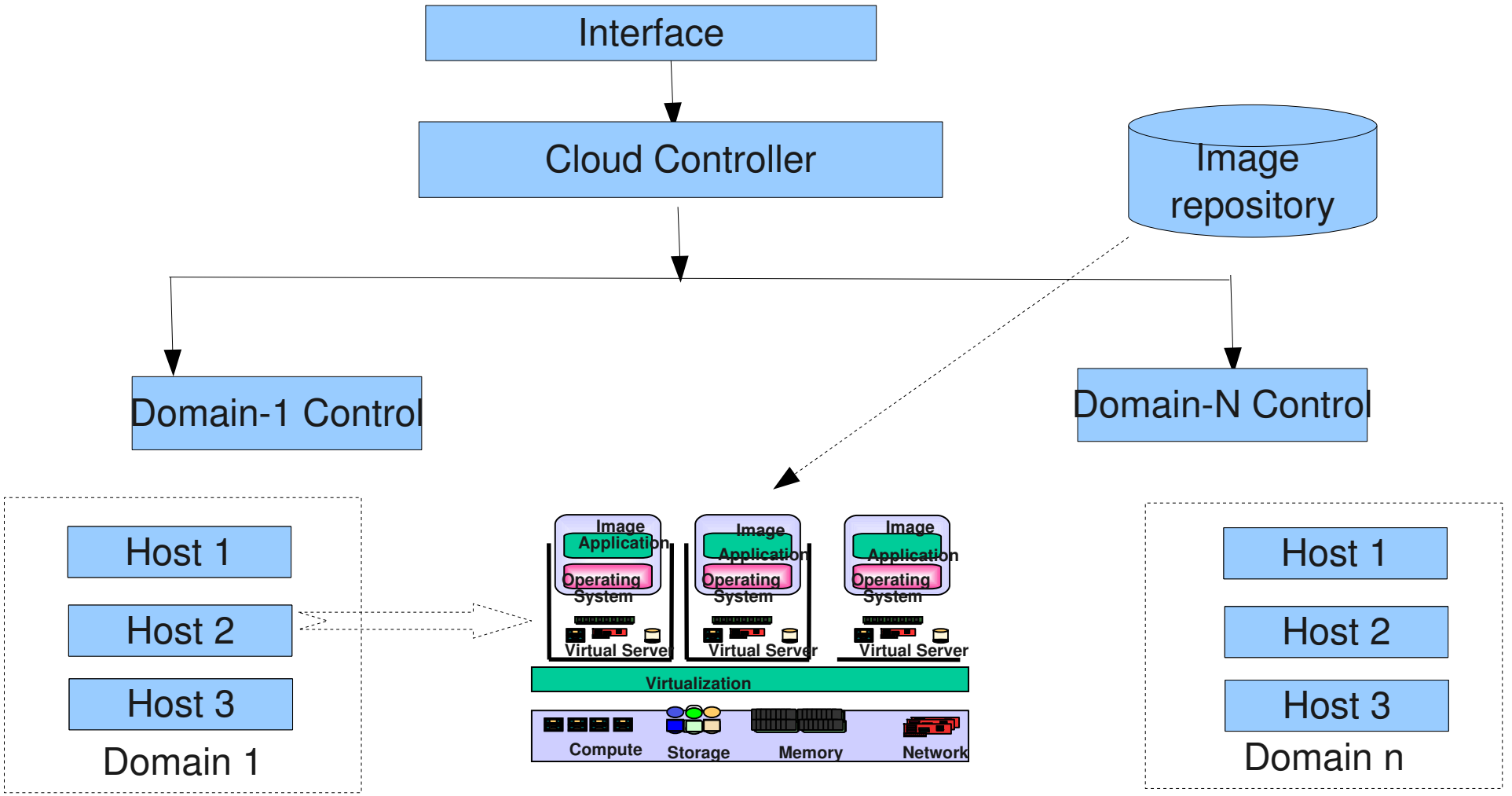
Automating Network Security Profiles

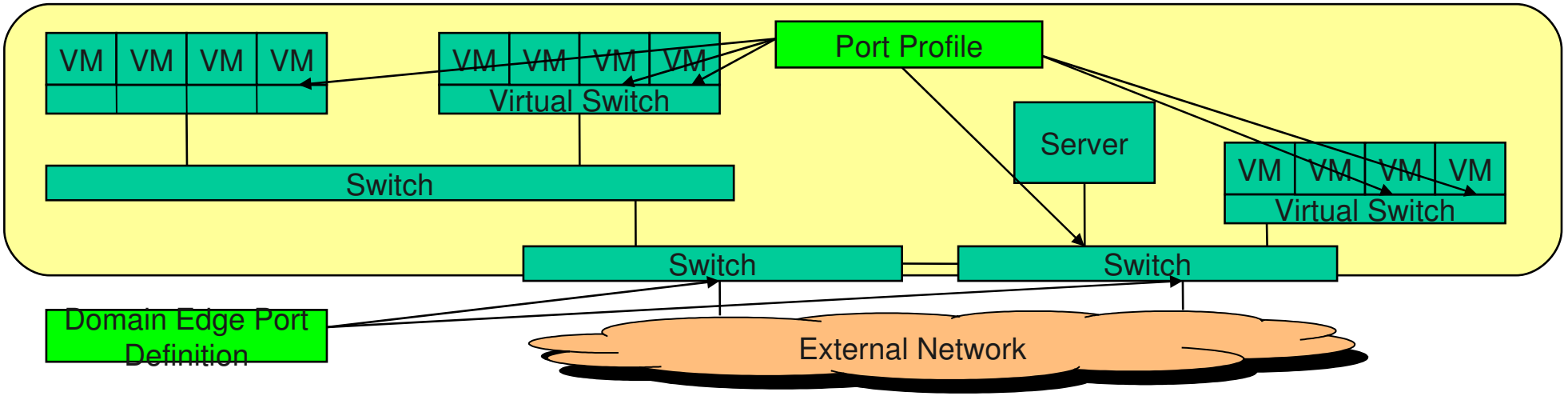
Vivek Kashyap
Senior Technical Staff Member
Linux Technology Center
IBM

Cloud



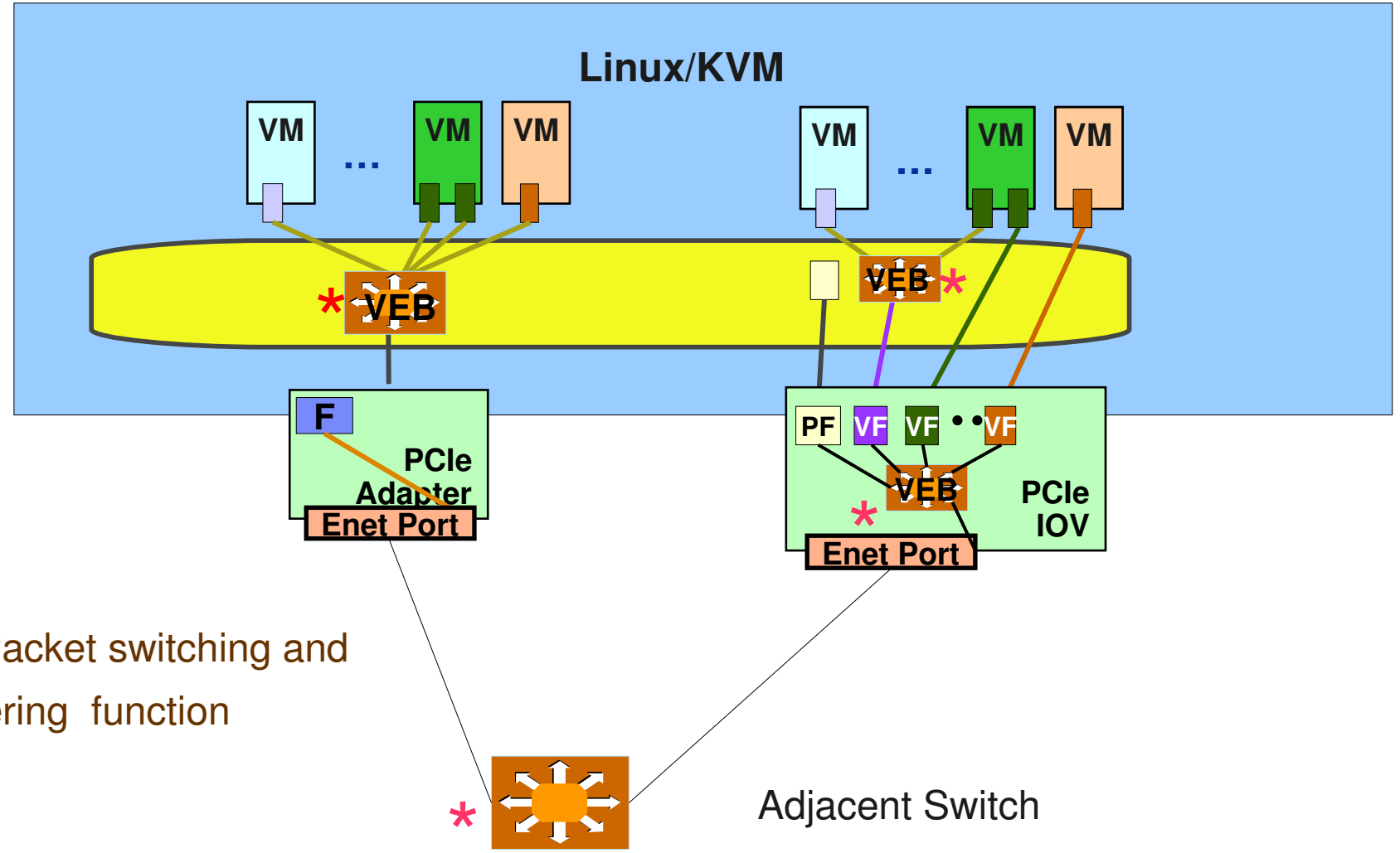
- Multiple images deployed on physical nodes in the DataCenter/Cloud
- Network isolation a must for a viable multi-tenant solution
- Collaborative applications may be on same virtual network

Network View



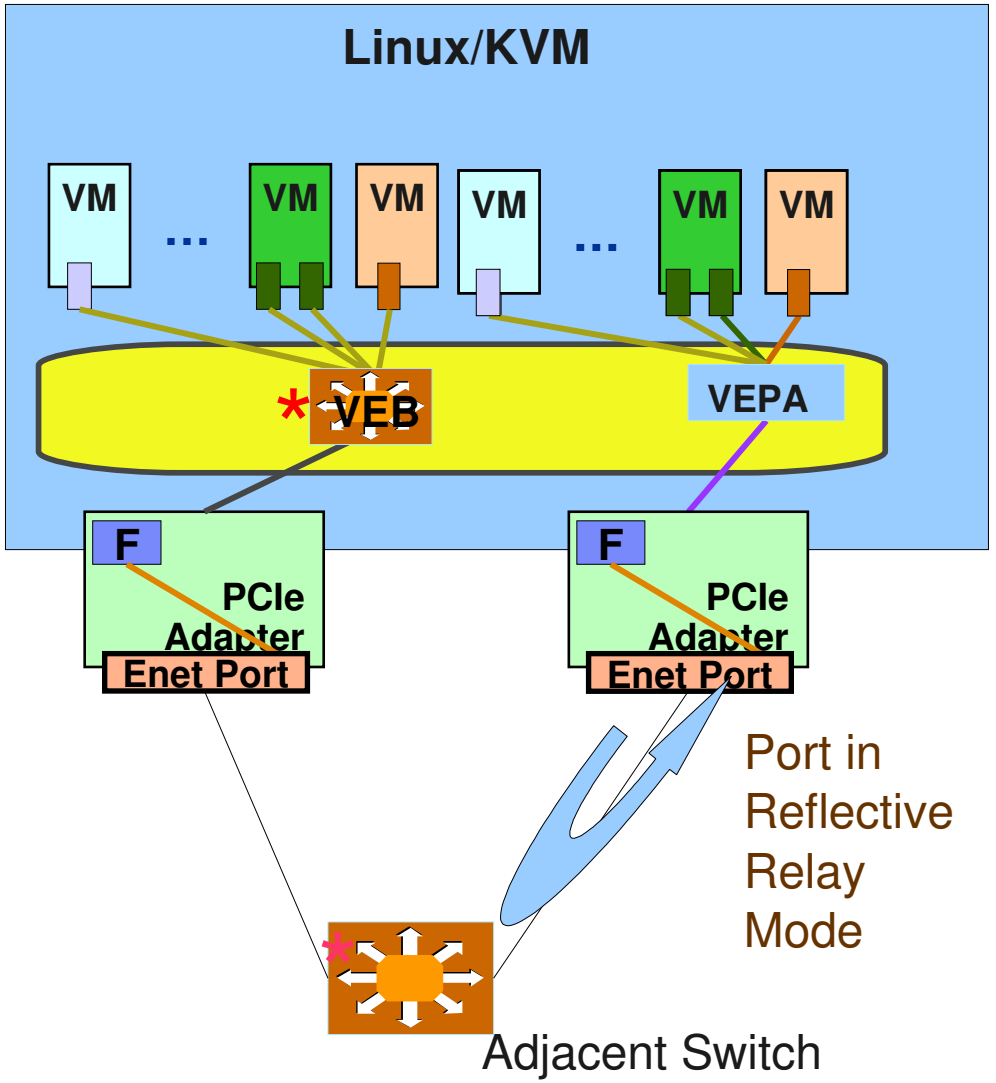
- DataCenter consists of large number of physical and virtual switches
  - Applications with different network profiles
- Host vswitch provides guest-guest switching, filtering, bandwidth control
  - Error-prone managing large deployments
  - Virtual switch is not integrated with network fabric management
    - inconsistencies and manual verification
- For load-balancing, resiliency the KVM guests are mobile
  - Network policies must continue to be applied to the KVM guest workload
    - Manually ensure target are correctly configured to support network profiles
    - Physical port security/profiles need to be re-programmed with mobility

Linux Virtual Networking



* Packet switching and filtering function

Automating Physical/Virtual Switching



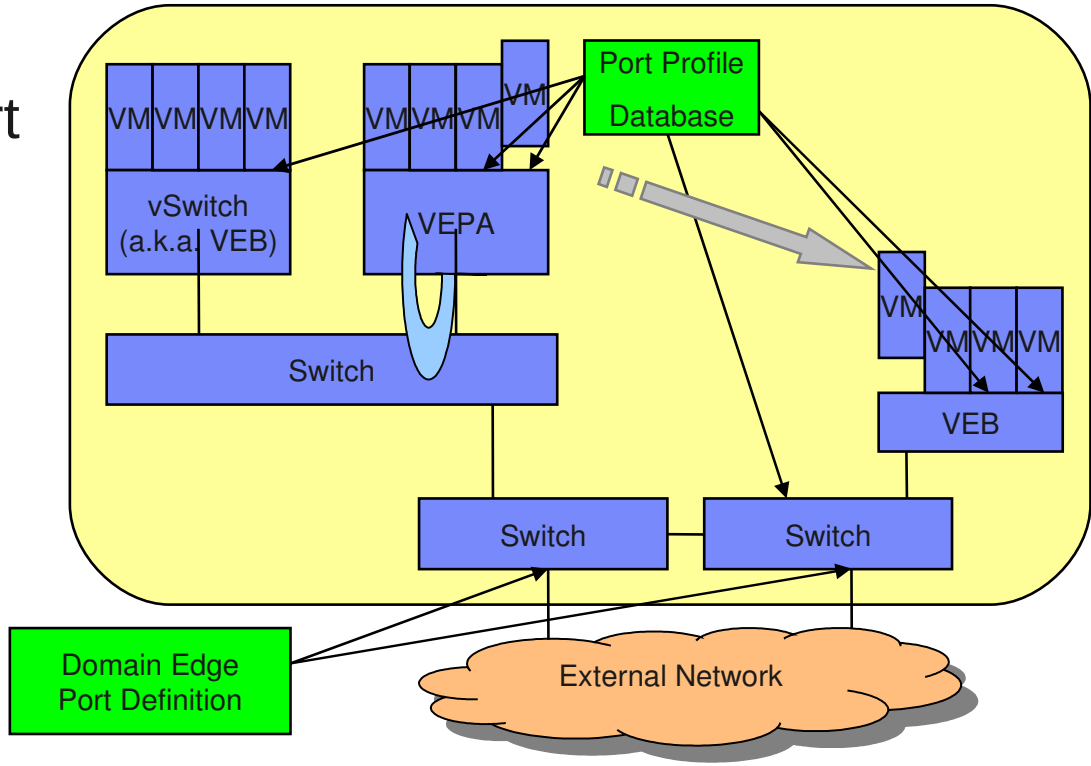
Edge Virtual Bridging: IEEE 802.1Qbg

- Offload switching function to external bridge
  - VM's virtual interface directly tied to physical switch port policies
  - Simplified VEPA (Virtual Ethernet Port Aggregator) bridging in hypervisor to send all packets to adjacent bridge.
    - Provides packet replication of inbound frames.
  - Physical switch port put in 'reflective relay' mode
    - Sends packets back over same port
- **Con**: Introduces limited latency

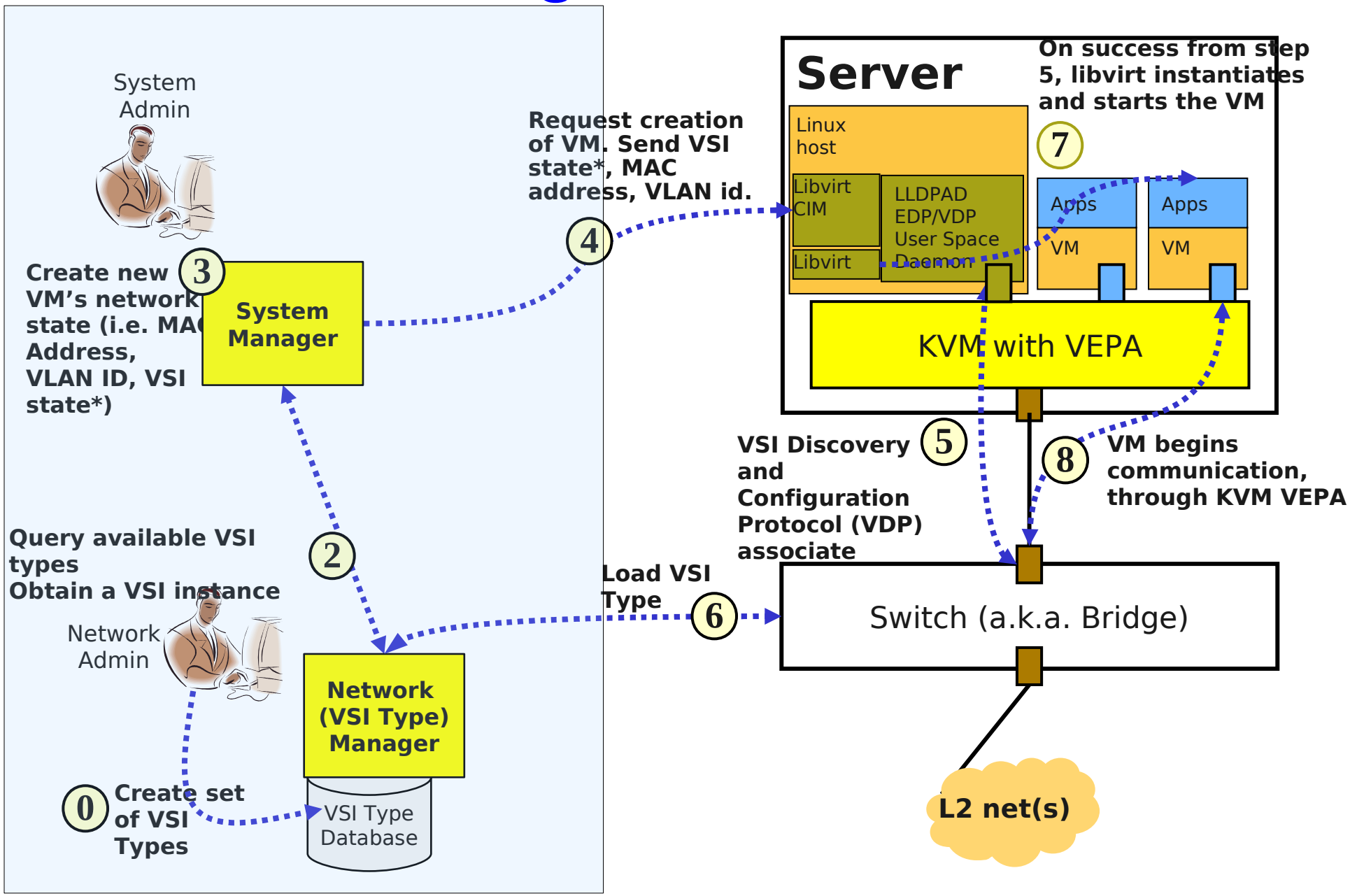
VEPA: Virtual Ethernet Port Aggregator
VEB: Virtual Ethernet Bridge (or, Linux bridge)

How Does it Work?

- The network profile
  - Used by one or more VMs
  - Unique id
  - Stored in database
- Switch advertises 802.1Qbg support
- Linux/KVM host receives advertisement
  - Configures switch port in VEPA (hairpin_mode)
  - Offloads switching function
- Linux/KVM sends to switch
  - unique id of network profile
  - MAC/VLAN information
- Switch retrieves profile
  - Enforces bandwidth, Access controls



Creating a KVM Guest



*VSI state consists of the following: VSI Manager ID, VSI Instance ID, VSI Type ID, VSI Type Version.

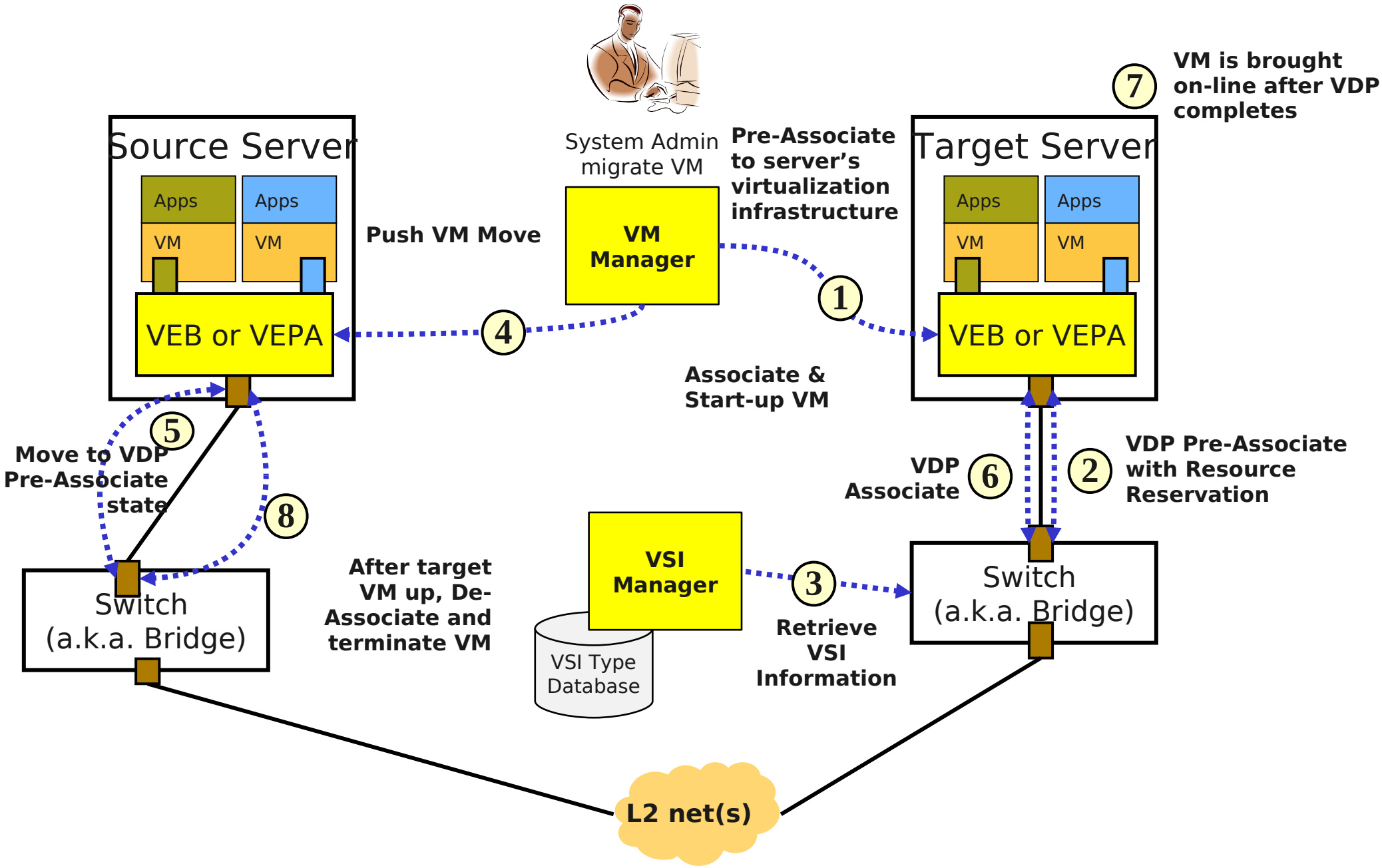
Simple extension to libvirt

- The VSI state is specified using the following domain XML extension

```
<interface type='direct'/>
    <source dev='device name' mode='vepa' />
     <model type='virtio'/>
     <virtualport type='802.1Qbg'>
          <parameters managerid='12'  typeid='0x123456'
  typeidversion='1'          instanceid='insert-uuid-here' />
     </virtualport>
</interface>
```

- Libvirt parses 'virtualport type'
  - Sends netlink message with 'ASSOCIATE' request to LLDPAD
  - LLDPAD sends ASSOCIATE VDP message
  - Returns success or failure
  - On success KVM guest is created

Migration Steps



Automatic Host based Virtual Switching

- Host vswitch
  - Linux bridge + ebtables/iptables + tc
  - OpenVswitch (not in mainline)

- Administrative simplification: Associate filter rules to Virtual machines
  - Rules enforced in the kernel when guest started
  - Rules torn down when the guest is terminated
  - Rules may be modified at runtime
  - Rules may contain macros which get instantiated at runtime
    - IP Address, MAC address , more possible
      - DHCP Snooping/first packet to determine IP Address

Example Filter

```
<filter name='no-ip-spoofing' chain='ipv4'>
  <uuid>fce8ae33-e69e-83bf-262e-30786c1f8072</uuid>
  <rule action='drop' direction='out' priority='500'>
    <ip match='no' srcipaddr='$IP'/>
  </rule>
</filter>
```

This filter may now be referenced with any guest by adding to the 'interface' element in the guest domain:

```
<interface type='bridge'>
  <mac address='52:54:00:56:44:32'/>
  <source bridge='br1'/>
  <ip address=$IP/>
  <target dev='vnet0'/>
  <model type='virtio'/>
  <filterref filter='no-ip-spoofing'/>
</interface>
```

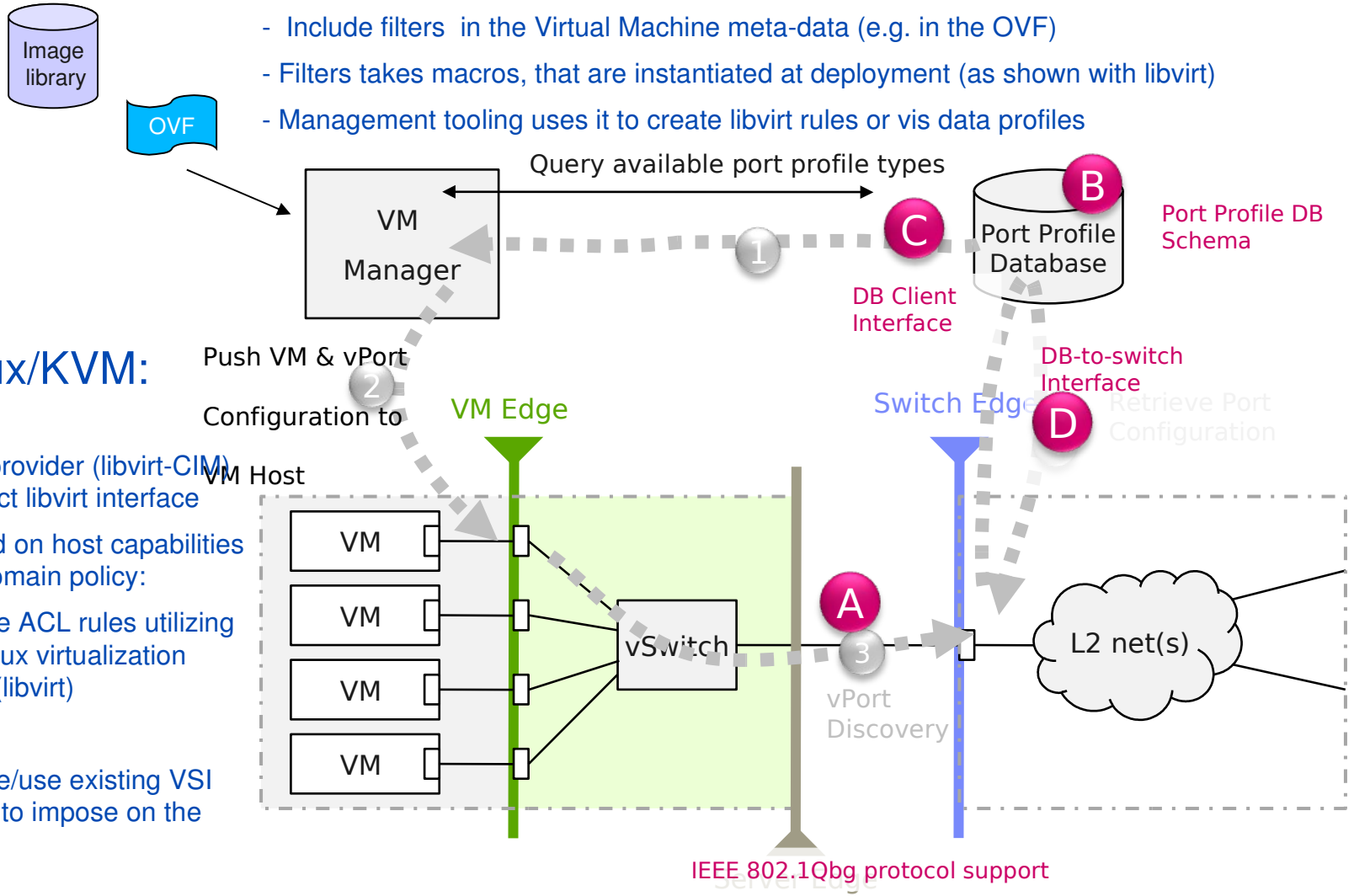
Status

- Linux 2.6.34 onwards
  - VEPA mode (for IEEE 802.1Qbg support)
  - Bridging between virtual interfaces
  - Vhost-net interface for Qemu
  - GSO/GRO acceleration for macvtap
- Libvirt 0.8.7 (http://libvirt.org/formatnwfilter.html)
  - VEPA and VSI support
    - Netlink notifications for VDP protocol (to LLDPAD)
  - Support for host based filter rules
- LLDPAD:
  - Version lldpad 0.9.41 (open-lldp.org)
    - EVB TLVs, ECP/VDP
- Libvirt-CIM: 0.5.12
  - Support for specifying VEPA, VSI state

A peek into the Future: Network Profile Automation

Network extensions

- Include filters in the Virtual Machine meta-data (e.g. in the OVF)
- Filters takes macros, that are instantiated at deployment (as shown with libvirt)
- Management tooling uses it to create libvirt rules or vis data profiles



Linux/KVM:

- CIM provider (libvirt-CIM) or direct libvirt interface
- Based on host capabilities and domain policy:
- Create ACL rules utilizing the linux virtualization library(libvirt)

or

- Create/use existing VSI profile to impose on the guest

Legal Statement

This work represents the view of the author and does not necessarily represent the view of IBM.

IBM is a registered trademark of International Business Machines Corporation in the United States and/or other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries .

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, and service names may be trademarks or service marks of others.

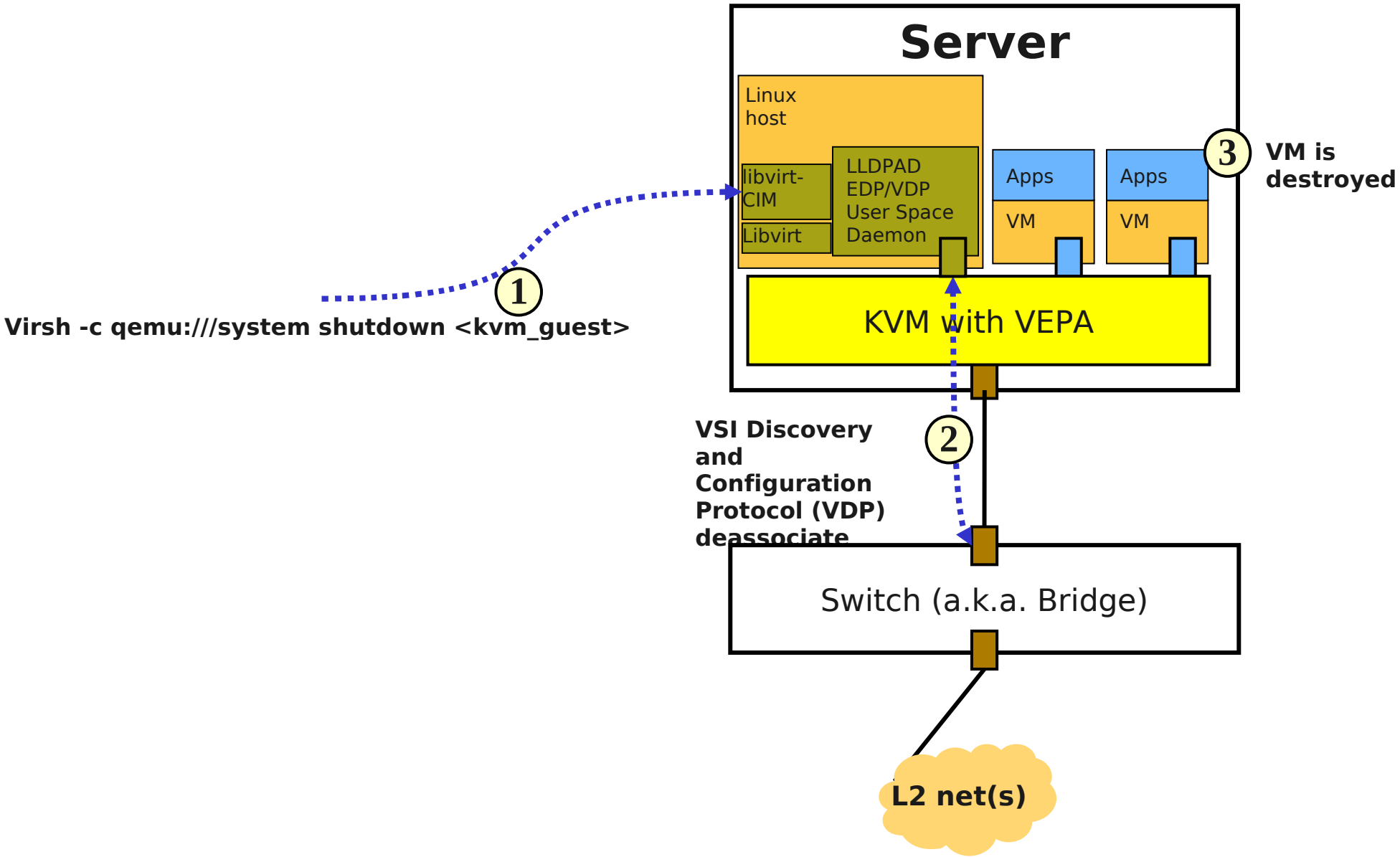
Questions?

802.1Qbg Protocols

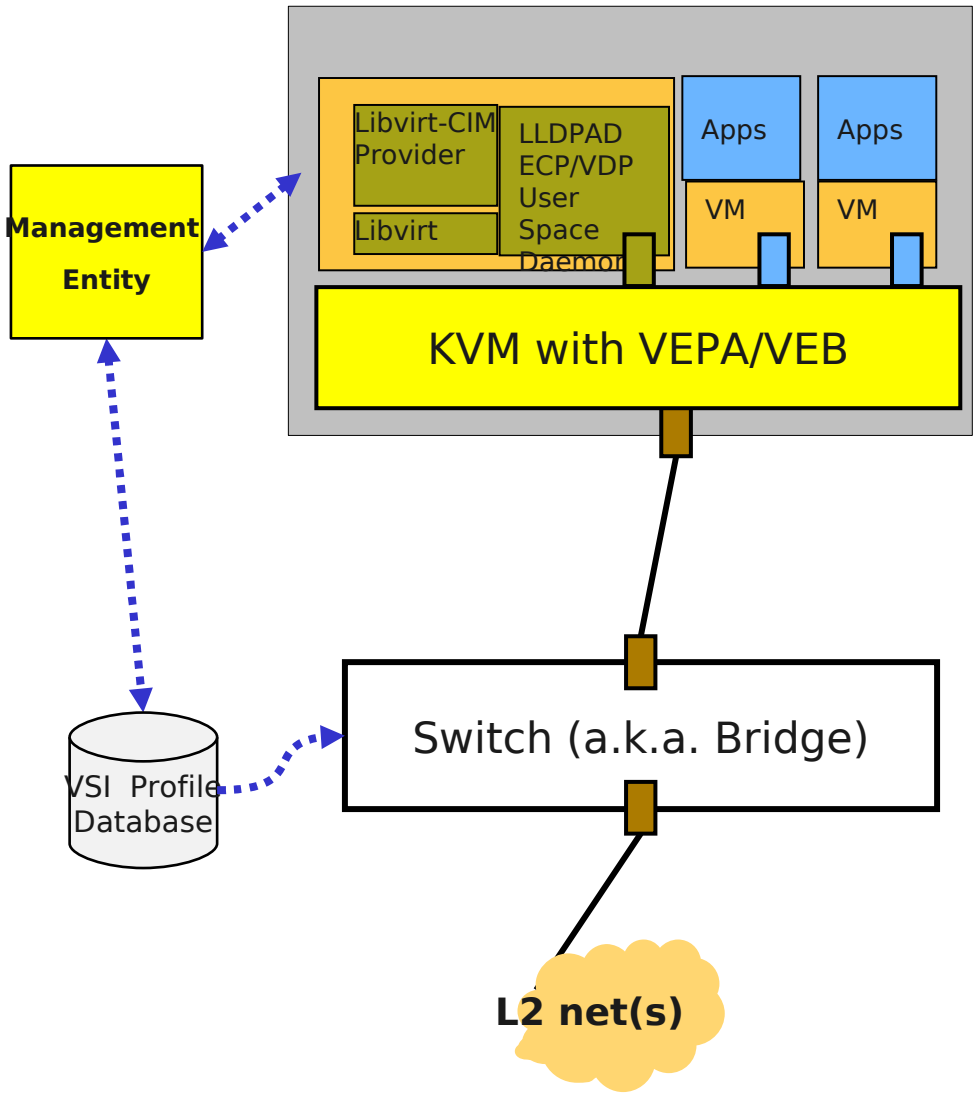
- External Virtual Bridging protocol (EVB) uses LLDP as transport
  - Defines locus of VM to VM switching
    - Set VEPA or Bridge mode
- Channel Discovery and Configuration Protocol (CDCP) [Not Implemented]
  - Virtualize the physical link to simultaneously support multiple VEPA/VEB components
- Edge Control Protocol (ECP)
  - ECP provides a reliable, acknowledged protocol for VDP rather than using LLDP.
- Virtual Station Interface (VSI) Discovery Protoco(VDP)
  - Associate and De-associate interface MAC/VLAN to port profile

For specification go to: IEEE 802.1Qbg version 0 Specification

Destroy Guest



Linux KVM Components



- Enablement of 'macvtap'
- Linux virtualization library framework (libvirt)
  - Configure interfaces in VEPA or VEB (Bridge)Mode
  - Enforce ACLs on the VEB virtual port
  - Support for creation, destroy and migration of KVM guests (VM)
  - Trigger of IEEE 802.1Qbg protocols with KVM guest life-cycle events
- 802.1Qbg protocols implemented in LLDPAD daemon
  - VDP : VSI Discovery Protocol
  - ECP: Edge Control Protocol
  - LLDP extensions for switch mode (Ethernet Virtual Bridging TLVs)

VSI: Virtual station interface (a.k.a virtual interface or vNIC)

***VSI state consists of the following: VSI Manager ID, VSI Instance ID, VSI Type ID, VSI Type Version.**

MACVTAP: VEPA interface

- **Goal**: Share a single Ethernet NIC between KVM guest interfaces with no bridging function except replication of incoming multicast/broadcast packets
- **Implementation**:
  - Utilize existing 'macvlan' driver already supported in Linux
  - Create a 'macvtap' device driver that plugs into a macvlan driver to interface to KVM guest
    - Macvtap driver implements tun/tap-like interface
    - Frames sent from guest put directly into queue of outbound interface
    - Frames received put in guests receive path

Libvirt: VEPA interface

- Macvtap is tied specifically to an interface that provides uplink connectivity
- Define it in guest's domain xml as:

```
<interface type='direct'/>
    <source dev='device name' mode='vepa' />
    <model type='virtio'/>
</interface>
```

  - Additional modes defined: *bridge and pepa*

- Libvirt creates (and destroys) macvtap devices
  - Passes macvtap file-descriptor to QEMU (similar to the case with tap interface)
  - AF_NETLINK socket to create (and destroy) macvtap interface