Protocol for Extending LLDP

Paul Bottorff
Paul.Bottorff@hpe.com

Discovery Protocols

- 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-00
  - 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
- Proposed new IEEE project on LLDPv2 (802.1ABdh)
  - Purposal is to extend LLDP to scale existing LLDP applications and add enhancements for router and TSN applications
  - The LLDPv2 project will be an amendment of 802.1AB-2016 (P802.1ABdh)
  - The LLDPv2 project will allow LLDPv2 databases consisting 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

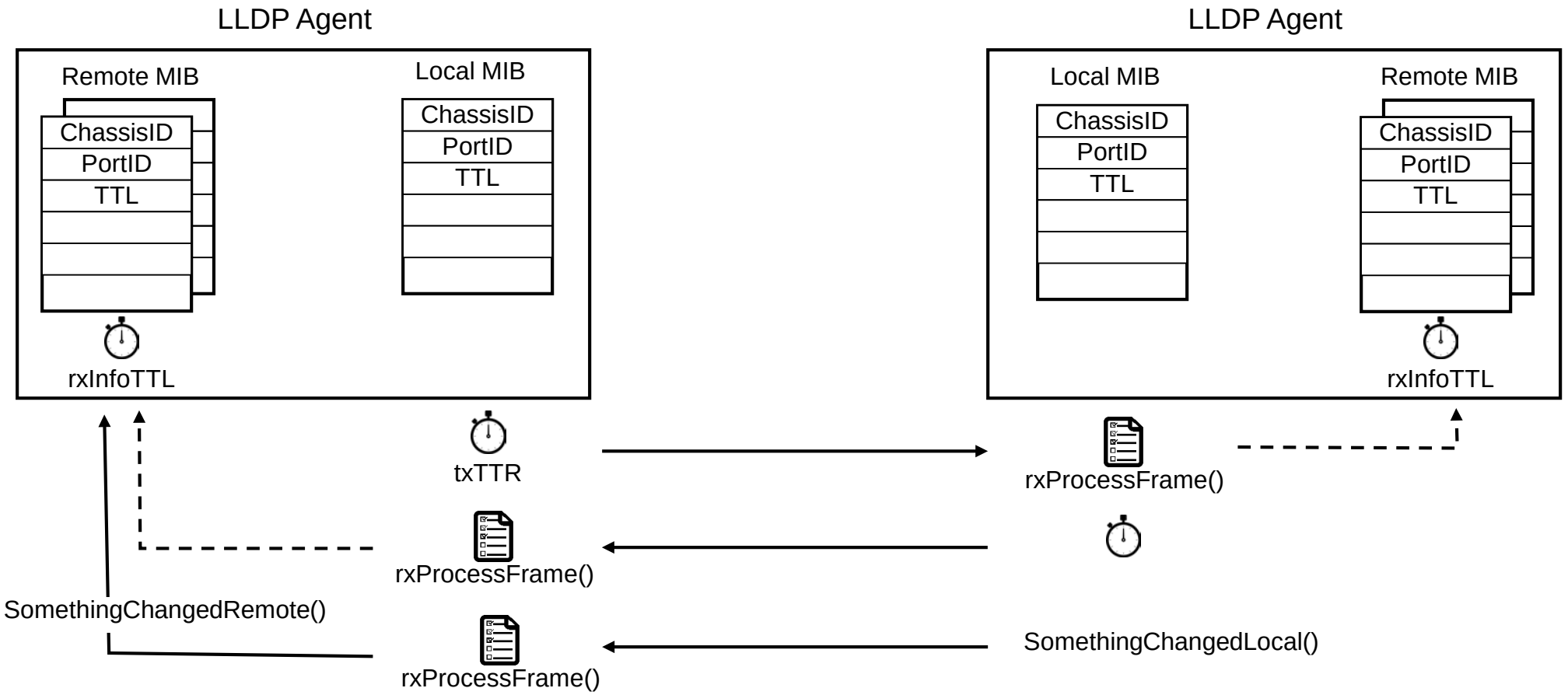
Objectives for New LLDPv2 Method

- Support LLDP databases larger than a single frame
  - IETF is working on 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 (base) LLDPDU would be exchanged between and LLDPv1 and LLDPv2 implementation
- Support shared media, this proposal optimizes for point-to-point though allows shared
  - Both for the base database and extension database PDUs
- Ensure the integrity of the full set of TLVs is received by partner
  - This can be useful in v1 implementations as well
  - Do we also need to provide a means to authenticate the LLDP database? The IETF has this requirement.

Objectives for New LLDPv2 Method

- 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
  - 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
  - Computational load requirements for LLDPv2 receivers to update and validate the database?
  - Multiple manifests, feature could allow application specific manifests
  - Larger TLVs, using multiple TLVs appears sufficient?
  - TLVs spanning multiple extension database PDUs, is this required for TSN?
  - Database authentication, is high want for IETF and other applications
    - Part of base standard or separate authentication extensions
    - Key exchange requirements

Current LLDP Operation



NOTE: Remote and Local MIBs are databases that must fit within a single frame length PDU
Replace all values of the Remote MIB with contents of LLDPDU when something changes

aruba
a Hewlett Packard Enterprise company

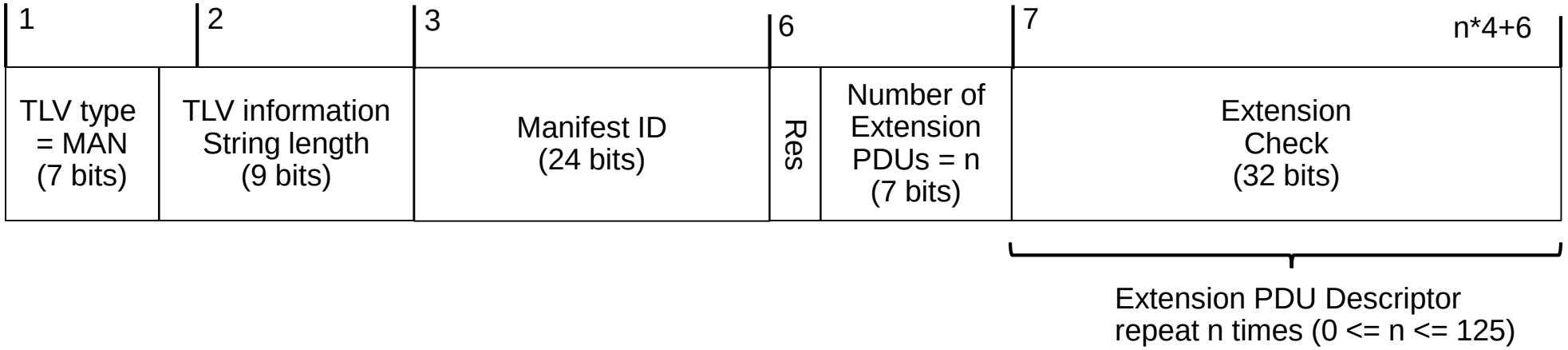
Proposal

- Define the current LLDPv1 database as the base database
  - Has a size not to exceed a single frame (an LLDPDU is a single frame)
  - The base database is exchanged as a single LLDPDUv1
  - Define an LLDPv2 extension database as a database of size 1-n frames described by an LLDPv2 manifest
  - An extension database is exchanged by a set of PDUs identified by the LLDPv2 manifest
  - An LLDPv2 manifest is encoded in an manifest TLV
  - If no manifest TLV is present in the base database then no extension databases exist
  - The upper limit to the number of frames is determined by the LLDPv1 TLV size limit (512) and the format of the manifest
    - Note: In the case of TSN the manifest TLV size may be limited by the max allowed frame size
- The manifest TLV defines:
  - A way to uniquely identify each frame in the extension 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
  - The receiving agent may use multiple extension PDU requests to pace frame reception

Proposal Continued

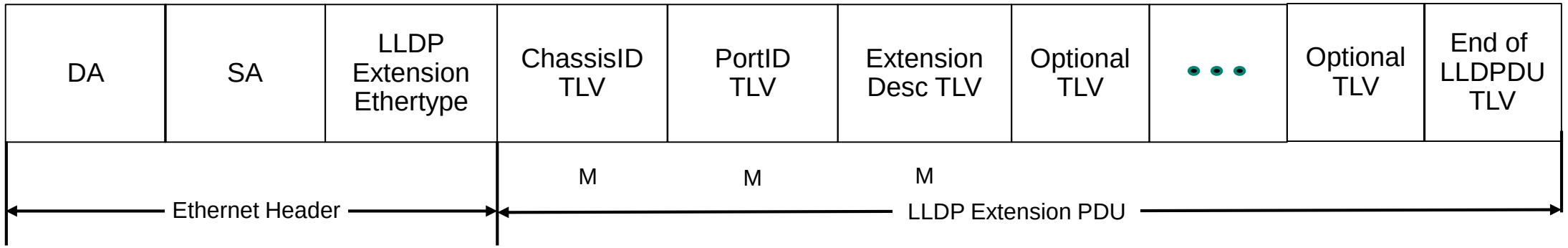
- The new LLDPDUv2s will be ignored by LLDPv1
  - Since the extension PDUs are unicast, LLDPv1 may filter out LLDPv2 by destination address
  - In addition, an alternate Ethertype may be used for LLDPv2 to guarantee frames are never directed to LLDPv1
- Each extension PDU needs to have a mandatory format:
  - Each extension PDU contains the first two mandatory TLVs of a LLDPDUv1 (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 database
  - Support multiple peers on a shared media
  - Loading an extension database at the receiver LLDPv2 is at the systems discretion
  - Extension databases are not multicast and are loaded based on receiver paced RFE message
  - The receiver only loads the out of date frames of an extension database of interest when it determines it's current database is out of date as determined by the manifest TLV
  - Transmitters only periodically send the 1st LLDPv1 PDU
  - TTL in 1st PDU relates to all extension PDUs
- Options
  - Multiple Extensions databases could be supported

Example Manifest TLV: Added to LLDPv1 Database



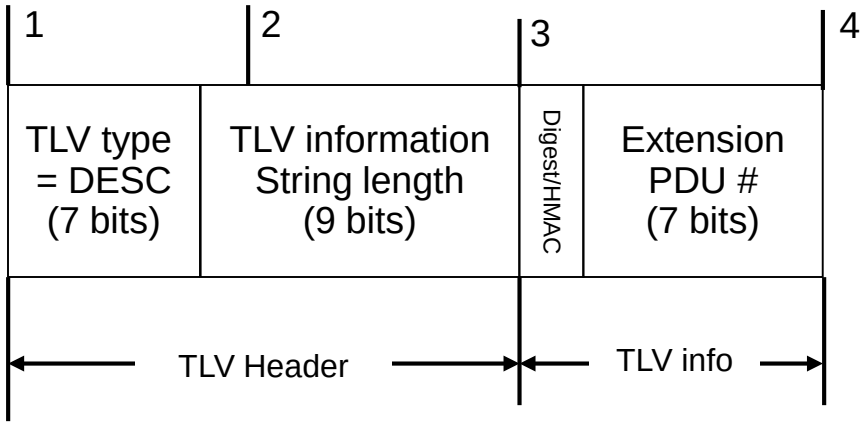
- Manifest ID identifies the extension database (for present a single constant chosen by committee (i.e. an CID, 0x1, etc))
  - This may be used in the future for determining if a receiver wishes to load the extension database
- Number of extension PDUs indicates the number of valid PDU descriptors in 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.

Extension PDU Format



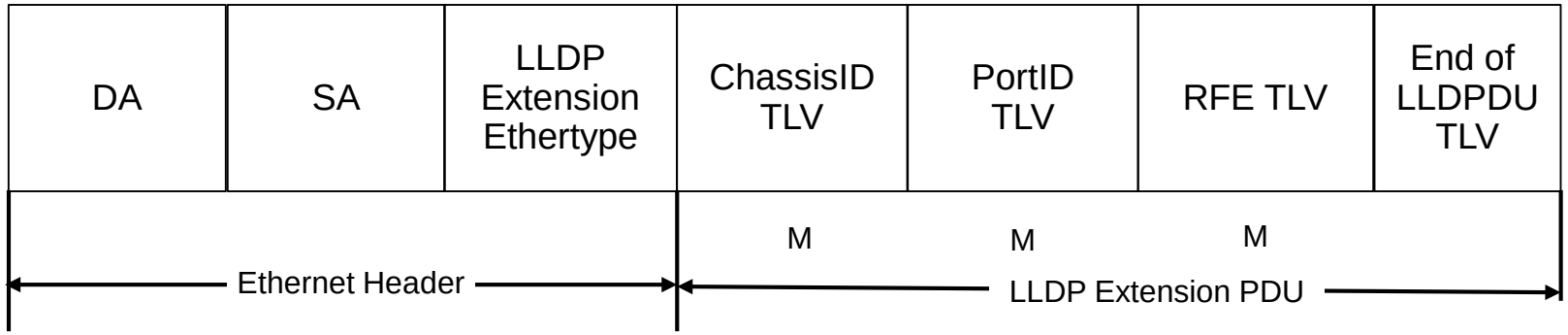
– LLDP Extension Ethertype
  – Since extensions are not multicast and only delivered on request no new Ethertype is required, though one could be used if desired
– Chassis ID + Port ID are mandatory
  – Note TTL from 1st PDU should apply and is not needed here
– Extension Description TLV is mandatory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

Extension Description TLV



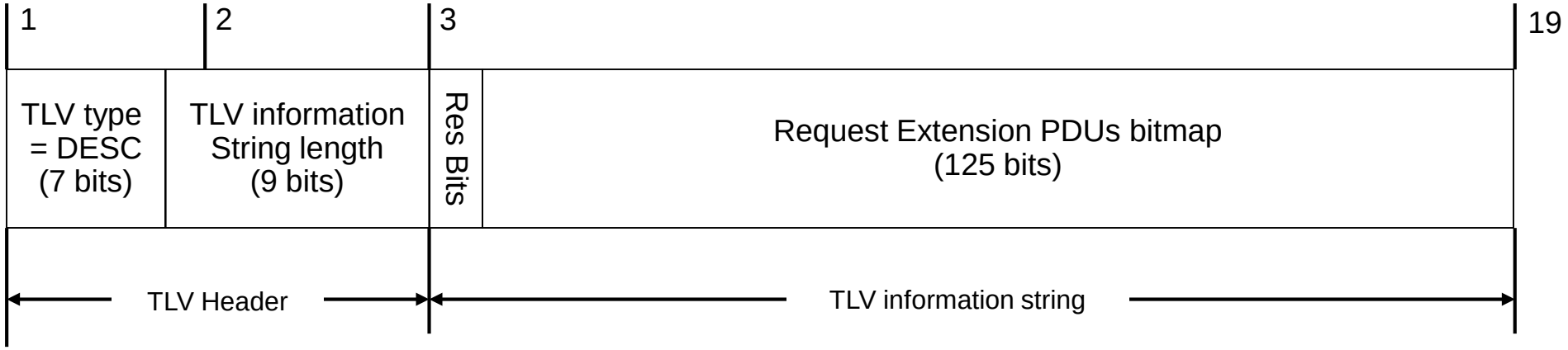
- Extension PDU # is the designation number for this PDU
  - The PDU number is in the range from 1 – 126
- Extension PDU Hash
  - The check is computed over all TLVs within the PDU including the extension description TLV

Request For Extension PDUs



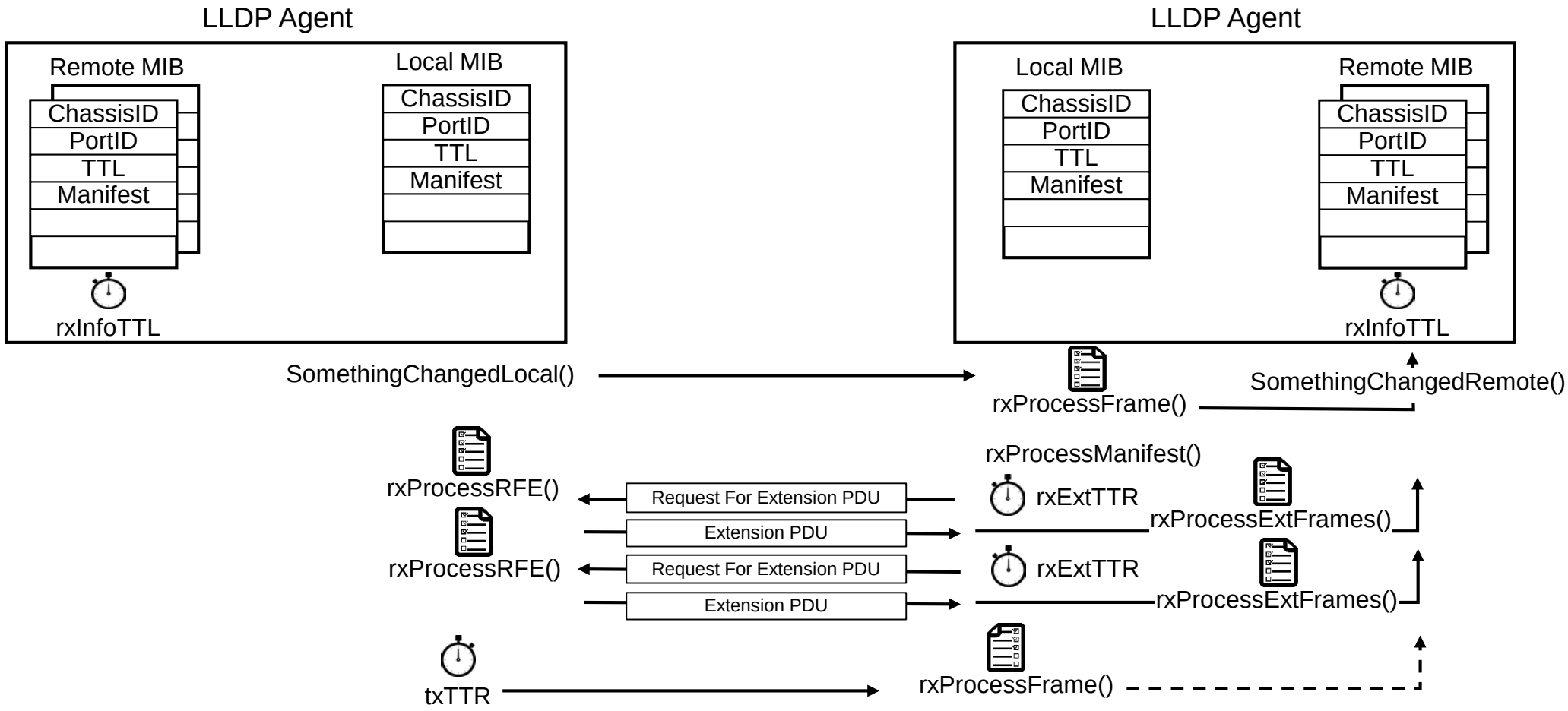
– Since these are unicast to the source of the base database they would only arrive at an LLDPv1 agent as a result of a bug
  – Using a new Ethertype will prevent a transmission error for corrupting an LLDPv1 database
– ChasssisID and PortID are mandatory
– Request for Extension PDU TLVs
  – Identifies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

LLDPv2 Operation Proposal: Receiver Pacing



NOTE: Send LLDPDUv1 as specified by LLDPv1 when something changes and periodically
Only send extension LLDPDUv2s when explicitly requested by a RFE
Only issue RFE when manifest shows the local copy is out of date

aruba
a Hewlett Packard Enterprise company

Summary

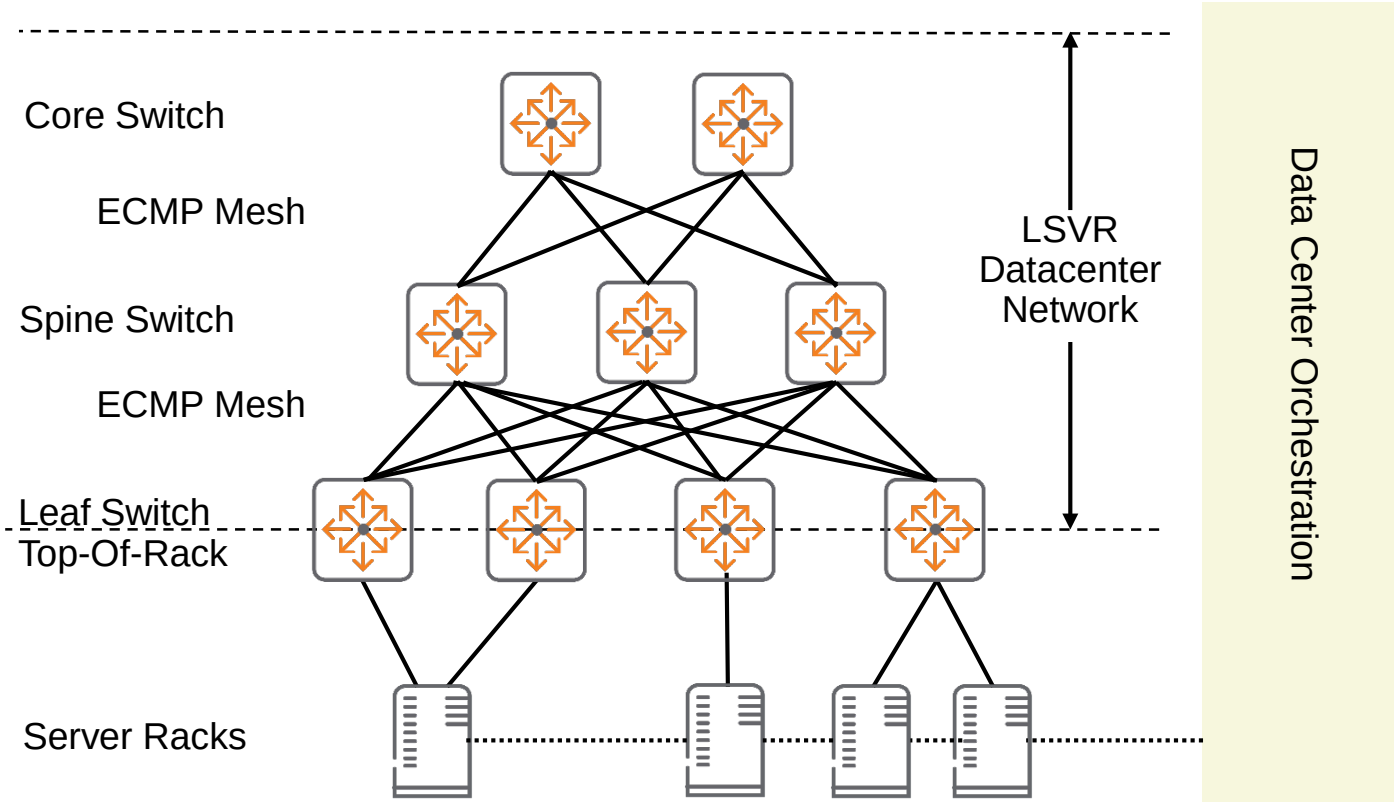
- Since the Manifest TLV implicitly includes the PDU number a 32 bit PDU check code is sufficient to validate the PDU revision
  - Using a 32 bit identifier allows up to 125 frames in the manifest
- Adding an identifier to the manifest TLV provides for specializing manifests to specific applications such as router discovery, TSN configuration, bridge configuration
- Using a unicast protocol optimizes extension updates for point-to-point, however allows for both point-to-point and shared media
  - The unicast protocol allows receiver pacing for frame reception
  - The receiver may also control re-transmission for reliable delivery of extension frames
- Extension frame updates are only required when the current revision is out of date
  - The base LLDPv1 database which includes the Manifest TLV is updated periodically and whenever something changes
  - If any extension PDU changes then a change will occur in the Manifest TLV in the base LLDPv1 database
  - LLDPv2 can determine if an extension TLV needs to be updated by comparing the manifest check code with the current database

Thank You

aruba
a Hewlett Packard Enterprise company

Backup Slides

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

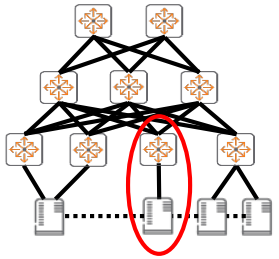
– Typically these networks provide an IPv4/IPv6 topology organized with ToR and Spine switches within Pods (around 8-128 racks)

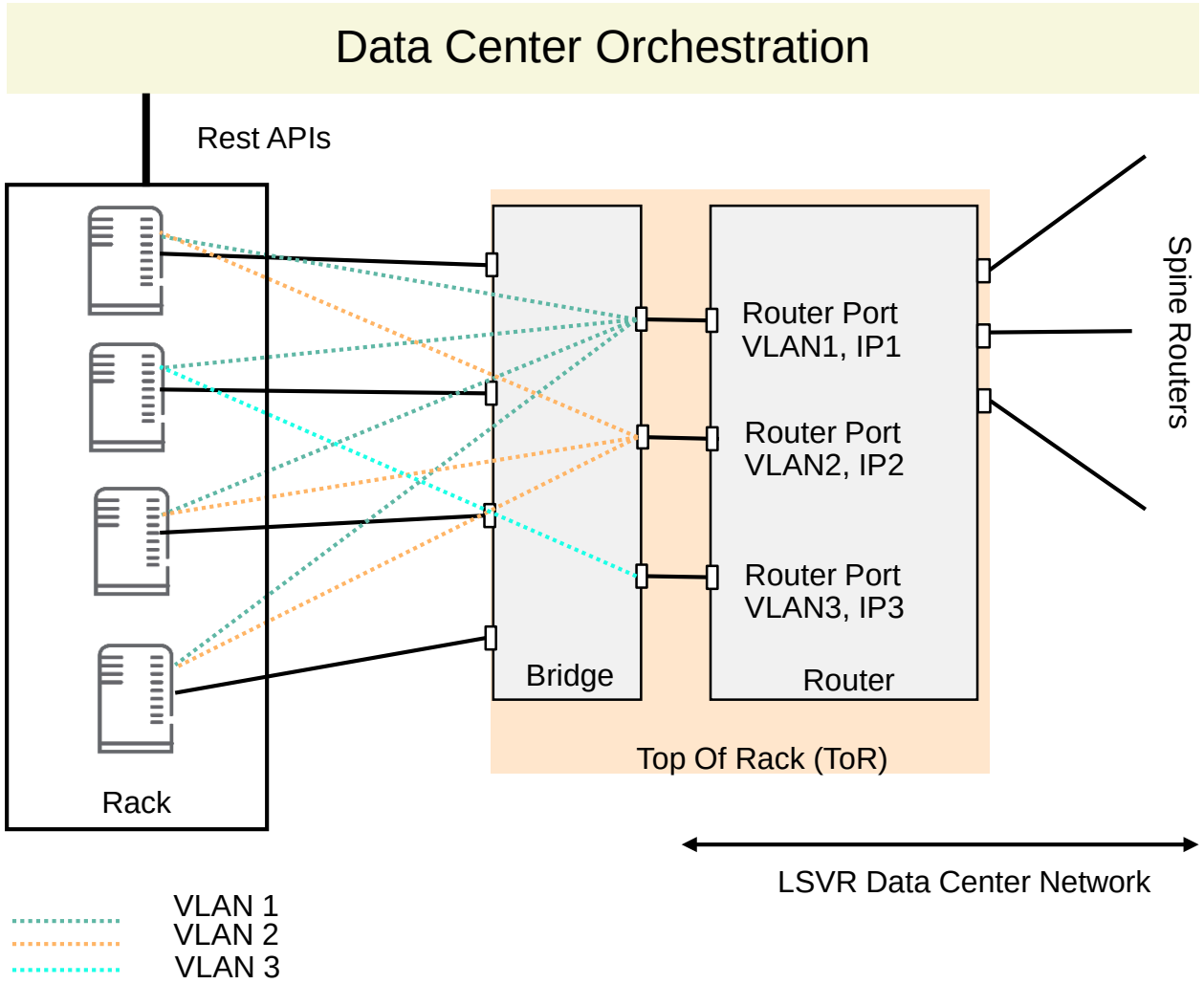
– Servers at the network edge manage virtual and tenant networks which are encapsulated into the IP packets for transmission over the data center

– The orchestrator controls the creation of the virtual and tenant networks along with coupling to services

aruba
a Hewlett Packard Enterprise company

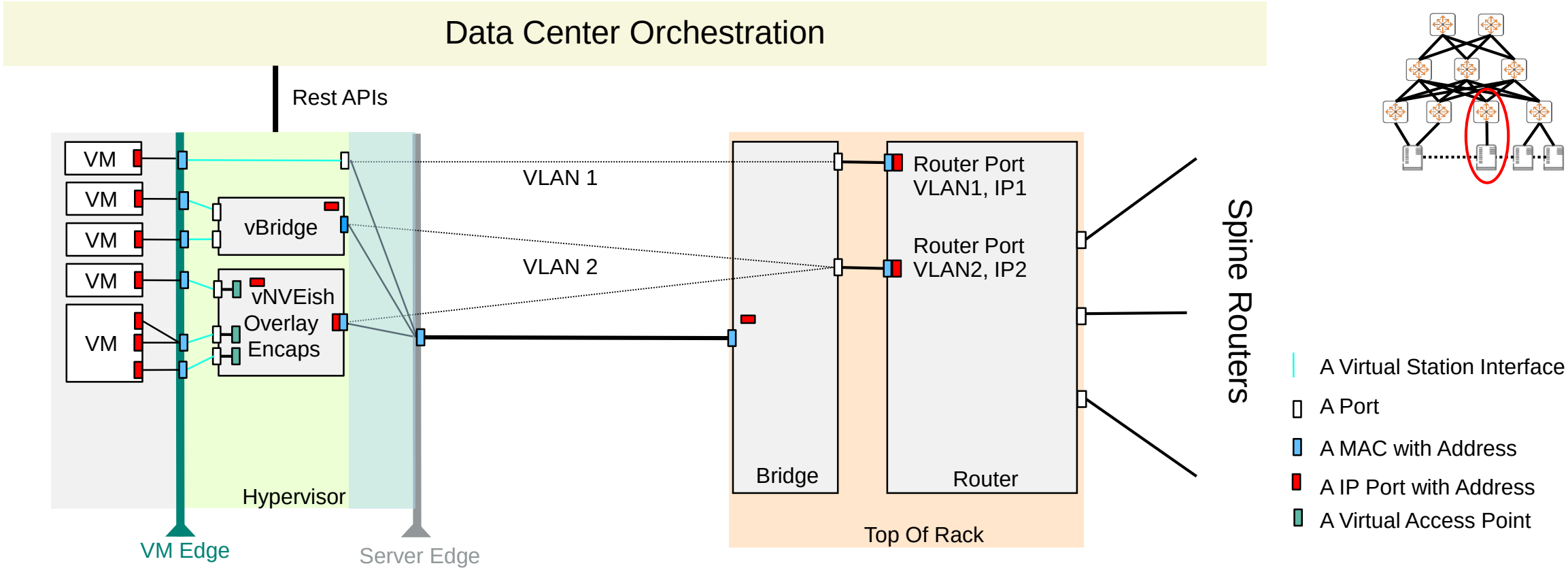
Typical Server and Switch Rack Configuration





- 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

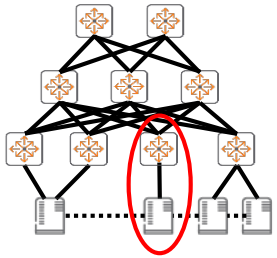
Server Network Interfaces – Virtual Machines (i.e. VMWare)

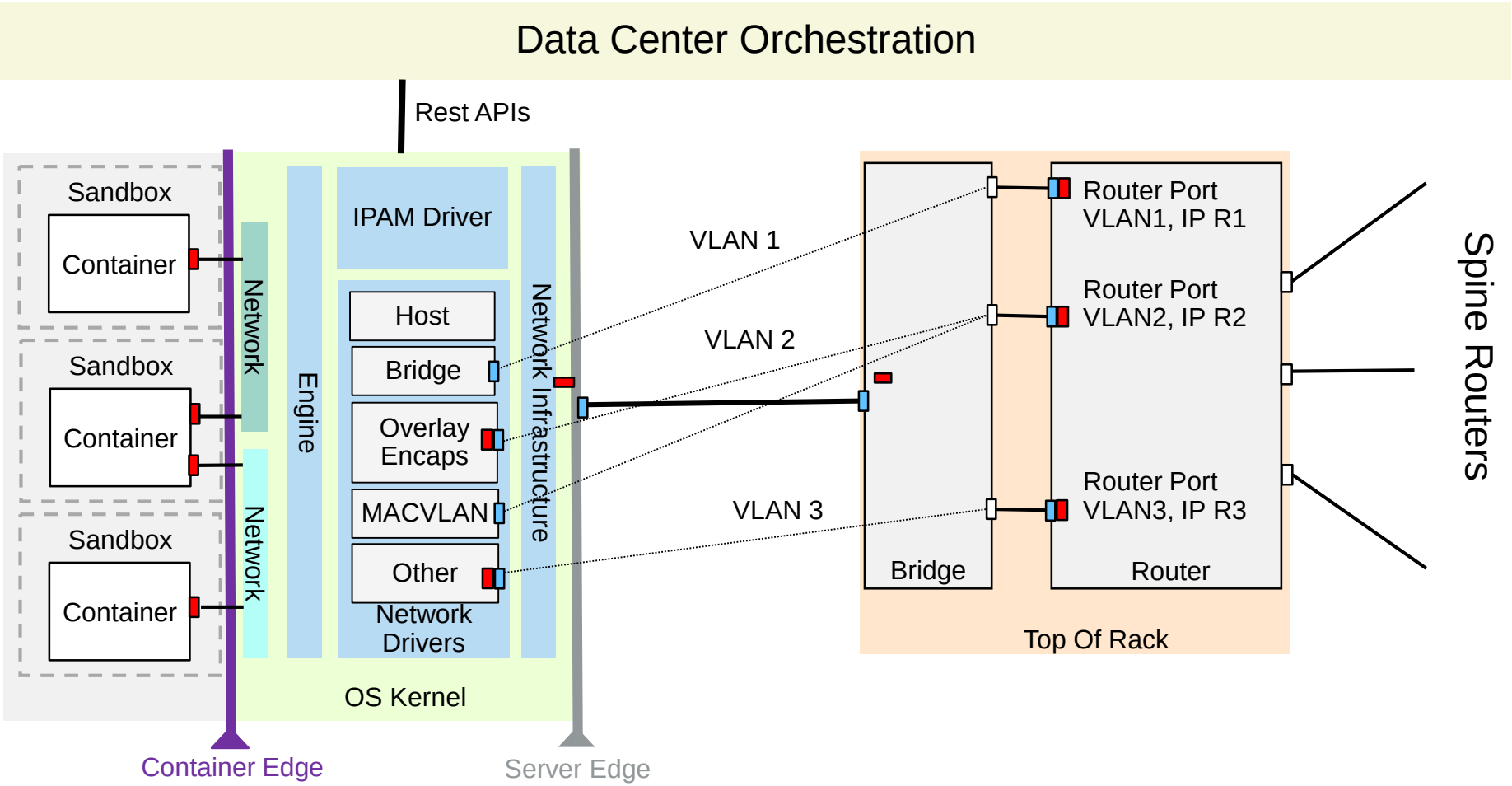


- **Virtual Station Interface (VSI, defined in IEEE Std 802.1Q-2018): is an internal LAN which connects between a virtual NIC and a virtual Bridge Port**
- **Virtual Access Point (VAP): A logical connection point on the Network Virtualization Edge (NVE) for connecting a Tenant System to a virtual network**
- **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**

aruba
a Hewlett Packard Enterprise company

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

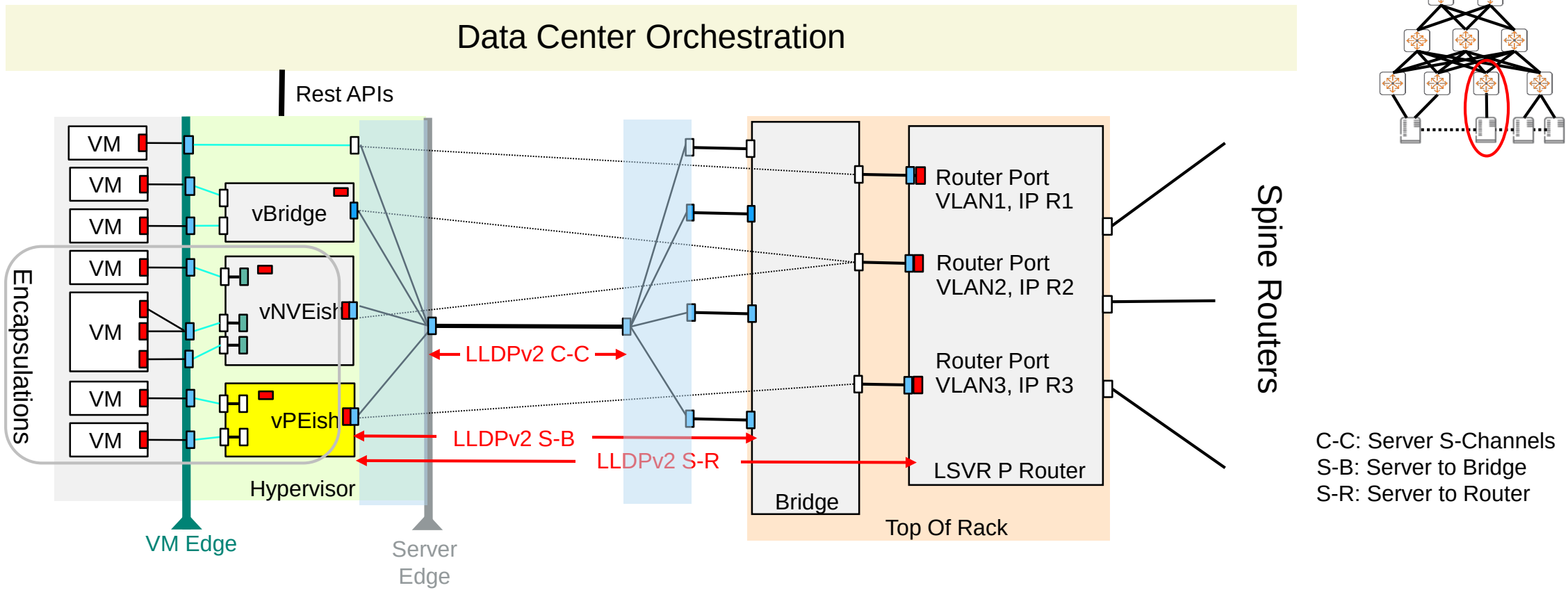




- **Container Solutions use Linux Namespaces and Groups to isolate containers**
- **These solutions provide a variety of network connections, though use an overlay for large scale datacenters**
- **DC network is a simple IP network. For scaling L3 encapsulations are supported using "NVE like" procedures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

aruba
a Hewlett Packard Enterprise company

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.**