MINT 709

Capstone Project Report

COMPARATIVE ANALYSIS OF OVERLAY TECHNOLOGIES

Submitted by

USMAN RASHEED

in partial fulfillment for the award of the degree

Of

Master of Science

In

Internetworking

University of Alberta, Canada

==

Declaration

I confirm that the capstone project report I am submitting is entirely my own work and that any material used from other sources has been clearly referenced at the end of this report.

Project Title: COMPARATIVE ANALYSIS OF OVERLAY TECHNOLOGIES
Student Name: Usman Rasheed
Date: 15/04/2015

==

ACKNOWLEDGEMENT

A special thanks goes to the head of the project, **Dr. Mike Macgregor** for showing his confidence in me by assigning me to this project and putting full effort in guiding the team to achieve the goal.

I would like to express my gratitude and appreciation to System Engineer, Cisco Systems **Ms. Kanwal Cheema** whose guidance, help, stimulating suggestions and encouragement, helped me to complete my project. She played the most crucial role by attending my phone calls at all times and by always being available to help whenever I got stuck. Wonder of thanks to her for giving me the permission to use all required valuable equipment and providing me the necessary material to complete this project.

I would also like to thanks MINT coordinator, **Mr. Shahnawaz Meer**, who gave me all the available resource in MINT Lab and gave me suggestions to carry out this project successfully.

Last not least, thanks to student coordinator **Ms. Sharon Gannon** for carrying out all the official administrative work regarding registration and other necessary work related to this project.

Table of Contents

List of Figures

Project Title

COMPARATIVE ANALYSIS OF OVERLAY TECHNOLOGIES

Project Problem Description

Every service provider and company has three words for network architects looking to grow their data centers: Scalable, Simpler, Faster and Flatter whereas on the other hand applications within Data Centers are becoming more and more bandwidth hungry and complex. Layer 2 is an important requirement within Data Centers. Current Protocols like Spanning Tree and other layer 2 protocols are not sufficient to handle required amount of traffic. STP provides redundancy and prevents loops, but it uses an active/standby model to do so. As a result, STP networks offer two potential paths for any flow, only one of which can forward traffic. Spending billions of dollars on redundant devices, ports and cables does not make sense when you can't use them. This not only increases CAPEX costs, it also significantly increases OPEX by requiring more power and cooling.

Layer 2 is mandatory especially as the DC interconnect but has many limitations – it is imperative that we find a way to use possibly the combination of both layer 2 and layer 3 in a way that employs their strengths but masks their weaknesses. Deploying multiple active paths in a data center network is also very important to reduce total cost of ownership and protect investment. To address these problems different vendors have come up with different solutions but none of them are complete or satisfactory. Virtually every vendor addressing the data center fabric switching market proposes different and its own solutions examples of which are:

1) Overlay Transport Virtualization (OTV)

2) Network Virtualization using Generic Routing Encapsulation (NVGRE)

3) Virtual Extensible LAN (VXLAN)

Some of these protocols are hardware dependent and some are based on the SDN (Software Defined Network) concept. The intention of this report is to do a comparative analysis of these protocols from the perspective of scalability as well as usability within an SDN framework.

Scope of this Project:

Scope of this project is to do a theoretical analysis of these protocols and then objectively decide which of them have the characteristics to be used in the real world in large Service Provider and Enterprise DC environments. The ones that qualify will then be extensively tested in terms of their bandwidth utilization, design versatility and network performance. The methodology to do so will be to run many different practical tests and highlight the problems and strengths of each of these protocols.

The project aims to test these protocols at all DC layers including the access, aggregation, core and in the DCI as well encompassing both the logical and physical layers. The stretch goal is to break the tie among different vendors and propose one complete solution which can be implemented by Service Providers to make their data center more efficient, flexible, resilient, manageable and above all best use of their resources.

Protocols as Solution to Network Evolutions

Network Evolution # 1 - Network Virtualization:

Server virtualization is the division of a physical server into small-scale virtual servers to optimize server resources. This technique of server virtualization helps us to operate the data centre more efficiently as several applications can be all be hosted on one server running multiple virtual environments. It helps maximize utilization of resources and prevents any waste in investment. As a result, fewer bare-metal servers are used and both OPEX and CAPEX are significantly lowered.

There was a need of a similar concept in the field of networking as well. Many networking devices were under-utilized. Also, each node had its own control functions, which bought up cost per node and also meant that every network node had to be managed separately. Moreover, there was no uniformity among networking devices in terms of their processor utilization and packet forwarding. To overcome this challenge, a similar concept to virtualization was introduced in the field of networking as well. It encompassed many different things including Network Functions Virtualization or NFV in which many network functions or services could be hosted on x86 servers. To enable some of these transitions, Overlay Protocols were a necessity. They virtualized packets by encapsulating layer 3 in layer 2 or vice versa.

Network Evolution # 2 - Large Cloud Networks:

When networking was first born, in a layer 2 construct, the identity of a host was its MAC address while in a layer 3 construct, the identity and location was determined via an IP address. This was pretty much decoupled from whatever was running or hosted on these devices. However, it was soon realised that segmentation was a necessity due to security and hence VLANs came into existence. However, they were limited to 4096 VLANs per network domain. With the advent of cloud networking, there was a need for SPs to host many different customers/ applications/ tenants and the restriction of 4096 quickly became a bottleneck.

So, there was a need for new protocols that provided VLAN type segmentation but were much more scalable. With the evolution of some Overlay Protocols i.e VXLAN the scalability to create new segments has increased to 16 million.

Network Evolution # 3 - Next Generation Data Centres:

The next wave in DC evolution is the Software Defined Networking. SDN proposes certain new concepts that break away from the traditional approach to networking. Some of these ideas include the separation of the control and forwarding functions within the network and in some cases centralizing the control functions while keeping the forwarding distributed. They also include the reuse of hardware resources by employing virtualization within networking. This means the use of protocols for layer2 and layer3 encapsulation and virtualization as well which is achieved through the protocols that are being extensively discussed in this report.

For example, Cisco Systems has jumped on to the SDN bandwagon and brought in Application Centric Infrastructure that uses Nexus 9K series switches as stateless networking and APIC-Controller for the centralized policy functionality. All traffic within the Nexus 9K fabric is tunnelled using VXLAN.

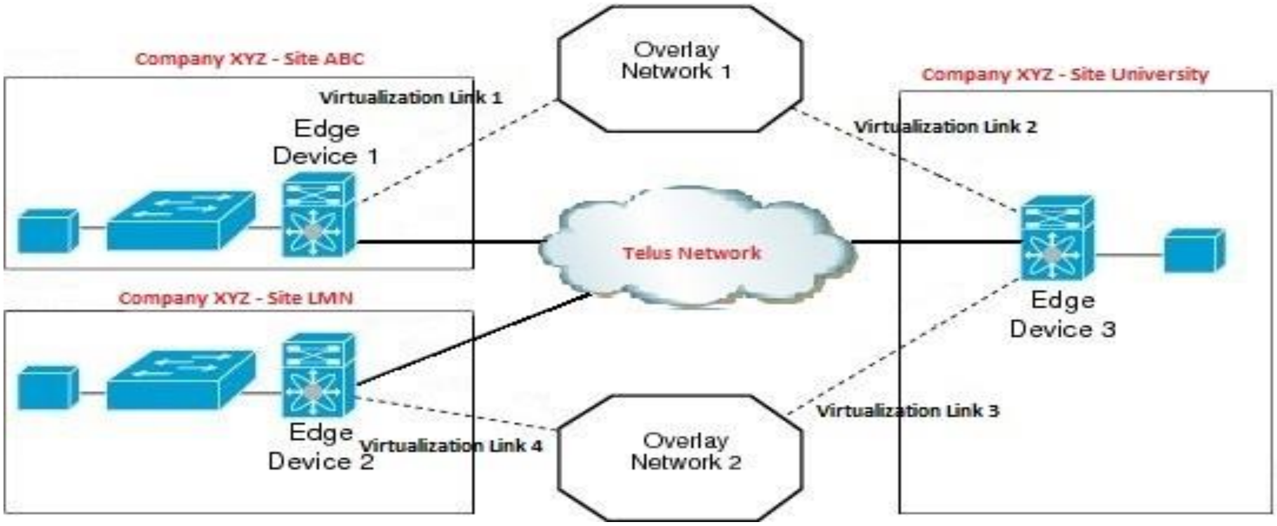


Figure 1 Overlay Protocols Virtual Channels of Communications

I will attempt to discuss one by one each of these protocols and share with you my observations and findings.

Theoretical Study of Protocols:

Overlay Transport Virtualization:

OTV stands for Overlay Transport Virtualization and it is a technique that extends layer-2 between data centres that are geographically separated. Simply put, it is Data Centre Interconnect Technology that routes MAC address based information by encapsulating the traffic in IP packet for transit.

Overview:

Present technologies of Layer-2 VPNs work and rely heavily on tunnels. Technologies like EoMPLS, VPLS and VPLS+ all employ tunnels to route traffic between Data Centers. On the contrary, OTV (Overlay Transport Virtualization) works by encapsulating layer-2 traffic with an IP header and does not form any stateful tunnel.

OTV supports layer2, layer3 or even MPLS packets. All it requires is a layer-3 connectivity between different data centres. It does not require any design changes; however it is only supported in Cisco 7k switches with M1 line cards.

I will break down the topology in small parts and elaborate it one by one.

OTV Edge Device:

OTV edge device can be either Nexus 7000 or nexus 7k VDC that is placed at the edge of Data Centre carrying out all the virtualization functions for the sake to connect two different data centre sites. This edge device is connected to both layer-2 data centre domain and layer-3 IP network domain. The nexus OS 5.1 and above allows maximum two OTV edge devices to be deployed for redundancy

Internal Interfaces:

Internal interfaces of OTV edge device will participate in STP domain and also learns the mac address.

The layer-2 interface configured either trunk or access port.

Join Interface:

- Join interface is a layer-3 interface on the OTV edge device and it connects layer-3 IP network.
- This interface is last point where the outgoing traffic leaves the OTV edge device after getting encapsulated.
- Loopback or logical interfaces are not allowed for this. With Nexus OS this have to be the physical interface or port channel.
- A single interface can also be used with a pre-given overlay.
- More than one overlays can also be shared with the same Join interface.

Overlay Interface:

- Overlay interface is logical interface which is capable of multi-access and multicast and this is an interface where all the Overlay transport Virtualization configuration are explicitly stated by user.
- This interface actually tells that which layer-2 traffic has to be encapsulated before sending it to the join interface.

OTV Control-Group

In an overlay network, the OTV control group is used as a speaker. It is the multicast group.

For each overlay group there is always a unique multicast address is required.

OTV Data-Group:

Any frames of multicast traffic that is extended across the overlay is encapsulated by OTV Data-group.

Extended VLANs:

This is going to check that are VLANs are permitted as an extension across different sites over the overlay. The MAC addresses of the VLAN's will not be allowed to be advertised on overlay.

Site-VLAN:

Site-VLAN should be define and active. It should also use default configuration.

OTV Operation:

For the advertisement of MAC reachability the OTV primarily use control plane.IS-IS is a control protocol which is used as an underlying protocol for routing. Its hellos are encapsulated in multicast header of OTV. These packets usually use a particular multicast address. The use of IS-IS is basically for two main reasons:

1. The only routing protocol which does not use IP for routing but instead of this it actually uses CLNS.
2. The second reason is with the use of TYPE, LENGTH and VALUE (TLV) is easily extended to carry out informational fields since this type of encoding is actually used for optional information.

Prior to the exchange of any informational reachability regarding MAC, all of the OTV devices must be adjacent. This can be possible by the application of OTV Control-Group.

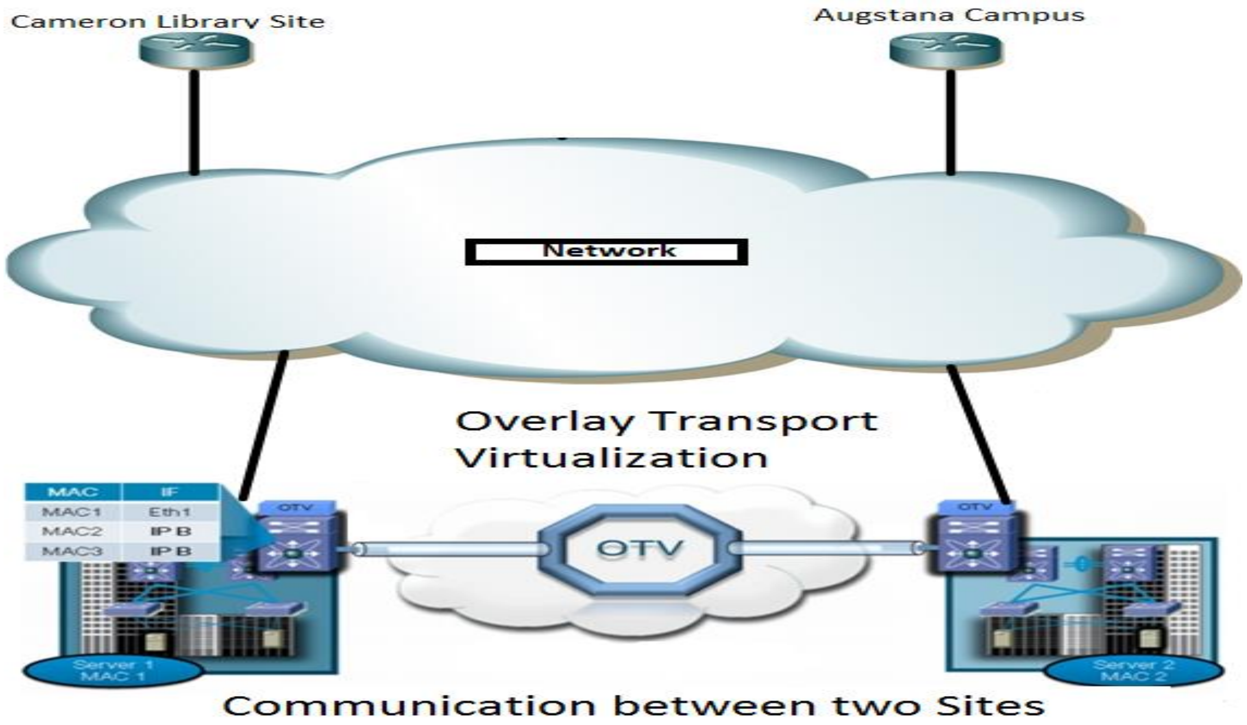


Figure 2 OTV – Communication between two sites

Neighbor Discovery Control Plane:

In order to perform control protocol exchanges the edge device join the ASM group by sending IGMP report. These OTV edge devices joint it as a hosts. Keeping in mind that this is not PIM. In order to establish the adjacencies for control plane and for the communication of local devices, the hello packets are sent to every OTV edge device. With the use of overlay networks these hello packets sent to remote machines. As I mentioned earlier, the destination address will be the address ASM multicast for traffic control whereas the source address will be the IP address of a joint interface. On the other end, the device which will receive will strip off all those encapsulated headers before passing it to the c-plane. After once the process is complete the OTV end devices will have each other’s information in their tables and for any incoming new connection they perform the same procedure.

MAC Address Advertisement Control Plane:

In Mac address advertisement there will be a look up on Ethernet frame which is going to determine the last interface for the destination.

Header Format for the OTV:

The total size for the header is 42 bytes which is a overhead. The Don't Fragment is set on usually on all OTV ongoing packets. The main purpose being to set this field is that the Nexus7k does not support the feature of reassembling the fragmentation packet. Like VXLAN, we have to set the increase in MTU size on all of the transport interfaces. This is just due to the fact that fragmentation is not possible. Increasing the MTU size and eradication of fragmentation has its own pros and cons. This technique makes the forwarding faster but it has the overhead of doing some extra configuration. So, we have to increase the size of MTU in order to make it work.

Spanning Tree Protocol Working in OTV:

All the devices participate in Spanning Tree Protocol by transmitting and accepting packets on internal interfaces except the OTV edge device. Because OTV actually confined the domains of STP on both sides. So, the biggest advantage the OTV has on other techniques is that OTV does not use MAC flooding to learn MAC reachability instead it use control-plane protocol. This is the unique behavior of OTV which makes it much efficient in terms of it functionality and resource utilization.

Multi Homing in OTV:

OTV also offers the co-exist behavior in same site with the help of multiple edge devices. OTV offers multi homing for the purpose of load sharing.

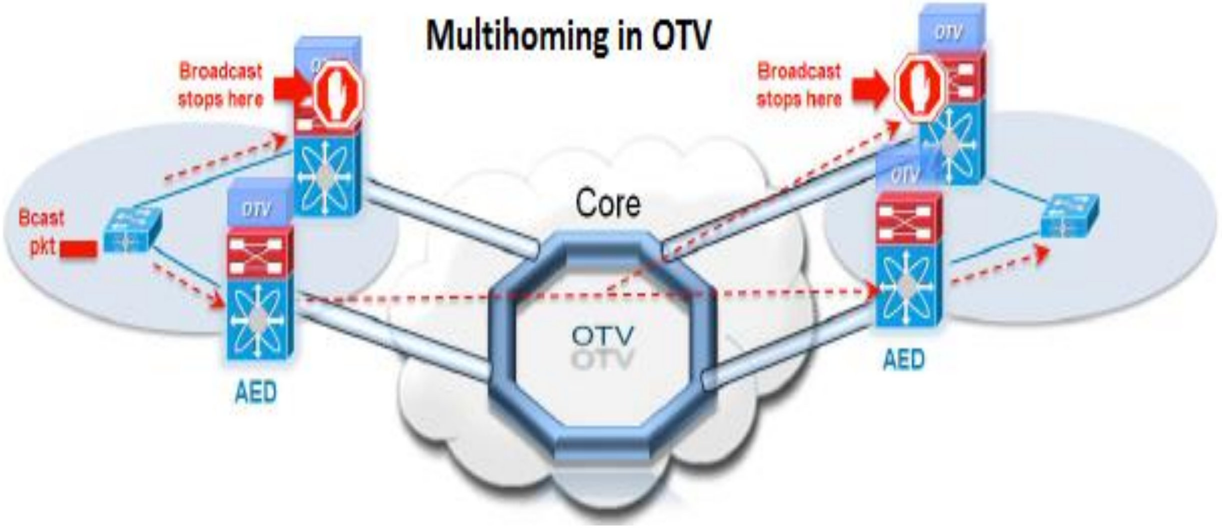


Figure 3 Multi Homing in OTV

Observation and Findings:

According to my observation, the control plane functionality of advertising MAC is the biggest advantage that OTV brings to the DCI realm. The traditional data plane learning with the help of MAC flooding is feeble and resource intensive. OTV also introduces the new concept of MAC routing or MAC in IP routing. This is actually done by encapsulating a layer-2 frame inside a layer-3 frame before it actually transmits over a traditional IP network. This technique of encapsulation is called an overlay between different data sites.

OTV is usually deployed on the data center edge devices called Overlay Transport Virtualization end devices. These devices perform layer-2 learning and forwarding on their site facing internal interfaces and carry out IP virtualization functions on interfaces that are towards Core via logical bridge interface. Each Overlay Transport Virtualization end device must have an IP address which is unique to that ISP/network provider for reachability. This allows OTV to be placed in any kind of network in a very simple manner.

Network Virtualization using Generic Routing Encapsulation:

NVGRE stands for **Network Virtualization using Generic Routing Encapsulation.** As the name suggests, it is a virtualization technology in which Layer-2 packets are encapsulated within layer-3 packets using Generic Routing Encapsulation Technique. It helps to eradicate the problems associated with scalability. Microsoft is the main supporter of NVGRE.

NVGRE is very similar to VMware technology called VXLAN. NVGRE uses multicast for logical broadcast just like VXLAN. Porting to NVGRE is much easier as compared to others as many of the switching chip already support GRE than a non-supported format of tunneling. Open vSwitch already support NVGRE.

NVGRE Ethernet Frame Encapsulation:

Tenant Network Identifier (TNI) is added to the L2 which is a unique 24-bit ID. Ethernet frame use lower 24 bits of the GRE-Key field. This new 24-bit Tenant Network Identifier enables more than 16 million Layer 2 logical networks to function within the same administrative domain. It is also a scalability improvement of many orders of magnitude over the 4,094 VLAN segment limit. The

Layer 2 packet (frame) with **Generic Routing Encapsulation** is then encapsulated with an outlying IP header and finally an exterior MAC address.

Observation and Findings:

NVGRE, a new mechanism from Emulex and Microsoft to form Overlay Networks. By stretching the routing inside the server site, Emulex and Microsoft are enabling the production of programmatic dynamic Overlay Networks that will easily scale from small to large to cloud able multi-tenant architectures. This is a very important initiative towards bringing down the cost of the data center infrastructure. NVGRE restrains the capability of using Receive Side Scaling (RSS) to redistribute traffic across all cores based on the inner packet, which means that there is a severe reduction in bandwidth. This downgrade of performance and increase in CPU overhead significantly cancel out many benefits of using NVGRE.

Improvements in NVGRE can be achieved by supporting all existing hardware offloads in the network controllers. Which includes:

a. Allowing checksum to be done on both the inner and outer headers
b. Have to perform huge segmentation offloading

Virtual Extensible LAN:

So here is strip down topology to match this idea, there is a tunneling mechanism identical to this which is called VXLAN. The basic idea of VXLAN is to connect several layer -3 networks and make it seems as if they use the same layer-2 domain. This will actually allow the virtual machines to exist in different networks but still remain on the same layer 2 domain.

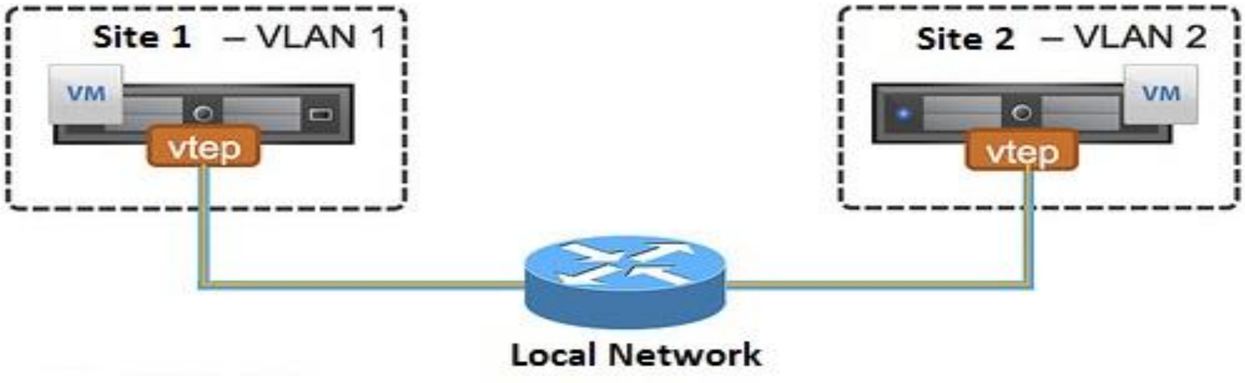


Figure 4 what is VXLAN

The encapsulation is consist of following changes from standard frames:

Ethernet Header:

VLAN:

This field is optional in VXLAN and can be designated as an Ethernet type of 0x8100 and an associated VLAN tag ID.

Destination Address:

When a destination VTEP is on different Layer 3 network, this destination address is set to the MAC address of destination VTEP.

Header:

Protocol:

This is to set as 0x11 for the indication that the frame is a UDP packet.

Source IP:

This is an IP address of originating VTEP.

Destination IP:

This contains the IP address of VTEP target. If this is not known in advance as in a case when the virtual machine has never targeted for, couple of steps need to be done by originating VTEP which are as follow:

- The IP address of the destination is going to replace by the multicast IP group similar to the VNI of the originating VM.
- Those VTEP's which have subscribed to the multicast group will be going to receive the frame and then it will uncover and will learn the host VTEP and mapping of virtual machine mac address.

UDP Header:

Source Port:

This is usually set by transmitting VTEP.

VXLAN Port:

VXLAN port number has been assigned by IANA.

UDP Checksum:

This has to be set as 0x0000.If the In case the checksum is not set to this value by the VTEP source then the receiving VTEP will inspect the checksum and if its not correct , it should drop the frame and must not de-encapsulate it at any case.

VXLAN Header:

VXLAN Flags:

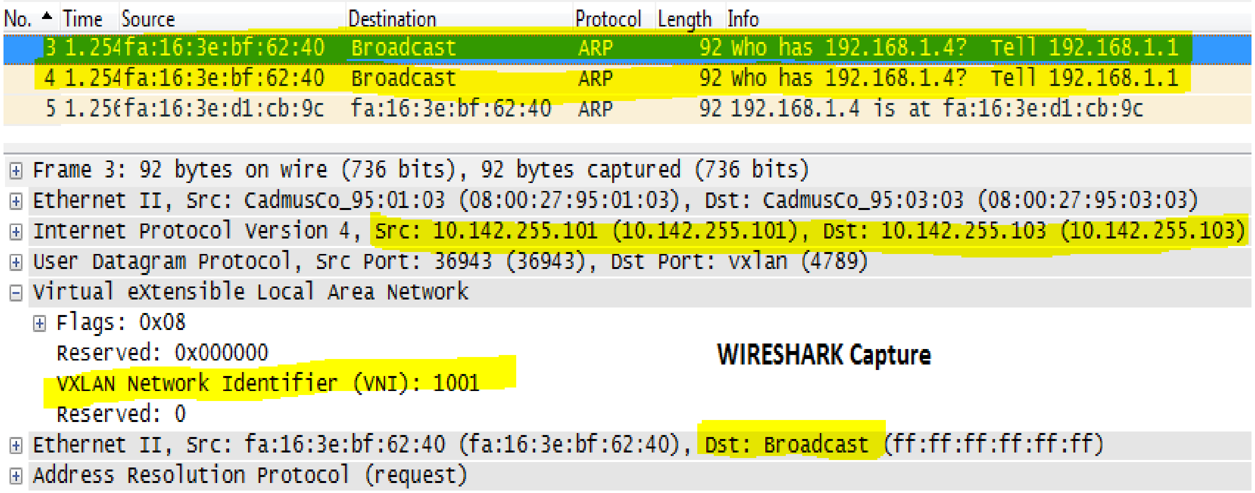
The one bit which has to set to 1 is for valid VNI. All other reserve bits should set to zero except the third bit.

VNI:

The VNI is twenty four bit field that is the VXLAN Identifier.

Reserved:

Twenty four and eight bit fields that are reserved to zero.

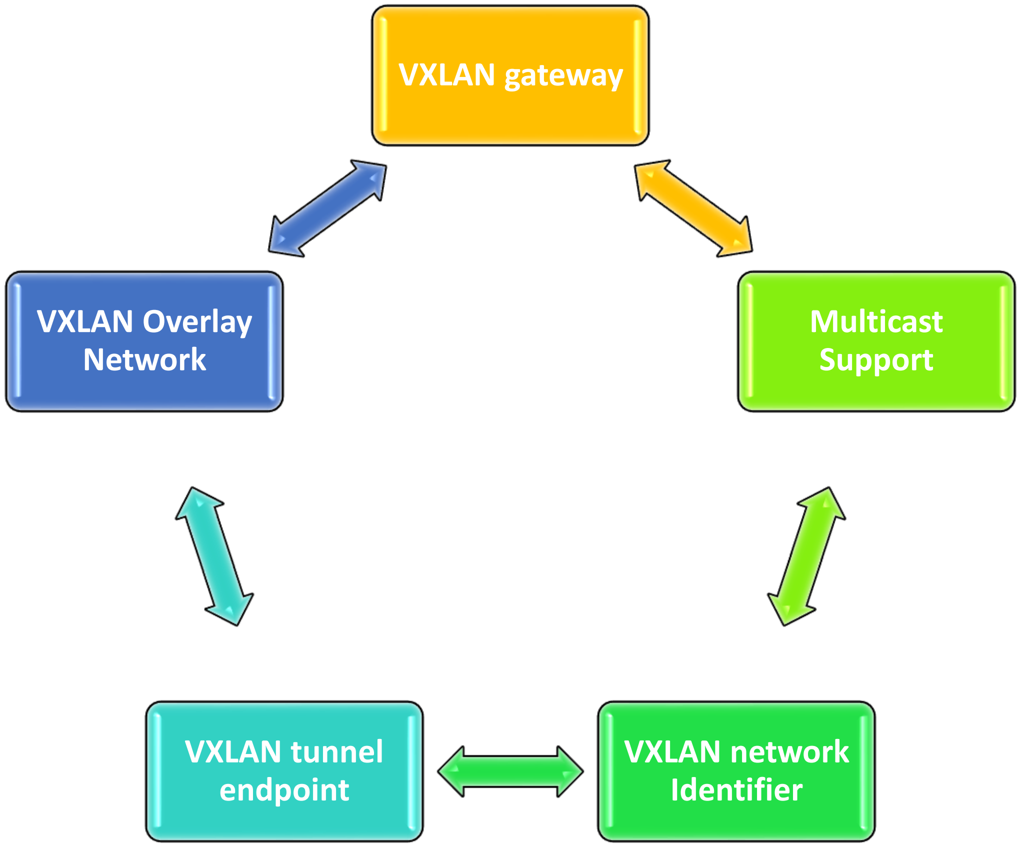


```
No. ▲ Time  Source             Destination        Protocol Length Info
    3 1.254fa:16:3e:bf:62:40   Broadcast          ARP          92 Who has 192.168.1.4?  Tell 192.168.1.1
    4 1.254fa:16:3e:bf:62:40   Broadcast          ARP          92 Who has 192.168.1.4?  Tell 192.168.1.1
    5 1.256fa:16:3e:d1:cb:9c   fa:16:3e:bf:62:40  ARP          92 192.168.1.4 is at fa:16:3e:d1:cb:9c

⊞ Frame 3: 92 bytes on wire (736 bits), 92 bytes captured (736 bits)
⊞ Ethernet II, Src: CadmusCo_95:01:03 (08:00:27:95:01:03), Dst: CadmusCo_95:03:03 (08:00:27:95:03:03)
⊞ Internet Protocol Version 4, Src: 10.142.255.101 (10.142.255.101), Dst: 10.142.255.103 (10.142.255.103)
⊞ User Datagram Protocol, Src Port: 36943 (36943), Dst Port: vxlan (4789)
⊟ Virtual eXtensible Local Area Network
  ⊞ Flags: 0x08
    Reserved: 0x000000
    VXLAN Network Identifier (VNI): 1001
    Reserved: 0
⊞ Ethernet II, Src: fa:16:3e:bf:62:40 (fa:16:3e:bf:62:40), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
⊞ Address Resolution Protocol (request)
```

Figure 5 VXLAN Header

Operation:

To operate a VXLAN we need to have following things aligned:



By combining the VNI and mac addresses make a unique combination for the identification of the VXLAN. The layer-2 packet which is send by virtual machines is actually encapsulated and includes its VNI in its VXLAN header. This packet is than wrapped in a UDP. The UDP packet is then wrapped in a IP packet which then converted to an Ethernet packet for final delivery on the transport network. So, because of this encapsulation the VXLAN is considered kind of tunneling scheme along with ESX hosts which make the tunnel end points (VTEP).The VTEP is responsible for the encapsulation of each and every packet that leaves the Virtual machines in a VXLAN header. The VTEP is also responsible for the de-encapsulation of the VXLAN header on the header side and handing the original layer 2 packet to the virtual machines.

Working and Internal Mechanism:

When one Virtual Machine wants to communicate to another Virtual Machine it need to know its Mac Address. Below is stated the process that is followed:

- ❖ At first the Virtual Machine 1 sends the ARP packet request for the MAC address associated with the IP address 192.168.0.101.

- VTEP 1 will encapsulate that ARP packet into multicast packet to the multicast group having association for VNI 864.
- All other VTEPs will see the packet and put the entry for VTEP 1 and VXLAN in their VXLAN table.
- VTEP 2 will receive the multicast packet and de-encapsulate it for broadcasting it to the port groups associated it to VNI 864.
- Virtual Machine responds with its mac address after seeing the ARP packet.
- VTEP 2 since it has every information in his VXLAN table will encapsulate the responding IP packet as a unicast IP packet and sends it back to VTEP 1 using IP routing.
- VTEP 1 will receive the unicast packet from VTEP 2 , it de-encapsulate the packet and sends it to the Virtual Machine 1.

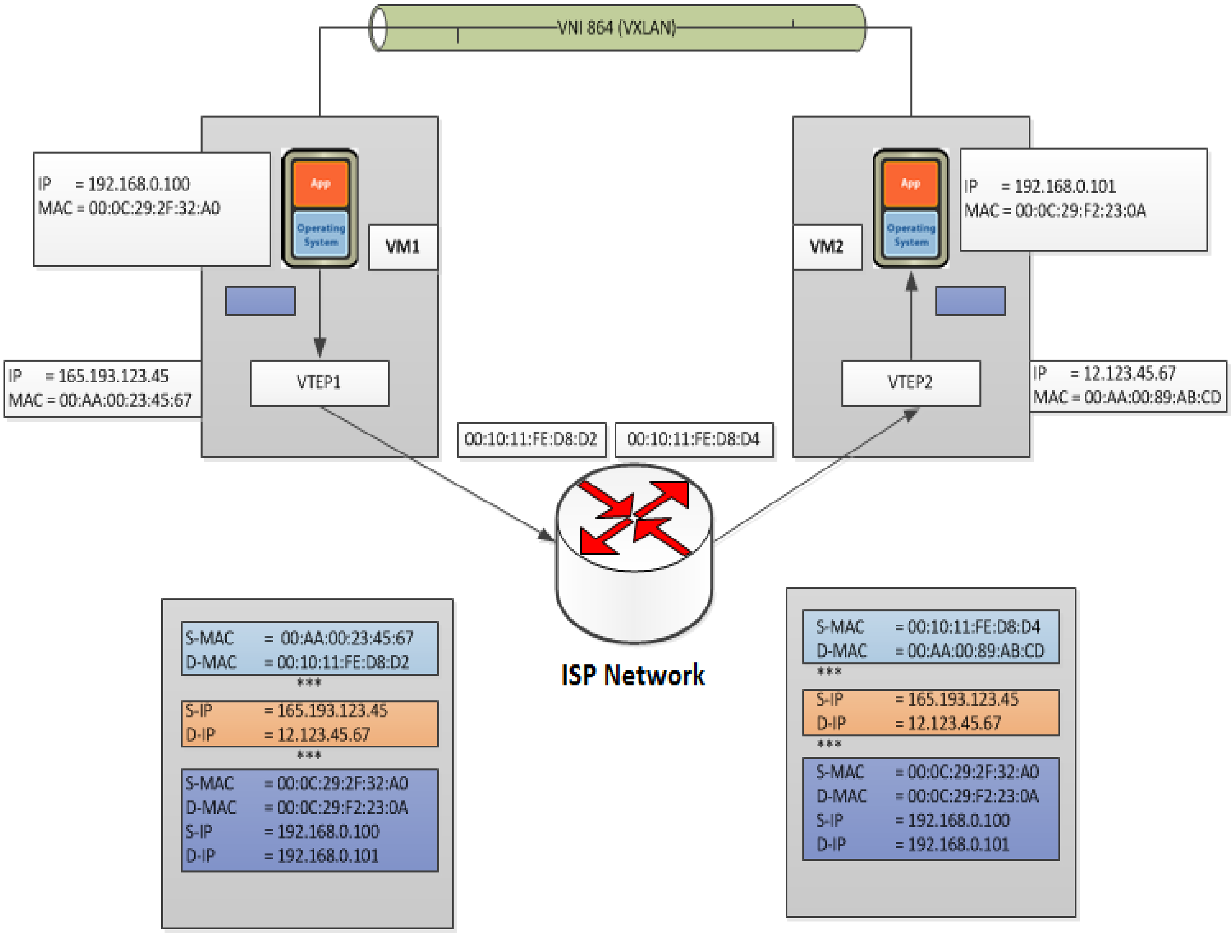


Figure 6 Working and Internal Mechanism of VXLAN

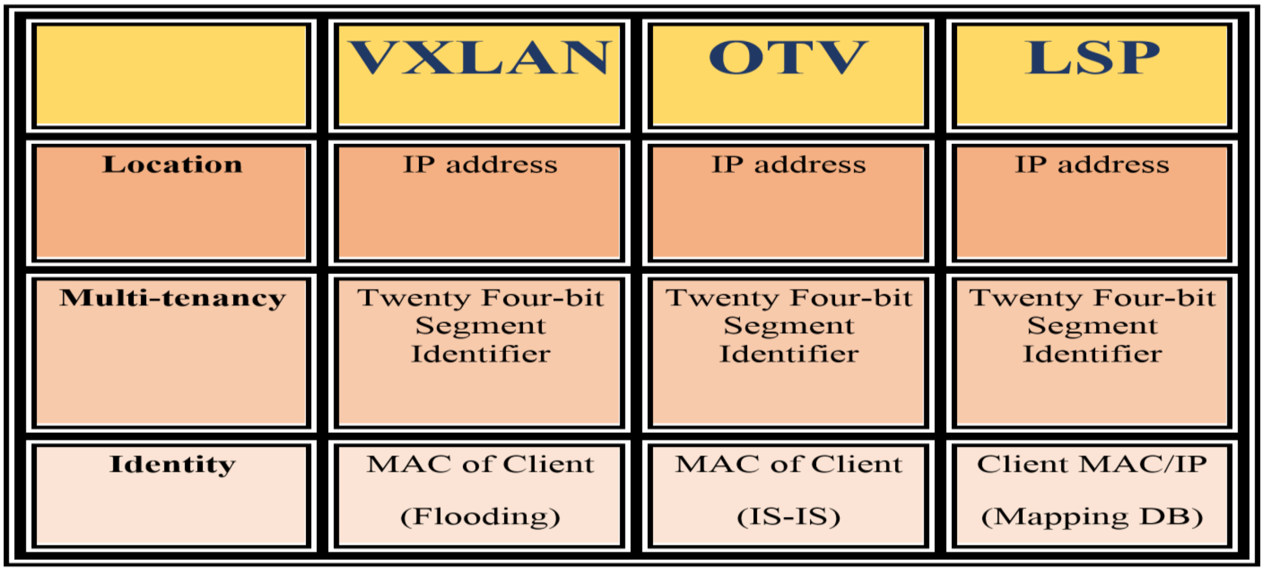
Now at this stage Virtual Machine 1 actually knows the MAC address of Virtual Machine 2 and they can actually exchange the packets directly without any hindrance.Virtual Machine 1 exchanges the ip packet to Virtual Machine 2 from IP address 192.168.0.100 to 192.168.0.101.

VTEP 1 receive the packets and make following amendments in the form of appending headers:

1. It append the VXLAN header with the VNI number of 864.
2. It append the standard UDP header by setting the UDP checksum to 0x0000.It also sets the designation port to IANA given VXLAN designation port.
3. In my topology for this project the Cisco Nexus 1000v will use a port id of 8472.
4. It then append the standard IP header.it will set the IP address of VTEP 2 as a destination and the protocol field is set to 0x011 for the delivery of UDP packet used.

It will append the standard MAC header with the mac address of next hop. In my case it is the router interface. Now the destination MAC address in the packet is for VTEP 2 so it will receive the packet. The VTEP 2 will de-encapsulate the packet and come to know that the packet is a VXLAN packet due to the UDP destination port. The VTEP2 will search the port groups associated for VNI 864 found in VXLAN header. After verifying the target, Virtual Machine 2 in this particular scenario will allow to receive frame because of its port group memberships and pass on the packet after verification.

Virtual Machine 2 will receive the packet and treat it like any other IP packet.

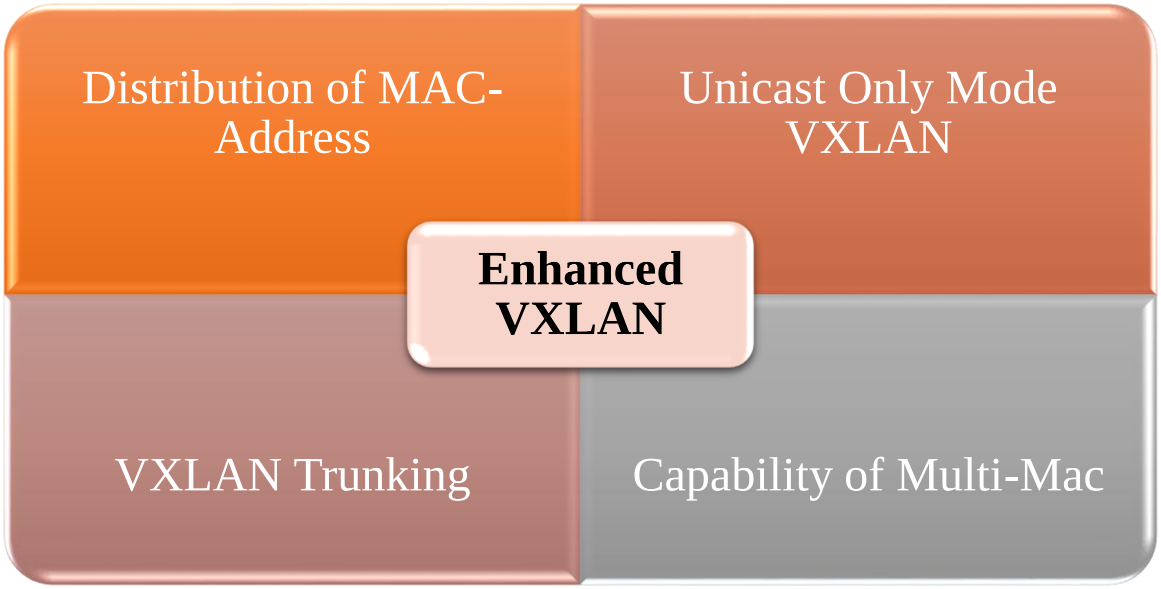


	VXLAN	OTV	LSP
Location	IP address	IP address	IP address
Multi-tenancy	Twenty Four-bit Segment Identifier	Twenty Four-bit Segment Identifier	Twenty Four-bit Segment Identifier
Identity	MAC of Client (Flooding)	MAC of Client (IS-IS)	Client MAC/IP (Mapping DB)

Figure 7 VXLAN Solution

Enhanced VXLAN Sol:

The enhanced VXLAN provides following improvement and innovations:

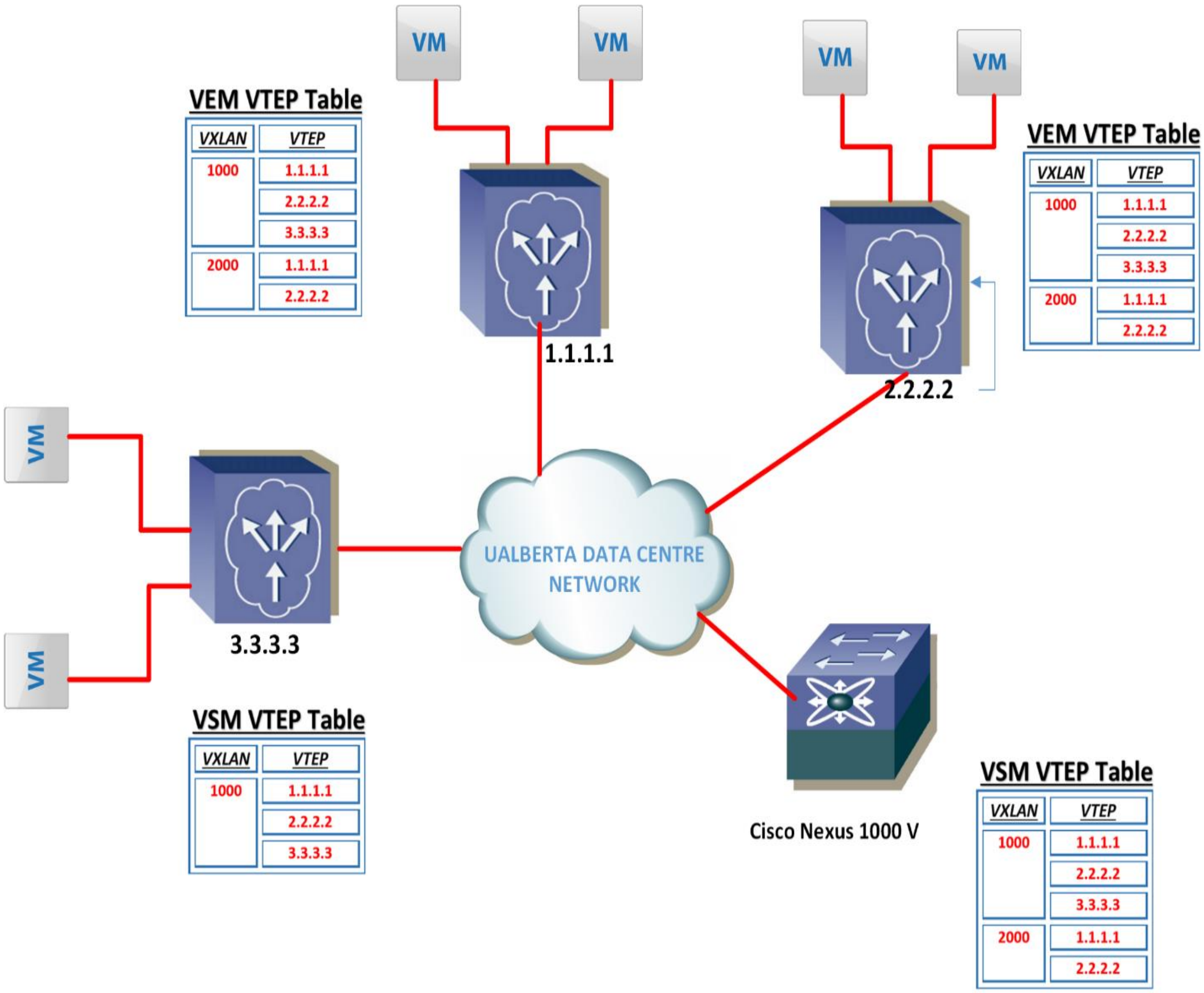


On the contrary to the VXLAN, the enhanced solution of VXLAN gives us the capability either to run only on mac address distribution mode or unicast only mode.

Unicast Only Mode:

Traditional Implementation of VXLAN requires multicast support. Though the VXLAN solves all the issues which have not addressed by OTV,LSP and VPLS even than the many of the industry customers and large enterprises do not want to implement the multicast in their core networks just because of security reasons and company policies. So an alternate was required. To fulfill the needs Cisco System come with a new solution Unicast Only mode of VXLAN which does not rely on the VXLAN and primarily use the Unicast mode to fulfill all the tasks which are need and requirements of today's industry.

When the VMs sends a layer 2 frame in a VXLAN configured in unicast mode, a table lookup is done in the MAC address table using the destination MAC of VXLAN identifier and the frame. If the outcome is a hit, the entry of layer 2 will contain the remote IP address VTEP to use to enclose the frame, and the layer 2 packet will be sent within an IP unicast packet finally destined for the remote VTEP. If the outcome is a miss the frame is copied for each VTEP that has active VMs in exactly the same VXLAN. The replica packet is enclose and is transmit as an IP unicast packet to the VTEPs destination.



VEM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3
2000	1.1.1.1
	2.2.2.2

VEM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3
2000	1.1.1.1
	2.2.2.2

VSM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3

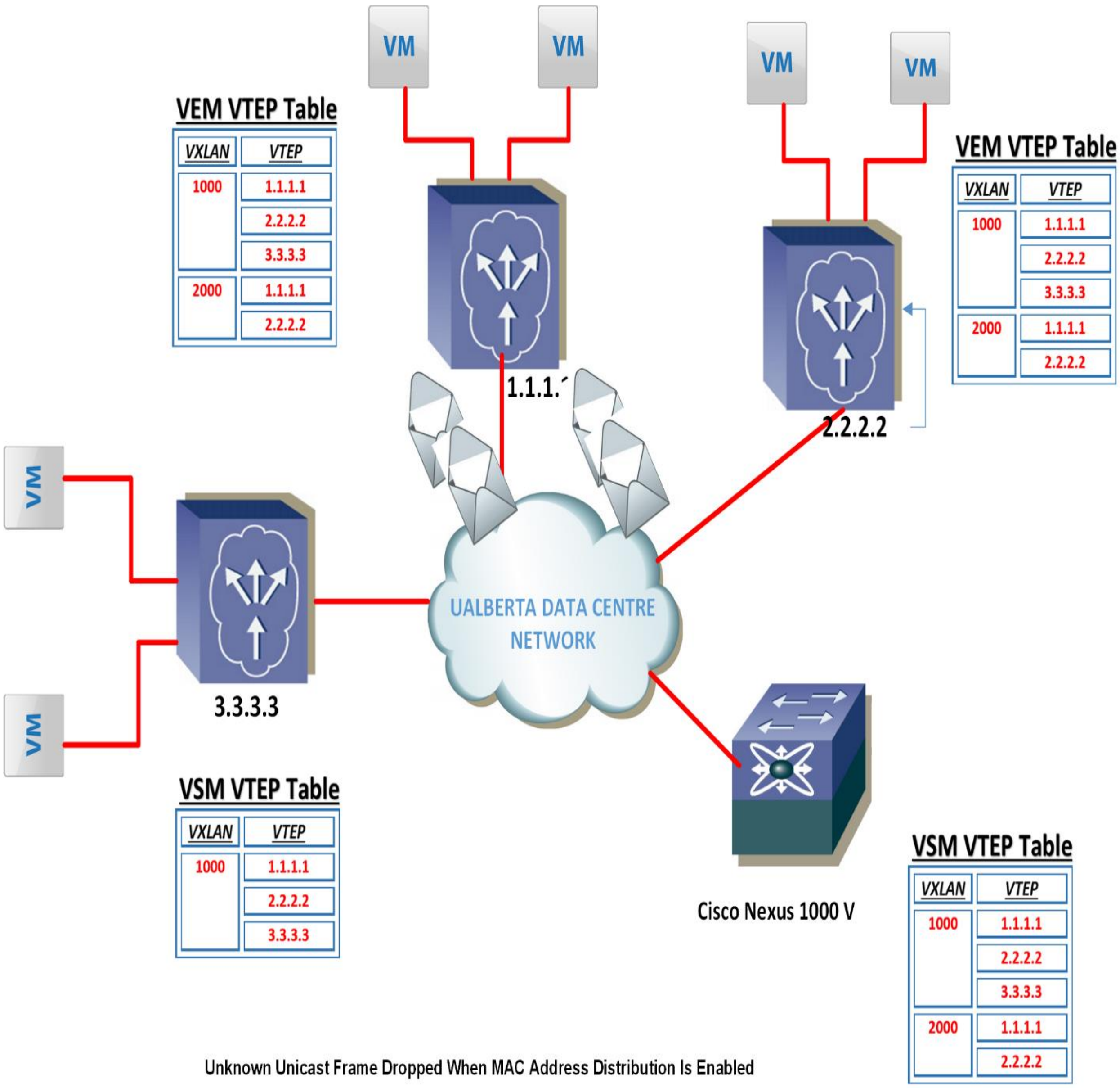
VSM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3
2000	1.1.1.1
	2.2.2.2

Figure 8. VEM, VTEP and VSM Table in VXLAN

The Nexus 1000V VEM attain this efficient replication by making a table of VTEP. When a VM is configured on VXLAN, as the VEM on the host site append VTEP IP and VXLAN entry in this table. The Nexus 1000V VSM keep on aggregates and spread all the VTEPs to all the VEMs for a given

VXLAN. As a result, the Cisco Nexus 1KV VEM must has the VTEP of all the VEMs IP addresses concerned in traffic for a specified VXLAN. Every VEM keeps a VXLAN VTEP table. In the example in Figure, VXLAN 1000 has VMs in VTEPs 1.1.1.1, 2.2.2.2, and 3.3.3.3. VXLAN 2000 has VM only in VTEPs 1.1.1.1 and 2.2.2.2.



VEM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3
2000	1.1.1.1
	2.2.2.2

VEM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3
2000	1.1.1.1
	2.2.2.2

VSM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3

VSM VTEP Table

VXLAN	VTEP
1000	1.1.1.1
	2.2.2.2
	3.3.3.3
2000	1.1.1.1
	2.2.2.2

Unknown Unicast Frame Dropped When MAC Address Distribution Is Enabled

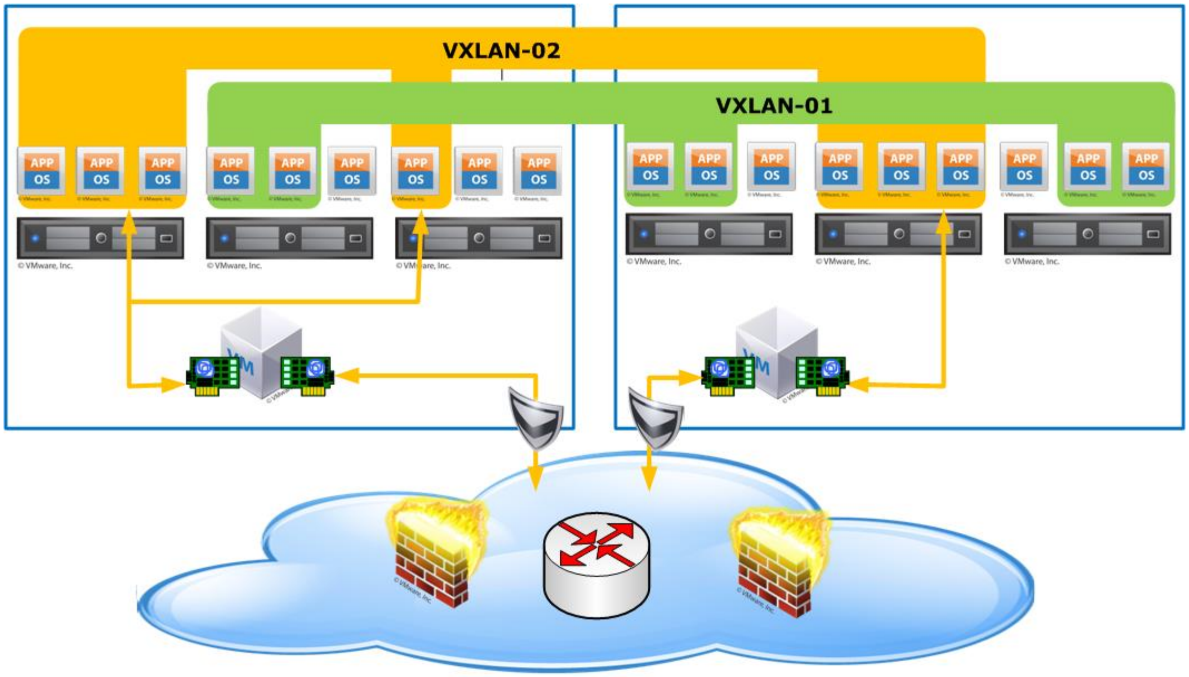
Figure 9.Unknown Unicast Frame dropped when MAC Address Distribution is enabled

Observation and Findings:

VXLAN actually give resolution to the challenges by encapsulation technique of MAC in UDP.The basic use of VXLAN is two connect several layer -3 networks and make them look like that they share the same layer-2 domain. This will actually permit the virtual machines to stay at two different networks and still work and function like as if they are on the same layer-2 domain.in case of unknown unicast instead of broadcasting the layer 2 frame the UDP packet is sent to those virtual machines which reside on the same segment.

The VXLAN solution has following advantages:

- Those virtual machines which reside in different subnets will be extended to one large logical network.
- New Servers can be added in various subnets to make the cloud architecture more scalable and flexible.
- Possibility of migration of virtual machines among servers in various subnets.



Conclusion from Theoretical Overview:

NVGRE offers good benefits in terms of scalability and security of virtualized networks even than it is not as valuable until and unless it maintains the availability of the Central Processing Unit and a similar rate of throughput. In order for NVGRE to be of real value, the extra CPU overhead it creates have to be eliminated. This is the reason the VXLAN and OTV are much predominant in industry as compared to NVGRE. Hence, it is evident that NVGRE has to remove its deficiencies in order to compete with VXLAN and OTV. So, therefore I will perform tests among OTV and VXLAN in order to break the tie between them.

Practical Study of Protocols:

LAB Implementation & Analysis for Test-Scenario 1:

I have implemented the OTV on Cisco Nexus 7000 k to test the features and to see that to how much extent the claims made by Cisco are useful for end customer and how much it is different from VXLAN and its comparative analysis with VXLAN and NVGRE.

Cisco Nexus 7000 Series Switches:

Cisco Nexus 7000 Series Switches actually sets the network fundamentals for our campus core and next-generation Unified Fabric data center. Modular switches like Cisco Nexus 7700 and 7000 Series, possess a excellent Cisco NX-OS feature set and free ware programmable tools for software-defined network (SDN) integrity. They have high-density 40, and 100 Gigabit Ethernet with performance analytics and application awareness.

The Cisco Nexus 7000 Series is designed based three principles:

Scalability of Infrastructure:

Virtualization, high density, cloud scale with automation, efficient power and cooling and performance for all sort of data centers.

Continuity of Operations:

The design integrates NX-OS software features, hardware and management to support outage free environments in data center.

Flexibility for Transport:

We can cost-effectively shift to new networking technologies.

SDN Support, Programmability Operations:

Open-source tools and platforms for open standards prominently boost Cisco Nexus 7000 SDN capabilities and programmability for cloud deployments and virtualize environments.

Topolgy for Test Scenario 1

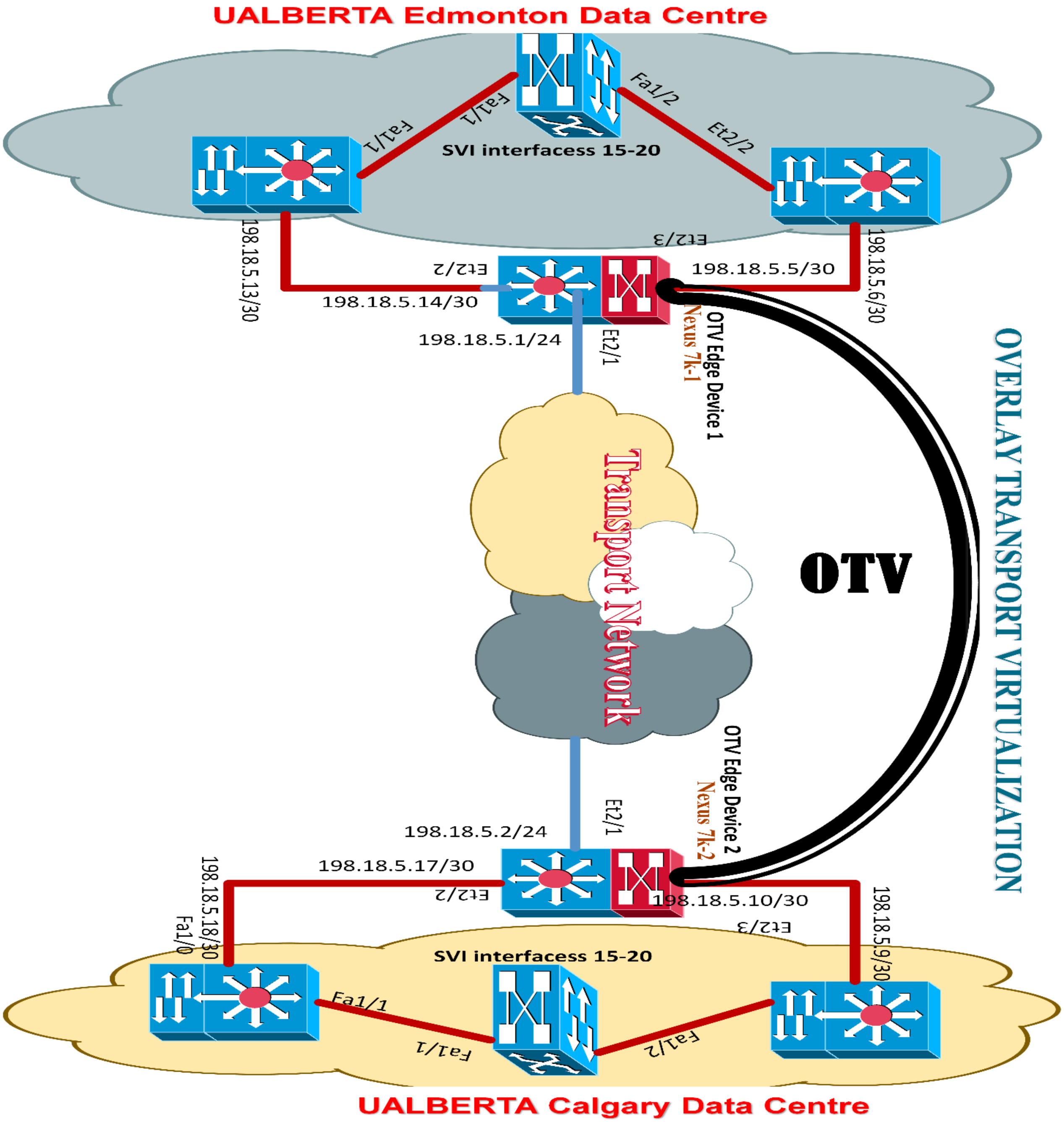


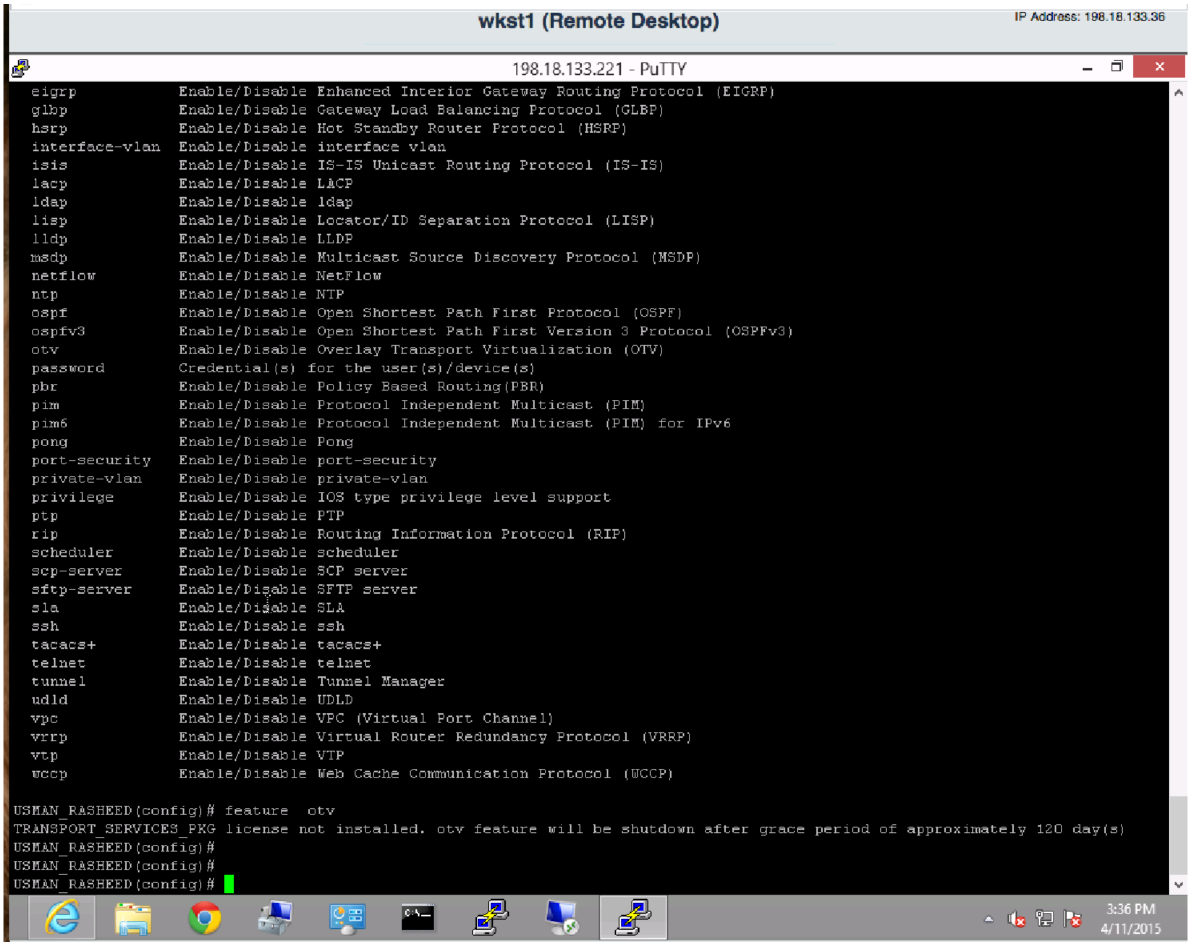
Figure 10 Physical Topology for Test Scenario 1

__NOTE:__ The names "UALBERTA Edmonton Data Centre" and "UALBERTA Calgary Data Centre" are fictitious and only used for understanding purposes.

OTV Configuration Steps on Nexus 7k-1:

Step 1:
OTV Feature Enable on Nexus 7k-1:
First we have to on this feature. For this we have to give the following commands.

USMAN_RASHEED(config)# feature otv



```
  eigrp            Enable/Disable Enhanced Interior Gateway Routing Protocol (EIGRP)
  glbp             Enable/Disable Gateway Load Balancing Protocol (GLBP)
  hsrp             Enable/Disable Hot Standby Router Protocol (HSRP)
  interface-vlan   Enable/Disable interface vlan
  isis             Enable/Disable IS-IS Unicast Routing Protocol (IS-IS)
  lacp             Enable/Disable LACP
  ldap             Enable/Disable ldap
  lisp             Enable/Disable Locator/ID Separation Protocol (LISP)
  lldp             Enable/Disable LLDP
  msdp             Enable/Disable Multicast Source Discovery Protocol (MSDP)
  netflow          Enable/Disable NetFlow
  ntp              Enable/Disable NTP
  ospf             Enable/Disable Open Shortest Path First Protocol (OSPF)
  ospfv3           Enable/Disable Open Shortest Path First Version 3 Protocol (OSPFv3)
  otv              Enable/Disable Overlay Transport Virtualization (OTV)
  password         Credential(s) for the user(s)/device(s)
  pbr              Enable/Disable Policy Based Routing(PBR)
  pim              Enable/Disable Protocol Independent Multicast (PIM)
  pim6             Enable/Disable Protocol Independent Multicast (PIM) for IPv6
  pong             Enable/Disable Pong
  port-security    Enable/Disable port-security
  private-vlan     Enable/Disable private-vlan
  privilege        Enable/Disable IOS type privilege level support
  ptp              Enable/Disable PTP
  rip              Enable/Disable Routing Information Protocol (RIP)
  scheduler        Enable/Disable scheduler
  scp-server       Enable/Disable SCP server
  sftp-server      Enable/Disable SFTP server
  sla              Enable/Disable SLA
  ssh              Enable/Disable ssh
  tacacs+          Enable/Disable tacacs+
  telnet           Enable/Disable telnet
  tunnel           Enable/Disable Tunnel Manager
  udld             Enable/Disable UDLD
  vpc              Enable/Disable VPC (Virtual Port Channel)
  vrrp             Enable/Disable Virtual Router Redundancy Protocol (VRRP)
  vtp              Enable/Disable VTP
  wccp             Enable/Disable Web Cache Communication Protocol (WCCP)

USMAN_RASHEED(config)# feature  otv
TRANSPORT_SERVICES_PKG license not installed. otv feature will be shutdown after grace period of approximately 120 day(s)
USMAN_RASHEED(config)#
USMAN_RASHEED(config)#
USMAN_RASHEED(config)#
```

Figure 11

Step 2:
VLANS Creation on Nexus 7k-1:

We have to make VLANs that is going to be extended among sites across the Overlay Network.

# vlan 2,5-10

Step 3:
Creation of OTV Site VLAN on Nexus 7k-1:

It is always good to use a dedicated VLAN. The definedSite VLAN should not be extended on the other side of the overlay networks. So, that same Site VLAN can be used at both sites. If you will not create any site vlan than by default VLAN 1 will be used as OTV site vlan. otv site-vlan

Step 4:
Configuration of Join Interface on Nexus 7k-1:

```
interface ethernet 2/1

 ip address 198.18.5.1/24

 ip igmp version 3

 no shutdown
```

Step 5:
Creation of Overlay Interface on Nexus 7k-1:

```
otv site-identifier 256

interface Overlay1

 otv control-group 239.1.1.1

 otv data-group 232.1.1.0/28

 otv join-interface ethernet 2/1
```

Step 6:
Addition of Extended VLANs on Nexus 7k-1:

```
otv extend-vlan 5-10

 no shutdown
```

Step 7:
Verify Connectivity on Nexus 7k-1:

```
n7k-1#ping 198.18.5.2
```

OTV Configuration Steps on Nexus 7k-2:

Step 1:
OTV Feature Enable on Nexus 7k-2:

```
n7k-2(config)# feature otv
```

Step 2:
VLANS Creation on Nexus 7k-2:

```
# vlan 2,5-10
```

Step 3:
Creation of OTV Site VLAN on Nexus 7k-2:

```
otv site-vlan 2
```

Step 4:
Configuration of Join Interface on Nexus 7k-2:

```
interface ethernet 2/1

 ip address 198.18.5.2/24

 ip igmp version 3

 no shutdown
```

Step 5:
Creation of Overlay Interface on Nexus 7k-2:

```
otv site-identifier 256

interface Overlay1

 otv control-group 239.1.1.1

 otv data-group 232.1.1.0/28
 otv join-interface ethernet 2/1
```

Step 6:
Addition of Extended VLANs on Nexus 7k-2:

```
otv extend-vlan 5-10

 no shutdown
```

LAB Implementation & Analysis for Test-Scenario 2:

I have divided the lab implementation in following phases:

- ➔ Connection between VXLAN and VMs
- ➔ Configuration of Upstream device
- ➔ Configuration of Cisco Nexus 1000V
- ➔ Troubleshooting for VXLAN Functions

Non-Disruptive Operational Model:

The key feature of this lab is that it has non-disruptive operational models for both server administration and network administration. This feature is actually supported by Cisco nexus 1KV.So, this means that in practical models where Cisco Nexus 1K is going to implement it gives own management perspective with their own tools and views to network engineers and VMware engineers.

I have purposely implement this lab in a manner in which I can test both of these prospective which have been claimed by Cisco. These prospective are:

- Cisco NX-OS will be used as a primary management tool for Network Administration
- vCentre will be the primary management tool for VMware Administration Prospective

I have done it so that I can better test and expose the every prospective of Cisco Nexus 1000v.

Lap Equipment:

The following equipment has been used:

- ✷ 2 virtual VMware ESX servers.

- 1 VMware vCenter, can be reachable by vshpere client
- 1 Cisco Nexus 1000V Virtual Supervisor
- 1 upstream switch

Topology for Test Scenario 2

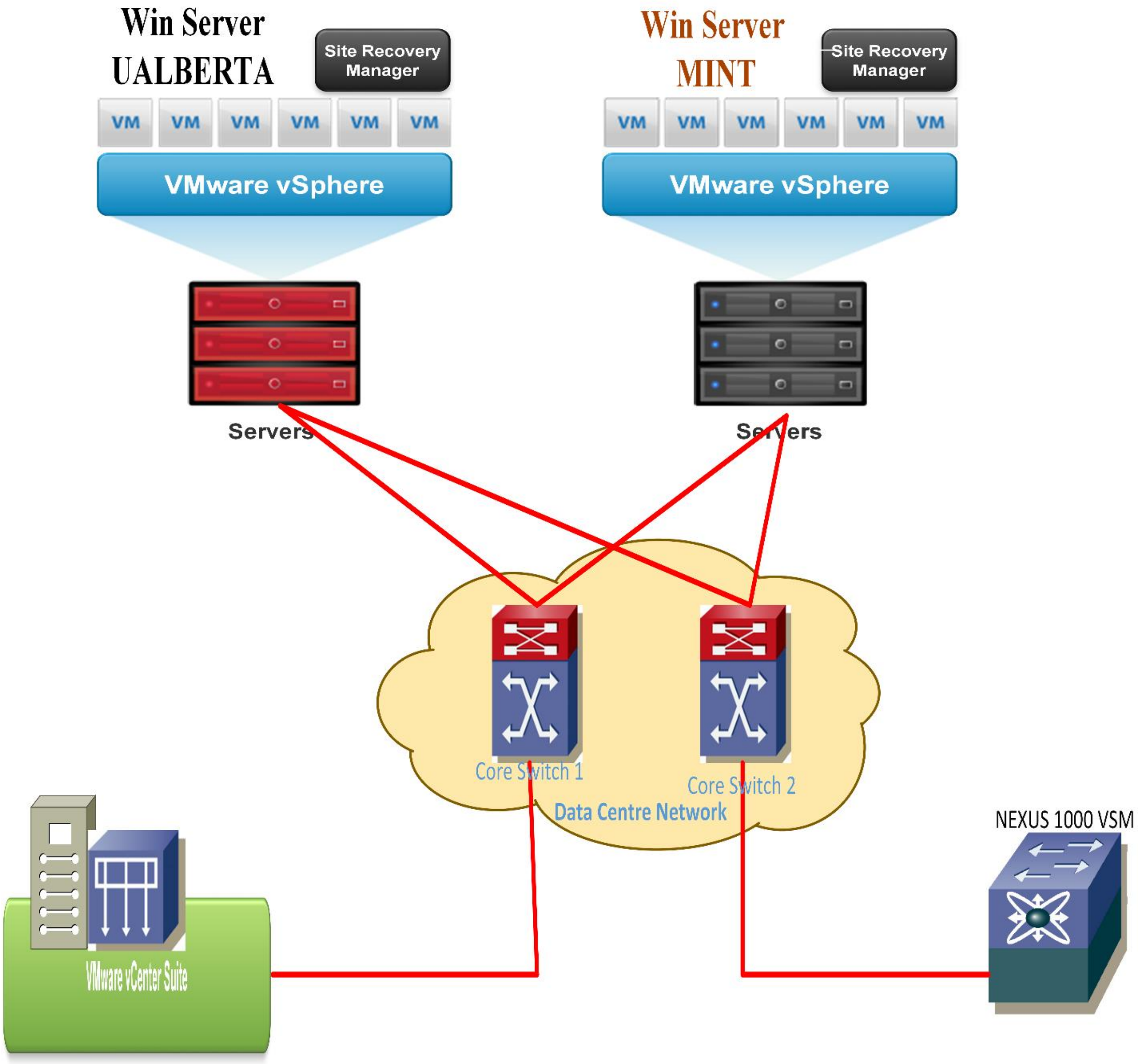


Figure 12

Strategy Note (Challenge):

Please note that I deliberately put both ESX hosts on different subnets. This will show us one of the key differentiators about VXLAN.

PROCEDURE:

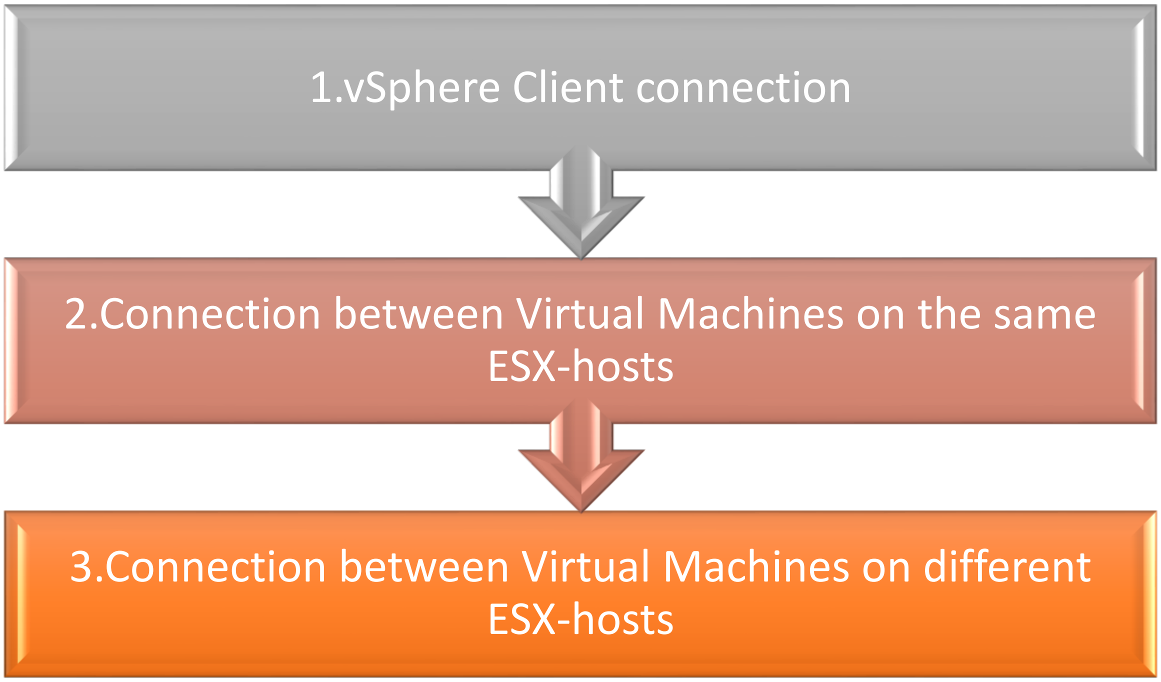
At first the lab configuration was performed on the VMware vCentre. The vSphere client was used in order access the VMware vCentre. SSH was used to access VSM.

Step A:

Connection of VXLAN and connected Virtual Machines:

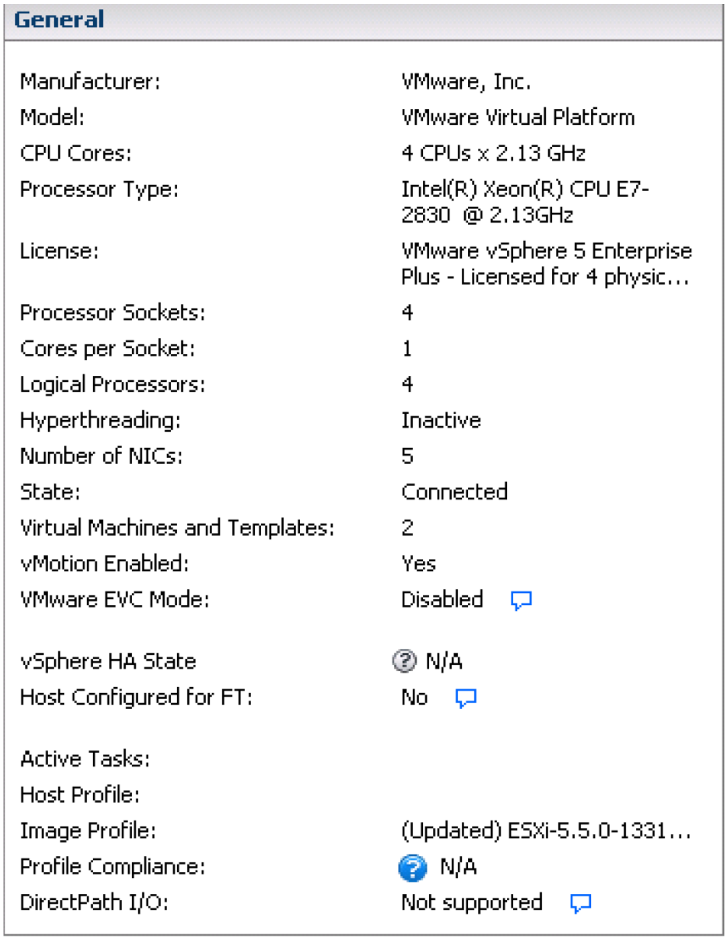
ESX hosts VMKernel ports used for VXLAN connectivity reside in different subnets. This exercise will demonstrate L2 connectivity between the VMs WebServer, Win Server MINT and Win Server UALBERTA using ping.

In order for the connection between VXLAN and VMS, I have followed these steps:


1.vSphere Client connection
2.Connection between Virtual Machines on the same ESX-hosts
3.Connection between Virtual Machines on different ESX-hosts

1. vSphere Client Connection

At first before any other connections I checked the connection of vSphere from GUI based client:



General

Manufacturer:	VMware, Inc.
Model:	VMware Virtual Platform
CPU Cores:	4 CPUs x 2.13 GHz
Processor Type:	Intel(R) Xeon(R) CPU E7-2830 @ 2.13GHz
License:	VMware vSphere 5 Enterprise Plus - Licensed for 4 physic...
Processor Sockets:	4
Cores per Socket:	1
Logical Processors:	4
Hyperthreading:	Inactive
Number of NICs:	5
State:	Connected
Virtual Machines and Templates:	2
vMotion Enabled:	Yes
VMware EVC Mode:	Disabled
vSphere HA State	N/A
Host Configured for FT:	No
Active Tasks:	
Host Profile:	
Image Profile:	(Updated) ESXi-5.5.0-1331...
Profile Compliance:	N/A
DirectPath I/O:	Not supported

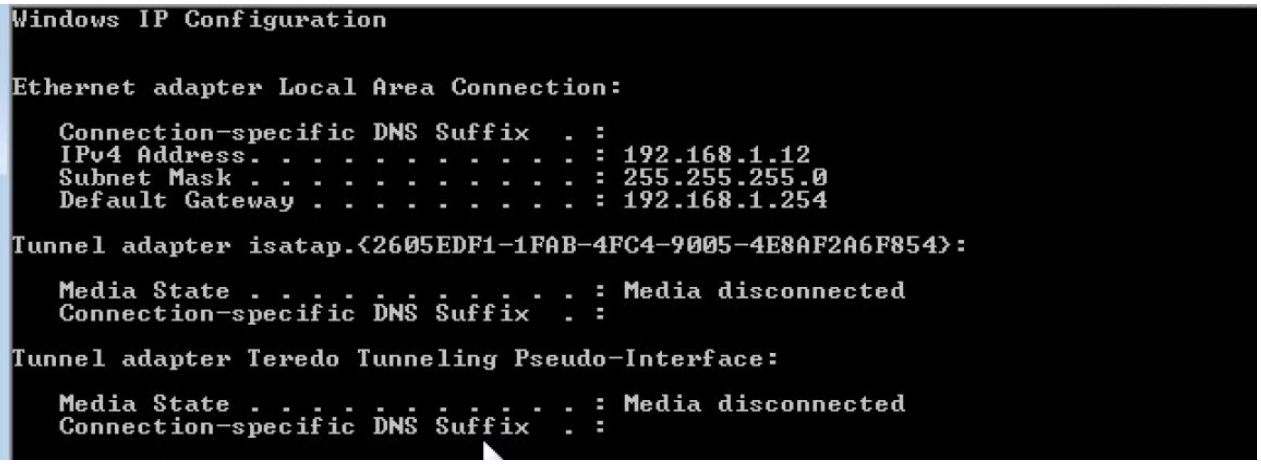
Figure 13 vSphere Client Connection

2. Verify network connectivity between VMs on the same ESX-hosts:

Both VMs *WebServer* and Win Server UALBERTA reside on the same ESX host. Network traffic between these VMs will therefore not traverse the physical network –

1. I have first verified the Virtual Machine Win Server UALBERTA that this VM is connected to the port-profile called *VXLAN_Network*. We will later verify that this is a VXLAN backed port profile. Note: I later did the verification that this is actually VXLAN backed port profile.

2.Within the VM open the command prompt and lookup the VM’s IP address with the command *ipconfig*



```
Windows IP Configuration

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.12
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.254

Tunnel adapter isatap.{2605EDF1-1FAB-4FC4-9005-4E8AF2A6F854}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Notice that the IP address is within the 192.168.1.0/24 network.

3.Within the VM, test connectivity to the Webserver VM.

Use the command *ping web-server-a* to do so.

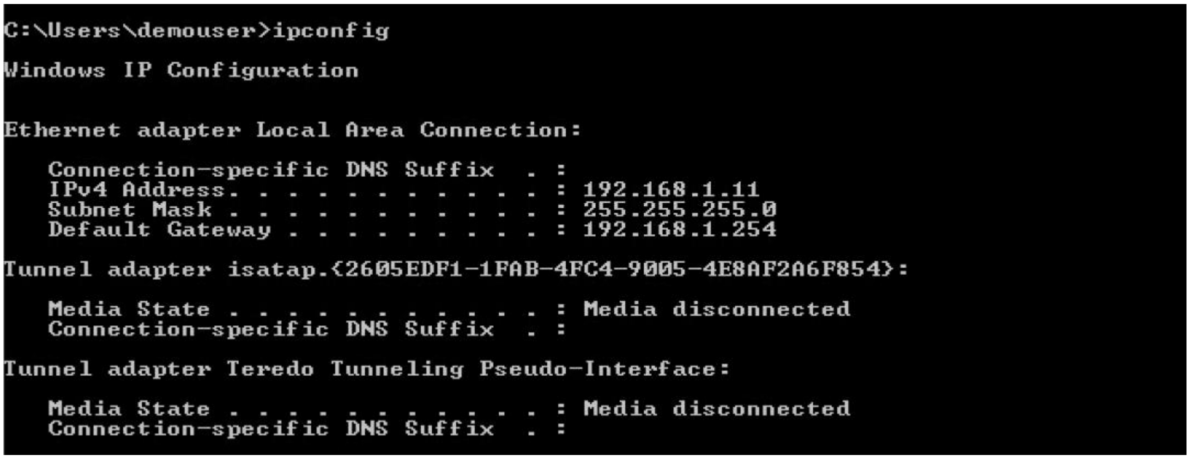
```
Pinging web-server-a [192.168.1.100] with 32 bytes of data:
Reply from 192.168.1.100: bytes=32 time=2ms TTL=64
Reply from 192.168.1.100: bytes=32 time=1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64
Reply from 192.168.1.100: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.100:
  Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
```

3. Verify network connectivity between VMs on different ESX-hosts:

As the Virtual Machines *WebServer* and Win Server MINT both rather than same ESX Host actually located on different ESX host . So, packets generated by the network between these VMs will therefore traverse the physical network.

So because of this I got a need to verify that these two Virtual Machines can ping each other via the VXLAN_Network port-profile.

1. I have first verified the Virtual Machine Win Server MINT that this VM is connected to the port-profile called *VXLAN_Network*. We will later verify that this is a VXLAN backed port profile.
2. Within the VM open the command prompt and lookup the VM’s IP address with the command *ipconfig*



```
C:\Users\demouser>ipconfig
Windows IP Configuration

Ethernet adapter Local Area Connection:

   Connection-specific DNS Suffix  . :
   IPv4 Address. . . . . . . . . . . : 192.168.1.11
   Subnet Mask . . . . . . . . . . . : 255.255.255.0
   Default Gateway . . . . . . . . . : 192.168.1.254

Tunnel adapter isatap.{2605EDF1-1FAB-4FC4-9005-4E8AF2A6F854}:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :

Tunnel adapter Teredo Tunneling Pseudo-Interface:

   Media State . . . . . . . . . . . : Media disconnected
   Connection-specific DNS Suffix  . :
```

Notice that the IP address is within the 192.168.1.0/24 network

Step B:
Upstream Device Configurations

Though VXLAN allows connection among Virtual Machine, where the hosts of ESX are not connected via Layer -2, the upstream network have to attain some specific requirements like **ProxyARP** support.

As I used Layer 3 connection between VMKernel ports so VXLAN requires ProxyARP to be configured for every Layer 3 interface the VMKernel ports will have connection to. So, I configured it via CLI commands to fulfill the requirement.

Verify the interface configuration

As it is apparent from the figure, the two int connected to my ESX hosts VMKernel VXLAN interfaces are Giga1 and Giga2.

Both of these interfaces are Layer 3 ports and are configure by default for Proxy ARP .It is a mechanism by which the interface replies the ARP queries for a net address that is not present on that network. The ARP Proxy is fully aware of the place for traffic destination, and gives its own MAC-address in answer.

Step C:
Cisco Nexus (Unicast-only mode) Configuration:

1. Bridge-Domain per VXLAN segment:

I configured Bridge-domain because it needs to be configured for every segment of a VXLAN, This actually created a bonding between the numeric Nexus 1000V internal name and VXLAN ID. Implementing enhanced VXLAN is straightforward as by default unicast-only mode is configured.

2. VMKernel port-profile:

The VEM which stand for Virtual Ethernet Module in every ESX hosts encapsulates traffic from VXLAN backed port-profiles into User Datagram Protocol and sends it to the corresponding target ESX host. VMKernel interface is used for one ESX host to another ESX host communication. So, therefore I made a a special port profile on Cisco Nexus 1Kv.

3. VXLAN-backed port-profile:

VXLAN-backed port-profile is the port-profile on which Virtual Machines will be connected. All VMs will communicate with each other via VXLAN which are going to connect on this port-profile. Each port-profile will only belong to a single VXLAN.

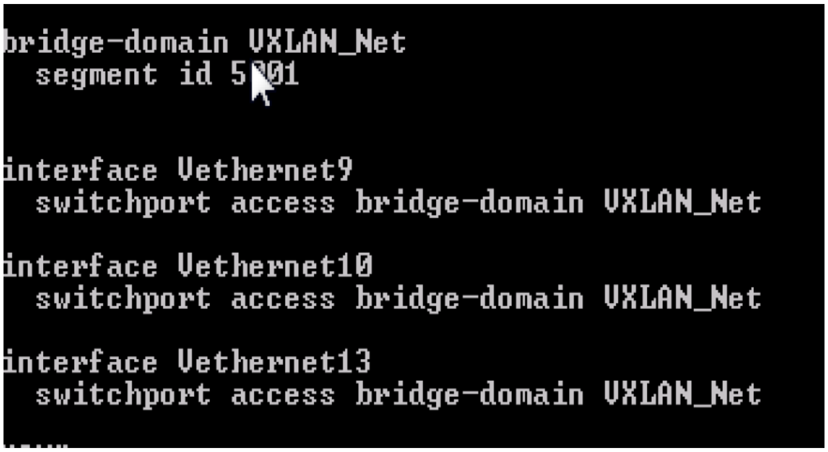
I have configured all of these configuration on Nexus. It was challenging. To achieve this I performed these steps:

- Bridge-domain setup
- VXLAN VMKernel port-profile
- VXLAN-backed port-profile

1. Bridge-domain setup:

The bridge-domain configures a logical binding between the numeric VXLAN ID and a Nexus 1000V internal name. Let’s verify that everything is setup as expected.

Just for demonstration when I gave the command show running-config bridge-domai following output displayed:



```
bridge-domain VXLAN_Net
  segment id 5001


interface Vethernet9
  switchport access bridge-domain VXLAN_Net

interface Vethernet10
  switchport access bridge-domain VXLAN_Net

interface Vethernet13
  switchport access bridge-domain VXLAN_Net
```

Note: segment mode unicast-only will actually determine that VXLAN will be on UNICAST ONLY domain.

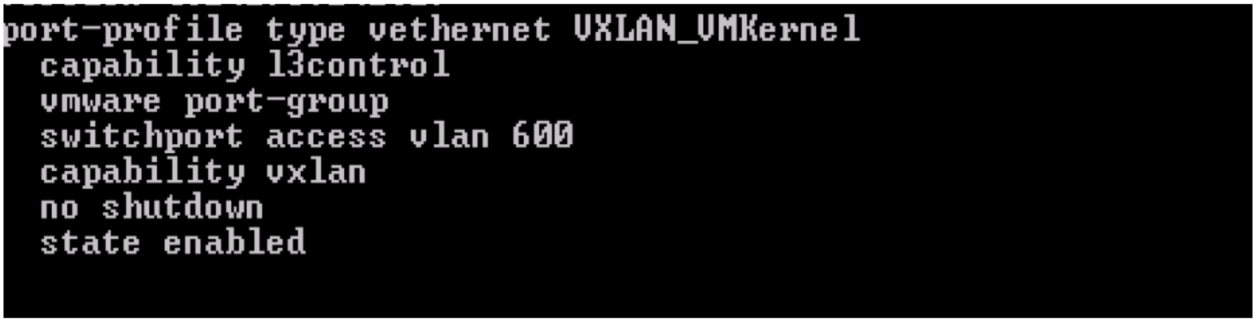
Before any further VXLAN configuration can be performed the command feature segmentation enables the VXLAN feature.It has to be given in any case on Cisco 1Kv before going further.

The command bridge-domain creates a Cisco Nexus 1000V internal VXLAN configuration with the name VXLAN_Net and binds the VXLAN ID to it.

2. VMKernel port-profile for VXLAN:

The VEM which stand for Virtual Ethernet Module in every ESX hosts encapsulates traffic from VXLAN backed port-profiles into User Datagram Protocol and sends it to the corresponding target ESX host. VMKernel interface is used for one ESX host to another ESX host communication. So, therefore I made a a special port profile on Cisco Nexus 1Kv.

show running-config port-profile VXLAN_VMKernel. The output was this :



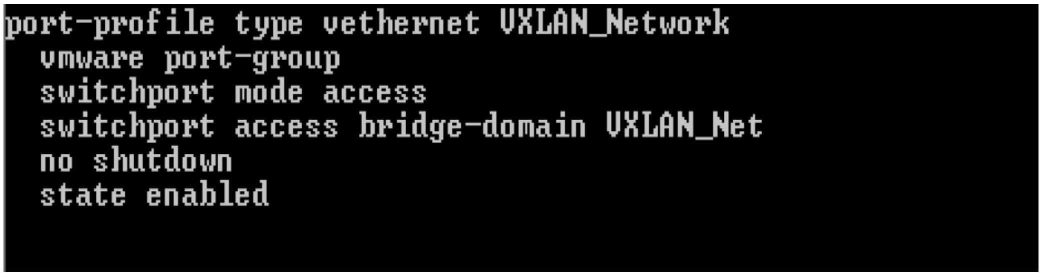
```
port-profile type vethernet VXLAN_VMKernel
  capability l3control
  vmware port-group
  switchport access vlan 600
  capability vxlan
  no shutdown
  state enabled
```

Within vCenter verify that one of the VMKernel ports of the ESX hosts is connected to the port-profile *VXLAN_VMKernel*. Click on the ESX host *vesx1.dcloud.cisco.com*, choose the *Configuration* tab and in the Hardware section the menu item *Networking*. Chose the *vNetwork Distributed Switch* view and click on *Manage Virtual Adapters*. Notice that *vmk4* is connected to the port-profile *VXLAN_VMKernel*.

3.VXLAN_Network port-profile:

The last task is configuring of the VXLAN_Network port-profile on the Nexus 1KV VSM and that VMs which I have used before are connected to this port-profile.

Command *show running-config port-profile VXLAN_Network*. The output will look as follows.



```
port-profile type vethernet VXLAN_Network
  vmware port-group
  switchport mode access
  switchport access bridge-domain VXLAN_Net
  no shutdown
  state enabled
```

Port-profile *VXLAN_Network does not use VLAN ID* even the port-profile *VXLAN_Network* is an access port. In place of this, it actually uses the internal VXLAN identifier *VXLAN_Net*.

Within vCenter I verified that the Virtual Machine *WebServer*, Win Server UALBERTA and Win Server MINT all of them connected to the port-profile VXLAN_Network. Note that this Virtual Machine has a connection to the port-profile *VXLAN_Network*

DEMAND	INTER DATA CENTRE COMMUNICATION	INTERA DATA CENTRE COMMUNICATION
Layer-2 Connection Requirement	VXLAN	OTV/VPLS
Mobilization	LISP	LISP
Security	VXLAN	LAYER-3 VPNS

Conclusion

OTV is a Datacenter Interconnect technology that extends layer 2 between geographically separated DCs. It can be clearly observed from the tests of this report that OTV is more powerful in terms of packet routing, traffic directed in/out/around or across the network because it runs on the physical network equipment. This results in best utilization of Data Centre inter communication. It reduces Traffic Tromboning as well as providing excellent redundancy and efficient failover at the same time as compared to VXLAN and NVGRE. But there are potential problems associated with OTV as discussed earlier like hardware dependencies, broadcast suppression and the fact that VLAN SVI's can't exist on the same VDC or device as OTV transport of those VLANs. NVGRE on the other hand cannot do load balancing effectively and utilize CPU efficiently. These problems of OTV and NVGRE are so relentless and severe for existing Data Centre infrastructure that they cannot be overlooked. Contrary to the OTV and NVGRE, the VXLAN deployment is much simpler and no hardware dependencies are involved. It can be observed from lab tests that one should not have to touch existing designed Layer-3 data center network to implement VXLAN functionalities. Moreover, VXLAN gives complete separation between the physical network and virtualized segments. VXLAN Unicast mode also solves the concern for those who are not willing to implement multicast in their core network. Above all, it provides excellent load sharing of links.

Recommendation:

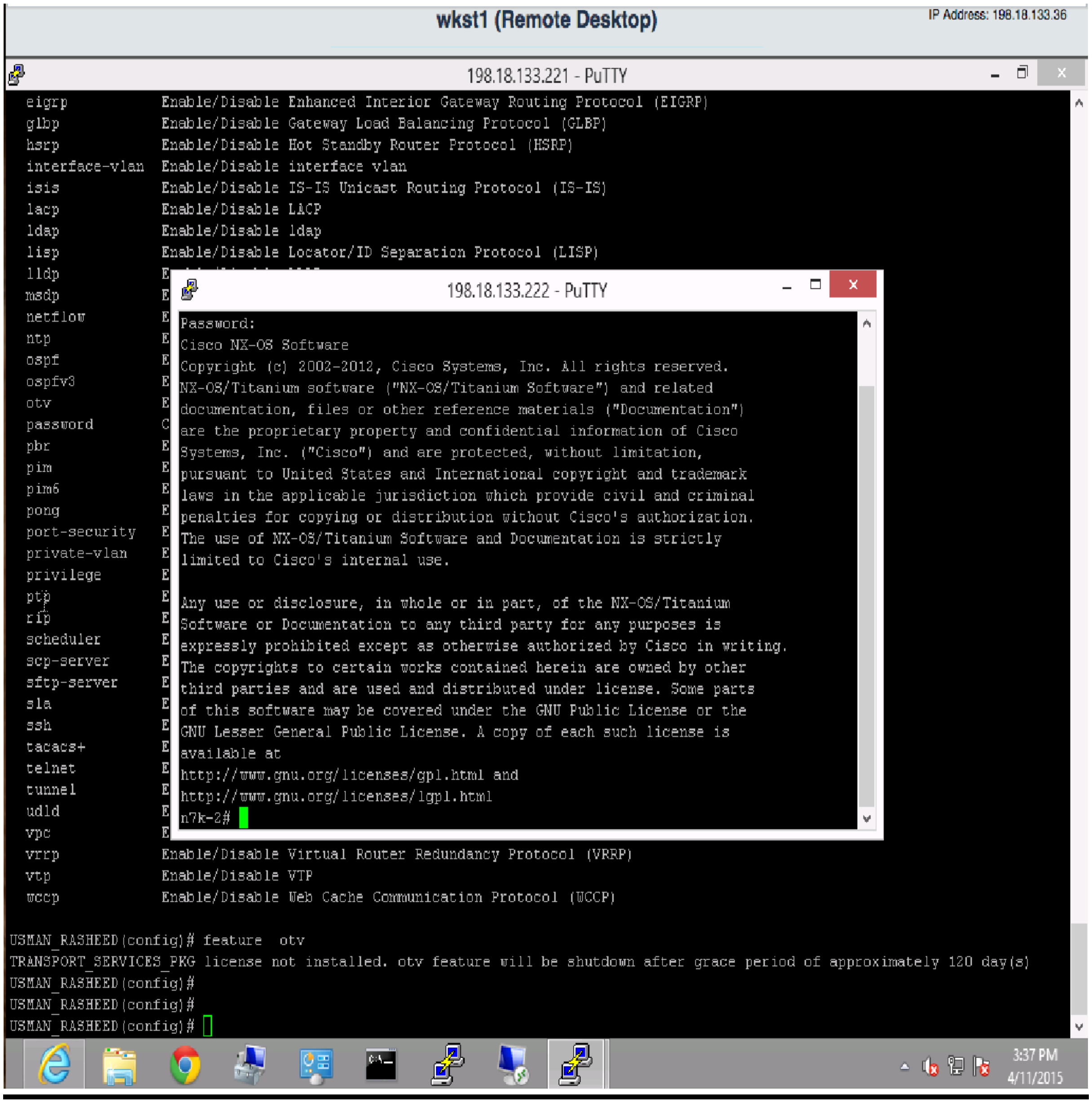
In short, at the moment, OTV, NVGRE, and VXLAN are living in different parallel universes. Unfortunately, right now we cannot merge all of these technologies to gain the best of all worlds. Based on my tests and findings it is clearly evident that VXLAN is highly recommendable to solve inter-Data Centre mobility issues better than any other Overlay Protocol available. The advantages are clear.

1. It can solve the security problems that arise due to VLAN groupings.
2. It gives the feature to make new networks on the fly when used in combination with different VMware solutions.
3. It solves the issue for those customers who do not want to enable multicast in their network domain.
4. Totally hardware independent solution.
5. Excellent Load Sharing of links
6. It deceives Virtual Machines into a notion that they are part of one flat, large network, and it's tremendously scalable. Ideal and excellent for massive virtual networks.

National Science Foundation predicts that internet traffic will increase to five billion by 2020. With such a huge and massive increase, we need new solutions for large virtual networks on urgent basis. I think the VXLAN protocol is the start of a new revolution in the field of software-driven networks.

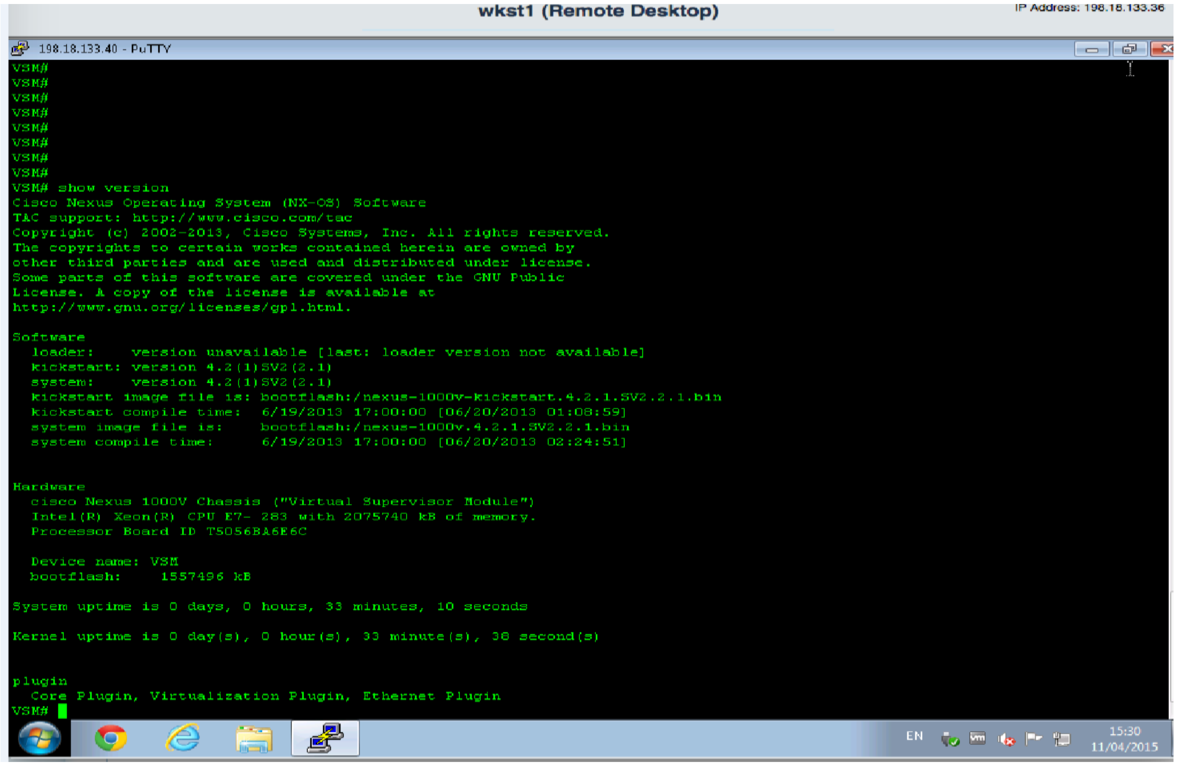
Appendices:

Rather than putting the running configuration here. I am putting in the separate folder for the ease. Which I will attach with the report.
Some random screen shots.

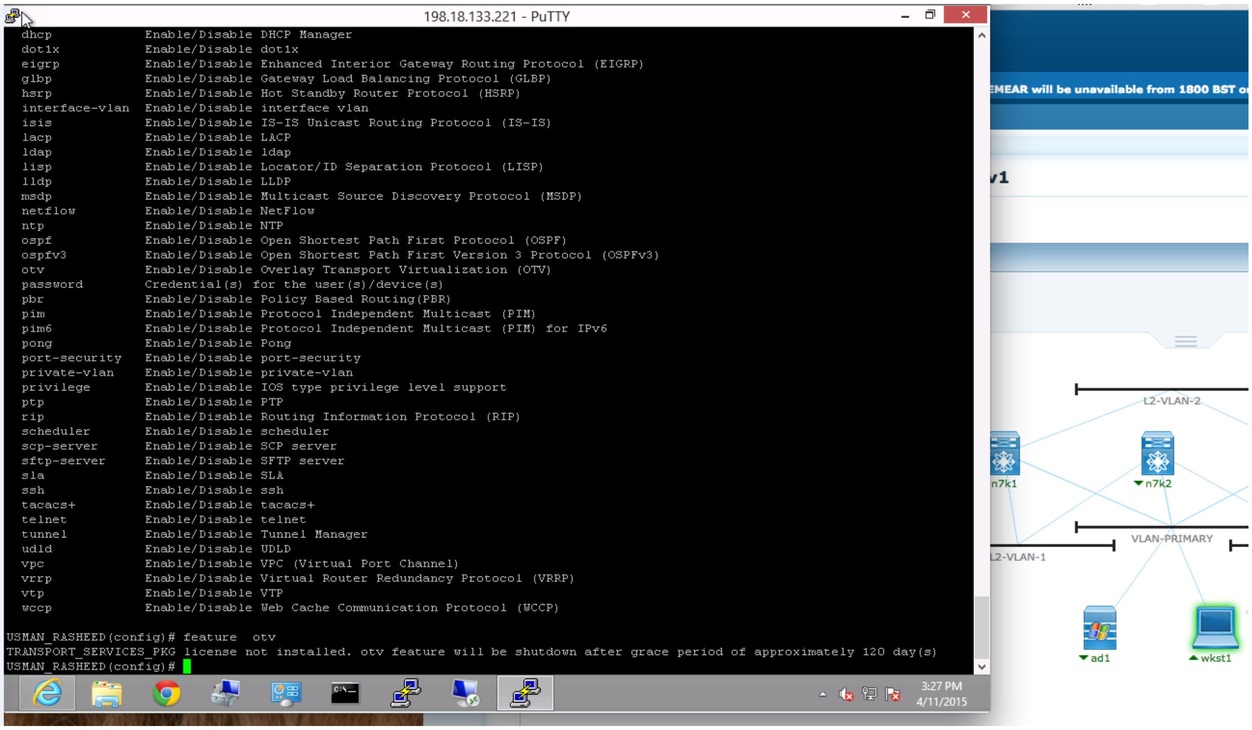


```
  eigrp          Enable/Disable Enhanced Interior Gateway Routing Protocol (EIGRP)
  glbp           Enable/Disable Gateway Load Balancing Protocol (GLBP)
  hsrp           Enable/Disable Hot Standby Router Protocol (HSRP)
  interface-vlan Enable/Disable interface vlan
  isis           Enable/Disable IS-IS Unicast Routing Protocol (IS-IS)
  lacp           Enable/Disable LACP
  ldap           Enable/Disable ldap
  lisp           Enable/Disable Locator/ID Separation Protocol (LISP)
  lldp           E
  msdp           E
  netflow        E
  ntp            E
  ospf           E
  ospfv3         E
  otv            E
  password       C
  pbr            E
  pim            E
  pim6           E
  pong           E
  port-security  E
  private-vlan   E
  privilege      E
  ptp            E
  rip            E
  scheduler      E
  scp-server     E
  sftp-server    E
  sla            E
  ssh            E
  tacacs+        E
  telnet         E
  tunnel         E
  udld           E
  vpc            E
  vrrp           Enable/Disable Virtual Router Redundancy Protocol (VRRP)
  vtp            Enable/Disable VTP
  wccp           Enable/Disable Web Cache Communication Protocol (WCCP)

USMAN_RASHEED(config)# feature  otv
TRANSPORT_SERVICES_PKG license not installed. otv feature will be shutdown after grace period of approximately 120 day(s)
USMAN_RASHEED(config)#
USMAN_RASHEED(config)#
USMAN_RASHEED(config)#
```

```
Password:
Cisco NX-OS Software
Copyright (c) 2002-2012, Cisco Systems, Inc. All rights reserved.
NX-OS/Titanium software ("NX-OS/Titanium Software") and related
documentation, files or other reference materials ("Documentation")
are the proprietary property and confidential information of Cisco
Systems, Inc. ("Cisco") and are protected, without limitation,
pursuant to United States and International copyright and trademark
laws in the applicable jurisdiction which provide civil and criminal
penalties for copying or distribution without Cisco's authorization.
The use of NX-OS/Titanium Software and Documentation is strictly
limited to Cisco's internal use.

Any use or disclosure, in whole or in part, of the NX-OS/Titanium
Software or Documentation to any third party for any purposes is
expressly prohibited except as otherwise authorized by Cisco in writing.
The copyrights to certain works contained herein are owned by other
third parties and are used and distributed under license. Some parts
of this software may be covered under the GNU Public License or the
GNU Lesser General Public License. A copy of each such license is
available at
http://www.gnu.org/licenses/gpl.html and
http://www.gnu.org/licenses/lgpl.html
n7k-2#
```



```
VSM#
VSM#
VSM#
VSM#
VSM#
VSM#
VSM#
VSM#
VSM# show version
Cisco Nexus Operating System (NX-OS) Software
TAC support: http://www.cisco.com/tac
Copyright (c) 2002-2013, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained herein are owned by
other third parties and are used and distributed under license.
Some parts of this software are covered under the GNU Public
License. A copy of the license is available at
http://www.gnu.org/licenses/gpl.html.

Software
  loader:    version unavailable [last: loader version not available]
  kickstart: version 4.2(1)SV2(2.1)
  system:    version 4.2(1)SV2(2.1)
  kickstart image file is: bootflash:/nexus-1000v-kickstart.4.2.1.SV2.2.1.bin
  kickstart compile time:  6/19/2013 17:00:00 [06/20/2013 01:08:59]
  system image file is:    bootflash:/nexus-1000v.4.2.1.SV2.2.1.bin
  system compile time:     6/19/2013 17:00:00 [06/20/2013 02:24:51]


Hardware
  cisco Nexus 1000V Chassis ("Virtual Supervisor Module")
  Intel(R) Xeon(R) CPU E7- 283 with 2075740 kB of memory.
  Processor Board ID T5056BA6E6C

  Device name: VSM
  bootflash:    1557496 kB

System uptime is 0 days, 0 hours, 33 minutes, 10 seconds

Kernel uptime is 0 day(s), 0 hour(s), 33 minute(s), 38 second(s)


plugin
  Core Plugin, Virtualization Plugin, Ethernet Plugin
VSM#
```



```
  dhcp            Enable/Disable DHCP Manager
  dot1x           Enable/Disable dot1x
  eigrp           Enable/Disable Enhanced Interior Gateway Routing Protocol (EIGRP)
  glbp            Enable/Disable Gateway Load Balancing Protocol (GLBP)
  hsrp            Enable/Disable Hot Standby Router Protocol (HSRP)
  interface-vlan  Enable/Disable interface vlan
  isis            Enable/Disable IS-IS Unicast Routing Protocol (IS-IS)
  lacp            Enable/Disable LACP
  ldap            Enable/Disable ldap
  lisp            Enable/Disable Locator/ID Separation Protocol (LISP)
  lldp            Enable/Disable LLDP
  msdp            Enable/Disable Multicast Source Discovery Protocol (MSDP)
  netflow         Enable/Disable NetFlow
  ntp             Enable/Disable NTP
  ospf            Enable/Disable Open Shortest Path First Protocol (OSPF)
  ospfv3          Enable/Disable Open Shortest Path First Version 3 Protocol (OSPFv3)
  otv             Enable/Disable Overlay Transport Virtualization (OTV)
  password        Credential(s) for the user(s)/device(s)
  pbr             Enable/Disable Policy Based Routing(PBR)
  pim             Enable/Disable Protocol Independent Multicast (PIM)
  pim6            Enable/Disable Protocol Independent Multicast (PIM) for IPv6
  pong            Enable/Disable Pong
  port-security   Enable/Disable port-security
  private-vlan    Enable/Disable private-vlan
  privilege       Enable/Disable IOS type privilege level support
  ptp             Enable/Disable PTP
  rip             Enable/Disable Routing Information Protocol (RIP)
  scheduler       Enable/Disable scheduler
  scp-server      Enable/Disable SCP server
  sftp-server     Enable/Disable SFTP server
  sla             Enable/Disable SLA
  ssh             Enable/Disable ssh
  tacacs+         Enable/Disable tacacs+
  telnet          Enable/Disable telnet
  tunnel          Enable/Disable Tunnel Manager
  udld            Enable/Disable UDLD
  vpc             Enable/Disable VPC (Virtual Port Channel)
  vrrp            Enable/Disable Virtual Router Redundancy Protocol (VRRP)
  vtp             Enable/Disable VTP
  wccp            Enable/Disable Web Cache Communication Protocol (WCCP)

USMAN_RASHEED(config)# feature  otv
TRANSPORT_SERVICES_PKG license not installed. otv feature will be shutdown after grace period of approximately 120 day(s)
USMAN_RASHEED(config)#
```

References:

- ✓ White paper NVGRE Overlay Networks: Enabling Network Scalability for a Cloud Infrastructure
- ✓ http://www.google.ca/url?sa=t&rct=j&q=&esrc=s&source=web&cd=3&ved=0CCoQFjAC&url=http%3A%2F%2Fwww.cisco.com%2Fweb%2FCA%2Fplus%2Fassets%2Fpdf%2FFabricPath-RFULLER.pdf&ei=CAAqVYHXHILloATg6YG4Cw&usg=AFQjCNFVMRTFBBpvYk2xWqP9Dd9BRk9bbg&bvm=bv.90491159,d.cGU
- ✓ https://dcloud-rtp-web-1.cisco.com/dCloud/
- ✓ http://routing-bits.com/2011/06/16/cisco-otv-part-i/
- ✓ http://www.borgcube.com/blogs/2011/11/vxlan-primer-part-1/
- ✓ Configuring Enhanced VXLAN in Unicast Mode Lab v1:Cisco Dcloud
- ✓ http://www.networkheresy.com/2011/10/03/nvgre-vlxan-and-what-microsoft-is-doing-right/
- ✓ http://www.cisco.com/c/en/us/products/collateral/switches/nexus-7000-series-switches
- ✓ Cisco OTV configuration Guide
- ✓ http://www.enterprisenetworkingplanet.com/netsp/vxlan-beyond-the-hype.html