IEEE P802.1ABdh Update to LSVP
(Note: P802.1ABdh == LLDPv2)

IETF-105

Montreal

Paul Congdon (Tallac Networks)

Paul Bottorff (Aruba)

July 24, 2019

Disclaimer

- This presentation should be considered as the personal view of the presenter not as a formal position, explanation, or interpretation of IEEE.

- Per IEEE-SA Standards Board Bylaws, December 2017
  - “At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position of IEEE.”

Background - References

- Initial LLDPv2 proposal - presented in January 2019
  - http://www.ieee802.org/1/files/public/docs2019/new-congdon-lldpv2-consideration-0119-v01.pdf
- Evaluation of LLDPv2 proposal against LSVR requirements – Presented at IEEE 802.1 Interim in January 2019
  - http://www.ieee802.org/1/files/public/docs2019/new-congdon-lsvr-disco-requirements-for-LLDPv2-0119-v01.pdf
- IEEE 802.1 WG responded to an LSVR liaison in March 2019
  - http://www.ieee802.org/1/files/public/docs2019/liaison-response-IETF-LSVR-Link-State-over-Ethernet-0319-v01.pdf
- Most recent technical proposal was presented in July 2019
  - http://www.ieee802.org/1/files/public/docs2019/dh-bottorff-alt-0719-v03.pdf
- IEEE 802.1 WG approved forwarding the new project request in July 2019
  - http://www.ieee802.org/1/files/public/minutes/2019-07-closing-plenary-slides-v6.pdf

Discovery protocols running wild…

- New IETF work on Link State Vector Routing (lsvr) has resulting in development of a discovery protocol called Layer 3 Data Link (l3dl) also IETF bgp group has a contribution for neighbor discovery protocol
  - The lsvr draft in progress draft-ietf-lsvr-l3dl-02
  - The idr contribution draft-xu-idr-neighbor-autodiscovery-11
- Work recently completed at IEEE on extensions to Virtual Station Interface Discovery and Configuration Protocol (VDP, 802.1Q-2018 clauses 40, 41, and 43) extends VDP to cover IP addressing for split NVE of NVO3 (802.1Qcy-2019)
- Work in progress at IEEE on Auto Attach (P802.1Qcj) which is currently described for Provider Backbone Bridges
  - Open source for LLDP auto attach is at: https://github.com/auto-attach/aa-lldpd
  - Provides discovery of VID to I-SID mapping for BEBs attaching to servers

Time to revise/update LLDP

- LLDP is widely deployed and people have (ab)used its facilities for many years, in many different environments
- The number of Vendor and Organizational specific TLV definitions continues to rise
- A new IEEE project on LLDPv2 (802.1ABdh) is underway
  - Propose to extend LLDP to scale existing LLDP applications and add enhancements for router (i.e. LSVR) and Time Sensitive Networking (TSN) applications
  - The LLDPv2 project will be an Amendment to 802.1AB-2016 (P802.1ABdh)
  - The LLDPv2 project will allow transmission of LLDPv2 TLV databases to consist of multiple frames
  - LLDPv2 and LLDPv1 nodes will interoperate as though they were LLDPv1 with a single frame database
  - LLDPv2 should be sufficient to fill most discovery needs without the additional protocols

Objectives for a new LLDPv2 method

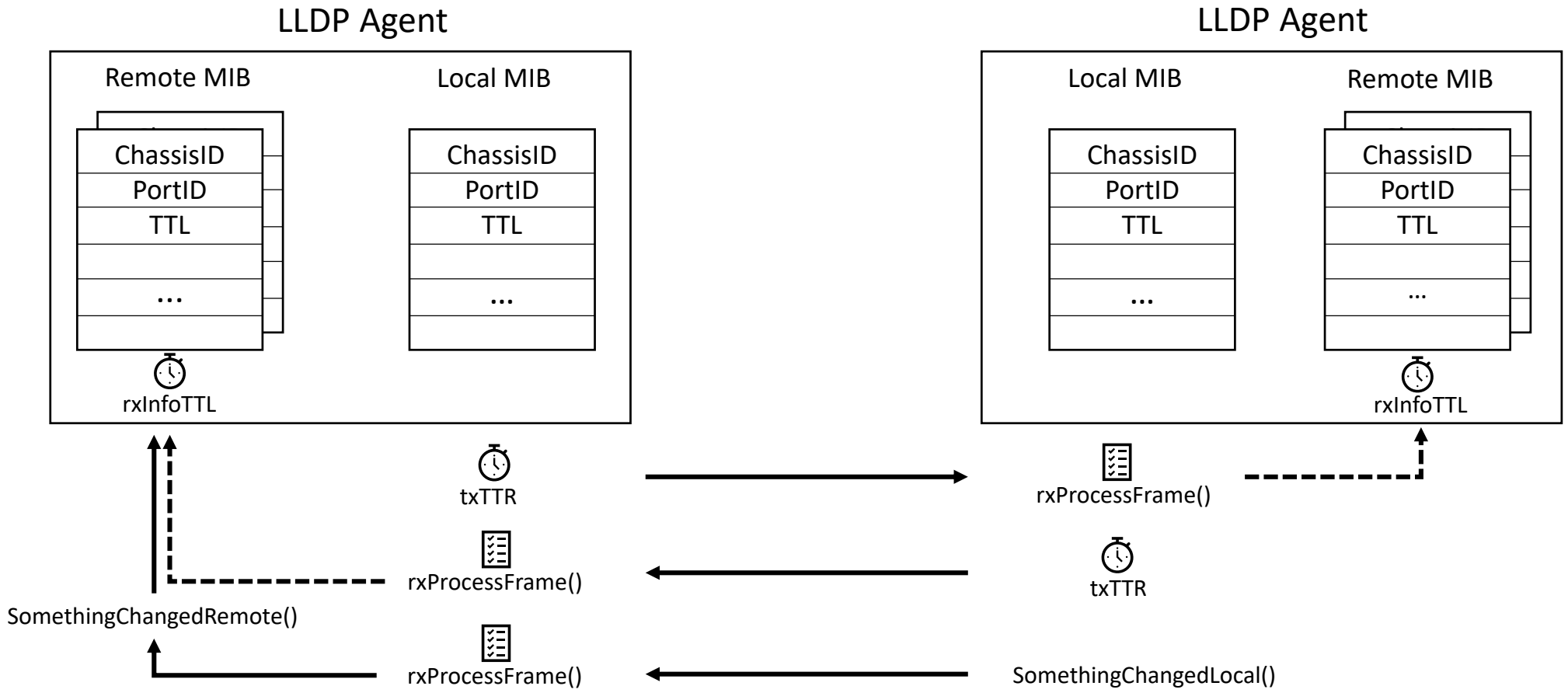
- Transparent to current users of LLDP, they just have a larger database of TLVs
- Support LLDP TLV databases larger than a single frame
  - IETF needs discovery database sizes around 64K octets
- Support the ability to limit the LLDP frame size to meet timing constraints imposed by some TSN applications
  - Do we need to split TLVs over multiple PDUs?
- Support the ability to communicate with an LLDPv1 implementation
  - Only the first LLDPDU would be exchanged between an LLDPv1 and LLDPv2 implementation
- Support shared media, however, optimize for point-to-point links

- Ensure the integrity of the full set of TLVs received by a peer
  - This can be useful in v1 implementations as well
  - NOTE: Do we also need to provide a means to authenticate the LLDP database? The IETF has this requirement.

Objectives for a new LLDPv2 method (cont)

- Support pacing of frames to receivers to prevent overloading low-level network firmware
  - Historically OSPF and IS-IS have had problems from lack of flow and congestion management
- Reduce network traffic by reducing periodic transmission to the minimum
  - Only update the base LLDPv1 PDU periodically (same as LLDP today)
  - Extension PDUs are only updated on demand from receivers
  - Update other PDUs only when they have changed
- Other optimizations and considerations which may be useful
  - Minimize computational load on LLDPv2 receivers to update and validate the database
  - Supporting larger TLVs
  - TLVs spanning multiple extension PDUs
  - Database authentication, is high want for IETF and other applications
    - Use a separate authentication extension TLV or secure other ways?
    - Address key exchange requirements

Current LLDP operation reminder



NOTE: Think of the Remote and Local MIBs as a database that must fit into a single PDU
Replace all values of the Remote MIB with contents of LLDPDU when something changes

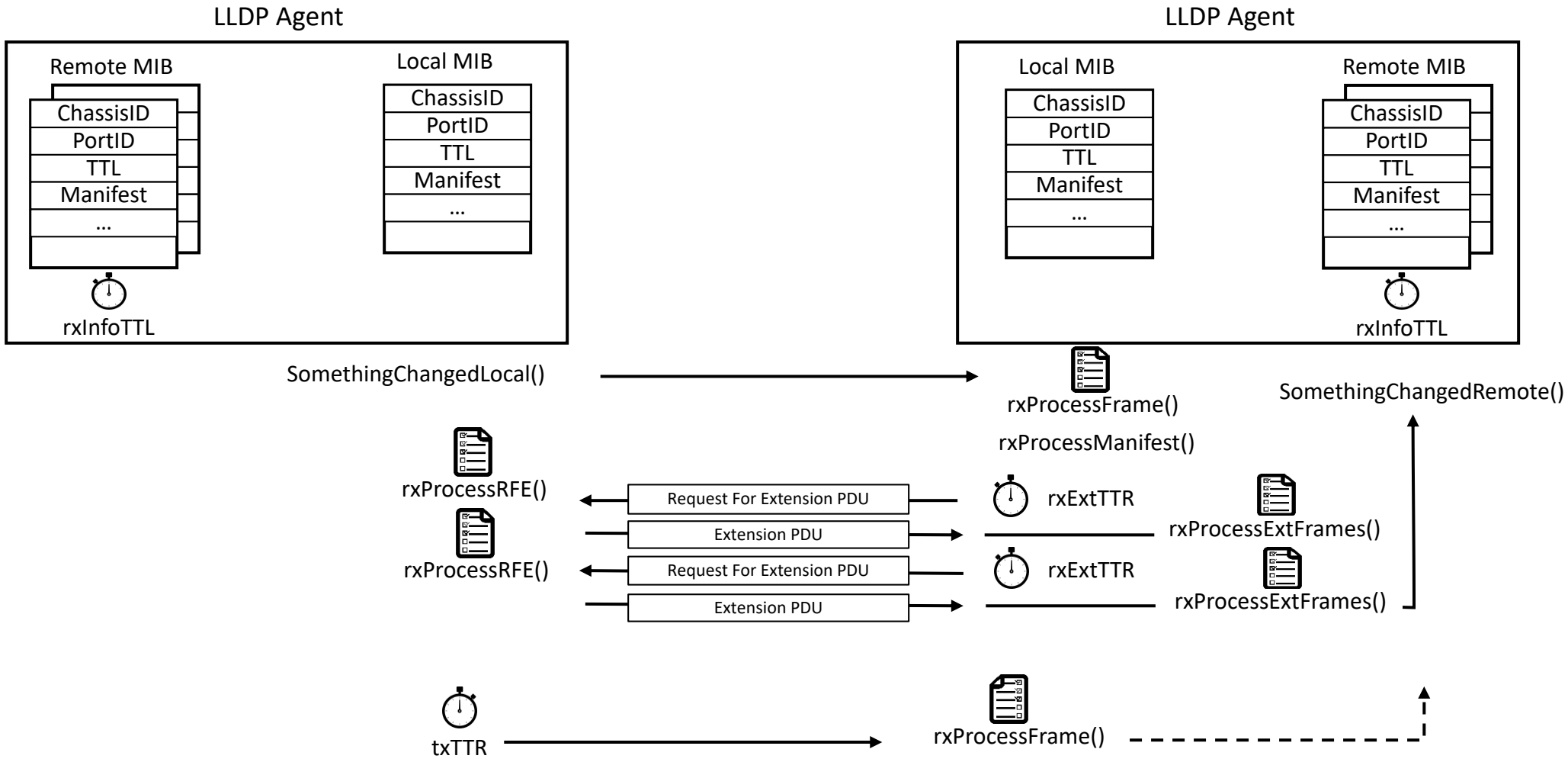
Proposal

- Define the current LLDPv1 PDU as the base PDU
  - Size not to exceed a single frame (current LLDPDU is a single frame)
  - LLDPv1 implementations can receive and interpret the base PDU
  - Define an LLDPv2 extension PDU as an extension to the database
  - Describe the PDU by an LLDPv2 manifest
  - The extension to the database is exchanged by a set of PDUs identified by the LLDPv2 manifest
  - An LLDPv2 manifest is encoded in a manifest TLV
  - If no manifest TLV is present in the base PDU then no extensions to the database exist
  - The upper limit to the number of frames is determined by the LLDPv1 TLV size limit (512 bytes) and the format of the manifest
- The manifest TLV defines:
  - A way to uniquely identify each frame in the extension to the database
- Transmission of extension PDUs is controlled by the receiver using explicit requests to the sender
  - Extension PDUs are transmitted from the source LLDPv2 Agent to a **unicast** destination determined by the request
  - An LLDPv2 receiving agent may only have a single extension PDU request pending at any point in time
  - Each extension PDU requests may ask for as many PDUs as desired (i.e. receiver pacing).

Proposal (cont)

- The new LLDPv2 PDUs will be ignored by LLDPv1
  - Since the extension PDUs are unicast, LLDPv1 may filter out LLDPv2 by destination address
  - In addition, an alternate EtherType may be used for LLDPv2 to guarantee frames are never directed to LLDPv1
- Each extension PDU needs to have a mandatory format:
  - Each extension PDU contains the first two mandatory TLVs of a LLDPv1 PDU (ChassisID + PortID)
  - Each extension PDU contains a new TLV that identifies the PDU
- A new Request for Extension (RFE) message is sent from receiving peer to load an extension PDU
  - Supports multiple peers on a shared media
  - Loading an extension PDU at the receiver LLDPv2 is at the systems discretion (i.e. receiver pacing)
  - The receiver only requests the out of date frames when it determines the current extension PDUs are out of date
  - Transmitters only periodically send the 1st LLDPv1 PDU
  - TTL in 1st PDU relates to all extension PDUs

Proposed LLDPv2 Operation: Receiver Pacing



NOTE: Send LLDPDUv1 as specified by LLDPv1 when something changes and periodically
Only send extension LLDPv2 PDUs when explicitly requested by an RFE
Only issue an RFE when manifest shows the local copy is out of date

Summary of proposed LLDPv2 support for L3DL exchange

Information Exchanged	Supportable by an LLDP TLV and Protocol	Comments
My LLEI	☑	Can be defined by an IETF Organizational Specific TLV
Attribute List	☑	Can be defined by an IETF Organizational Specific TLV
Authentication Data	☑	Should fit within current LLDP TLV length restrictions
Encapsulation & Addresses	☑	Must be split across multiple TLVs & PDUs
Keepalives	☒	Frequency may not be appropriate for LLDP
Acks	☑	Implicit part of LLDPv2 proposal
Vendor Extensions	☑	LLDP supports Vendor and Organizational Specific TLVs

What if LLDPv2 were used by LSVR?

What needs to be done?

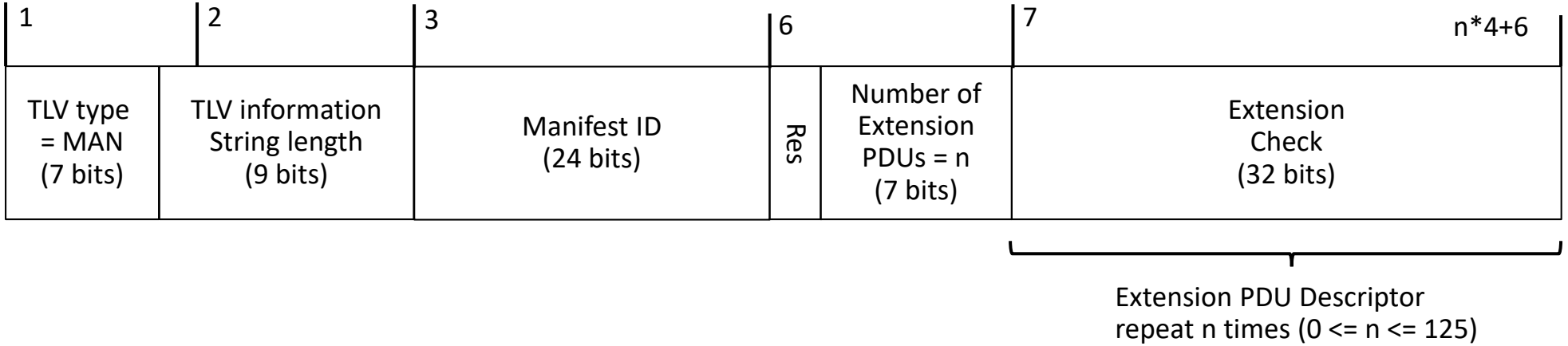
- Write a draft to define IETF Organizational Specific TLVs for LLDPv2
  - My LLEI
  - Attribute list
  - Encapsulations and Addresses
  - Authentication data
- Consider alternative solutions for L2 Liveness
  - Connectivity Fault Management (CFM) specified in Clauses 18-22 of IEEE Std 802.1Q-2018
- Provide technical input and review during P802.1ABdh

LLDPv2 Project (P802.1ABdh) Next Steps

- Continued technical contributions
  - Would love to have an Open Source implementation for evaluation
- Evaluation of changes to IEEE Std IEEE 802.1AB-2016 (aka LLDP)
- Initial draft by an individual contributor
- Formalization of project by IEEE New Standards Committee (NESCOM)
  - Expected on September 27th

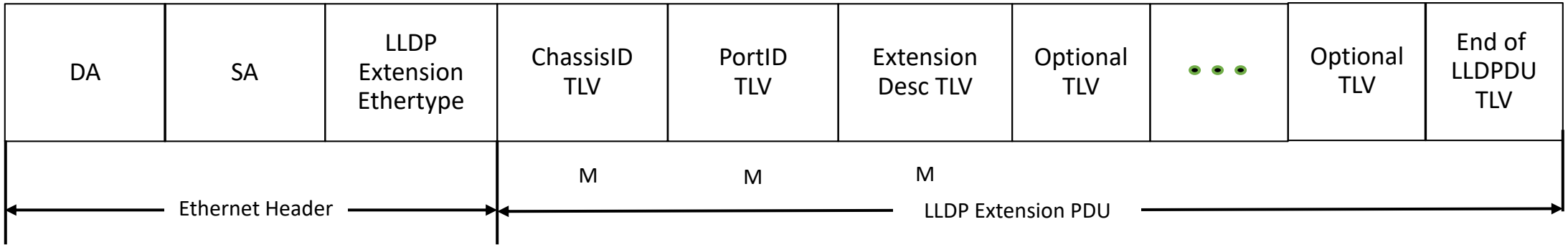
Backup - Details

Example Manifest TLV: Added to LLDPv1 Database



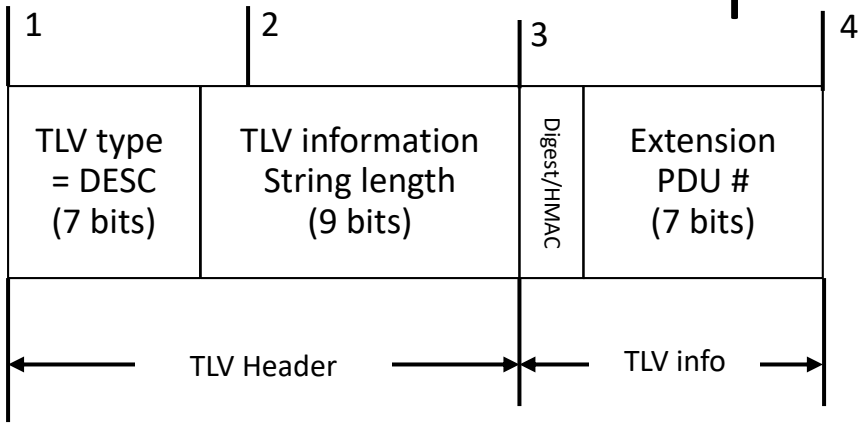
- Manifest ID identifies the extension database (for present a single constant chosen by committee (i.e. an CID, 0x1, etc))
  - This may be used in the future for determining if a receiver wishes to load the extension database
- Number of extension PDUs indicates the number of valid PDU descriptors in the manifest
  - Some implementations may fix the manifest TLV size however load it with a variable number of PDUs
- Each Extension PDU is identified by a:
  - PDU number which is implied by the index to the location of the PDU Descriptor
  - PDU check contains a 32 bit check.
    - Use of the low order 32 bits of a MD5 hash of the frame appears sufficient
- Implicit encoding of the PDU number provides the smallest possible extension PDU descriptor allowing the largest possible extension database size
  - Since the PDU number is implicitly encoded inserting or deleting a PDU from the middle of the extension database is a relatively expensive operation.

Format for LLDPv2 Extension PDUs



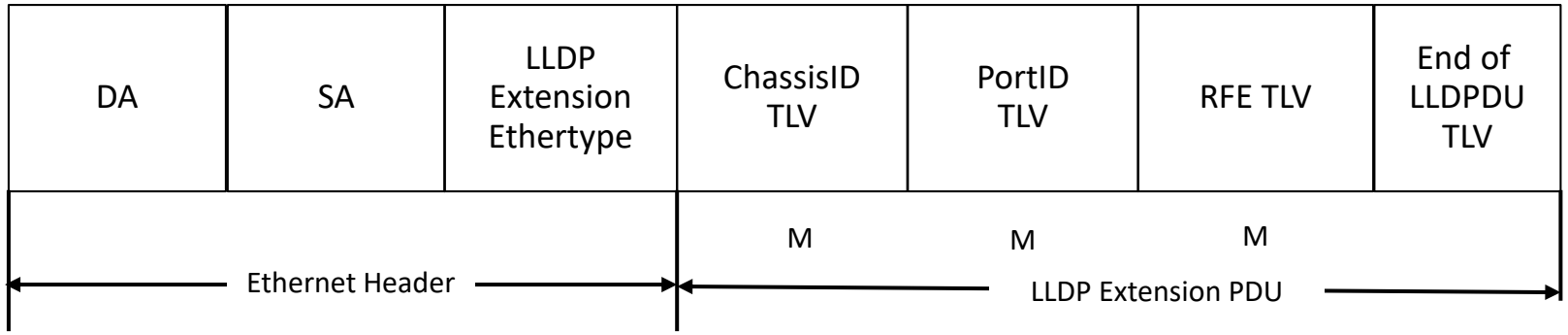
- LLDP Extension Ethertype
  - Since extensions are not multicast and only delivered on request no new Ethertype is required, though one could be used if desired
  - New Ethertype for Extension PDUs provides additional safety
- Chassis ID + Port ID are mandatory
  - Note TTL from 1st PDU should apply and is not needed here
- Extension Description TLV is mandatory
  - Identifies this Extension PDU, the PDU revision, and the PDU hash

Extension Description TLV



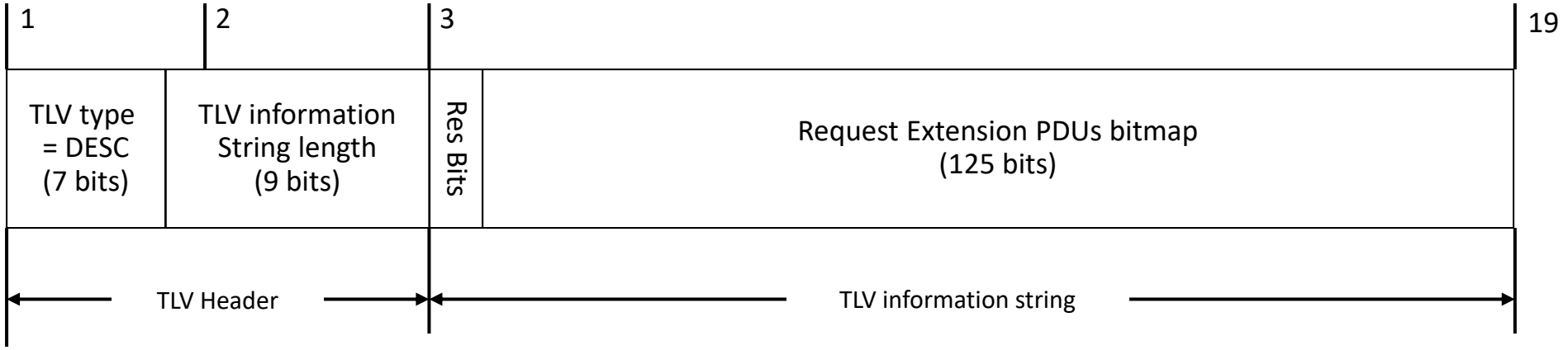
- Extension PDU # is the designation number for this PDU
  - The PDU number is in the range from 1 – 126

Request For Extension PDUs



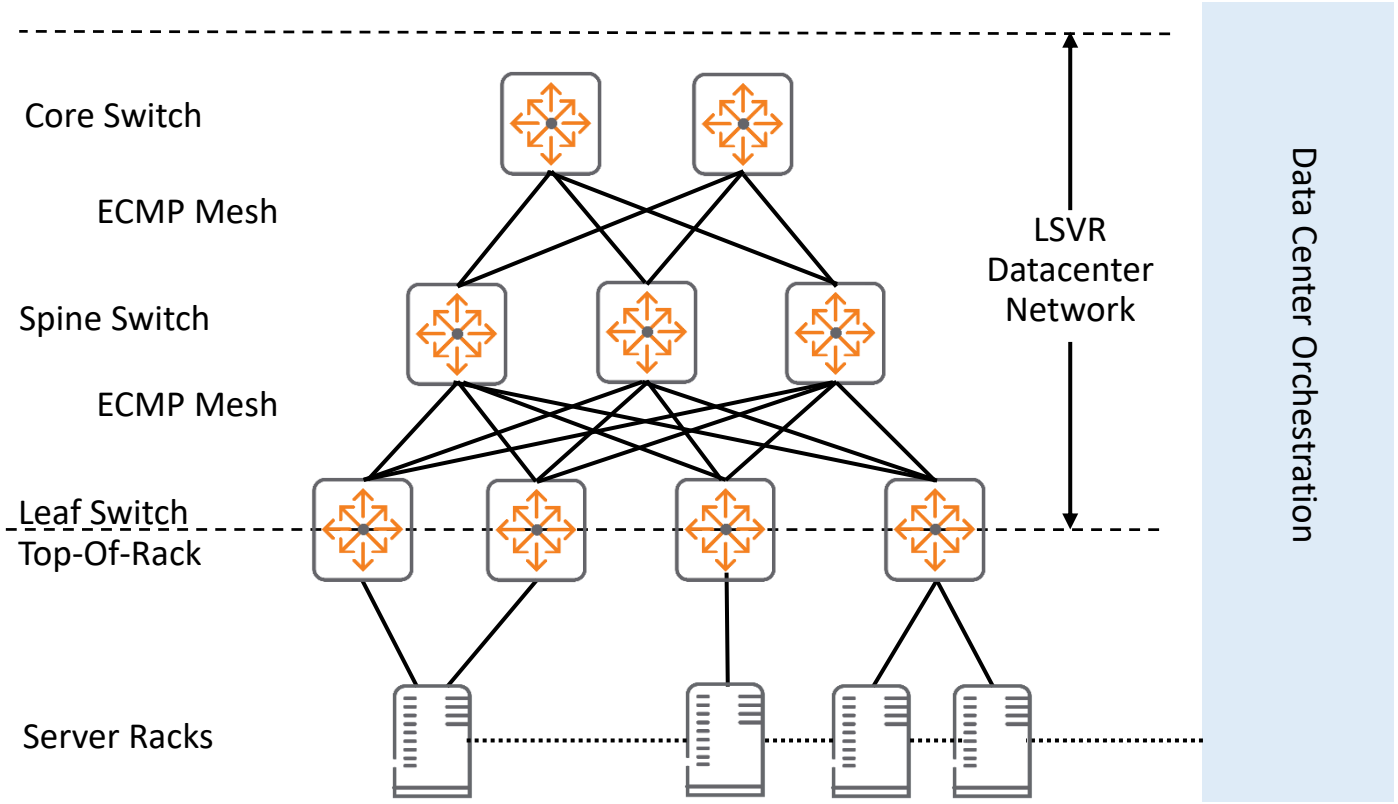
- Since these are unicast to the source of the base database they would only arrive at an LLDPv1 agent as a result of a bug
  - Using a new Ethertype will prevent a transmission error for corrupting an LLDPv1 database
- ChasssisID and PortID are mandatory
- Request for Extension PDU TLVs
  - Identifies extension PDUs that need to be set by peer

Request Extension PDUs TLV



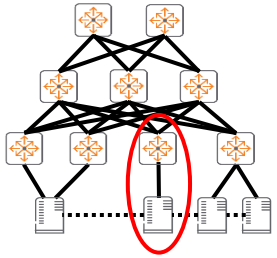
- Request for extension PDUs
  - Multiple RFEs may be used to pace the frames at the receiver by withholding RFEs
  - A single RFE may request multiple frames if the receiver has sufficient buffer for them
- Extension LLDPDUs are not multicast, instead they are unicast frames
  - The frames are sent to the SA address within the RFE PDU
  - On a shared media each individual LLDP Agent must provide independent requests for extension frames
  - This allows the individual receivers to pace PDUs at rates that match their ability to handle the reception
  - Since LLDPv2 Extension PDUs are unicast they will not interfere with LLDPv1 implementations which will never issue RFEs

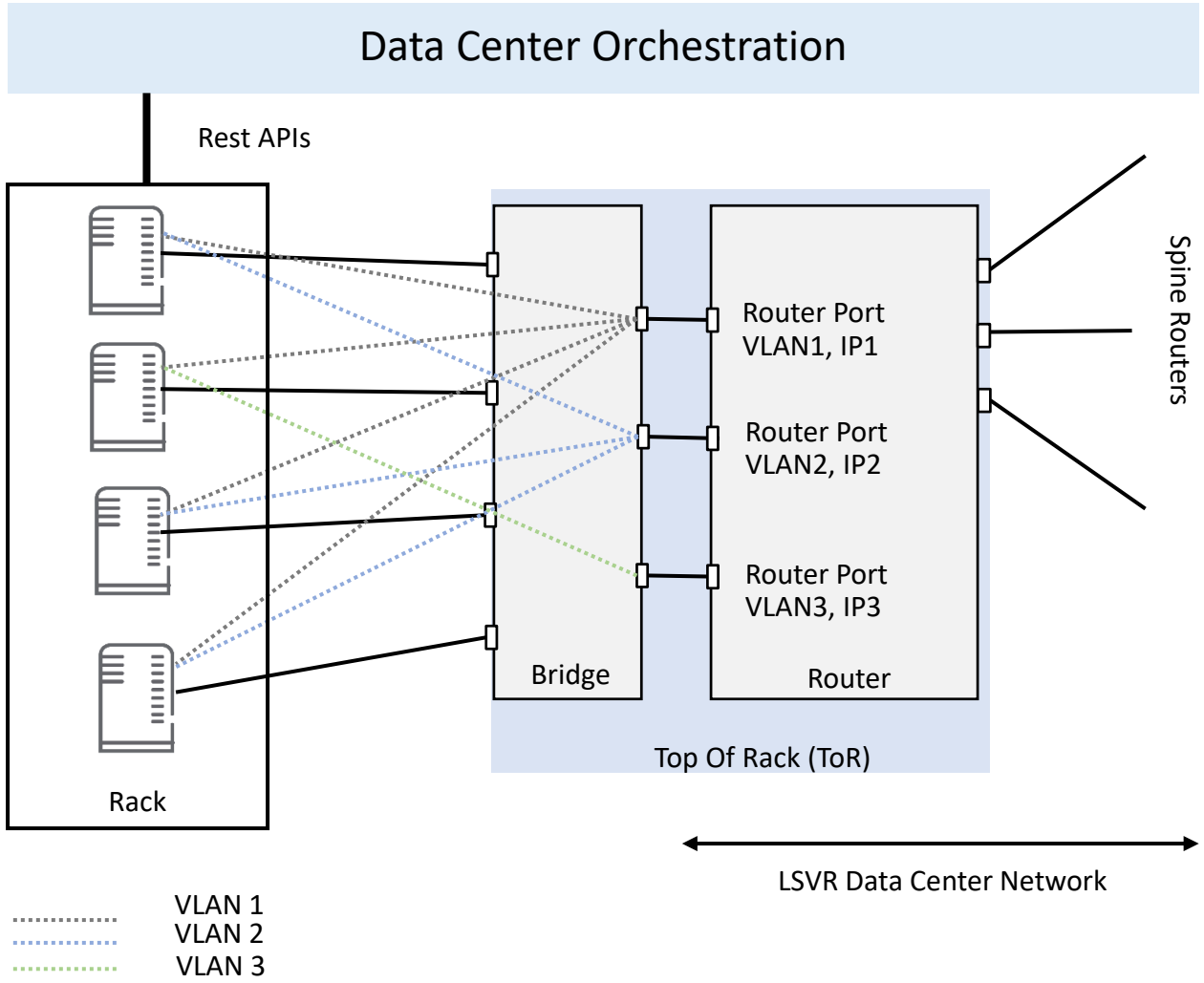
Datacenter Network Using LSVR



- Most datacenters are configured as 2-3 layer Clos networks using ECMP for distribution over the mesh and LAGs/M-LAGs for server attachment
- Typically these networks provide an IPv4/IPv6 topology organized with ToR and Spine switches within Pods (around 8-128 racks)
- Servers at the network edge manage virtual and tenant networks which are encapsulated into the IP packets for transmission over the data center
- The orchestrator controls the creation of the virtual and tenant networks along with coupling to services

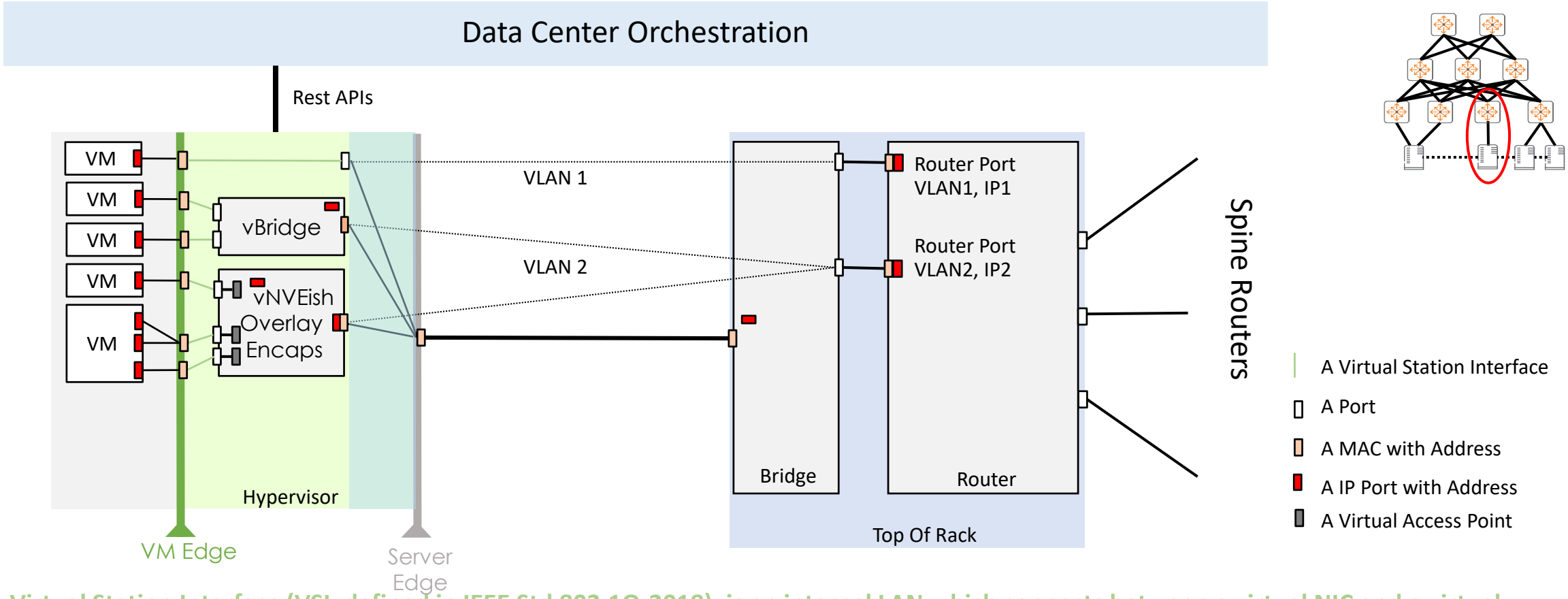
Typical Server and Switch Rack Configuration





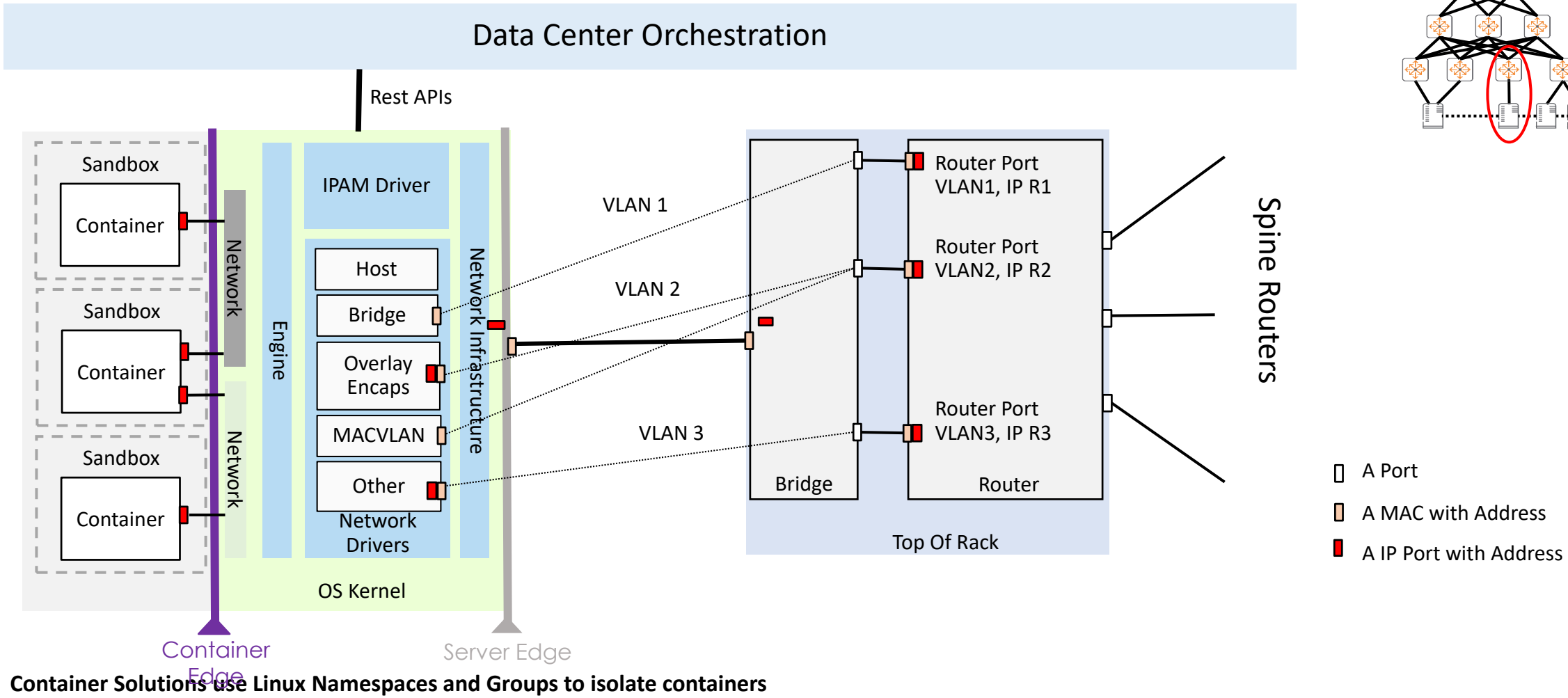
- Here the Bridge portion of the Top Of Rack Switch couples physical ports to each server in the rack
- Over the Bridge Ports VLANs are distributed to each server
- For each VLAN within the rack an IP subnet is assigned
- Each router port in the Top Of Rack is coupled to a single VLAN which is mapped onto an IP subnet
- Protocols within the switch (in this case LSVR) advertise the subnets available within the rack to the rest of the network

Server Network Interfaces – Virtual Machines (i.e. VMWare)



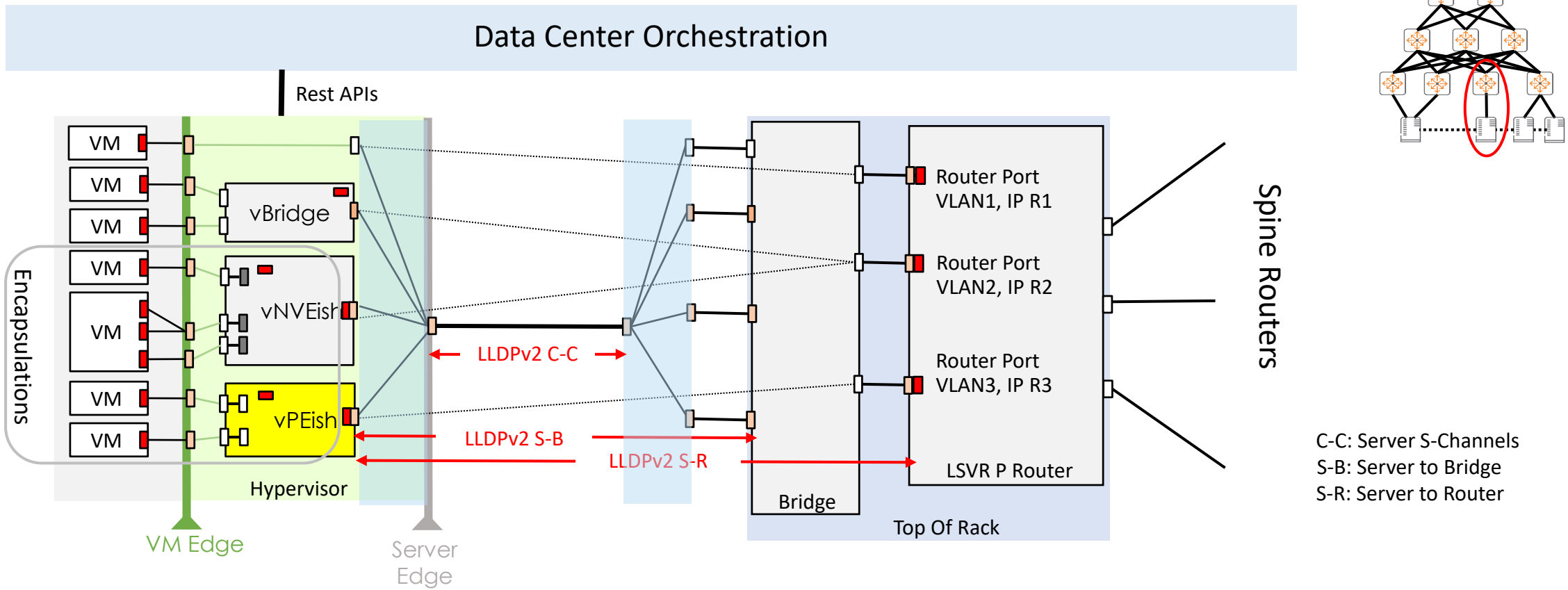
- Virtual Station Interface (VSI, defined in IEEE Std 802.1Q-2018): is an internal LAN which connects between a virtual NIC and a virtual Bridge Port
- Virtual Access Point (VAP): A logical connection point on the Network Virtualization Edge (NVE) for connecting a Tenant System to a virtual network
- **DC network is a simple IP underlay network. For scaling L3 encapsulations are supported using "NVE like" procedures within the server controlled by Data Center Orchestration**

Server Network Interfaces – Containers (i.e. Docker)



- **Container Solutions use Linux Namespaces and Groups to isolate containers**
- **These solutions provide a variety of network connections, though use an overlay for large scale datacenters**
- **DC network is a simple IP network. For scaling L3 encapsulations are supported using "NVE like" procedures within the server controlled by Data Center Orchestration**

Discover Protocol Termination Points for LLDPv2



- **Currently LLDPv2 is specified to operate at two levels within a Server. These are between the Server and the adjacent Top Of Rack switch (S-B) and over an S-Channel to a Virtual Edge (PE-B).**
- **The IETF L3DL protocol is specified to operate between end system ports (PR-R). LLDPv2 could also take this path by choosing a destination MAC that passes through Bridges rather than contained at Bridges**
- **For the typical case where there are no other Bridges except those embedded in the Server and ToR it is un-necessary to pass LLDP through the Bridge layer. Instead, the Router control plane just needs an API to the LLDPv2 database.**