Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: March 31, 2019

R. Bush
Arrcus & IIJ
K. Patel
Arrcus

September 27, 2018

Link State Over Ethernet
draft-ymbk-lsvr-lsoe-02

Abstract

Used in Massive Data Centers (MDCs), BGP-SPF and similar protocols need link neighbor discovery, link encapsulation data, and Layer 2 liveness. The Link State Over Ethernet protocol provides link discovery, exchanges supported encapsulations (IPv4, IPv6, ...), discovers encapsulation addresses (Layer 3 / MPLS identifiers) over raw Ethernet, and provides layer 2 liveness checking. The interface data are pushed directly to a BGP-LS API, obviating the need for centralized controller architectures. This protocol is intended to be more widely applicable to other upper layer routing protocols which need link discovery and characterisation.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC2119] only when they appear in all upper case. They may also appear in lower or mixed case as English words, without normative meaning. See [RFC8174].

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 March 31, 2019.

Table of Contents

1. Introduction

The Massive Data Center (MDC) environment presents unusual problems of scale, e.g. O(10,000) devices, while its homogeneity presents opportunities for simple approaches. Approaches such as Jupiter Rising [JUPITER] 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 and similar environments. But BGP-SPF and similar higher level device-spanning protocols need link state and addressing data from the network to build the routing topology. LLDP has scaling issues, e.g. in extending a PDU beyond 1,500 bytes.

Link State Over Ethernet (LSOE) provides brutally simple mechanisms for devices to

- Discover each other's Layer 2 (MAC) Addresses,
- Run Layer 2 keep-alive messages for liveness continuity,
- Discover each other's unique IDs (ASN, RouterID, ...),
- Discover mutually supported encapsulations, e.g. IP/MPLS,
- Discover Layer 3 and/or MPLS addressing of interfaces of the link encapsulations,
- Enable layer 3 link liveness such as BFD, and finally
- 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.

This protocol may be more widely applicable to a range of routing and similar protocols which need link discovery and characterisation.

2. Terminology

Even though it concentrates on the Ethernet layer, this document relies heavily on routing terminology. The following are some possibly confusing terms:

Association: An established, vis OPEN messages, session between two LSOE capable devices,

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 SPF decision process. See [I-D.ietf-lsvr-bgp-spf].

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

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

Frame: An Ethernet Layer 2 packet.

MAC Address: Media Access Control Address, essentially an Ethernet address, six octets.

MDC: Massive Data Center, commonly thousands of TORs.

Message: A complete application layer message. These may be longer than will fit in a single Ethernet frame.

PDU: Protocol Data Unit, essentially an application layer packet. A Message may need to be broken into multiple PDUs to make it through MTU or other restrictions.

RouterID: An 32-bit identifier unique in the current routing domain, see [RFC4271] updated by [RFC6286].

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

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

LSOE assumes a datacenter scale and topology, but can accommodate richer topologies which contain potential cycles.

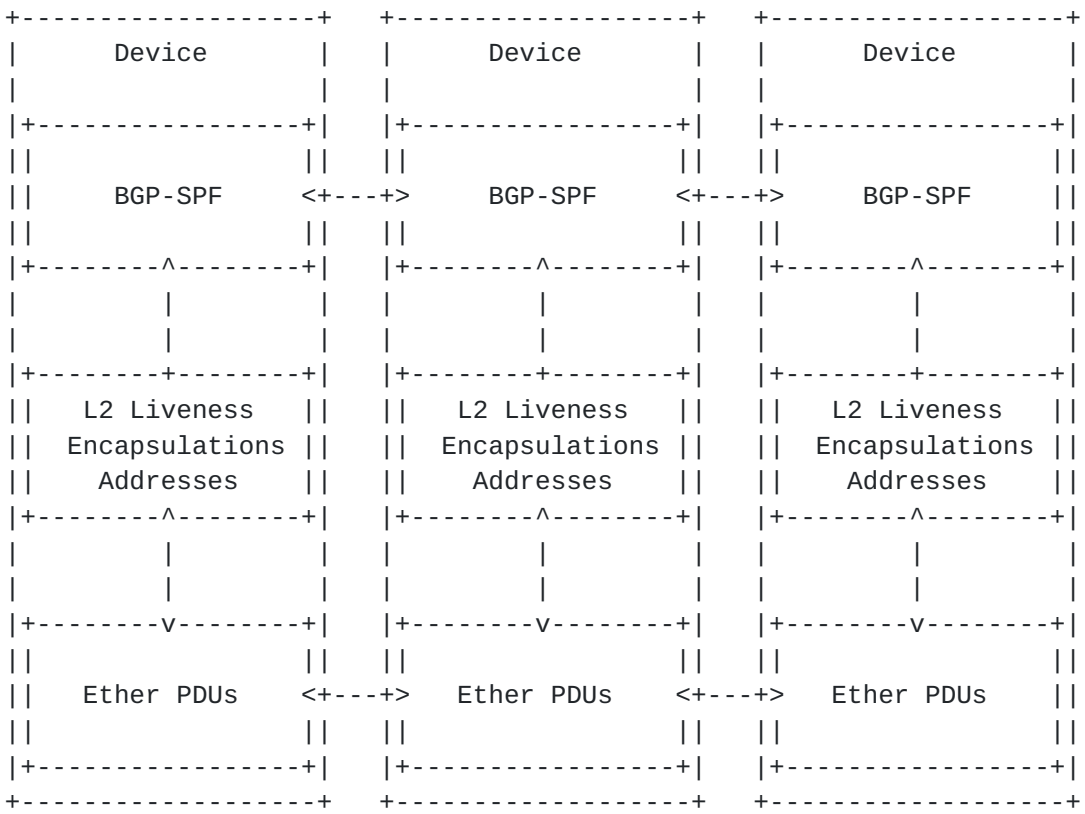
While LSOE is designed for the MDC, there are no inherent reasons it could not run on a WAN; though, as it is simply a discovery protocol, it is not clear that this would be useful. The authentication and

authorisation needed to run safely on the WAN are not provided in detail in this version of the protocol, although future versions/ extensions could expend on them.

LSOE assumes a new IEEE assigned EtherType (TBD).

4. Top Level Overview

- Devices discover each other on Ethernet links
- MAC addresses and Link State are exchanged over Ethernet
- Layer 2 Liveness Checks are begun
- Encapsulation data are exchanged and IP-Level Liveness Checks done
- A BGP-like protocol is assumed to use these data to discover and build a topology database



There are two protocols, the Ethernet discovery and the interface to the upper level BGP-like protocol:

- Layer 2 Ethernet protocols are used to exchange Layer 2 data, i.e. MAC addresses, and layer 2.5 and 3 identifiers (not payloads), i.e. ASNs, Encapsulations, and interface 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 LSOE protocols.

To simplify this document, Layer 2 Ethernet framing is not shown.

5. Ethernet to Ethernet Protocols

Two devices discover each other and their respective MAC addresses by sending multicast HELLO messages (Section 5.3). To allow discovery of new devices coming up on a multi-link topology, devices send periodic HELLOs forever, see Section 8.1.

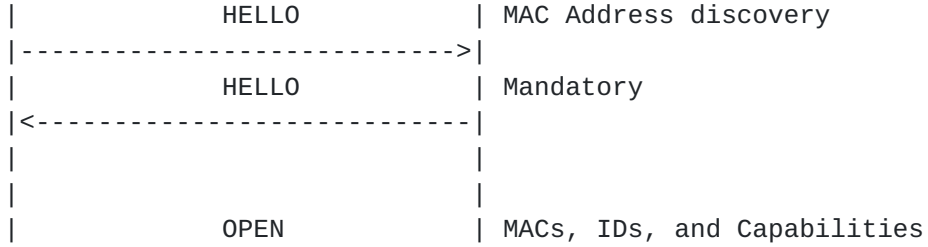
Once a new device is recognized, both devices attempt to negotiate and establish peering by sending unicast OPEN messages (Section 5.4). In an established peering, Encapsulations (Section 5.5) may be announced and modified. When two devices on a link have compatible Encapsulations and addresses, i.e. the same AFI/SAFI and the same subnet, the link is announced via the BGP-LS API.

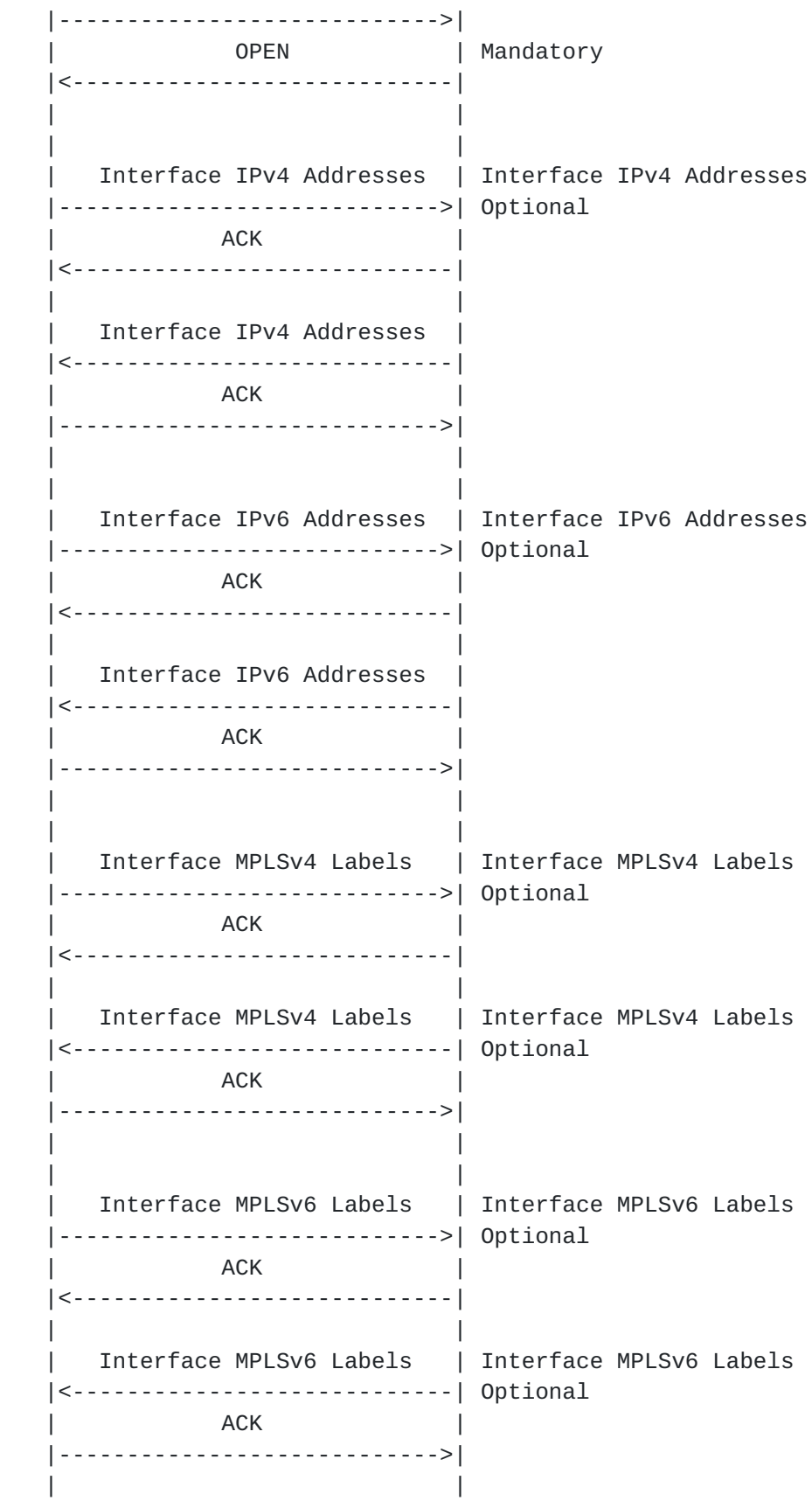
5.1. Inter-Link Ether Protocol Overview

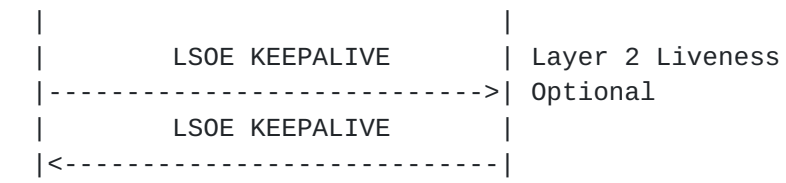
The HELLO, Section 5.3, is a priming message. It is an Ethernet multicast of a minimal message with the simple goal of discovering the Ethernet MAC address(es) of devices reachable via an interface.

The HELLO and OPEN, Section 5.4, messages, which are used to discover and exchange MAC address and IDs, are mandatory; other messages are optional; though at least one encapsulation MUST be agreed at some point.

The following is a ladder-style sketch of the Ethernet protocol exchanges:




OPEN
Mandatory
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



5.2. Messages, PDUs, and Frames

An LSOE Message occupies one or more Ethernet Frames. We refer to the payload in an Ethernet frame as a PDU (Protocol Data Unit).

If a message will not fit in a single frame/PDU, likely due to MTU issues, then the message is broken into multiple PDUs each in its own frame, each with a full header. The OPEN message and the Encapsulation messages provide for multi-PDU packaging.

The Flags field indicates whether the message required multiple PDUs and if the current PDU is the last of a multi-PDU sequence; see Section 5.2.1.

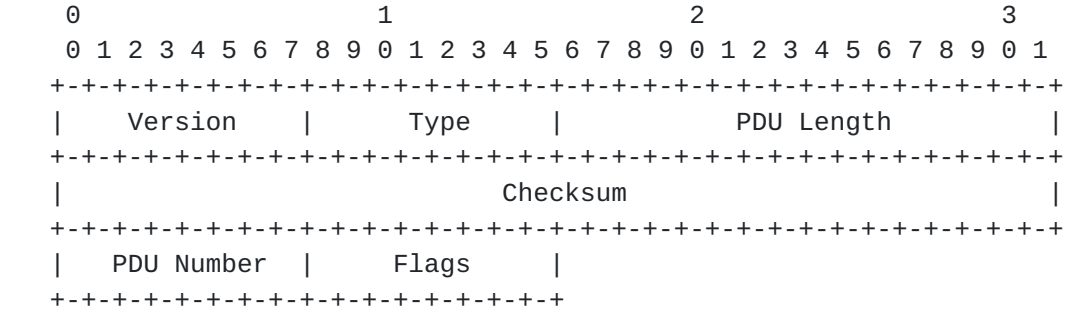
The PDU Number field of a multi-PDU message must monotonically increase, modulo 256, see [RFC1982]. This and the Flags field may be used to re-assemble long messages.

5.2.1. Frame TLV

The basic LSOE message is a typical TLV (Type Length Value) PDU with a Version number in front.

Aside from the HELLO, Section 5.3, and KEEPALIVE, Section 5.7, an LSOE PDU has a PDU Sequence Number and a Flags field to allow assembly of out order PDUs/frames of messages too long to fit in one Ethernet frame.

All LSOE frames/PDUs, other than the HELLO and KEEPALIVE, have the following header:



The fields of the basic LSOE header are as follows:

Version: Version number of the protocol, currently 0.

Type: An integer differentiating message payload types

0 - HELLO
1 - OPEN
2 - KEEPALIVE
3 - Encapsulation ACK
4 - IPv4 Announcement
5 - IPv6 Announcement
6 - MPLS IPv4 Announcement
7 - MPLS IPv6 Announcement
8-255 Reserved

PDU Length: Total number of octets in the PDU including all payloads and fields

Checksum: A 32 bit hash over the message to detect bit flips, see Section 5.2.1.1

PDU Number: 0..255, a monotonically increasing value, modulo 256, see [RFC1982].

Flags: A bit field, the low order bit is number 0

0 - One of a multi-Frame sequence
1 - last of a multi-Frame sequence
2-7 - Reserved

5.2.1.1. 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 conservative approach.

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

Sum up 32-bit unsigned ints in a 64-bit long, then take the high-order section, shift it right, rotate, add it in, repeat until zero.

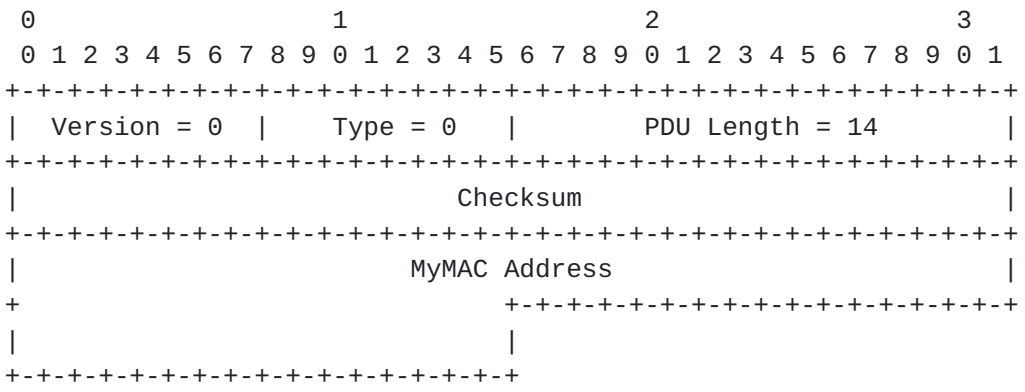
```
#include <stddef.h>
#include <stdint.h>

/* 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 & 0xFFFFFFFF);
    result = (result >> 32) + (result & 0xFFFFFFFF);
    return (uint32_t) result;
  }
```

5.3. HELLO

The HELLO message is unique in that it is a multicast Ethernet frame. It solicits response(s) from other device(s) on the link. See Section 8.1 for why multicast is used.

All other LSOE messages are unicast Ethernet frames, as the peer's MAC Address is known after the HELLO exchange.

When an interface is turned up on a device, it SHOULD issue a HELLO periodically. The interval is set by configuration.



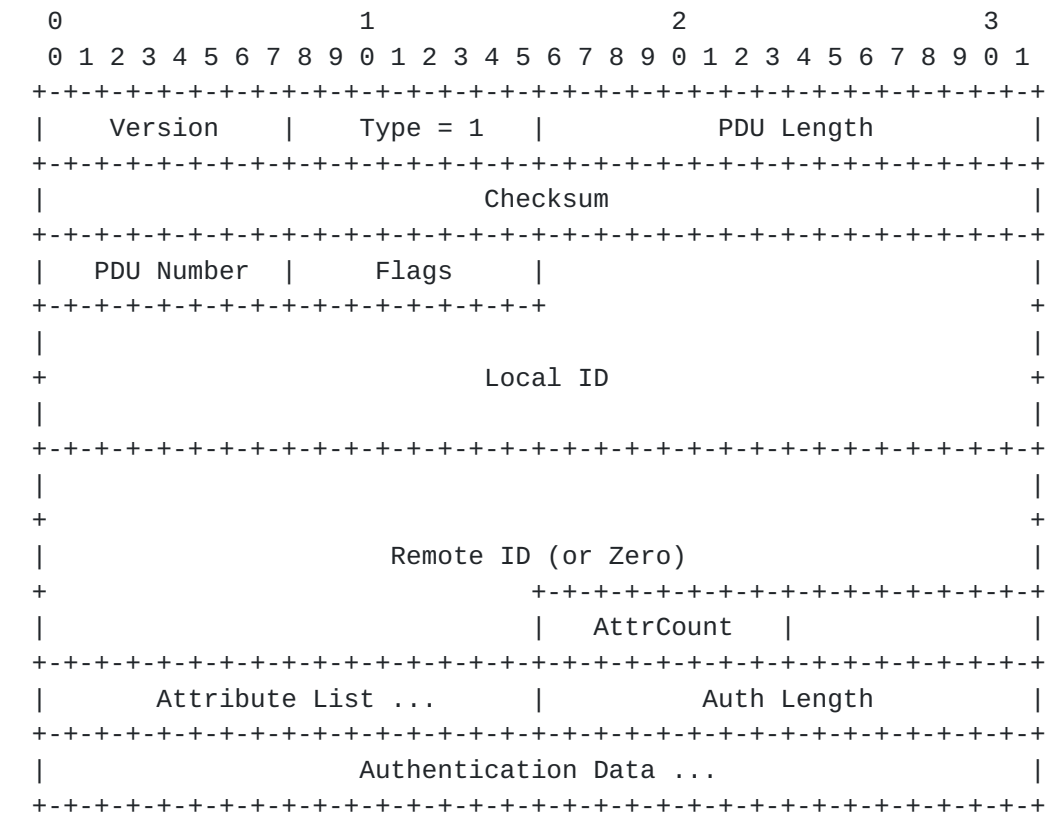
If more than one device responds, one adjacency is formed for each unique (MAC address) response. LSOE treats the adjacencies as separate links.

When a HELLO is received from a MAC address where there is no established LSOE adjacency, the receiver SHOULD respond with an OPEN message. The two devices establish an LSOE adjacency by exchanging OPEN messages.

The PDU Length is the octet count of the entire PDU, including the Version, the Type, the PDU Length field itself, the Checksum, and the following payload.

5.4. OPEN

Each device has learned the other's MAC address from the HELLO exchange, see Section 5.3. Therefore the OPEN and subsequent messages are unicast, as opposed to the HELLO's multicast, Ethernet frames.



An ID can be an ASN with high order bits zero, a classic RouterID with high order bits zero, a catenation of the two, a 80-bit ISO System-ID, or any other identifier unique to a single device in the current routing space. IDs are big-endian.

When the local device sends an OPEN without knowing the remote device's ID, the Remote ID MUST be zero. The Local ID MUST NOT be zero.

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

A node may have zero or more user-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 message.

Auth Length is the length in octets of the Authentication Data, not including the Auth Length itself. If there are no Authentication Data, the Auth Length MUST BE zero.

The Authentication Data are specific to the operational environment. A failure to authenticate is a failure to start the LSOE association, and HELLOs MUST BE restarted.

Once two devices know each other's MAC addresses, and have exchanged OPEN messages, Layer 2 KEEPALIVEs (see Section 5.7) SHOULD be started to ensure Layer 2 liveness and keep the association semantics alive. The timing and acceptable drop of the KEEPALIVE messages SHOULD be configured.

If a properly authenticated OPEN arrives from a device with which the receiving device believes it already has an LSOE association (OPENs have already been exchanged), the receiver MUST assume that the sending device has been reset. All discovered data MUST BE withdrawn via the BGP-LS API and the recipient MUST respond with a new OPEN.

5.5. The Encapsulations

Once the devices know each other's MAC addresses, know each other's upper layer identities, have means to ensure link state, etc., the LSOE 'association' is considered established, and the devices SHOULD announce their interface encapsulation, addresses, (and labels).

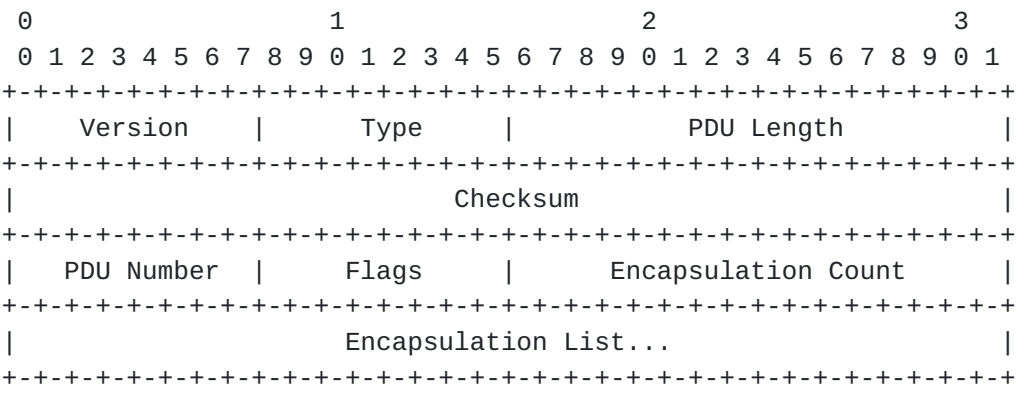
The Encapsulation types the peers exchange may be IPv4 Announcement (Section 5.5.3), IPv6 Announcement (Section 5.5.4), MPLS IPv4 Announcement (Section 5.5.6), MPLS IPv6 Announcement (Section 5.5.7), or possibly others not defined here.

The sender of an Encapsulation message MUST NOT assume that the peer is capable of the same Encapsulation Type. An Encapsulation ACK (Section 5.6) merely acknowledges receipt. Only if both peers have sent the same Encapsulation Type is it safe to assume that they are compatible for that type.

Further, to consider a 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, i.e. on the same IPvX subnet.

5.5.1. The Encapsulation Message Skeleton

The header for all encapsulation messages is as follows:



If the length of an Encapsulation message exceeds the PDU size limit on media, the message is broken into multiple PDUs. See Section 5.2.

The Encapsulation Exchange is over an unreliable transport so the PDU Number and ACKs are used for reassembly.

The Receiver MUST acknowledge the Encapsulation PDU with a Type=3, Encapsulation ACK PDU (Section 5.6) with the Encapsulation Type being that of the encapsulation being announced, see Section 5.6 and the PDU Number being the PDU Number of the Encapsulation PDU being ACKnowledged.

If the Sender does not receive an ACK in one second, they SHOULD retransmit. After a user configurable number of failures, the LSOE association should be considered dead and the OPEN process SHOULD be restarted.

An Encapsulation message MAY describe multiple addresses of the encapsulation type.

An Encapsulation message of Type T replaces all previous encapsulations of Type T.

To remove all encapsulations of Type T, the sender uses and Encapsulation Count of zero.

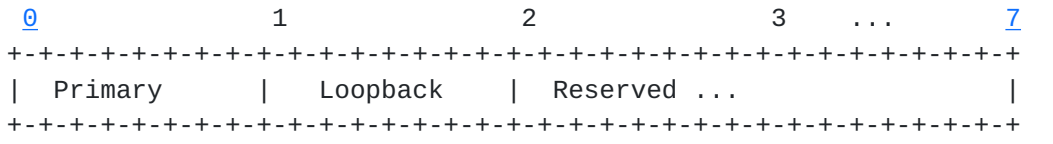
If an interface has multiple addresses for an encapsulation type, one address SHOULD be marked as primary, see Section 5.5.2.

If a loopback address needs to be exposed, e.g. for iBGP peering, then it should be marked as such, see Section 5.5.2.

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

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

5.5.2. Prim/Loop Flags

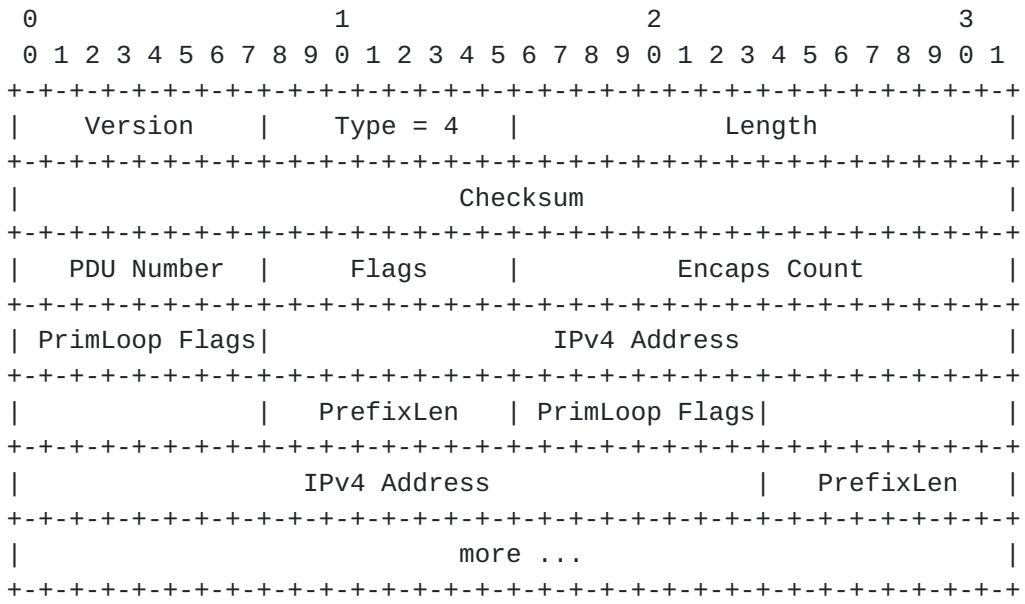


Each Encapsulation interface address MAY be marked as a primary address, and/or a loopback, in which case the respective bit is set to one.

Only one address MAY be marked as primary for an encapsulation type.

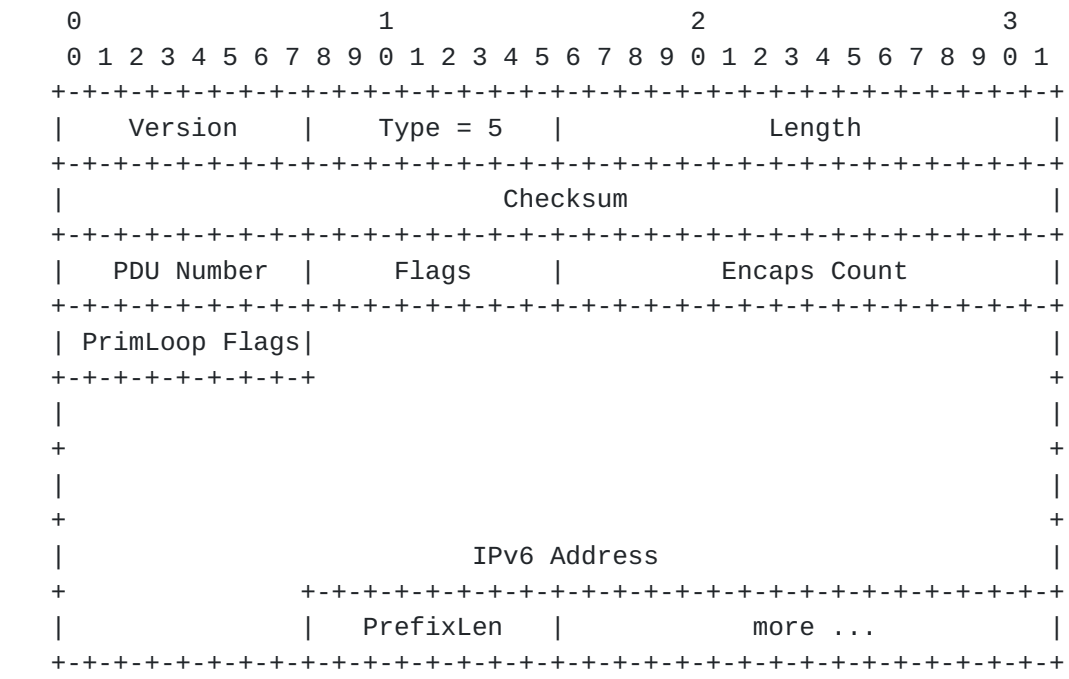
5.5.3. IPv4 Encapsulation

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 address and the prefix length.



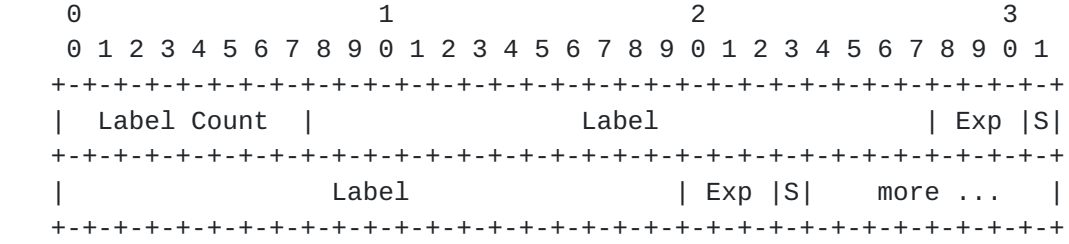
5.5.4. IPv6 Encapsulation

The IPv6 Encapsulation describes a device's ability to exchange IPv6 packets on one or more subnets. It does so by stating the interface's address and the prefix length.



5.5.5. MPLS Label List

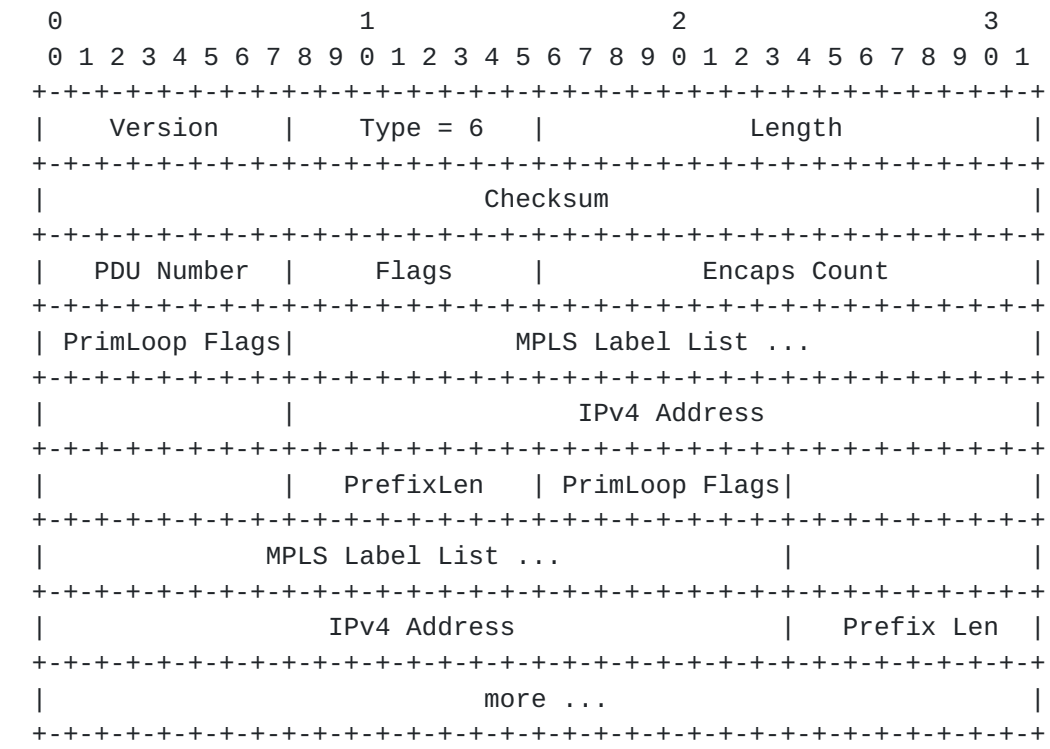
As an MPLS enabled interface may have a label stack, see [RFC3032], a variable length list of labels is needed.



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

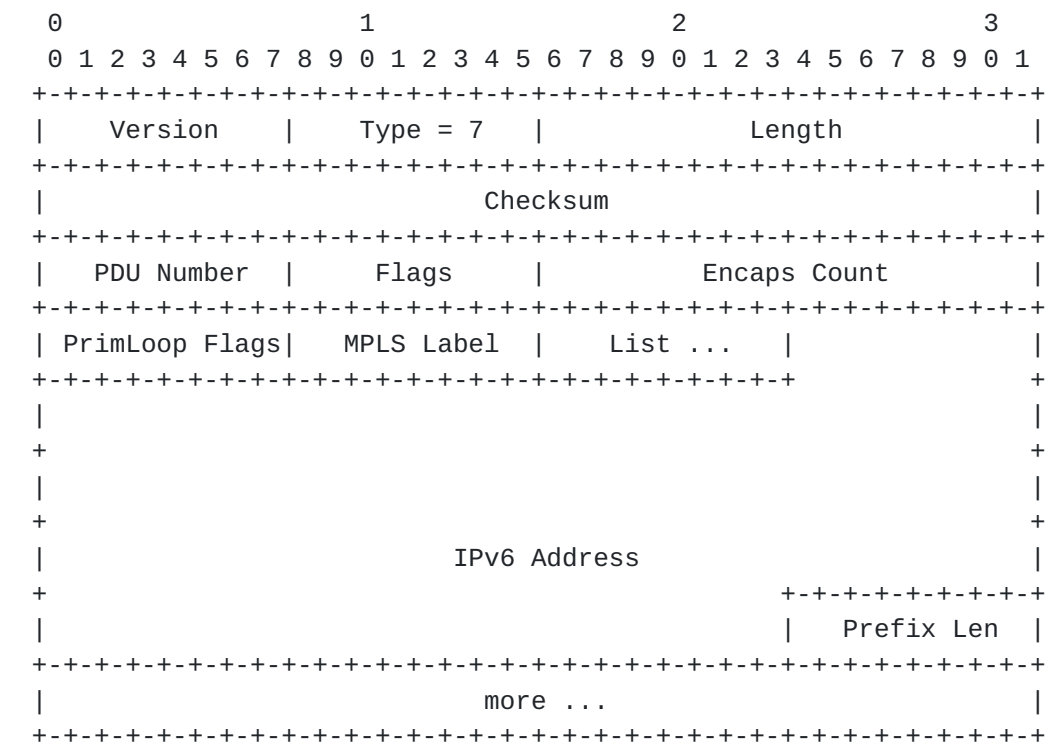
5.5.6. MPLS IPv4 Encapsulation

The MPLS IPv4 Encapsulation describes a device's ability to exchange labeled IPv4 packets on one or more subnets. It does so by stating the interface's address and the prefix length.

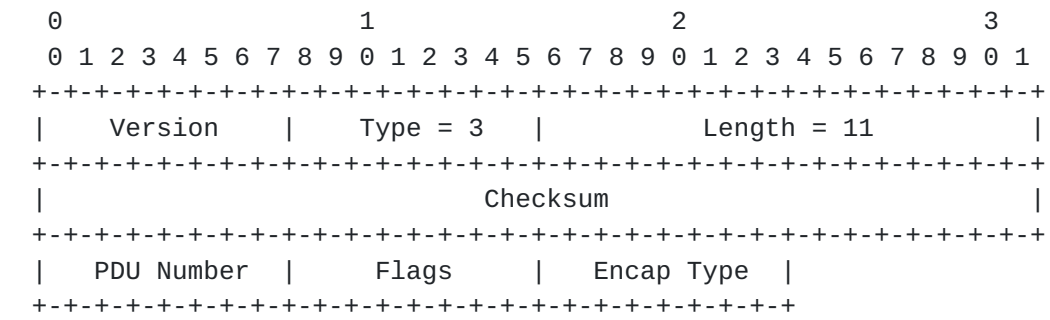


5.5.7. MPLS IPv6 Encapsulation

The MPLS IPv6 Encapsulation describes a device's ability to exchange labeled IPv6 packets on one or more subnets. It does so by stating the interface's address and the prefix length.



5.6. Encapsulation ACK



The header is as described in Section 5.5.

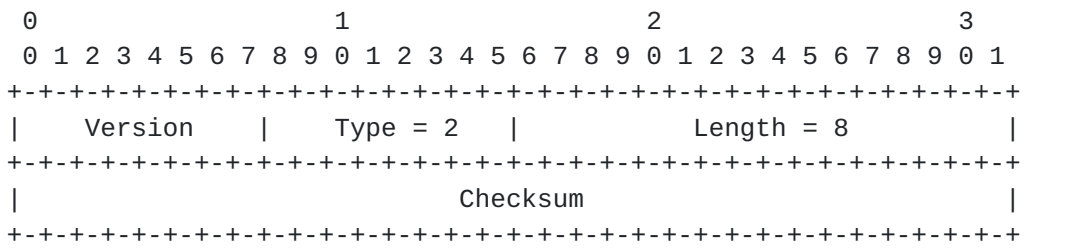
The PDU Number is the number of the received PDU being acknowledged. This provides is a lock-step exchange of an Encapsulation message.

The Encap Type is the Type of the Encapsulation Message being acknowledged.

5.7. KEEPALIVE - Layer 2 Liveness

LSOE devices MUST exchange occasional Layer 2 KEEPALIVE messages to ensure association continuity.

They SHOULD be exchanged at a configured frequency. One per second is the default. Layer 3 liveness, such as BFD, will likely be more aggressive.



6. Layers 2.5 and 3 Liveness

Ethernet liveness is continuously tested by KEEPALIVE messages, see Section 5.7. As layer 2.5 or layer 3 connectivity could still break, liveness above layer 2 SHOULD be frequently tested using BFD ([RFC5880]) or a similar technique.

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

7. The North/South Protocol

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

The nodes know the unique node identifiers (ASNs, RouterIDs, ...) and Encapsulations on each link interface.

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

Therefore the node identifiers, 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.

7.1. Use BGP-LS as Much as Possible

BGP-LS [RFC7752] defines BGP-like PDUs describing 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.

7.2. 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 ID's described in Section 5.4. 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.

8. Discussion

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

8.1. HELLO Discussion

There is the question of whether to allow an intermediate switch to be transparent to discovery. We consider that an interface on a device is a Layer 2 or a Layer 3 interface. In theory it could be a Layer 3 interface with no encapsulation or Layer 3 addressing currently configured.

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 interface, I, on an LSOE 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, interfaces MUST continue to send HELLOs as long as they are turned up.

8.2. HELLO versus KEEPALIVE

Both HELLO and KEEPALIVE are periodic. KEEPALIVE might be eliminated in favor of keeping only HELLOs. But currently KEEPALIVE is unicast, has a checksum, is acknowledged, and thus more firmly verifies association existence.

This warrants discussion.

8.3. Open Issues

VLANs/SVIs/Subinterfaces

9. Security Considerations

The protocol as is MUST NOT be used outside a datacenter or similarly closed environment due to lack of formal definition of the authentication and authorisation mechanism. These will be worked on in a later effort, likely using credentials configured using ZTP.

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.

It is generally unwise to assume that on the wire Ethernet 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.

Malicious nodes/devices could mis-announce addressing, form malicious associations, etc.

For these reasons, the OPEN message's authentication data exchange SHOULD be used. [A mandatory to implement authentication is in development.]

10. IANA Considerations

This document requests the IANA create a registry for LSOE Message Type, which may range from 0 to 255. The name of the registry should be LSOE-Message-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:

Message Code	Message Name
0	HELLO
1	OPEN
2	KEEPALIVE
3	Encapsulation ACK
4	IPv4 Announce / Withdraw
5	IPv6 Announce / Withdraw
6	MPLS IPv4 Announce / Withdraw
7	MPLS IPv6 Announce / Withdraw
8-255	Reserved

This document requests the IANA create a registry for LSOE Flag Bits, which may range from 0 to 7. The name of the registry should be LSOE-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	One of a multi-Frame sequence
1	last of a multi-Frame sequence
2-7	Reserved

This document requests the IANA create a registry for LSOE PL Flag Bits, which may range from 0 to 7. The name of the registry should be LSOE-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	Primary
1	Loopback
2-7	Reserved

11. IEEE Considerations

This document needs a new EtherType.

12. Acknowledgments

The authors thank Cristel Pelsser for multiple reviews, Jeff Haas for review and comments, Joe Clarke for a useful review, John Scudder deeply serious review and comments, Larry Kreeger for a lot of layer 2 clue, Martijn Schmidt for his contribution, Rob Austein for reviews

and checksum code, Russ Housley for checksum discussion and sBox, and Steve Bellovin for checksum advice.

13. References

13.1. Normative References

[I-D.ietf-idr-bgp-ls-segment-routing-ext]
Previdi, S., Talaulikar, K., Filsfils, C., Gredler, H., and M. Chen, "BGP Link-State extensions for Segment Routing", draft-ietf-idr-bgp-ls-segment-routing-ext-08 (work in progress), May 2018.

[I-D.ietf-idr-bgpls-segment-routing-epe]
Previdi, S., Filsfils, C., Patel, K., Ray, S., and J. Dong, "BGP-LS extensions for Segment Routing BGP Egress Peer Engineering", draft-ietf-idr-bgpls-segment-routing-epe-15 (work in progress), March 2018.

[I-D.ietf-lsvr-bgp-spf]
Patel, K., Lindem, A., Zandi, S., and W. Henderickx, "Shortest Path Routing Extensions for BGP Protocol", draft-ietf-lsvr-bgp-spf-02 (work in progress), August 2018.

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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc4271>.

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

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <http://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, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc8174>.

13.2. Informative References

[JUPITER] Singh, A., Germano, P., Kanagala, A., Liu, H., Provost, J., Simmons, J., Tanda, E., Wanderer, J., HAP.lzle, U., Stuart, S., Vahdat, A., Ong, J., Agarwal, A., Anderson, G., Armistead, A., Bannon, R., Boving, S., Desai, G., and B. Felderman, "Jupiter rising", Communications of the ACM Vol. 59, pp. 88-97, DOI 10.1145/2975159, August 2016.

Authors' Addresses

Randy Bush
Arrcus & IIJ
5147 Crystal Springs
Bainbridge Island, WA 98110
United States of America

Email: randy@psg.com

Keyur Patel
Arrcus
2077 Gateway Place, Suite #250
San Jose, CA 95119
United States of America

Email: keyur@arrcus.com