. The Encapsulation PDU Skeleton

The header for all encapsulation PDUs is as follows:



An Encapsulation PDU describes zero or more addresses of the encapsulation type.

The 24-bit Count is the number of Encapsulations in the Encapsulation list.

The Serial Number is a monotonically increasing 32-bit value representing the sender's state in time. It may be an integer, a timestamp, etc. On session restart (new OPEN), a receiver MAY send the last received Session Number to tell the sender to only send newer data.

If a sender has multiple links on the same interface, separate state: data, ACKs, etc. must be kept for each peer session.

Over time, multiple Encapsulation PDUs may be sent for an interface as configuration changes.

If the length of an Encapsulation PDU exceeds the Datagram size limit on media, the PDU is broken into multiple Datagrams. See Section 8.

The Signature fields are described in Section 8.

The Receiver MUST acknowledge the Encapsulation PDU with a Type=3, ACK PDU (Section 12) with the Encapsulation Type being that of the encapsulation being announced, see Section 12.

If the Sender does not receive an ACK in a configurable interval (default one second), and the interface is live at layer-2, they SHOULD retransmit. After a user configurable number of failures

(default three), the L3DL session should be considered dead and the OPEN process SHOULD be restarted.

If the link is broken at layer-2, retransmission MAY BE retried if data have not changed in the interim.

13.2. Encapsulaion Flags

The Encapsulation Flags are a sequence of bit fields as follows:

Each encapsulation in an Encapsulation PDU of Type T may announce new and/or withdraw old encapsulations of Type T. It indicates this with the Ann/With Encapsulation Flag, Announce == 1, Withdraw == 0.

Each Encapsulation interface address in an Encapsulation PDU is either a new encapsulation be announced (Ann/With == 1) (yes, a la BGP) or requests one be withdrawn (Ann/With == 0). Adding an encapsulation which already exists SHOULD raise an Announce/Withdraw Error (see Section 24.5); the EType SHOULD be 2, suggesting a session restart (see Section 12 so all encapsulations will be resent.

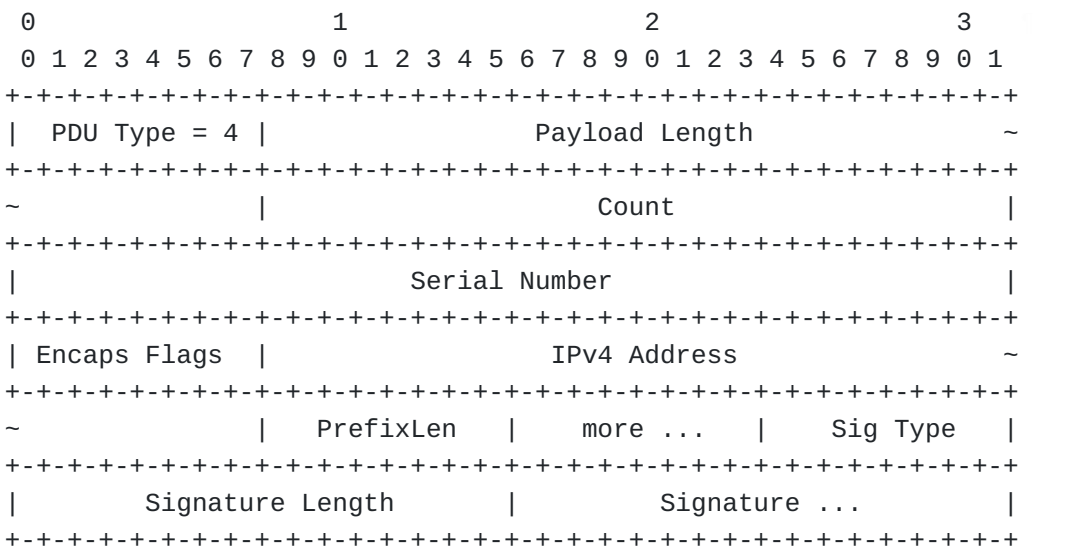
If an LLEI has multiple addresses for an encapsulation type, one and only one address MAY be marked as primary (Primary Flag == 1) for that Encapsulation Type.

An Encapsulation interface address in an Encapsulation PDU MAY be marked as a loopback, in which case the Loopback bit is set. Loopback addresses are generally not seen directly on an external interface. One or more loopback addresses MAY be exposed by configuration on one or more L3DL speaking external interfaces, e.g. for iBGP peering. They SHOULD be marked as such, Loopback Flag == 1.

Each Encapsulation interface address in an Encapsulation PDU is that of the direct 'underlay interface (Under/Over == 1), or an 'overlay' address (Under/Over == 0), likely that of a VM or container guest bridged or configured on to the interface already having an underlay address.

13.3. IPv4 Encapsulation

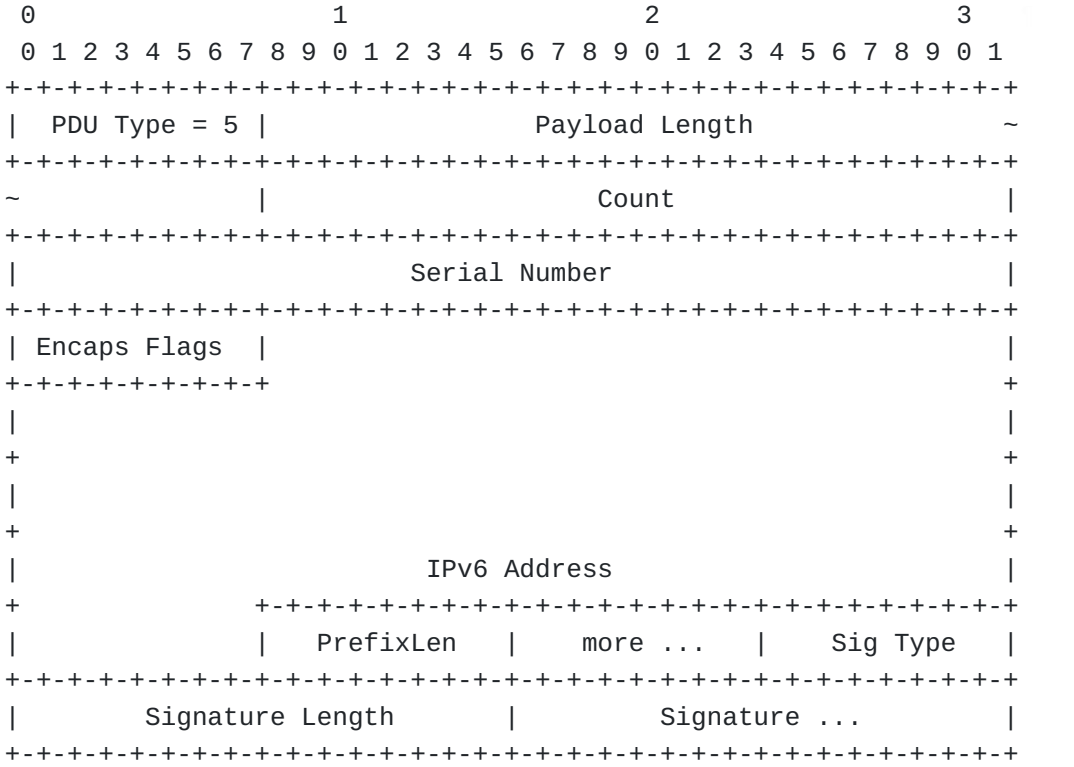
The IPv4 Encapsulation describes a device's ability to exchange IPv4 packets on one or more subnets. It does so by stating the interface's addresses and the corresponding prefix lengths.



The 24-bit Count is the sum of the number of IPv4 Encapsulations being announced and/or withdrawn.

13.4. IPv6 Encapsulation

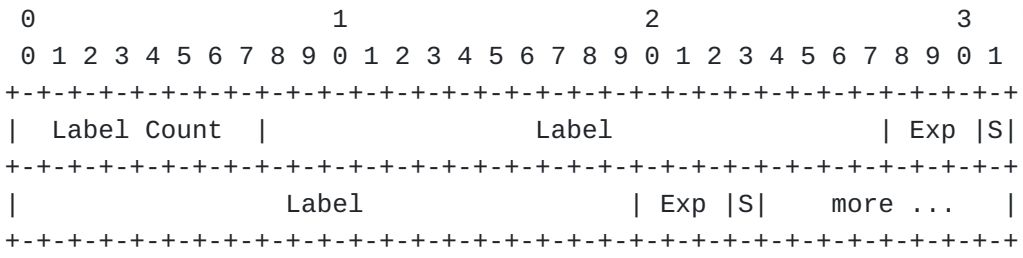
The IPv6 Encapsulation describes a logical link's ability to exchange IPv6 packets on one or more subnets. It does so by stating the interface's addresses and the corresponding prefix lengths.



The 24-bit Count is the sum of the number of IPv6 Encapsulations being announced and/or withdrawn.

13.5. MPLS Label List

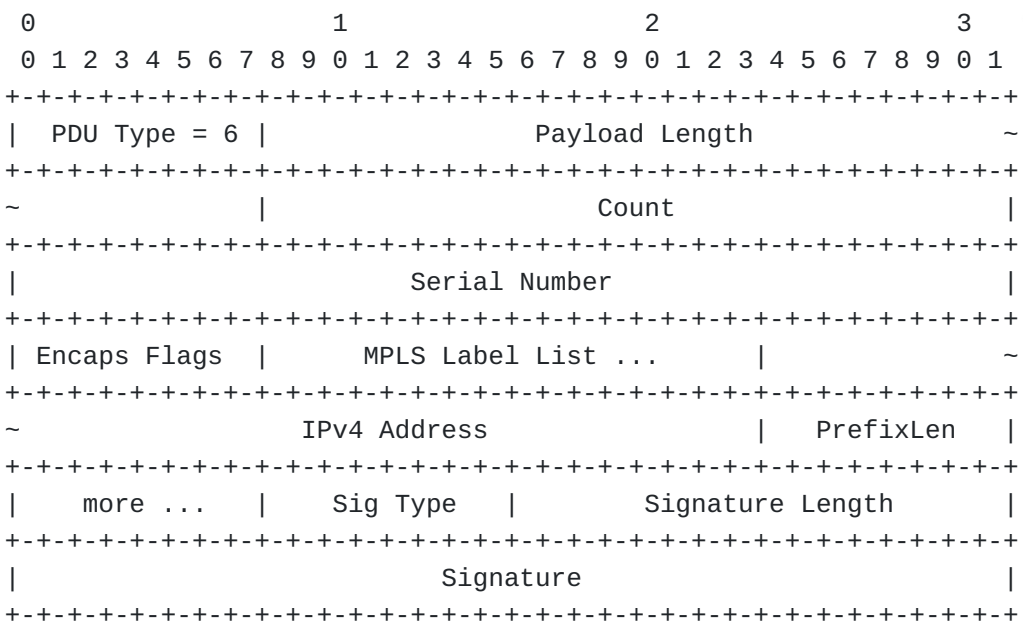
As an MPLS enabled interface may have a label stack, see [RFC3032], a variable length list of labels is needed. These are the labels the sender will accept for the prefix to which the list is attached.



A Label Count of zero is an implicit withdraw of all labels for that prefix on that interface.

13.6. MPLS IPv4 Encapsulation

The MPLS IPv4 Encapsulation describes a logical link's ability to exchange labeled IPv4 packets on one or more subnets. It does so by stating the interface's addresses the corresponding prefix lengths, and the corresponding labels which will be accepted for each address.

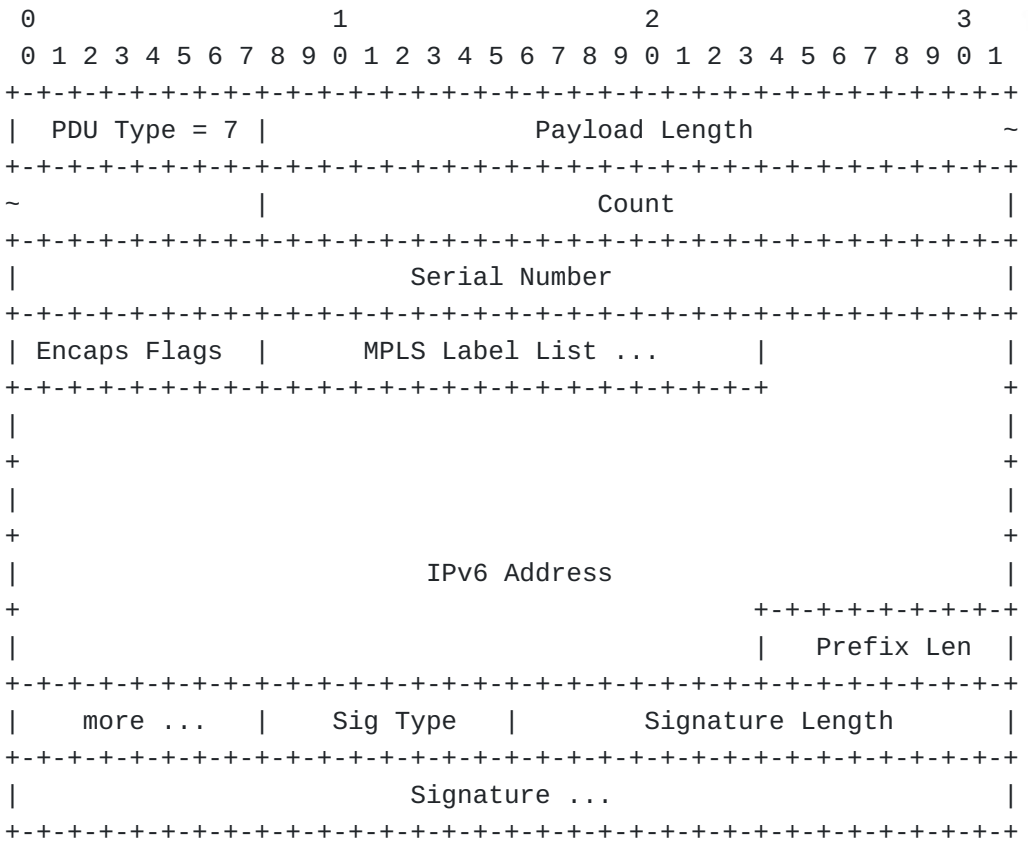


The 24-bit Count is the sum of the number of MPLSv4 Encapsulation being announced and/or withdrawn.

13.7. MPLS IPv6 Encapsulation

The MPLS IPv6 Encapsulation describes a logical link's ability to exchange labeled IPv6 packets on one or more subnets. It does so by

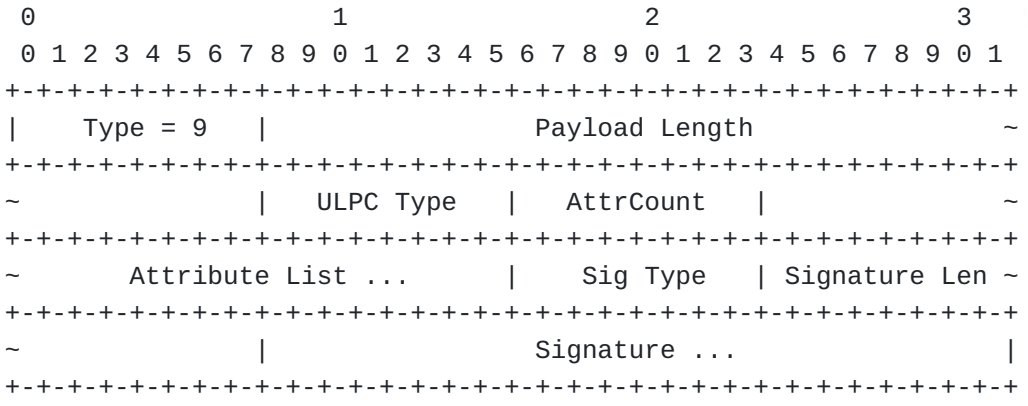
stating the interface's addresses, the corresponding prefix lengths, and the corresponding labels which will be accepted for each address.



The 24-bit Count is the sum of the number of MPLSv6 Encapsulations being announced and/or withdrawn.

14. Upper-Layer Protocol Configuration PDU

To communicate parameters required to configure peering and operation of Upper-Layer Protocols at IP layer-3 and above, e.g., BGP sessions on a link, a neutral sub-TLV based Upper-Layer Protocol PDU is defined as follows:



The Type and Payload Length are defined in Section 8.

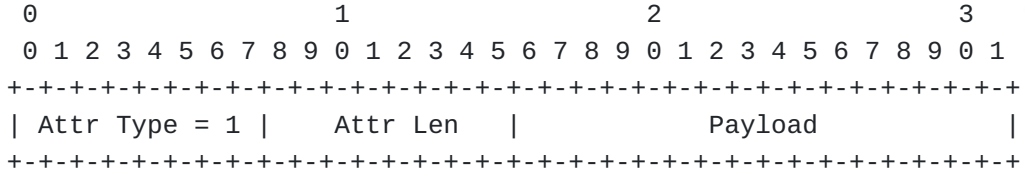
ULPC Type: A one octet integer denoting the type of the upper-layer protocol

0 : Reserved
1 : BGP
2-255 : Reserved

The one octet AttrCount is the number of attribute sub-TLVs in the Attribute List.

The Attribute List is a, possibly null, set of sub-TLVs describing the configuration attributes of the specific upper-layer protocol.

An Attribute consists of a one octet Attribute Type, a one octet Attribute Length of the number of octets in the Attribute, and a Payload of arbitrary length up to 253 octets.



14.1. ULPC BGP Attribute sub-TLVs

The parameters needed for BGP peering on a link are exchanged in sub-TLVs within an Upper-Layer Protocol PDU. The following describe the various sub-TLVs for BGP.

The goal is to provide the minimal set of configuration parameters needed by BGP OPEN to successfully start a BGP peering. The goal is specifically not to replace or conflict with data exchanged during BGP OPEN. Multiple sources of truth are a recipe for complexity and hence pain.

If there are multiple BGP sessions on a link, e.g., IPv4 and IPv6, then separate BGP ULPC PDUs should be sent, one for each address family.

A peer receiving BGP ULPC PDUs has only one active BGP ULPC PDU for an particular address family on a specific link at any point in time; receipt of a new BGP ULPC PDU for a particular address family replaces the data any previous one; but does not actually affect the session.

If there are one or more open BGP sessions, receipt of a new BGP ULPC PDU SHOULD NOT affect these sessions. The received data are stored for a future session restart.

As a link may have multiple encapsulations and multiple addresses for an IP encapsulation, which address of which encapsulation is to be used for the BGP session MUST be specified.

For each BGP peering on a link here MUST be one agreed encapsulation, and the addresses used MUST be in the corresponding L3DL IPv4/IPv6 Announcement PDUs. If the choice is ambiguous, an Attribute may be used to signal preferences.

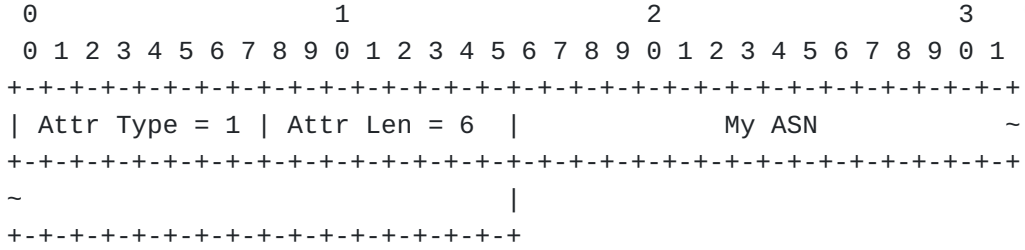
If a peering address has been announced as a loopback, i.e. MUST BE flagged as such in the L3DL Encapsulation PDU (see Section 13.2), a two or three hop BGP session will be established. Otherwise a direct one hop session is used. The BGP session to a loopback will forward to the peer's address which was marked as Primary in the L3DL Encapsulation Flags, iff it is in a subnet which is shared with both BGP speakers. If the primary is not in a common subnet, then the BGP speaker MAY pick a forwarding next hop that is in a subnet they share. If there are multiple choices, the BGP speaker SHOULD have signaled which subnet to choose in an Upper-Layer Protocol Configuration PDU Attribute.

Attributes MUST be unique in the Attribute List. I.e. a particular Attr Type MUST NOT occur more than once in the Attribute List. If a ULPC PDU is received with more than one occurrence of a particular Attr Type, an Error ACK MUST be returned.

As there are separate PDU Attr Types for IPv4 and IPv6 peering addresses, separate sessions for the two AFIs MAY be created for the same ASN in one ULPC PDU.

14.1.1. BGP ASN

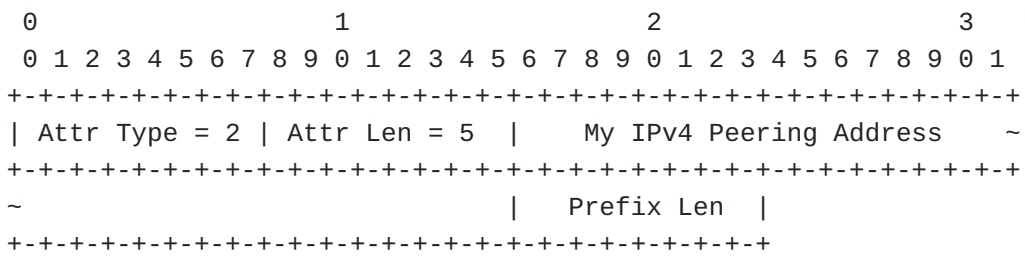
The four octet Autonomous System number MUST be specified. If the AS Number is less than 32 bits, it is padded with high order zeros.



14.1.2. BGP IPv4 Address

The four octet BGP IPv4 Address sub-TLV announces the sender's IPv4 BGP peering source address to be used by the receiver. At least one of IPv4 or IPv6 BGP source addresses MUST be announced.

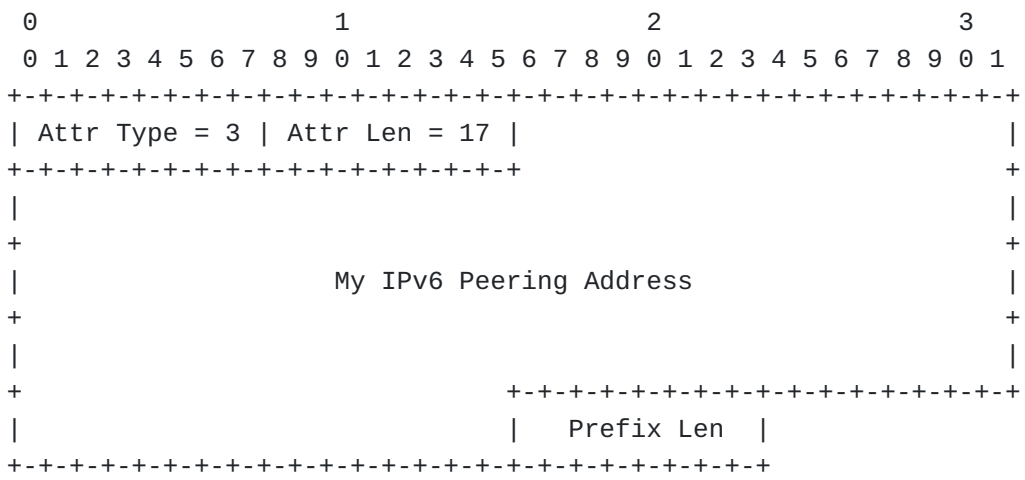
As usual, the BGP OPEN capability negotiation will determine the AFI/SAFIs to be transported over the peering, see [RFC4760] .



14.1.3. BGP IPv6 Address

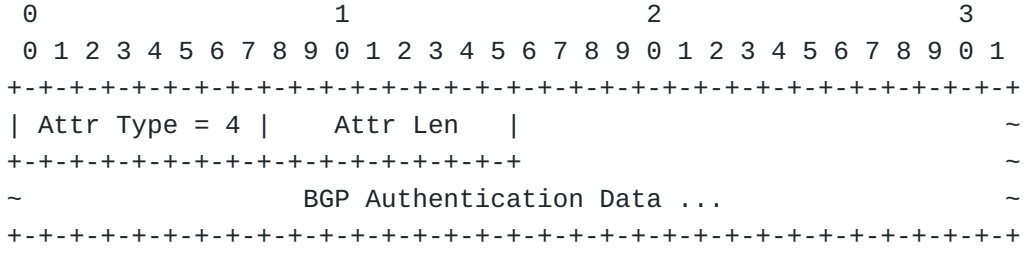
The BGP IPv6 Address sub-TLV announces the sender's 16 octet IPv6 BGP peering source address and one octet Prefix Length to be used by the receiver. At least one of IPv4 or IPv6 BGP source addresses MUST be announced.

As usual, the BGP OPEN capability negotiation will determine the AFI/SAFIs to be transported over the peering, see [RFC4760].



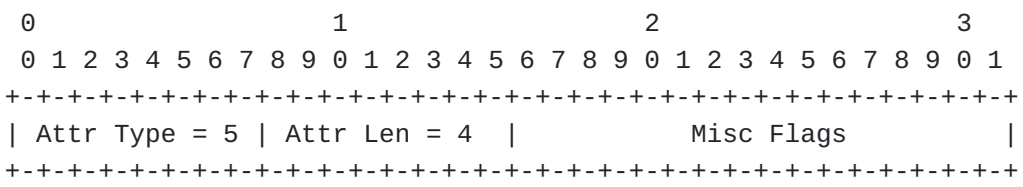
14.1.4. BGP Authentication sub-TLV

The BGP Authentication sub-TLV provides any authentication data needed to OPEN the BGP session. Depending on operator configuration of the environment, it might be a simple MD5 key (see [RFC2385]), the name of a key chain in a KARP database (see [RFC7210]), or one of multiple Authentication sub-TLVs to support [RFC4808].



14.1.5. BGP Miscellaneous Flags

The BGP session OPEN has extensive, and a bit complex, capability negotiation facilities. In case one or more extra attributes might be needed, the two octet BGP Miscellaneous Flags sub-TLV may be used. No flags are currently defined.



Misc Flags:

Bit 0: GTSM

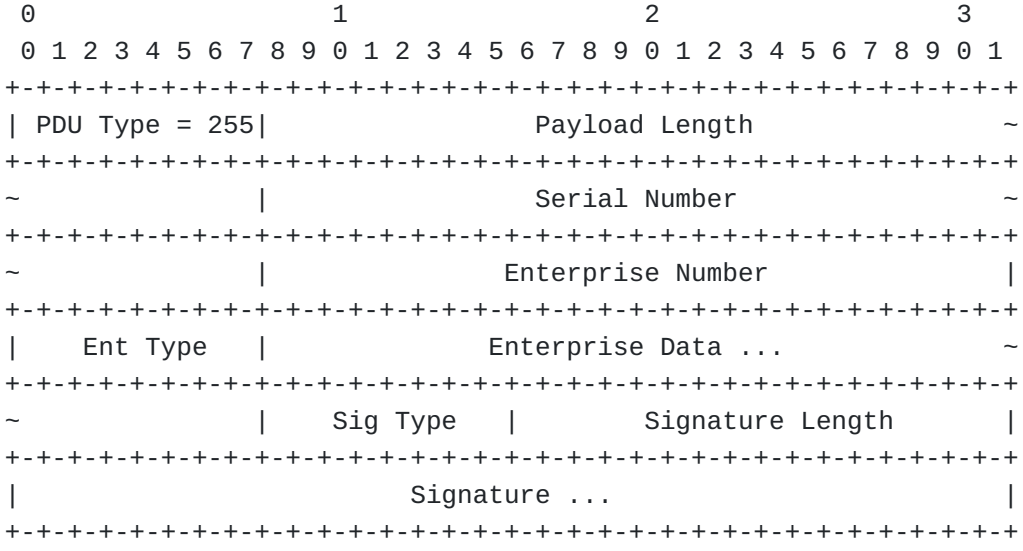
Bit 1: BFD

Bit 2-15: Must be zero

The GTSM flag, when 1, indicates that the sender wishes to enable the [RFC5082] Generalized TTL Security Mechanism for the session.

The BFD flag, when 1, indicates that the sender wishes to enable the [RFC5880] Bidirectional Forwarding Detection for the session.

15. VENDOR - Vendor Extensions

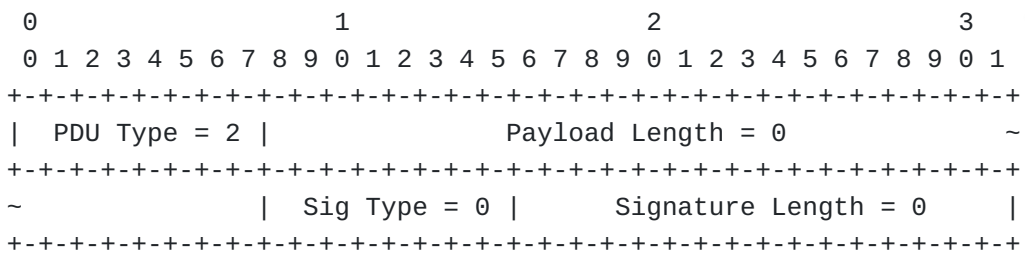


Vendors or enterprises may define TLVs beyond the scope of L3DL standards. This is done using a Private Enterprise Number [IANA-PEN] followed by Enterprise Data in a format defined for that Enterprise Number and Ent Type.

Ent Type allows a VENDOR PDU to be sub-typed in the event that the vendor/enterprise needs multiple PDU types.

As with Encapsulation PDUs, a receiver of a VENDOR PDU MUST respond with an ACK or an ERROR PDU. Similarly, a VENDOR PDU MUST only be sent over an open session.

16. KEEPALIVE - Layer-2 Liveness



L3DL devices SHOULD beacon frequent Layer-2 KEEPALIVE PDUs to ensure session continuity. The inter-KEEPALIVE interval is configurable, with a default of ten seconds. A receiver may choose to ignore KEEPALIVE PDUs.

An operational deployment MUST BE configured whether to use KEEPALIVEs or not, either globally, or as finely as to per-link granularity. Disagreement MAY result in repeated session failure and reestablishment.

KEEPALIVEs SHOULD be beaconed at a configured frequency. One per second is the default. Layer-3 liveness, such as BFD, may be more (or less) aggressive.

When a sender transmits a PDU which is not a KEEPALIVE, the sender SHOULD reset the KEEPALIVE timer. I.e. sending any PDU acts as a keepalive. Once the last fragment has been sent, the KEEPALIVE timer SHOULD be restarted. Do not wait for the ACK.

If a KEEPALIVE or other PDUs have not been received from a peer with which a receiver has an open session for a configurable time (default 30 seconds), the link SHOULD be presumed down. The devices MAY keep configuration state and restore it without retransmission if no data have changed. Otherwise, a new session SHOULD be established and new Encapsulation PDUs exchanged.

17. Layers-2.5 and 3 Liveness

Layer-2 liveness may be continuously tested by KEEPALIVE PDUs, see Section 16. As layer-2.5 or layer-3 connectivity could still break, liveness above layer-2 MAY be frequently tested using BFD ([RFC5880]) or a similar technique.

This protocol assumes that one or more Encapsulation addresses may be used to ping, run BFD, or whatever the operator configures.

18. The North/South Protocol

Thus far, a one-hop point-to-point logical link discovery protocol has been defined.

The devices know their unique LLEIs and know the unique peer LLEIs and Encapsulations on each logical link interface.

Full topology discovery is not appropriate at the L3DL layer, so Dijkstra a la IS-IS etc. is assumed to be done by higher level protocols such as BGP-SPF.

Therefore the LLEIs, link Encapsulations, and state changes are pushed North via a small subset of the BGP-LS API. The upper layer routing protocol(s), e.g. BGP-SPF, learn and maintain the topology, run Dijkstra, and build the routing database(s).

For example, if a neighbor's IPv4 Encapsulation address changes, the devices seeing the change push that change Northbound.

18.1. Use BGP-LS as Much as Possible

BGP-LS [RFC7752] defines BGP-like Datagrams describing logical link state (links, nodes, link prefixes, and many other things), and a new BGP path attribute providing Northbound transport, all of which can be ingested by upper layer protocols such as BGP-SPF; see Section 4 of [I-D.ietf-lsvr-bgp-spf].

For IPv4 links, TLVs 259 and 260 are used. For IPv6 links, TLVs 261 and 262. If there are multiple addresses on a link, multiple TLV pairs are pushed North, having the same ID pairs.

18.2. Extensions to BGP-LS

The Northbound protocol needs a few minor extensions to BGP-LS. Luckily, others have needed the same extensions.

Similarly to BGP-SPF, the BGP protocol is used in the Protocol-ID field specified in table 1 of [I-D.ietf-idr-bgpls-segment-routing-epe]. The local and remote node descriptors for all NLRI are the IDs described in Section 11. This is equivalent to an adjacency SID or a node SID if the address is a loopback address.

Label Sub-TLVs from [I-D.ietf-idr-bgp-ls-segment-routing-ext] Section 2.1.1, are used to associate one or more MPLS Labels with a link.

19. Discussion

This section explores some trade-offs taken and some considerations.

19.1. HELLO Discussion

A device with multiple Layer-2 interfaces, traditionally called a switch, may be used to forward frames and therefore packets from multiple devices to one logical interface (LLEI), I, on an L3DL speaking device. Interface I could discover a peer J across the switch. Later, a prospective peer K could come up across the switch. If I was not still sending and listening for HELLOs, the potential peering with K could not be discovered. Therefore, on multi-link interfaces, L3DL MUST continue to send HELLOs as long as they are turned up.

19.2. HELLO versus KEEPALIVE

Both HELLO and KEEPALIVE are periodic. KEEPALIVE might be eliminated in favor of keeping only HELLOs. But KEEPALIVEs are unicast, and thus less noisy on the network, especially if HELLO is configured to transit layer-2-only switches, see Section 19.1.

20. VLANs/SVIs/Sub-interfaces

One can think of the protocol as an instance (i.e. state machine) which runs on each logical link of a device.

As the upper routing layer must view VLAN topologies as separate graphs, L3DL treats VLAN ports as separate links.

L3DL PDUs learned over VLAN-ports may be interpreted by upper layer-3 routing protocols as being learned on the corresponding layer-3 SVI interface for the VLAN.

As Sub-Interfaces each have their own LLIEs, they act as separate interfaces, forming their own links.

21. Signature Types

The L3DL OPEN PDU contains an algorithm identifier, a key, and a L3DL certificate, which can be used to verify signatures on subsequent PDUs. This document describes two methods of key generation and signing for use by L3DL, Trust On First Use (TOFU) and a PKI-based mechanism to provide authentication as well as session integrity.

The Key in the OPEN PDU SHOULD be the public key of an asymmetric key pair. The sender signs with the private key, of course. The

device sending the OPEN PDU may use one key for all links, a different key for each link, or some mix(es) thereof.

In the TOFU method the key sent in the OPEN PDU is generated on the sending device, is believed without question by the receiver, and used to verify all subsequent PDUs from the same sender with the same public key and algorithm.

With the PKI method, an enrollment step is performed. The public key is signed by the the operational environment's trust anchor. In this way, the relying party can be confident that the public key is under control of the identified L3DL protocol entity.

As part of enrollment or before hand, all relying parties must have received the trust anchor in an authentic manner.

To the receiver verifying signatures on PDUs, the two methods are indistinguishable; the key provided in the OPEN PDU is used to verify the signatures of subsequent PDUs. The difference that PKI-based keys may be verified against the trust anchor when the OPEN PDU is received.

In the PKI method the public key in the OPEN PDU MUST be verified against the trust anchor for the operational domain. The OPEN PDU public key is then used to verify all subsequent PDUs in the session. A mechanism for 'rolling' from the current public key to a fresh one is described in Section 21.5.

21.1. Signature Algorithm Identifiers

To avoid the creation of yet another IANA registry for digital signature algorithm identifiers, this specification makes use of the existing IANA registry for "DNS Security Algorithm Numbers" [IANA]. In this registry, each signature algorithm is identified by an 8-bit value. The entries in this registry with "Y" in the "Zone Signing" column are appropriate for use with this protocol.

For interoperability, all implementations of this protocol MUST support the RSASHA256 algorithm (identified by the value 0x08). Implementation MAY support any other registered "Zone Signing" signature algorithms.

21.2. Trust On First Use Method

There are three parts to using a key: signing PDUs, verifying the OPEN PDU, and verifying subsequent PDUs.

21.2.1. Signing a PDU

All signed PDUs are generated in the same way:

- Compose the PDU, with all fields including "Sig Algo" and "Signature Length" set, but omitting the trailing "Signature" field itself. The Certificate Length should be zero and the Certificate field should be empty. This is the "message to be signed" for purposes of the signature algorithm.

- Generate the signature as specified for the chosen algorithm, using the private key of the asymmetric key pair. In general, this will involve first hashing the "message to be signed" then signing the hash, but the precise details may vary with the specific signature algorithm. The result will be a sequence of octets, the length of which MUST be equal to the value in the "Signature Length" field.

- Construct the complete message by appending the signature octets to the otherwise complete message composed above.

In the case of the OPEN PDU, the message to be signed will include the public member of the asymmetric keypair, but as far as the signature algorithm is concerned that's just payload, no different from any other PDU content.

21.2.2. Verifying the OPEN PDU

The process for verifying an OPEN PDU is slightly different from the process for verifying other PDU types, because the OPEN PDU also establishes the session key.

- Verify that the PDU is syntactically correct, and extract the Auth Type, Key, Sig Type, and Signature fields.

- Verify that Auth Type and Sig Type refer to the same algorithm suite, and that said algorithm suite is one that the implementation understands.

- Construct the "message to be verified" by truncating the PDU to remove the Signature field (in practice this should not require copying any data, just subtract the signature length from the PDU length).

- Verify the message constructed above against the public key using the rules for the specific signature suite.

- Record Auth Type and Key as this sessions's authentication type and session key, for use in verifying subseuqent PDUs.

If any of the above verification steps fail, generate an error using error code 2 ("Authorization failure in OPEN").

21.2.3. Verifying Other PDUs

The process for verifying non-OPEN PDUs is slightly simpler, but follows the same basic pattern as for OPEN PDUs.

- Verify that the PDU is syntactically correct, and extract the Sig Type and Signature fields.

- Verify that Sig Type refers to the same algorithm suite as the Auth Type recorded during verification of the OPEN PDU.

- Construct the "message to be verified" by truncating the PDU to remove the Signature field.

- Verify the message constructed above against the recorded session key using the rules for the specific signature suite.

If any of the above verification steps fail, generate an error using error code 3 ("Signature failure in PDU").

21.3. Public Key Infrastructure Method

Using a PKI is almost the same as using TOFU, but with one additional step: during verification of an OPEN PDU, after extracting the Key field from the PDU but before attempting to use it to verify the OPEN PDU signature, the receiver MUST verify the received key against the PKI to confirm that it's an authorized key.

Generating an OPEN PDU using the PKI method requires a certificate, which must be supplied via out of band configuration. The certificate is a signature of the public key to be sent in the Key field of the OPEN PDU, signed by the trust anchor private key.

Verifying an OPEN PDU using the PKI method requires the public key of the trust anchor, which the receiver uses to verify the certificate, thereby demonstrating that the supplied public key represents an authorized L3DL speaker in this administrative domain.

We use the term "certificate" here in the generic sense, not as defined in [RFC5280]. X.509 certificates are not used here; X.509 certificates are more complicated than needed for L3DL. The L3DL certificates are just signatures of one key (the public key supplied in the Key field of the OPEN PDU) that can be verified by another trusted public key (the trust anchor).

21.3.1. Signing OPEN PDU with PKI

Generating and signing the OPEN PDU with the PKI method is almost the same as in Section 21.2.1. The only difference is that the PKI method MUST supply the appropriate certificate in the Certificate field.

Note that the Auth Type field applies to both the Key and Certificate fields. That is: the certificate uses the same certificate suite as the session keys, L3DL does not support cross-algorithm-suite certification.

21.3.2. Verifying OPEN PDU with PKI

Verifying the OPEN PDU with PKI is similar to verifying with TOFU as described in Section 21.2.2, but includes one critical extra step:

After extracting the Key field from the PDU but before verifying the Signature, extract the Certificate field and verfiy that the Certificate is a valid signature of the Key field, according to the rules for the signature suite specified by Auth Type. If this step fails, handle as in Section 21.2.2.

21.4. Local Policy

Whether to use TOFU, PKI, or no signatures at all is a matter of local policy, to be decided by the operator. The useful policy combinations for Key and Certificate are probably:

*Not signing: sender need not sign, receiver does not check.

*Require TOFU: sender MUST supply key and receiver MUST check, but L3DL certificates not needed and ignored if sent.

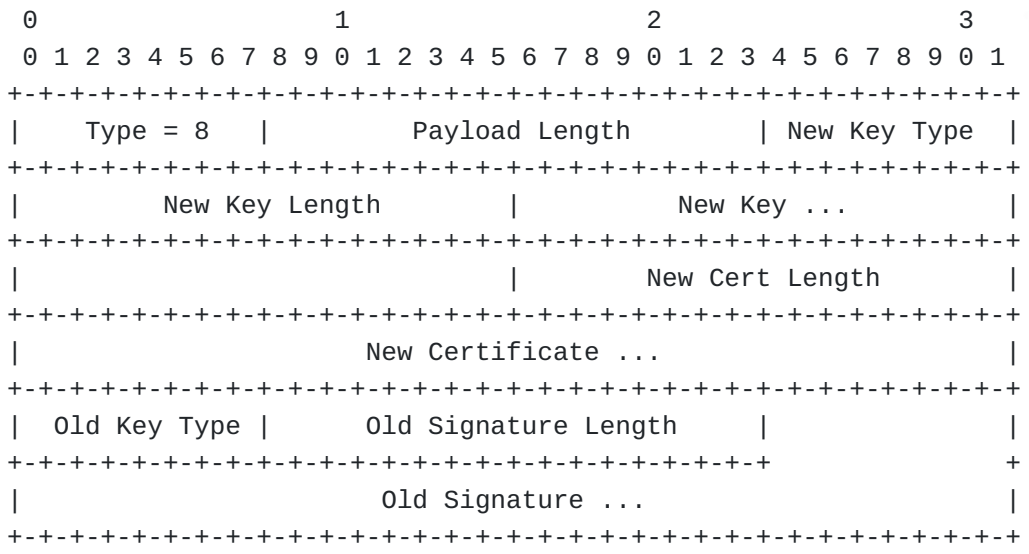
*Allow TOFU: sender MUST supply key and receiver MUST check, receiver SHOULD check certificate if supplyed by sender.

*Require PKI: sender MUST supply key and L3DL certificate, receiver MUST check signature and verify the L3DL certificate.

21.5. NEWKEY, Key Roll

Modern key management allows for agility in 'rolling' to a new key or even algorithm in case of key expiry, key compromise, or merely prudence. Declaring a new key with an L3DL OPEN PDU would cause serious churn in topology as a new OPEN PDU may cause a withdraw of previously announced encapsulations. Therefore, a gentler rekeying is needed.

Prior to 'rolling' to a new key or new algorithm, a new public/ private key pair is generated. If PKI is being used, then the trust anchor also signs the new public key to create a new L3DL certificate.



The New Key Type, New Key Length, New Key, New Cert Length, and New Certificate fields declare the replacement algorithm, key, and L3DL certificate.

The NEWKEY PDU is signed using the current (soon to be old) algorithm and key.

The sender and the receiver should be cautious of signature algorithm downgrade attacks.

To avoid possible race conditions, the receiver SHOULD accept signatures using either the new or old key for a configurable time (default 30 seconds). This is intended to accommodate situations such as senders with high peer out-degree and a single per-device asymmetric key.

If the sender does not receive an ACK in the normal window, including retransmission, then the sender MAY choose to allow a session reset by either issuing a new OPEN PDU or by letting the receiver eventually have a signature failure (error code 3) on a PDU.

The rekeying operation changes the session key and the associated algorithm described in Section 21.2.3. The NEWKEY PDU itself is verified using the old algorithm and session key. After the NEWKEY PDU has been accepted, subsequent PDUs are verified with the new algorithm and the new session key.

22. Implementation Considerations

An implementation SHOULD provide the ability to configure each logical interface as L3DL speaking or not.

An implementation SHOULD provide the ability to configure whether HELLOs on an L3DL enabled interface send Nearest Bridge or the MAC which is propagated by switches from that interface; see Section 10.

An implementation SHOULD provide the ability to distribute one or more loopback addresses or interfaces into L3DL on an external L3DL speaking interface.

An implementation SHOULD provide the ability to distribute one or more overlay and/or underlay addresses or interfaces into L3DL on an external L3DL speaking interface.

An implementation SHOULD provide the ability to configure one of the addresses of an encapsulation as primary on an L3DL speaking interface. If there is only one address for a particular encapsulation, the implementation MAY mark it as primary by default.

An implementation MAY allow optional configuration which updates the local forwarding table with overlay and underlay data both learned from L3DL peers and configured locally.

23. Security Considerations

The protocol as is MUST NOT be used outside a datacenter or similarly closed environment without authentication and authorization mechanisms such as [I-D.ymbk-lsvr-l3dl-signing].

Many MDC operators have a strange belief that physical walls and firewalls provide sufficient security. This is not credible. All MDC protocols need to be examined for exposure and attack surface. In the case of L3DL, Authentication and Integrity as provided in [I-D.ymbk-lsvr-l3dl-signing] is strongly recommended.

It is generally unwise to assume that on the wire Layer-2 is secure. Strange/unauthorized devices may plug into a port. Mis-wiring is very common in datacenter installations. A poisoned laptop might be plugged into a device's port, form malicious sessions, etc. to divert, intercept, or drop traffic.

Similarly, malicious nodes/devices could mis-announce addressing.

If OPENs are not being authenticated, an attacker could forge an OPEN for an existing session and cause the session to be reset.

For these reasons, the OPEN PDU's authentication data exchange SHOULD be used.

If the KEEPALIVE PDU is not signed (as suggested in Section 8) to save computation, then a MITM could fake a session being alive.

As the ULPC PDU may contain keying material, see Section 14.1.4, it SHOULD BE signed.

Any keying material in the PDU SHOULD BE salted and hashed.

The BGP Authentication sub-TLV provides for provisioning MD5, which is a quite weak hash, horribly out of fashion, and kills puppies. But, like it or not, it has been sufficient against the kinds of attacks BGP TCP sessions have endured. So it is what BGP deployments use.

The TOFU method requires a leap of faith to accept the key in the OPEN PDU, as it can not be verified against any authority. Hence it is jokingly referred to as Married On First Date. The assurance it does provide is that subsequent signed PDUs are from the same peer. And data integrity is a positive side effect of the signature covering the payload.

The PKI method offers assurance that the L3DL certificate, and hence the public key, provided in the OPEN PDU are authorized by a central authority, e.g. the network's security team. The onward assurance of talking to the same peer and data integrity are the same as in the TOFU method.

With the PKI method, automated device provisioning could restrict which L3DL certificates are allowed from which peers on a per interface basis. This would complicate key rolls. Where one draws the line between rigidity, flexibility, and security varies.

The REKEY PDU is open to abuse to create a signature algorithm downgrade attack.

24. IANA Considerations

24.1. PDU Types

This document requests the IANA create a registry for L3DL PDU Type, which may range from 0 to 255. The name of the registry should be L3DL-PDU-Type. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

PDU Code	PDU Name
0	HELLO
1	OPEN
2	KEEPALIVE
3	ACK
4	IPv4 Announcement
5	IPv6 Announcement
6	MPLS IPv4 Announcement
7	MPLS IPv6 Announcement
8	NEWKEY
9	ULPC
10-254	Reserved
255	VENDOR

24.2. ULPC Type

This document requests the IANA create a registry for L3DL ULPC Type, which may range from 0 to 255. The name of the registry should be L3DL-ULPC-Type. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

Value	Name
0	Reserved
1	BGP
2-255	Reserved

24.3. Signature Type

This document requests the IANA create a registry for L3DL Signature Type, AKA Sig Type, which may range from 0 to 255. The name of the registry should be L3DL-Signature-Type. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

Number	Name
0	Null
1	TOFU - Trust On First Use
2	PKI
3-255	Reserved

24.4. Flag Bits

This document requests the IANA create a registry for L3DL PL Flag Bits, which may range from 0 to 7. The name of the registry should be L3DL-PL-Flag-Bits. The policy for adding to the registry is RFC

Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

Bit	Bit Name
0	Announce/Withdraw (ann == 0)
1	Primary
2	Underlay/Overlay (under == 0)
3	Loopback
4-7	Reserved

24.5. Error Codes

This document requests the IANA create a registry for L3DL Error Codes, a 16 bit integer. The name of the registry should be L3DL-Error-Codes. The policy for adding to the registry is RFC Required per [RFC5226], either standards track or experimental. The initial entries should be the following:

Error Code	Error Name
0	No Error
1	Checksum Error
2	Logical Link Addressing Conflict
3	Authorization Failure
4	Announce/Withdraw Error

25. IEEE Considerations

This document requires a new EtherType.

This document requires a new multicast MAC address that will be broadcast through a switch.

26. Acknowledgments

The authors thank Cristel Pelsser for multiple reviews, Harsha Kovuru for comments during implementation, Jeff Haas for review and comments, Jörg Ott for an early but deep transport review, Joe Clarke for a useful review, John Scudder for deeply serious review and comments, Larry Kreeger for a lot of layer-2 clue, Martijn Schmidt for his contribution, Nalinaksh Pai for transport discussions, Neeraj Malhotra for review, Paul Congdon for Ethernet hints, Russ Housley for checksum discussion and sBox, and Steve Bellovin for checksum advice.

27. References

27.1. Normative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing", Work in Progress, Internet-Draft, draft-ietf-idr-bgp-ls-segment-routing-ext-18, 15 April 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgp-ls-segment-routing-ext-18>.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Talaulikar, K., Filsfils, C., Patel, K., Ray, S., and J. Dong, "Border Gateway Protocol - Link State (BGP-LS) Extensions for Segment Routing BGP Egress Peer Engineering", Work in Progress, Internet-Draft, draft-ietf-idr-bgpls-segment-routing-epe-19, 16 May 2019, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-bgpls-segment-routing-epe-19>.

[I-D.ietf-lsvr-bgp-spf] Patel, K., Lindem, A., Zandi, S., and W. Henderickx, "BGP Link-State Shortest Path First (SPF) Routing", Work in Progress, Internet-Draft, draft-ietf-lsvr-bgp-spf-29, 25 November 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-lsvr-bgp-spf-29>.

[I-D.ymbk-lsvr-l3dl-signing] Bush, R. and R. Austein, "Layer 3 Discovery and Liveness Signing", Work in Progress, Internet-Draft, draft-ymbk-lsvr-l3dl-signing-01, 6 May 2020, <https://datatracker.ietf.org/doc/html/draft-ymbk-lsvr-l3dl-signing-01>.

[IANA] "DNS Security Algorithm Numbers", <https://www.iana.org/assignments/dns-sec-alg-numbers/dns-sec-alg-numbers.xhtml>.

[IANA-PEN] "IANA Private Enterprise Numbers", <https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers>.

[IEEE.802_2001] IEEE, "IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture", IEEE 802-2001, DOI 10.1109/ieeestd.2002.93395, 27 July 2002, <http://ieeexplore.ieee.org/servlet/opac?punumber=7732>.

[IEEE802-2014] Institute of Electrical and Electronics Engineers, "Local and Metropolitan Area Networks: Overview and Architecture", IEEE Std 802-2014, 2014.

[RFC1213] McCloghrie, K. and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991, <https://www.rfc-editor.org/info/rfc1213>.

[RFC1629] Colella, R., Callon, R., Gardner, E., and Y. Rekhter, "Guidelines for OSI NSAP Allocation in the Internet", RFC 1629, DOI 10.17487/RFC1629, May 1994, <https://www.rfc-editor.org/info/rfc1629>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, DOI 10.17487/RFC3032, January 2001, <https://www.rfc-editor.org/info/rfc3032>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[RFC5082] Gill, V., Heasley, J., Meyer, D., Savola, P., Ed., and C. Pignataro, "The Generalized TTL Security Mechanism (GTSM)", RFC 5082, DOI 10.17487/RFC5082, October 2007, <https://www.rfc-editor.org/info/rfc5082>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC6286] Chen, E. and J. Yuan, "Autonomous-System-Wide Unique BGP Identifier for BGP-4", RFC 6286, DOI 10.17487/RFC6286, June 2011, <https://www.rfc-editor.org/info/rfc6286>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

27.2. Informative References

[I-D.malhotra-bess-evpn-lsoe] Malhotra, N., Patel, K., and J. Rabadan, "LSoE-based PE-CE Control Plane for EVPN", Work in Progress, Internet-Draft, draft-malhotra-bess-evpn-lsoe-00, 11 March 2019, <https://datatracker.ietf.org/doc/html/draft-malhotra-bess-evpn-lsoe-00>.

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <https://www.rfc-editor.org/info/rfc791>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/info/rfc1122>.

[RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <https://www.rfc-editor.org/info/rfc1982>.

[RFC2385] Heffernan, A., "Protection of BGP Sessions via the TCP MD5 Signature Option", RFC 2385, DOI 10.17487/RFC2385, August 1998, <https://www.rfc-editor.org/info/rfc2385>.

[RFC4808] Bellovin, S., "Key Change Strategies for TCP-MD5", RFC 4808, DOI 10.17487/RFC4808, March 2007, <https://www.rfc-editor.org/info/rfc4808>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[RFC7210] Housley, R., Polk, T., Hartman, S., and D. Zhang, "Database of Long-Lived Symmetric Cryptographic Keys", RFC 7210, DOI 10.17487/RFC7210, April 2014, <https://www.rfc-editor.org/info/rfc7210>.

Authors' Addresses

Randy Bush
Arrcus & Internet Initiative Japan
5147 Crystal Springs

Bainbridge Island, WA 98110
United States of America

Email: randy@psg.com

Rob Austein
Arrcus, Inc

Email: sra@hactrn.net

Russ Housley
Vigil Security, LLC
516 Dranesville Road
Herndon, VA 20170
United States of America

Email: housley@vigilsec.com

Keyur Patel
Arrcus
2077 Gateway Place, Suite #400
San Jose, CA 95119
United States of America

Email: keyur@arrcus.com