Ethernet Virtual Bridging Automation Use Cases

Renato Recio, Sivakumar Krishnasamy and Rakesh Sharma

***Abstract*— Managing the Ethernet switches is a complex task in today's Data Centers, as a lot of network state[A] is manually created, applied and maintained. The complexity increases further with server virtualization, where a single switch port can connect 10s of Virtual Machines to the network. These issues multiply when a virtual machine (VM) migrates from one server node to another, as the network states are not migrated along with VM.**

This paper will describe several use cases for managing network state in today's networks. It will then describe how protocol contributions to the IEEE 802.1Qbg Edge Virtual Bridging (EVB) working group can be used to enable new, more automated network state management use cases.

***Index Terms*—Virtual Servers, Virtual Machines (VMs), Single-Root IO Virtualization (SR-IOV), Virtual Ethernet Bridging (VEB), Automated Port Profile Migration, Ethernet Virtual Bridging (EVB)**

I. Introduction

Today's Ethernet switches use network element management tools to manually apply and maintain network state to one or more ports on each switch. The network state includes VLAN Identifiers, port access controls, port traffic controls and possibly security flow control rules. The complexity increases further with server virtualization, because a single physical switch now connects multiple Virtual Machines (VMs), each with network state that must be maintained in the physical switch. These VMs may use virtual Ethernet switches, which also contain network state. Additionally as a VM migrates from one server to another, the network state associated with the VM must migrate with it.

This paper starts with a description of today's usage models for automating the application of network state to switches and the migration of that state as necessary to support VM migration. It will also describe some of the issues associated with today's approaches.

The paper provides a brief overview of proposed Ethernet Virtual Bridging (EVB) protocols from IEEE 802.1Qbg[1] contributors. This includes several options for how the proposed EVB protocols can be implemented. Finally, it describes use cases for how these proposed protocols enable automation of network state creation and maintenance.

[A] Network state represents one or more of the following: Access controls (e.g. IP address filters, MAC address filters), Quality of Service controls (e.g. rate limits) and/or security flow rules (e.g. Firewall and Intrusion Detection and Prevention, IDP, flow rules).

II. Today's EVB Use Cases

Virtual Ethernet Bridging (VEB[2,3,4]), a.k.a. Virtual Switching, has been around for decades (e.g. zVM, PowerVM). Virtual switching provides efficient VM to VM Communications. In most of today's Hypervisor VEB implementations (such as PowerVM's Virtual Switch, VMware's vSwitch and the Nexus 1000v[5,6]) the network state associated with a VM migrates with the VM. The network state may include: a VLAN Identifier, port access controls, port traffic controls and security flow controls. However, one of the major challenges with today's virtualization approaches is automating the association of external network state to a VM and migrating that network state during VM migration.

Following are the use cases that exist today, each with its own set of limitations.

A. *Homogeneous tenant network state*

Under this use case all VMs have the same network state. The typical usage model for this use case is one where all VMs are running the same tenant (a.k.a. application) type. For example, all VMs run the same Web Serving application.

Figure 1 below depicts an example of this use case. As shown in the figure a network manager is used to configure the same network state on each port used by servers to access the switch. All VMs are associated to the same network state (1) in the Physical Switch. Under this use case, when a VM migrates (2) below, no network state needs to migrate, because the network state associated with the VM is already resident in the external switch.

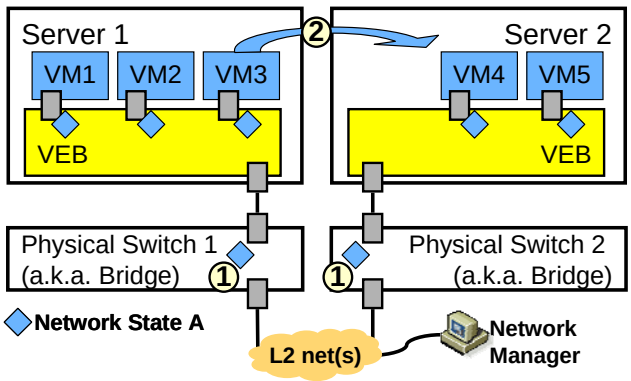

Figure 1: Homogeneous Tenant Use Case

For large enterprises, this use case can be applied to several racks of servers, where all servers are running the same application type. If one of those servers becomes over utilized, one or more of the VMs running in that server can migrate to another server.

The major limitation associated with this use case is that if the tenant type becomes over utilized and server hosting a different tenant type are under utilized, the VMs cannot migrate from the over utilized to the under utilized server, because each tenant type has different network state. For example, if servers hosting Web Serving VMs become over utilized, and servers hosting e-mail VMs are under utilized, the Web Serving VMs cannot be migrated to the server hosting the e-mail VMs, because their network state is not the same.

B. Post VM migration of heterogeneous tenant network state

Under this use case VMs host different tenant types. For example, some VMs run a Web Serving application and others an e-mail application. Network state is associated with each tenant (application) type and this network state resides in both the Hypervisor VEB[B] and the external switch. When a VM hosting a specific tenant type migrates, the associated network state must also migrate.

There are two approaches for how to satisfy this requirement today. The first is to use MAC Addresses to identify tenant types and automatically associate the network state to the MAC Address after migration. The second is to manually configure the network state prior to starting or migrating a VM, which is not automated. This section will describe the first approach.

Under this use case, the network manager is used to create network state that is unique for each tenant. The Hypervisor manager is used to manage (create, start-up, shut-down and destroy) the VMs, including the assignment of a MAC Address and VLAN Identifier for each virtual network interface used by the VM. Under this use case, the network manager creates network state for each tenant type. There are two possible ways to propogate the VM's tenant type, VLAN identifier and MAC addresses to the network manager:

1. Manual association
When a MAC Address is assigned to a VM, the VM's tenant type and MAC Address assignment along with associated VLAN identifier is made known to the network manager manually. The network manager then distributes this information to the access switches. When a VM begins communicating to an access switch, the switch automatically applies the network state associated with the VM's MAC Address.

Figure 2 below depicts an example of VM migration for this use case. In step 1, the network state is associated to the VM using the approach described above. In step 2, the Hypervisor performs the VM migration, including the network state migration for the Hypervisor's virtual switch. In step 3, when the VM begins to use the MAC Address on server 2, the switch will automatically apply the network state associated with that MAC Address.

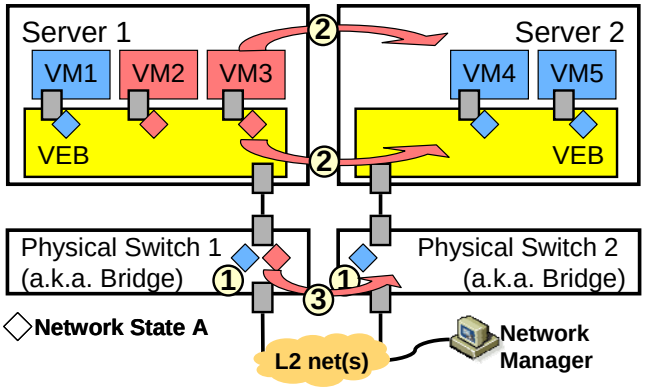

Figure 2: Post VM migration of heterogeneous tenant network state

2. Automatic association through an API
To automate the association and migration of network state to a VM's virtual network interface, this use case requires an API between the Hypervisor and network manager to communicate the: VM's tenant type, VM's MAC Addresses and the VLAN Identifier associated with each MAC Address. More importantly, without such an interface there is a risk that the switch will not be able to distinguish between a re-incarnated[C] and a migrated[D] MAC Address. This approach requires a per Hypervisor API investment.

In addition to the caveats above, this approach either requires switches to:

- Store the network state of all tenant types, which consumes switch memory resources;
- Allow the VM to communicate, while the switch retrieves the network state after a VM begins communication with a pre-configured MAC Address,

[B] VEB[4] - Virtual Ethernet Bridging, a.k.a. virtual switch (vSwitch).

[C] A re-incarnated MAC Address is one that was previously in use by a recently destroyed VM and is now in use by a different VM, which may require completely different network state.

[D] A migrated MAC Address is one that is associated with a VM that has been migrated across two physical servers in the fabric and retains the same network state association after the VM migration.

which creates an access control exposure window (i.e. where the VM can access the network without the switch applying the access/traffic controls defined by the network state associated with the VM tenant type); or

- Disallow the VM to communicate, while the switch retrieves the network state, resulting in lost VM packets, which can be detrimental for storage traffic.

C. *Open access for heterogeneous tenant network state*

Under this use case VMs host different tenant types and network state resides solely in the Hypervisor. That is, all physical access switch ports are configured as trunk ports and the Hypervisor's VEB is solely responsible for applying the necessary access and traffic controls.

The issue with this approach is all physical servers must be in the same security domain, which has the similar VM movement limitations as the "Homogeneous tenant network state" option. For example, a physical server cannot be managed by tenant A in the same layer-2 LAN as a physical server that is managed by tenant B.

D. *Summary of today's EVB[E] Use Cases*

Several use cases exist today for migrating the network state associated with a VM's virtual network interface. Unfortunately each of today's use cases comes with limitations that either cause security exposures (II.B and II.C) or migration restrictions (II.A)

III. Ethernet Virtual Bridging Principles of Operation

To automate the creation and migration of networking state, we need a control plane protocol that associates and de-associates a VM's virtual interface with state in nearest neighbor bridge adjacent to the physical network interface controller (NIC) used by the VM. This section provides a brief overview of the proposed EVB technologies used to provide this association. The final section of this paper will describe uses cases for how the EVB protocols automate the network configuration management.

A. *Defining network states and capabilities*

The Figure 3 depicts an example of a Server connected to an Adjacent Bridge (a.k.a. Switch). In this example the Server has two Virtual Machines (VMs), each VM has two virtual NICs (vNIC). Each vNIC connects to the Hypervisor virtual switch (vSwitch) through a Virtual Station Interface (VSI). The vSwitch can be implemented as a Virtual Ethernet Bridge (VEB) or as a Virtual Ethernet Port Aggregator (VEPA). A single Hypervisor instance may contain one or more such vSwitches. A VEB performs VSI to VSI packet forwarding, as well as VSI to adjacent bridge forwarding. A VEPA collaborates with an advanced bridge for all packet forwarding, including VSI to VSI packets. VEB and VEPA semantics are described in the version 0, Edge Virtual Bridge Proposal[7] to the IEEE 802.1Qbg PAR. Note, similar to the CEE Authors[8] version 0 Data Center Bridging capability eXchange (DCBX) protocol proposals[9,10,11], it is expected that vendors will implement the version 0, EVB proposal.

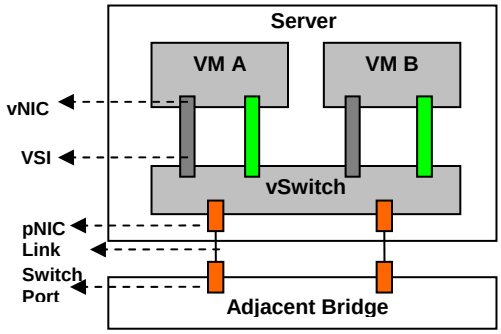

Figure 3: Virtual Station Components

The vSwitch shown in Figure 3 can also be implemented in physical NIC. An example of such an implementation is provided in Figure 4 below, which depicts a Single Root Input/Output Virtualization NIC. In this example, the SR-IOV [F]NIC supports one physical function (PFs) and a set of virtual functions (VFs). Each of these functions is connected to the SR-IOV NIC's vSwitch through a VSI. Similar to the Hypervisor based vSwitch, the SR-IOV NIC's vSwitch can be implemented as a VEB or VEPA and a single SR-IOV NIC may contain multiple such vSwitches.

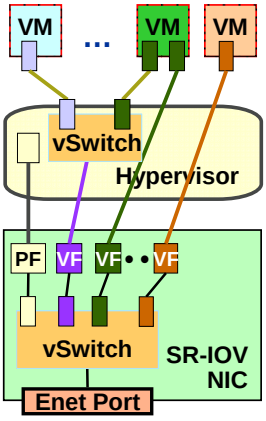

Figure 4: NIC based vSwitch

[E] EVB – Ethernet Virtual Bridging

[F] SR-IOV - Single-Root IO Virtualization

We next describe the proposed EVB protocols for automating the configuration of the VSI[G]s used by vSwitches.

B. Ethernet Virtual Bridging Protocols

This section provides a very brief overview of the proposed EVB protocols used in conjunction with Virtual Station Interface Discovery and Configuration Protocol (VDP). The detailed description of these proposed protocols is provided in the version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR.

There are three proposed protocols that are required to associate and de-associate a VSI with network state: Ethernet Virtual Bridging Type, Length and Value (EVB TLV) mechanism; Edge Control Protocol (ECP); and Virtual Station Interface Discovery and Configuration Protocol (VDP). The version 0, Edge Virtual Bridge Proposal also defines an S-Channel Discovery and Configuration Protocol (CDCP), which is used by the station (a.k.a. server or system) to support more than one VEB, VEPA or 2-Port VEB simultaneously. However, this protocol is not necessary if the station supports a single VEB, VEPA or 2-Port VEB.

The proposed EVB TLV is to be exchanged between station and bridge as part of the Link Layer Discovery Protocol (LLDP). The purpose of the EVB TLV is to enable the discovery and configuration of the station and bridge's EVB capabilities, including:

- The station's forwarding mode (i.e. VEB or VEPA).
- The ability of the station and bridge to use the Edge Control Protocol (ECP).
- The ability of the station and bridge to use the Virtual Station Interface Discovery and Configuration Protocol (VDP).
- If the station and bridge support VDP, the number of VSI Instances each is able to support and each has currently configured.
- The retransmission exponent used to calculate the minimum Upper Level Protocol Data Unit (ULPDU) retransmission time.

As noted above, EVB is used to determine if ECP and VDP are supported by both station and the adjacent bridge. If both are supported, then VDP modules are activated at the station and bridge to enable the seamless automation of network state configuration prior to starting up a VM's vNIC.

The proposed ECP is a control plane discovery and configuration protocol that provides acknowledgements, which are used by the receiver to signal to the sender that a ECP Upper Level Protocol (ULP) buffer is available for the reception an ECP Data Unit. The use of acknowledgements enables the sender to transmit discovery and configuration operations more frequently than possible with timer approaches, such as LLDP. The intent is to have the server's virtualization infrastructure (e.g. Hypervisor or a privileged guest VM) implement ECP, versus having the NIC implement ECP.

The proposed VSI Discovery and Configuration Protocol (VDP) is used to associate and de-associate a VSI Instance with a VSI Type Identifier (VTID), VSI Type Version, VSI Manager Identifier and one or more MAC Address and VLAN Identifier pairs. The VSI Type Identifier is used by the Adjacent Bridge to request a specific VSI Type from the VSI Type Database (VTDB) contained in the VSI Manager that is referenced by the VSI Manager Identifier. The version 0 EVB Proposal leaves the contents of a VSI Type opaque (i.e. to be standardized by another organization, such as the Distributed Management Task Force, DMTF, or left as vendor unique). That said, the contents of a VSI Type is expected to contain the network state of a VSI, such as access controls (e.g. IP Address filters), Quality of Service controls (e.g. rate limits) and/or security flow rules (e.g. Firewall and Intrusion Detection and Prevention, IDP, flow rules).

As shown in Figure 5 below, a network administrator uses a VSI Management user interface to create, change and destroy VSI Types in the VSI Manager's VTDB. The VSI Manager is expected to be a part of edge switch's Network Change and Configuration Manager (NCCM). A network administrator can create single VSI Type in the VTDB and use the VSI Type Version field to further refine the VSI Type's controls. Example uses cases for using the VSI Type Version will be provided in the next section.

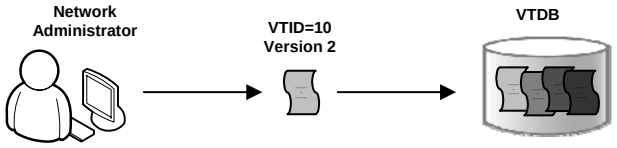

Figure 5: Virtual Station Components

The S-Channel Discovery and Configuration Protocol (CDCP) is a control plane protocol used to construct an S-VLAN for each VEB, VEPA or 2-Port VEB that will be used simultaneously by the station. The station and bridge must support an IEEE 802.1Qbc Port-mapping S-VLAN component. In the data plane, each S-channel carries an IEEE 802.1Qbc Port-mapping S-VLAN Tag, which identifies the S-Channel used by a specific VEB, VEPA or 2-port VEB instance. CDCP enables multiple virtual channels to be isolated and multiplexed over a single physical link.

[G] VSI – Virtual Station Interface

IV. Ethernet Virtual Bridging Use Cases

This section describes use cases associated with the version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR. This section will start with use cases for managing the VSI Manager and its associated VSI Type Database (VTDB), including models for how an access switch (a.k.a. adjacent bridge) can collaborates with a VSI Manager. It will then describe several use cases for performing network configuration operations between the server virtualization manager, the VSI Manager, the server's virtualization infrastructure and the access switch.

A. VSI Manager Use Cases

The VSI Manager can be implemented as a vendor specific tool or as a holistic tool, which spans multiple vendors. If the VSI Manager is implemented using a vendor specific tool, the virtualization infrastructure must know which VSI Manager Identifier to use in the VDP exchange. If a holistic tool is used, a single VSI Manager Identifier is used for all switch types.

If a vendor specific VSI Manager is used, the virtualization infrastructure can determine which VSI Manager Identifier is associated with the local switch through a management or control plane mechanism. Figure 6 depicts a high level overview of the management plane mechanism.

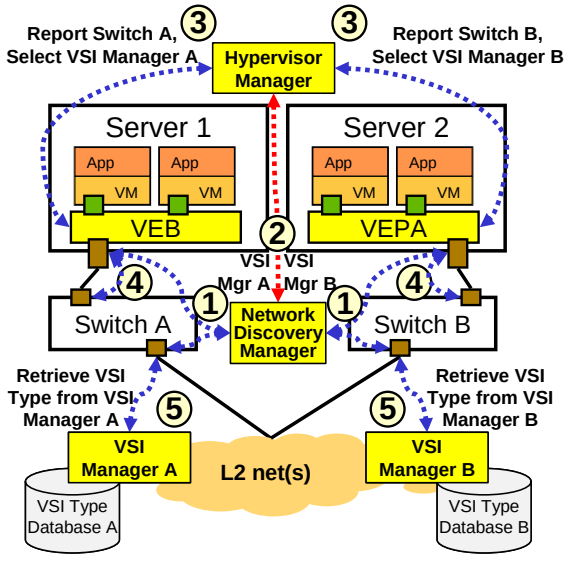

Figure 6: VSI Manager Selection via Management Plane

In figure 6, step 1 a Network Discovery Manager is used to determine the topology of the physical network, including what switch each NIC is connected to. In step 2, a management interface is used to inform the Hypervisor manager what VSI Manager Identifier must be used in conjunction with each NIC discovered by the Network Discovery Manager. In step 3, the Hypervisor manager informs the Hypervisor's VEB or VEPA control plane which VSI Manager must be used for each vNIC. In step 4, when the Hypervisor's VEB or VEPA control plane performs the VDP exchange, it then uses the VSI Manager Identifier passed from the Hypervisor and in step 5 the switch uses the VSI Manager Identifier passed through the VDP exchange to retrieve the VSI Type information.

Figure 7 depicts a high level overview for selection of a VSI Manager via a control plane mechanism. It's worth noting for this approach to work, the NIC must discover the VSI Manager Identifier from the adjacent switch. That is, a mechanism is needed for the switch to communicate its VSI Manager to the NIC. This type of mechanism would need to be added to the version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg.

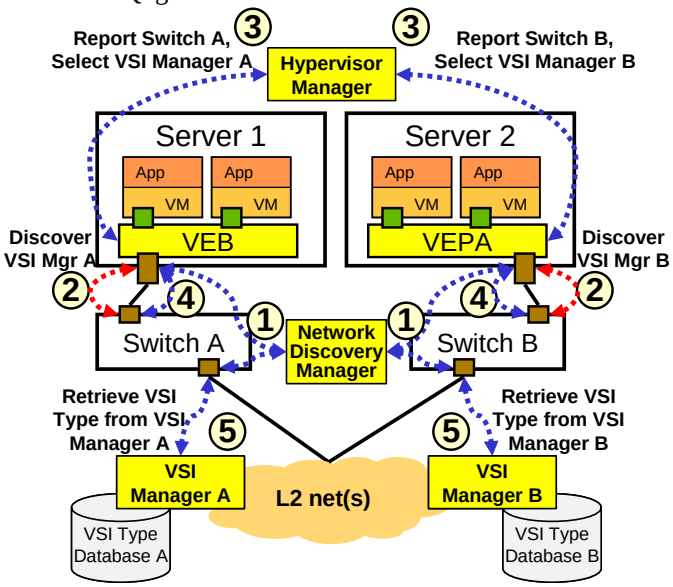

Figure 7: VSI Manager Selection via Control Plane

Following are the steps associated with a control plane mechanism depicted in figure 7. In step 1 a Network Discovery and Configuration Manager is used to determine the topology of the physical network and configure the VSI Manager Identifier and location on each switch used by servers to access the network. In step 2, a control plane protocol (e.g. EVB) is used by the server's virtualization infrastructure to discover the VSI Manager Identifier that must be used in a subsequent VDP exchange. In step 3, for each NIC, the Hypervisor's VEB or VEPA control plane reports to the Hypervisor manager the VSI Manager Identifiers it has discovered and the Hypervisor Manager returns the rest of the VSI state to use for the VDP exchange. In step 4, the Hypervisor's VEB or VEPA control plane performs the VDP exchange and in step 5, the switch uses the VSI Manager Identifier passed in the VDP exchange to look retrieve the

VSI Type information.

An alternative to the approach described in Figures 6 and 7 is to use a holistic Network Manager to host the VSI Manager, as well as the discovery and configuration management functions. That is, the network manager is capable of providing VSI Types to all the switches used in the network. In this case, the network manager loads the same VSI Manager Identifier on all the switches in the network and when each server's virtualization infrastructure performs the VDP exchange, the adjacent switch retrieves the VSI Type information from the same (single) VSI Manager.

B. VSI Type Database Use Cases

As described in the version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR, the VSI Type Database (VTDB) can use any combination of the VSI Type Identifier, VSI Type Version and VSI Instance Identifier to retrieve a VSI Type from the database. This enables a very flexible set of database schema use cases.

A simple VSI schema is to just use the VSI Type field as an index into the VTDB. Using this approach, a VSI Type can be associated to a VSI class. Figure 8 is an example of this use case, which depicts three VM types: Web Server, Application Server and Database Server. In this example, a specific VSI Type corresponds to each of these VM types, where the VSI Type contains the access, traffic and security controls that must be associated with each of these VM.

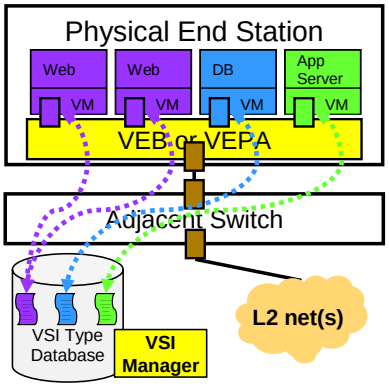

Figure 8: VSI Type schema

An alternative is to use the VSI Type to define a base set of access, traffic, and security controls and the VSI Type Version to define deltas from the base set of controls. For example, two types of web server application may be needed, where the only difference between the two is in the traffic rate limiting control. In this case, the same VSI Type can be used for both web server types and the VSI Version can be used to define the delta, rate limiting control. Figure 9 depicts this example

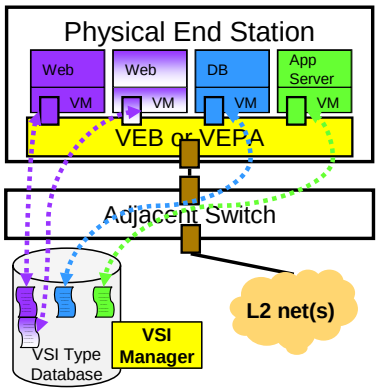

Figure 9: VSI Type Version, Delta schema

The VSI Type Version can also be used as a temporal attribute, where each VSI Type Version reflects a more recent version of the VSI Type. This allows network administrators to change a VSI Type's access, traffic, and security controls and use the VSI Type Version to roll out the change to new VMs. The change can also be attempted on existing VMs, by having each server's virtualization infrastructure re-execute a VDP Associate using the new VSI Type Version. If the adjacent switch is able to make the changes, the new VSI Type Version is used. Otherwise, the previously bound VSI Type Version remains in use.

Finally, the above use cases can be combined with the use of the VSI Instance as an index into the VTDB. This use case allows the VSI Manager to assure the VSI Instance can indeed be associated with the specific VSI Type and VSI Type Version.

C. Push Based VSI Type Use Cases

A VSI Type push model can be used if the amount of network state is small. For example, the VSI manager can push the network state onto all the switches if the set of VSI Types & Versions is small or if each VSI Type has a very small amount of access, traffic and security controls. This use case model essentially caches all the VSI Type state on each access switch for use during the VDP. Figure 10 on the next page depicts a high level use case for this model.

In Step 1, the VSI Manager creates a set of VSI Types. If VSI Versioning is used, each VSI Type version is assigned a VSI Version, which allows to the Network Manager to deploy one or more VSI Versions at any given time. As stated previously, the VSI Manager may be component of a larger networking function, such as a Network Change and Configuration Manager.

In Step 2 the VSI Manager discovers the access switches in the network to determine if they support a push model. If the switches support the push model, then the use case described

in figure 10 can be used. The VSI Manager pushes the VSI Types and VSI Versions to all access switches, which may be physical (e.g. top-of-rack of embedded blade switches) or virtual (e.g. Hypervisor VEB or IOV NIC's VEB).

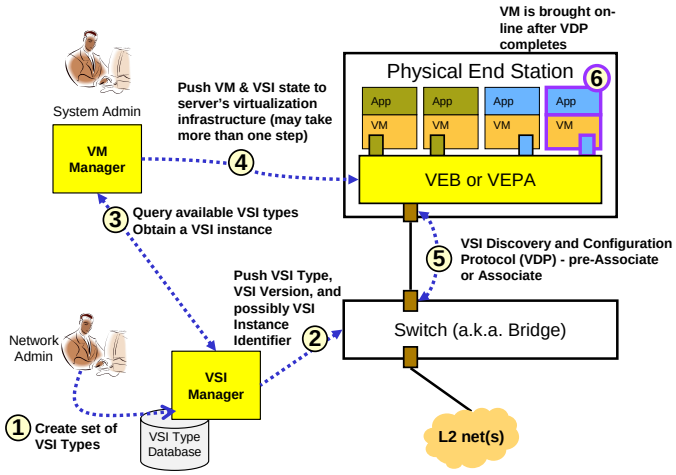

Figure 10: Push based VSI Management - VM Association

In step 3 the Hypervisor Manager queries the VSI Type Database to determine the usage of available Version Type IDs (VTIDs). For each NIC virtual port, the Hypervisor Manager selects a VSI Type, and if VSI versioning is used, a VSI version from the VTDB. The Hypervisor manager assigns a VSI Instance ID for each NIC virtual port. The Hypervisor manager may register the VSI Instance Identifier with the VTDB, including what VSI Type and VSI Version the Hypervisor plans to associate to the VSI Instance. If the Hypervisor registers the VSI Instance Identifier with the VTDB, the VSI Manager can check a pre-associate or associate VDP handshake to assure the correct combination are used.

In step 4, the Hypervisor manager creates a new Virtual Machine (VM) instance and pushes the VM's state into the server targeted to host the new VM. Part of the state the Hypervisor manager passes to the server virtualization infrastructure (Hypervisor or Hypervisor agent, such as a control partition), includes the VSI Type, VSI Version and VSI Instance Identifier.

In step 5 before VSI Instance (VM) activation, the VDP Module performs the VSI Discovery and Configuration Protocol exchanges that associate the VSI instance with a VTID, VSI Version, MAC Address and VLAN Identifier. The VDP Module is intended to be implemented as part of the server's virtualization infrastructure (e.g. in the Hypervisor or a service VM guest running on top of the Hypervisor). The VDP Module is also implemented in the adjacent bridge.

The VDP Module may use a Pre-Associate, Pre-Associate with Resource Reservation or Associate to begin the association process. The Pre-Associate enables the Hypervisor to probe a set of access switches prior to determining where a new VM (or set of VMs) is to be hosted. The first set of access switches to respond with a successful completion to the Pre-Associate can then be used for a subsequent Associate. However, the Pre-Associate operation doesn't reserve any resource on the access switches, so a subsequent Associate may completely unsuccessfully. The successful completion of a Pre-Associate with Resource Reservation guarantees the access switch has sufficient state for an Associate, while the VSI is kept alive.

In step 6, after the Associate VDP exchange has completed successfully the VM can start-up and the virtual NIC can be begin to use the virtual port associated the VSI Instance Identifier.

Figure 11 below describes the migration of a VM from the Source Server on the left to the Target Server on the right. Following are the associated steps. Note, this example assumes steps 1 and 2 of Figure 10 preceded the steps below.

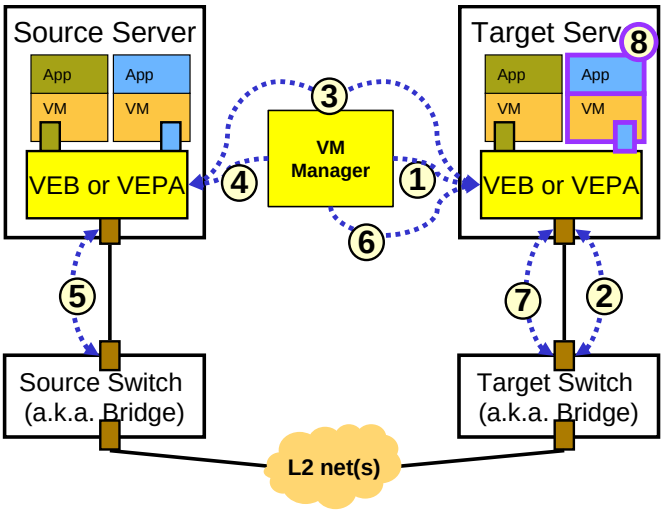

Figure 11: Push based VSI Management - VM Migration

In Step 1, the VSI Manager pushes either a Pre-Associate with Resource Reservation or Associate to assure the target server is attached to an access switch with sufficient resources to perform the VSI Association. Note, the VM being migrated from the source server is still associated to its access switch. The semantics in the version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR allow for the same VSI Instance to be associated on two switch ports simultaneously, for example during migrations. This dual residency can be eliminated by using a Pre-Associate with Resource Reservation on the Target server, which is the process that will be described in the remaining steps.

In step 2 the target's server virtualization infrastructure performs a Pre-Associate with Resource Reservation. If it completes successfully, the server virtualization infrastructure lets the Hypervisor manager know the target switch is capable of completing the migration. Otherwise the Hypervisor manager would have to select a different target switch, possibly on another server.

In step 3 the Hypervisor Manager begins the VM migration process by moving VM state from the Source to the Target Server. In step 4, when this process reaches completion, the Hypervisor Manager requests the server virtualization infrastructure to perform the shut down process on the Source Server. This process includes shutting down the VM on the Source Server and after the VM has been shutdown, performing a De-Associate, step 5 in Figure 10.

In step 6, once the De-Associate completes, the Hypervisor Manager requests the server virtualization infrastructure to perform the start up process on the Target Server. This process first performs an Associate VDP operation, step 7. After that operation completes successfully, the VM is started up on the Source Server, step 8. In the case of a fault on the switch that prevents the Associate from completing successfully, the Hypervisor Manager would have select either a different port, switch or server.

D. Pull Based VSI Type Use Cases

The version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR provided a high level overview of a VSI Type pull model, see Figure 12 below.

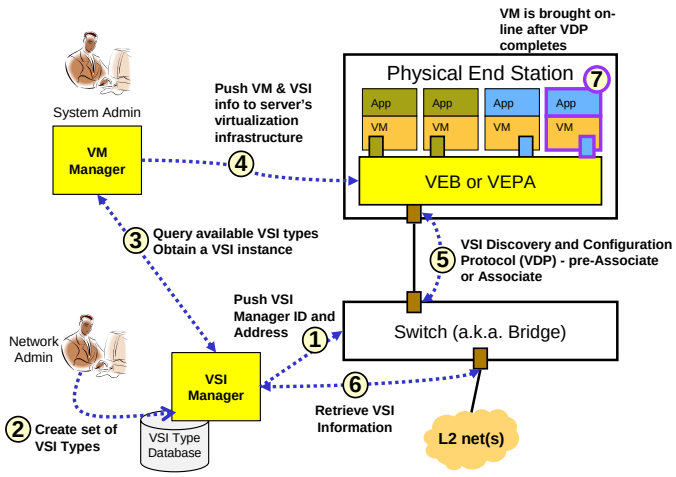

Figure 12: Pull based VSI Management - VM Association

The model shown in figure 12 enables switches to use less memory for VSI state (i.e. VSI Type and Version content), because each switch pulls only the VSI state in use by pre-associated or associated VSI Instances on that switch. For the case where a holistic VSI Manager is being used, this model also enables access switches from different vendors to pull VSI network state from the vendor's accompanying VSI Manager.

In Step 1 the VSI Manager discovers the access switches in the network and loads the VSI Manager Identifier as defined in the version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR. The VSI Manager also loads its location, e.g. IP Address, so the access switch knows what how to reach the VSI Manager.

In Step 2, the VSI Manager creates a set of VSI Types. If VSI Versioning is used, each VSI Type version is assigned a VSI Version, which allows to the Network Manager to deploy one or more VSI Versions at any given time. As stated previously, the VSI Manager may be component of a larger networking function, such as a Network Change and Configuration Manager.

In step 3 the Hypervisor Manager queries the VSI Type Database to determine the usage of available Version Type IDs (VTIDs). For each NIC virtual port, the Hypervisor Manager selects a VSI Type, and if VSI versioning is used, a VSI version from the VTDB. The Hypervisor manager assigns a VSI Instance ID for each NIC virtual port. The Hypervisor manager may register the VSI Instance Identifier with the VTDB, including what VSI Type and VSI Version the Hypervisor plans to associate to the VSI Instance. If the Hypervisor registers the VSI Instance Identifier with the VTDB, the VSI Manager can check a pre-associate or associate VDP handshake to assure the correct combination are used.

In step 4, the Hypervisor manager creates a new Virtual Machine (VM) instance and pushes the VM's state into the server targeted to host the new VM. Part of the state the Hypervisor manager passes to the server virtualization infrastructure (Hypervisor or Hypervisor agent, such as a control partition), includes the VSI Type, VSI Version and VSI Instance Identifier.

In step 5 before VSI Instance (VM) activation, the VDP Module performs the VSI Discovery and Configuration Protocol exchanges that associate the VSI instance with a VTID, VSI Version, MAC Address and VLAN Identifier. The VDP Module is intended to be implemented as part of the server's virtualization infrastructure (e.g. in the Hypervisor or a service VM guest running on top of the Hypervisor). The VDP Module is also implemented in the adjacent bridge.

In step 6, the access switch uses the VSI Manager Identifier

passed in the VDP exchange (step 5) to determine which VSI Manager contains the VSI state referenced in the VDP exchange. The access switch uses the VSI Type Identifier and VSI Version to retrieve the VSI Type from the VSI Manager. In step 7, after the Associate VDP exchange has completed successfully the VM can start-up and the virtual NIC can be begin to use the virtual port associated the VSI Instance Identifier.

Figure 13 below describes the migration of a VM from the Source Server on the left to the Target Server on the right. Given the similarity between the push and pull model, we'll just describe the delta between the two models, which is to perform steps 2.A & 2.B in the figure. In Step 2.B, the access switch uses the VSI Manager Identifier passed in the VDP exchange, step 2.A, to determine where to go for the VSI state. The access switch uses the VSI Type Identifier and VSI Version to retrieve the VSI Type from the VSI Manager. Note, if in step 7 any of the VDP variables are different from those use in the pre-Associate, the access switch should go back to the VSI Manager to determine if the change is appropriate. Again, the rest of the steps are the same as in Figure 11 above.

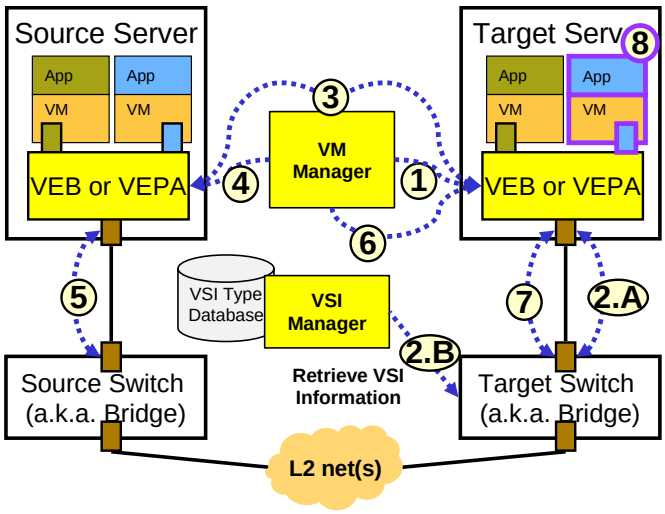

Figure 13: Pull based VSI Management - VM Migration

E. Secure VSI Type Control Use Cases

Today's server virtualization infrastructure does not provide a mechanism for assuring that a given Hypervisor has the authority to associate a given MAC Address to physical network state. Similarly, there is no way of assuring a non-virtualized server has the authority to associate a given MAC Address to physical network state.

The version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR can be used in conjunction with IEEE 802.1AE MAC Security to assure the Hypervisor, or OS for non-virtualized servers, has the authority to perform a VDP operation.

F. Cross-Data Center VSI Instance Migration Use Cases

The version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR can be used in conjunction with layer-2 extension technologies, such as Virtual Private LAN Service (VPLS) over a Multi-Protocol Label Switching service provider infrastructure. The intended usage model is planned workload migrations or workload redundancy.

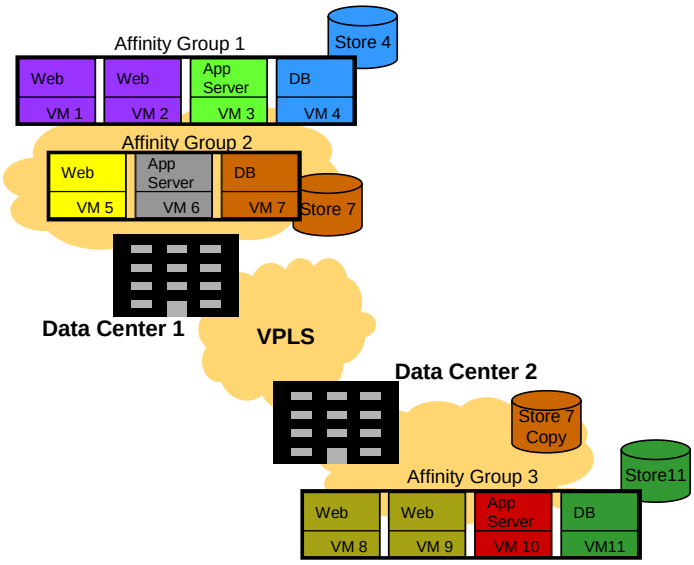

Figure 14: Cross Data Center VSI Migration

Figure 14 depicts two Data Centers, with a total of 11 VMs in 3 Affinity Groups. In this paper an affinity group represents a set of VMs that have frequent communication between VMs in the group. The colors in Figure 14 are meant to depict different VLANs. The disk cylinders are meant to represent storage associated with each affinity group.

The following considerations should be taken into account when architecting a cross Data Center (cross-DC) use case using the version 0, Edge Virtual Bridge Proposal and MPLS/VPLS:

- To keep multicast domains within a single Data Center, each VLAN should be kept within a single DC. That is, cross-DC migration should not split a VLAN multicast domain across Data Center boundaries. This can be accomplished by moving all VM in a VLAN together.
- To keep communications between VMs in an affinity group local, each affinity group should be kept within a

single Data Center. That is, cross-DC migration should not split an affinity group across Data Center boundaries, because VM-VM communications within the affinity group would then have to travel across Data Center boundaries. This can be accomplished by moving all VMs in an affinity group together.

- The migration of an affinity group across Data Centers, requires affinity group's storage to also be migrated. For example, this can be accomplished by using a remote site synchronous copy combined with dual active-active access to the remote site's synchronous copy. For example, in figure 13 a remote synchronous site copy mechanism is used, between Data Center 1 and Data Center 2, to maintain a copy of affinity group 2's store 7 in Data Center 2. Additionally, all the VMs in affinity group 2 that need to access store 7, would also have access to store 7's copy in Data Center 2. If affinity group 2 is migrated to Data Center 2, the VMs that access store 7 would switch over to store 7's copy in Data Center 2.

V. Conclusion

The version 0, Edge Virtual Bridge Proposal to the IEEE 802.1Qbg PAR is used as the basis for this paper. The use cases described in this paper enable the automation of network state configuration, such as port access, traffic and security flow controls. They also enable VM migration to include the automated migration of all the network state associated with a VM. In summary, the Ethernet Virtual Bridging use cases covered in section IV of this paper enable network state, as shown in Figure 15, to essentially be "moved" before the VM is moved. The version 0, Edge Virtual Bridge Proposal enables dynamic multi-tenant environments, with per tenant type network controls that automatically migrate in conjunction with tenant migration.

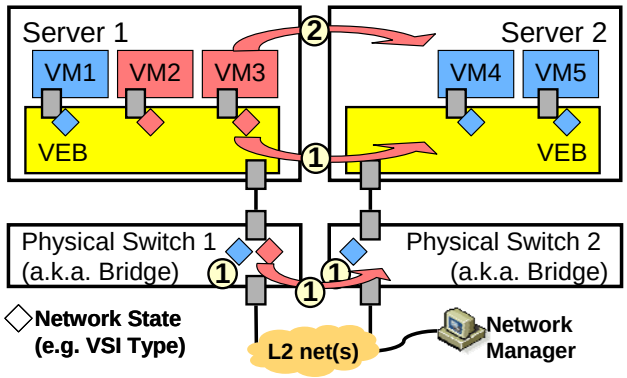

Figure 15: Cross Data Center VSI Migration

It is also worth noting that the Edge Control Protocol (ECP) and the Virtual Station Interface Discovery and Configuration Protocol (VDP) are complimentary to the IEEE 802.1Qbh Port Extender protocol. For example, IEEE 802.1Qbh may be able to use ECP and VDP to associate a VM virtual NIC's virtual port VSI Instance to a MAC Address, VLAN Identifier and VSI Type. However, such a proposal has not been brought forward at this time.

What's next? In our view, a key research focus area is liberating VM migration from layer-2 networking nuances.

VI. Acknowledgements

We are grateful to Daya Kamath, Vivek Kashyap, Jay Kidambi, Srikanth Kilaru, James Macon, Daniel Martin, Vijoy Pandey, Paul Congdon, Anoop Ghanwani, Pat Thaler, Chait Tumuluri, Manoj Wadekar and Yaron Haviv, for their insightful comments.

VII. References

[1] IEEE 802.1Qbg - Edge Virtual Bridging, http://www.ieee802.org/1/pages/802.1bg.html.

[2] Integrated Virtual Ethernet Adapter Technical Overview and Introduction, www.redbooks.ibm.com/redpapers/pdfs/redp4340.pdf.

[3] Ko, Mike; Recio, Renato; Virtual Ethernet Bridging, www.ieee802.org/1/files/public/docs2008/new-dcb-ko-VEB-0708.pdf

[4] Recio, Renato and Cardona, Omar; "Automated Ethernet Virtual Bridging", http://i-teletraffic.org/fileadmin/ITC21_files/DC-CAVES/DC-CAVES_-_AutomatedEthernetVirtualBridging.pdf; 9/14/2009.

[5] Cisco Nexus 1000V Virtual Switch Data Sheet, "http://www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9902/data_sheet_c78-492971.pdf" Cisco 3/2010

[6] Cisco, Virtual Networking Features of the VMware vNetwork Distributed Switch and Cisco Nexus 1000V Series Switches, www.cisco.com/en/US/prod/collateral/switches/ps9441/ps9902/solution_overview_c22-526262.pdf.

[7] HP, IBM, et. al.; "version 0, Edge Virtual Bridge Proposal", http://www.ieee802.org/1/files/public/docs2010/bg-joint-evb-0410v1.pdf; April 23, 2010.

[8] CEE Authors, Yahoo Group, http://tech.groups.yahoo.com/group/cee-authors/

[9] Priority Based Flow Control, CEE Author version 0 proposal to IEEE 802.1Qbb, http://www.ieee802.org/1/files/public/docs2008/bb-pelissier-pfc-proposal-0508.pdf

[10] Enhanced Transmission Selection, CEE Authors version 0 proposal to IEEE 802.1Qaz, http://www.ieee802.org/1/files/public/docs2008/az-wadekar-ets-proposal-0608-v1.01.pdf

[11] Data Center Bridging eXchange Protocol, CEE Authors version 0 proposal to IEEE 802.1Qaz, http://www.ieee802.org/1/files/public/docs2008/az-wadekar-dcbx-capability-exchange-discovery-protocol-1108-v1.01.pdf