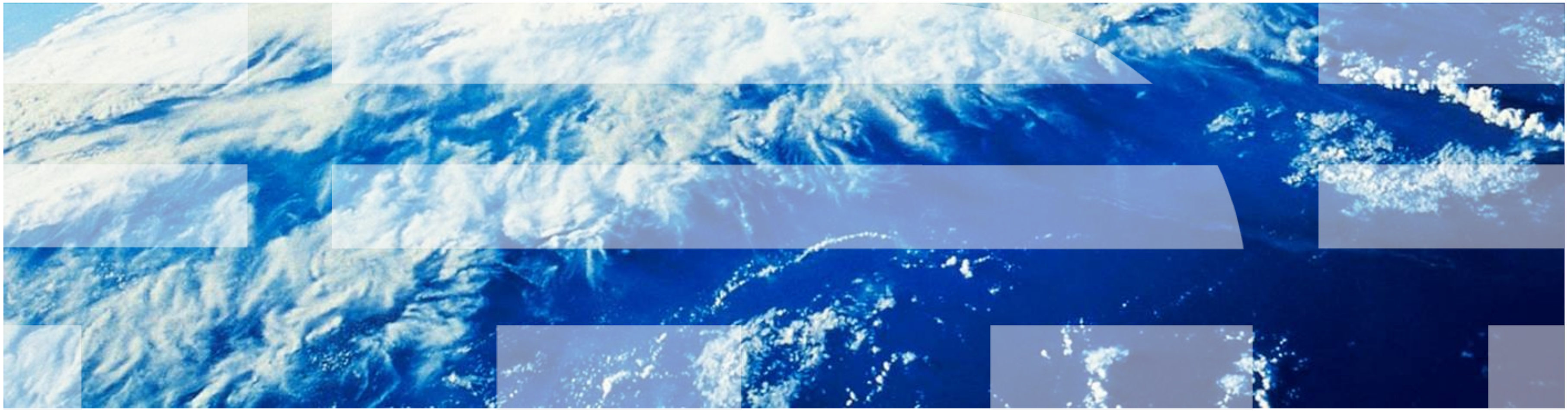
Software Defined Networking using VXLAN

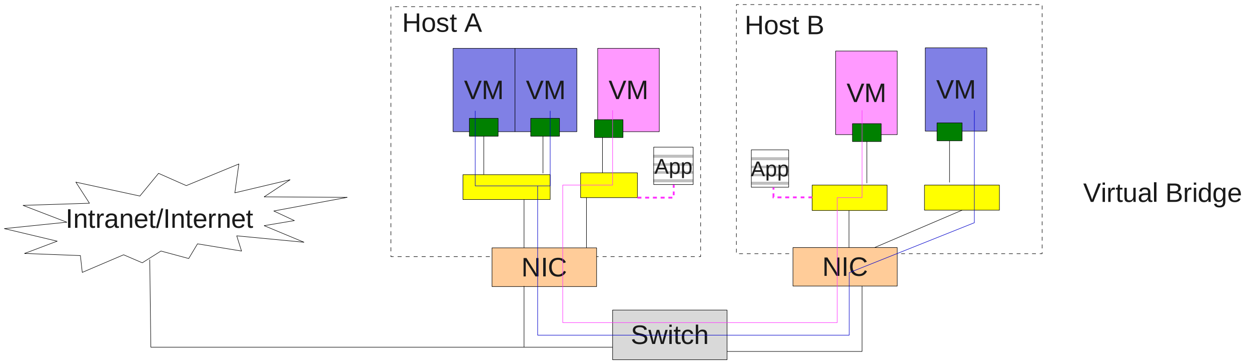
Thomas Richter



Agenda

- Vxlan
  - IETF Draft
  - VXLAN Features in Linux Kernel 3.8 (DOVE Extension)
- Principle of Operation
  - VM Creation, Migration, Removal
- Advanced Usage
  - Multicast, Broadcast, VM Detection
- Management Tools
- Related and Future Work

Virtualization in Data Center



Data centers host multiple customers

Customers require

- Own network (logical)
- Individual address space (IP and MAC)
- No interconnection with other customers
  - → Overlay Network
    - Logical network on top of existing network infrastructure

Targets

- Central management and control
- Reliability
- Cover long distance between data centers
- Define optional policies (compression, encryption, ...)

VXLAN (IETF Draft)

Virtual eXtensible Local Area Network

- Encapsulates data packets
- Connection between end points (VTEP)
- VTEP connection via existing IP infrastructure

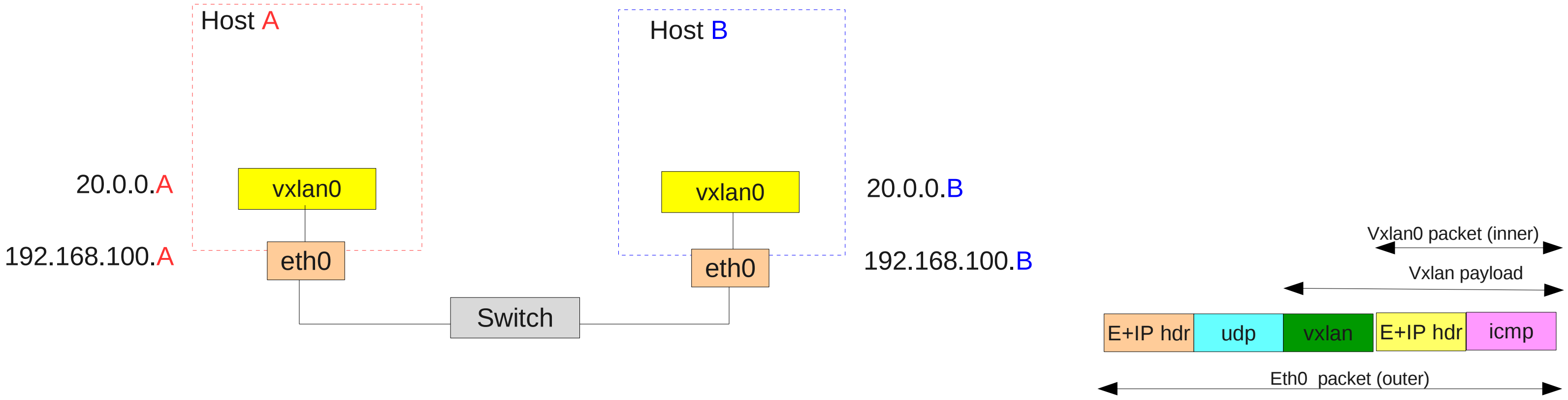
Provides

- 24 bit network identifier (VNI → defines VXLAN segment)
- VM to VM communication only within the same VXLAN segment
- VMs can use the same MAC/IP addresses in different VXLAN segments
- VM unaware of encapsulation

This Talk

- Explains recent extensions and typical traffic flow scenarios
- Mapping of VM addresses to VTEP
- Management of VTEP

VXLAN Details



Vxlan device:

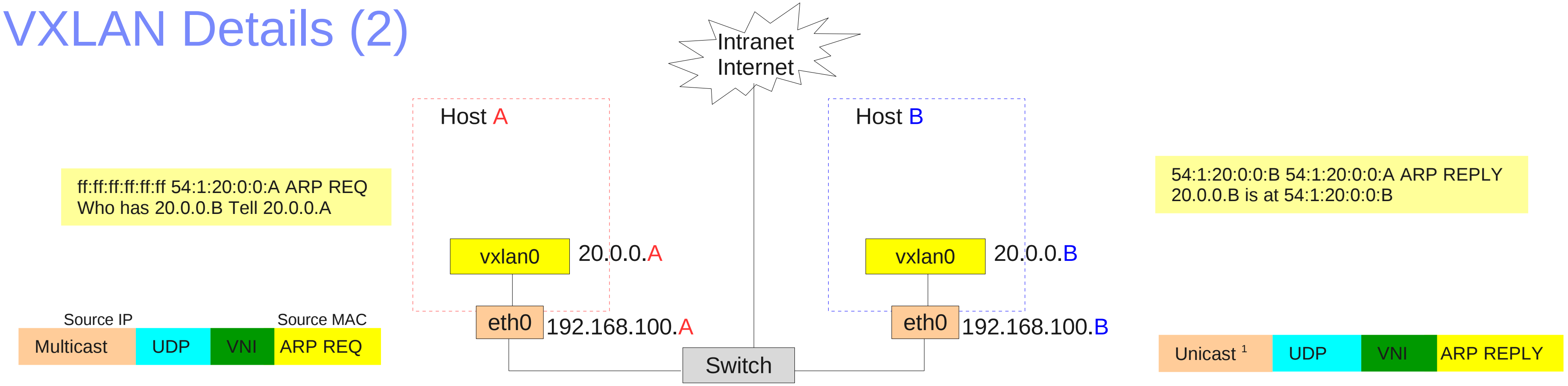
- Network device with IP, MAC address and VNI

```
# ip link add vxlan0 type vxlan id 42 group 239.1.1.1 dev eth0
# ip link set vxlan0 address 54:8:20:0:0:{A,B}
# ip address add 20.0.0.{A,B}/8 dev vxlan0
# ip link set up vxlan0
```

IANA Kernel

- Creates and connects to UDP socket endpoint (port 4789/8472 (vxlan))
- Joins multicast group
- Encapsulates all traffic with VXLAN header
- Uses UDP to forward traffic via eth0

VXLAN Details (2)



ff:ff:ff:ff:ff:ff 54:1:20:0:0:A ARP REQ
Who has 20.0.0.B Tell 20.0.0.A

54:1:20:0:0:B 54:1:20:0:0:A ARP REPLY
20.0.0.B is at 54:1:20:0:0:B

[1] With Learning enabled (multicast otherwise)

Host-A: # ping 20.0.0.B

- Find vxlan0 interface and send out ARP request
- Vxlan driver adds vxlan header (VNI)
  - No known destination MAC, use multicast address
- Eth0 sends packet

Host-B receives packet and forwards to udp port

- Vxlan driver verifies VNI and strips off vxlan header
- ARP request packet received and ARP reply packet generated
- Vxlan driver adds vxlan header (VNI)
  - No known destination MAC, use multicast address
- Eth0 sends packet

Host-B Vxlan device driver maintains a forwarding database (fdb)

Command *bridge fdb show dev vxlan0*
54:1:20:0:0:A dev vxlan0 dst 192.168.100.A self
0:0:0:0:0:0 dev vxlan0 dst 239.1.1.1 via eth0 self permanent → catch all

New VXLAN Features for Overlay Networks

Drawbacks:

- Missing control plane (no central control of VTEPs and table management)
- Depends on multicast routing support availability (wide area, routing table size)
- Mapping VNI to multicast address

Vxlan Features released into Linux Kernel 3.8 (DOVE extensions)

- L3MISS: Destination VM IP address not in Neighbor table
  - Trigger netlink message to user space
  - Expect netlink reply to add dst VM IP address into Neighbor table
- L2MISS: MAC address not in VXLAN FDB
  - Do not broadcast to any VTEP (multicast)
  - Trigger netlink message to user space
  - Expect netlink reply to add MAC address into VXLAN FDB
- NOLEARNING: Disable snooping of incoming packets
  - No entry of MAC and destination VTEP address to VXLAN FDB
- Optimization (for virtual bridges)
  - PROXY: Reply on Neighbor request when mapping found in VXLAN FDB
  - RSC: If dst MAC refers to router, replace with VM dst MAC address

    saves 1st hop

VXLAN Forwarding Database (FDB)

Maps destination VM MAC to VTEP IP

- Hashed, key is MAC address
- Size limitation possible

Contains destination

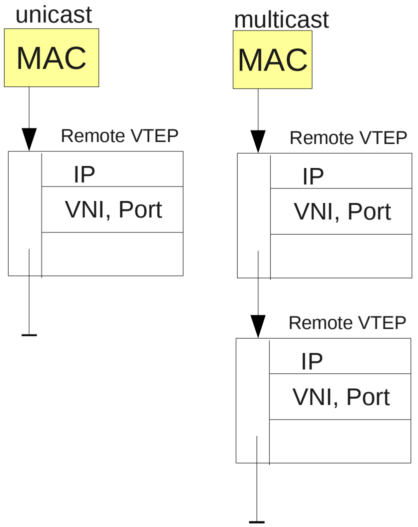
- IP Address
- VNI & port number
- Others: timestamps, flags
- Aging

Multiple destinations possible

- For multicast/all zero MAC address
- Transmit to several VTEP
- One copy per destination

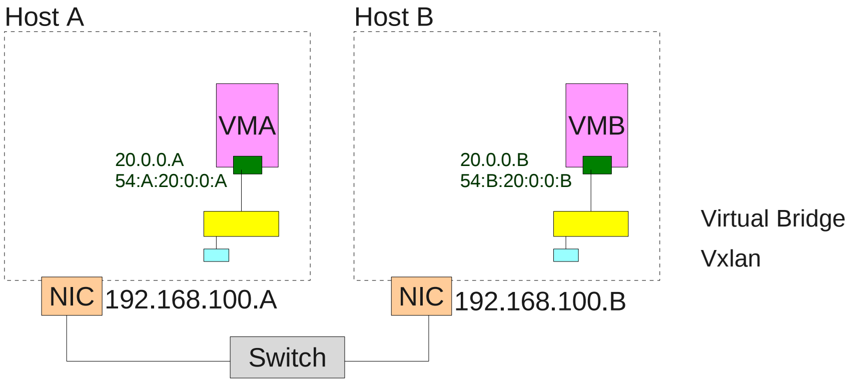
Use iproute2 tool to create/delete FDB entries

- Command: *bridge fdb add/del/append/replace …*



DOVE Extensions

VM Creation



Create virtual bridge with VXLAN device per VNI

```
# ip link add vxlan0 type vxlan id 1 l2miss l3miss rsc proxy nolearning
```

Neighbor & FDB Host A:

ARP: 20.0.0.B → 54:B:20:0:0:B (L3MISS netlink message)
FDB: 54:B:20:0:0:B → 192.168.100.B (L2MISS netlink message)

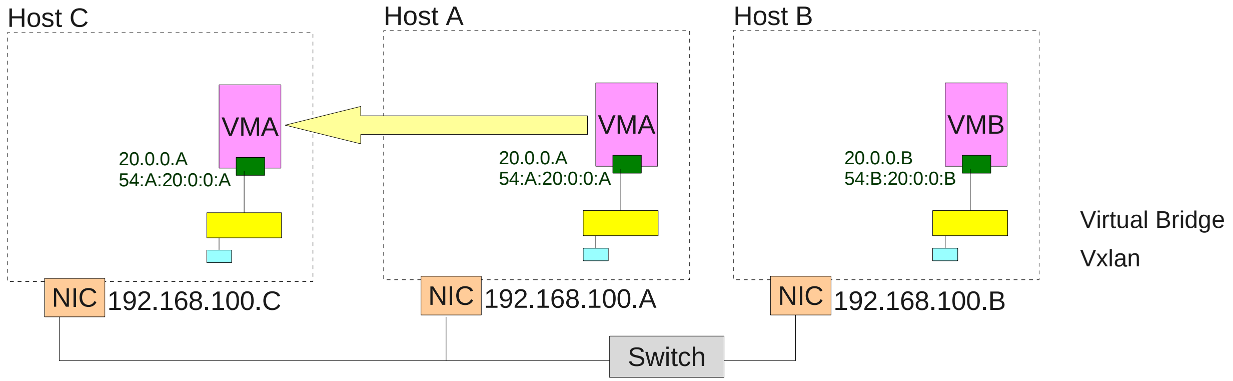
Neighbor & FDB Host B:

ARP: 20.0.0.A → 54:A:20:0:0:A (L3MISS netlink message)
FDB: 54:A:20:0:0:A → 192.168.100.A (L2MISS netlink message)

Traffic flow between VM A ↔ VM B

Can travel across internet

VM Migration



Create Virtual Bridge with VXLAN device per VNI

Host A: Delete Entries, Host C: Add Entries

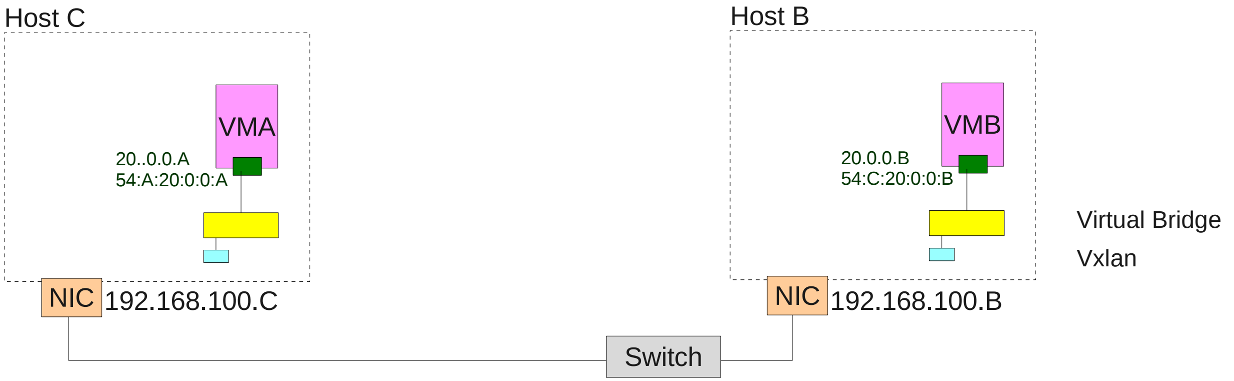
- ARP: 20.0.0.B → 54:B:20:0:0:B
- FDB: 54:B:20:0:0:B → 192.168.100.B

Host B: Modify Entries

- ARP: 20.0.0.A → 54:A:20:0:0:A
- FDB: 54:A:20:0:0:A → 192.168.100.C
- Modify on all hosts part of the 20.x.x.x overlay network

Traffic flow between VM A ↔ VM B

VM Removal



Delete Virtual Bridge with VXLAN device per VNI

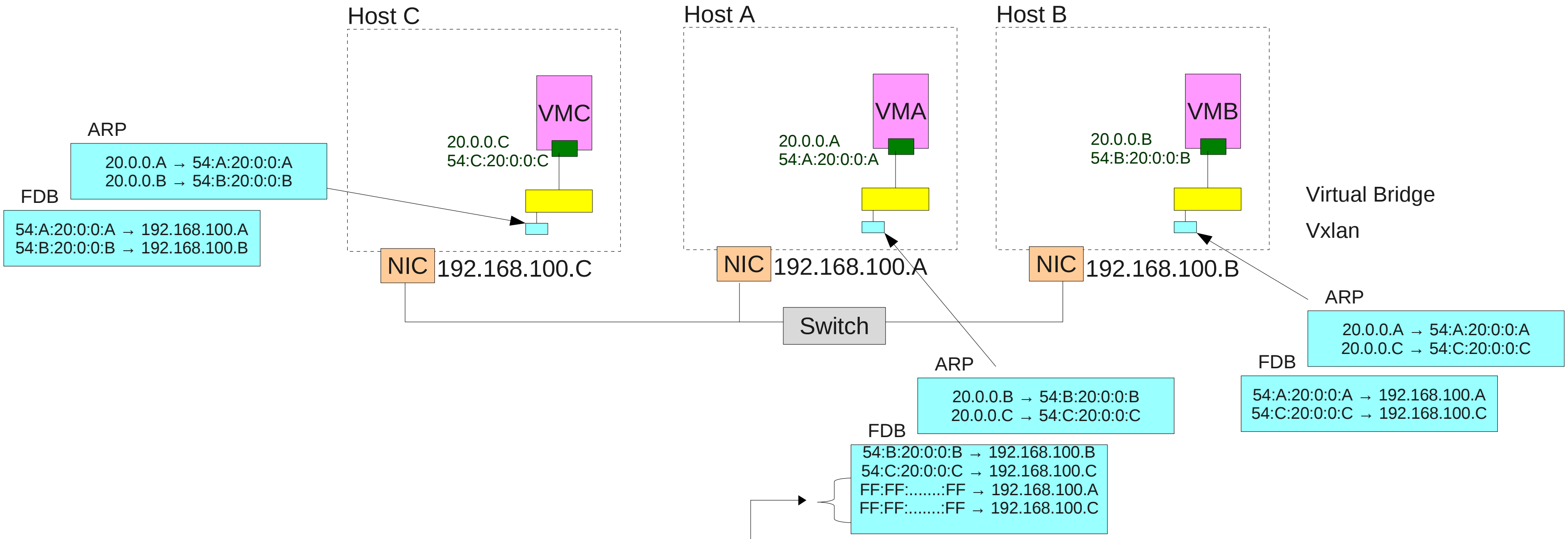
Host C: Delete Entries

- ARP: 20.0.0.B → 54:B:20:0:0:B
- FDB: 54:B:20:0:0:B → 192.168.100.B

Host B: Delete Entries

- ARP: 20.0.0.A → 54:A:20:0:0:A
- FDB: 54:A:20:0:0:A → 192.168.100.C
- Modify on all hosts part of the 20.x.x.x overlay network

VM Broadcast/Multicast



ARP
20.0.0.A → 54:A:20:0:0:A
20.0.0.B → 54:B:20:0:0:B

FDB
54:A:20:0:0:A → 192.168.100.A
54:B:20:0:0:B → 192.168.100.B

ARP
20.0.0.B → 54:B:20:0:0:B
20.0.0.C → 54:C:20:0:0:C

FDB
54:B:20:0:0:B → 192.168.100.B
54:C:20:0:0:C → 192.168.100.C
FF:FF:.......:FF → 192.168.100.A
FF:FF:.......:FF → 192.168.100.C

ARP
20.0.0.A → 54:A:20:0:0:A
20.0.0.C → 54:C:20:0:0:C

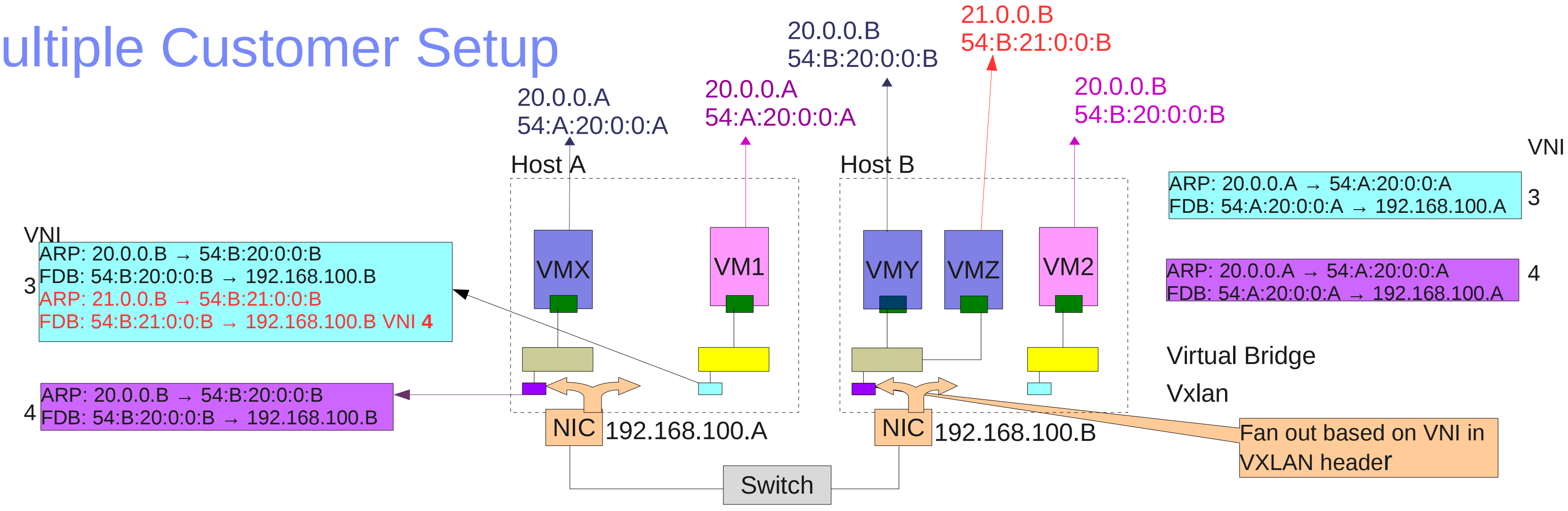
FDB
54:A:20:0:0:A → 192.168.100.A
54:C:20:0:0:C → 192.168.100.C

VM1: #ping -b 20.255.255.255

- Destination MAC ff:ff:ff:ff:ff:ff
- One entry per VTEP

Traffic flow between VM A ↔ VM B and VM C

Multiple Customer Setup



20.0.0.B
54:B:20:0:0:B

21.0.0.B
54:B:21:0:0:B

20.0.0.A
54:A:20:0:0:A

20.0.0.A
54:A:20:0:0:A

20.0.0.B
54:B:20:0:0:B

Host A

Host B

VNI

VNI

3
ARP: 20.0.0.B → 54:B:20:0:0:B
FDB: 54:B:20:0:0:B → 192.168.100.B
ARP: 21.0.0.B → 54:B:21:0:0:B
FDB: 54:B:21:0:0:B → 192.168.100.B VNI **4**

4
ARP: 20.0.0.B → 54:B:20:0:0:B
FDB: 54:B:20:0:0:B → 192.168.100.B

ARP: 20.0.0.A → 54:A:20:0:0:A
FDB: 54:A:20:0:0:A → 192.168.100.A
3

ARP: 20.0.0.A → 54:A:20:0:0:A
FDB: 54:A:20:0:0:A → 192.168.100.A
4

Fan out based on VNI in VXLAN header

Create Virtual Bridges with different VXLAN devices and VNIs

Traffic flow between VM1 ↔ VM2 and VMX ↔ VMY

- Isolation of logical networks (default configuration)

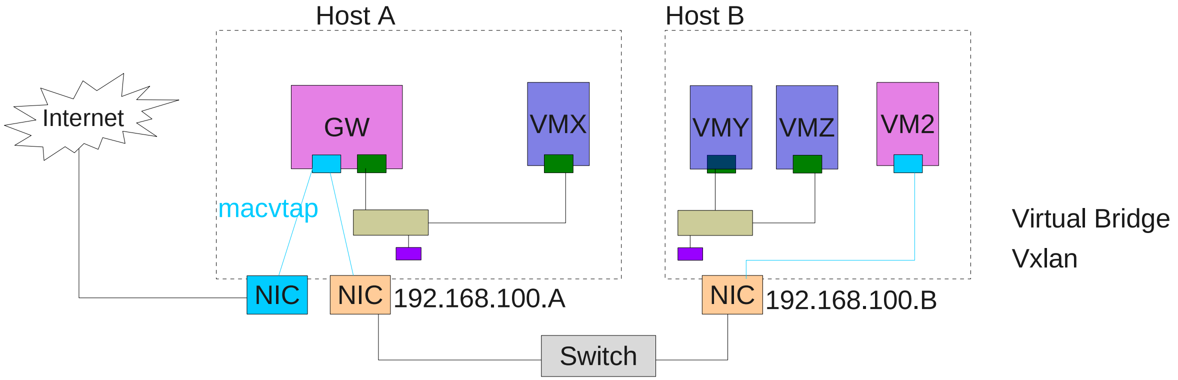
Cross logical network traffic possible (domain)

- Need configuration
- Add target VNI in VXLAN FDB

54:B:21:0:0:B → 192.168.100.B VNI 4

Multiple nets via IP routing VM X ↔ VM Z

External Connections

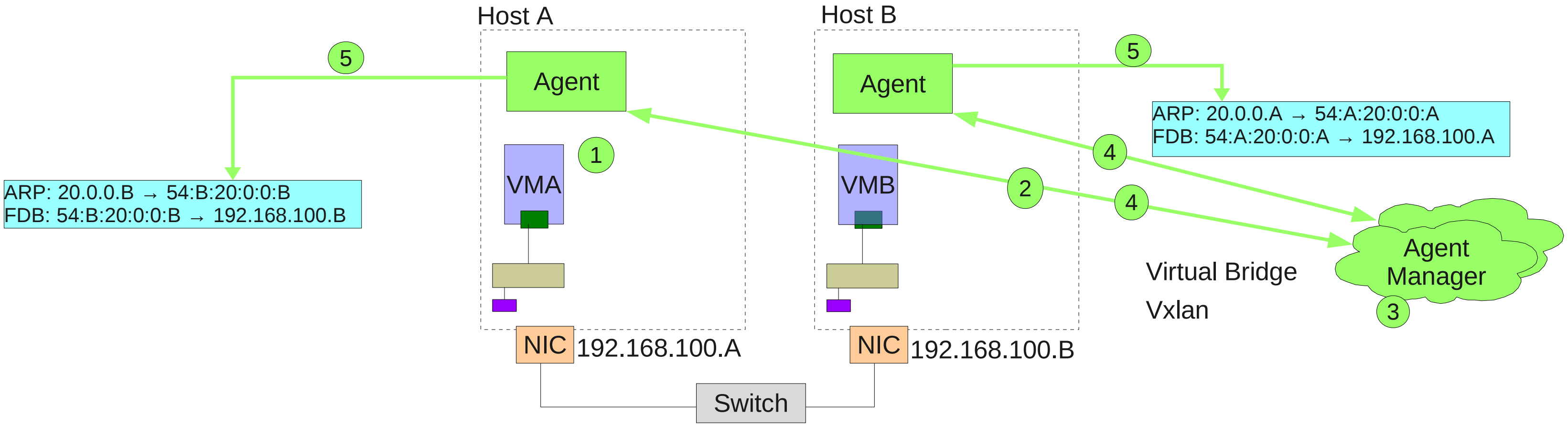


External Connections
- Legacy VM to Overlay Network VM
- Access to External Network

Create VM with access to both networks
- Configure as gateway

Traffic flow between
- VMX ↔ GW ↔ VM2/Internet

Control Plane: Neighbor and FDB Table Management


Host A
Host B
5
Agent
Agent
5
ARP: 20.0.0.A → 54:A:20:0:0:A
FDB: 54:A:20:0:0:A → 192.168.100.A
1
VMA
VMB
4
ARP: 20.0.0.B → 54:B:20:0:0:B
FDB: 54:B:20:0:0:B → 192.168.100.B
2
4
Agent
Manager
Virtual Bridge
Vxlan
3
NIC
192.168.100.A
NIC
192.168.100.B
Switch

Agent runs on each host

- – Manipulates Neighbor and FDB entries
- – Gets VM IP and MAC address
  - DHCP Snooping/Gratuitous ARP
  - IGMP Snooping
- – Data Exchange with AM
- – Agent registers for libvirtd migration events

1) VM boot detected by Agent

2)Agent forwards IP/MAC to AM

3) Check policy and permissions

4) Notifies Agents

5) Agents add entries

Agent Manager

- – Define logical networks
- – Connects to all agents
- – Multiple instances for reliability
- – Defines Policy (ACL, firewall, encryption, gateways, …)
- – Domains (One mngt for multiple VNI networks)

Remarks

Details

- Prevent fragmentation en route, set DF bit on VTEP
- UDP traffic between VTEP

Security

- Secure communication between Agent and Agent Manager
- Agent Manager data base protection
- Middle boxes (firewall, virus scanner) must be VXLAN aware

IP v6 support under work

- Multicast support missing

Iptables, ebtables, tc

- Available on host side

Alternatives (VLAN, IEEE 802.3 Qbg):

- Need hardware configuration on devices
- Export VM MAC addresses to physical network (table size, STP)

Summary

Location independent addressing

- VM assigned addresses retained while moved in overlay network

Logical network scaling

- Independent of underlying physical network and protocols
- Use existing IP network infrastructure
- No VM addresses in external switches → table size, STP
- No VLAN limitation
- No multicast dependency

Address space isolation

- Different tenants can use same addresses

Related and Future Work

Related Work

- Overlay transport:
  - Similar concept (encapsulation, inner and outer headers)
  - NVGRE:

    RFC 2784 and RFC2890

    GRE protocol (0x6558) over IP
  - STT:

    Designed for NIC with TSO, LRO

    STT protocol (similar to TCP) over IP

Future Work

- Integration into Open Stack (See Reference Nr. 4) and Open vSwitch

Questions?

Send to: tmricht@de.ibm.com

References

1)M. Mahalingam, D. Dutt et al, *VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks (Version 5),* 8-May-2013, http://datatracker.ietf.org, Note: This is work in progress

2)IBM: *IBM SND VE White Paper,* Jun-2013,

1)http://www-03.ibm.com/systems/networking/solutions/sdn.html

3)Rami Cohen, et al: *An intent-based approach for network virtualization,* IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 29-31-May-2013, pp 42-50

4)Rami Cohen, et al: *Distributed Overlay Virtual Ethernet (DOVE) integration with Openstack,* IFIP/IEEE International Symposium on Integrated Network Management (IM 2013), 29-31-May-2013, pp 1088-1089

5)Vivek Kashyap, *Network Overlays*, Network Virtualization and Lightning Talks, Linux Plumbers Conference, August 29-31, 2012, San Diego, CA, USA

References (2)

6)B. Davie, J. Gross, et al, *A Stateless Transport Tunneling Protocol for Network Virtualization (STT) (Version 4),* 13-Sep-2013, http://tools.ietf.org/html/draft-davie-stt, Note: This is work in progress

7)T. Narten, E. Gray, et al, *Problem Statement: Overlays for Network Virtualization*, 31-Jul-2013, http://tools.ietf.org/html/draft-nvo3-overlay-problem-statement-04,
Note: This is work in progress

8)M. Sridharan, et al, *NVGRE: Network Virtualization using Generic Routing Encapsulation (Version 3),* 8-Aug-2013, http://tools.ietf.org/html/draft-sridharan-virtualization-nvgre-03,
Note: This is work in progress

Acknowledgments

Vivek Kashyap, Gerhard Stenzel, Dirk Herrendörfer, Mijo Safradin, Sridhar Sumadrala, David Stevens, Vinit Jain: IBM Linux Technology Center, Data Center Networking

Trademarks

- 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.

Glossary

Agent: Application to maintain Neighbor/FDB tables

AM: Agent Manager

DOVE: Distributed Overlay Virtual Ethernet

FDB: Forwarding Data Base

Layer 2: OSI Data Link Layer (Reliable Link between directly connected nodes)

Layer 3: OSI Network Layer (IP addressing)

L2MISS: Destination MAC address unknown

L3MISS: Destination IP address unknown

LEARNING: Add new MAC/VTEP address in FDB

Multi Tenant: Software Instance used for several customers

NVGRE: Network Virtualization Generic Routing Encapsulation

OSI: Open Systems Interconnection

OTV: Overlay Transport Virtualization

RSC: Route Short Circuit

SDN: Software Defined Network

SST: Stateless Transport Tunneling

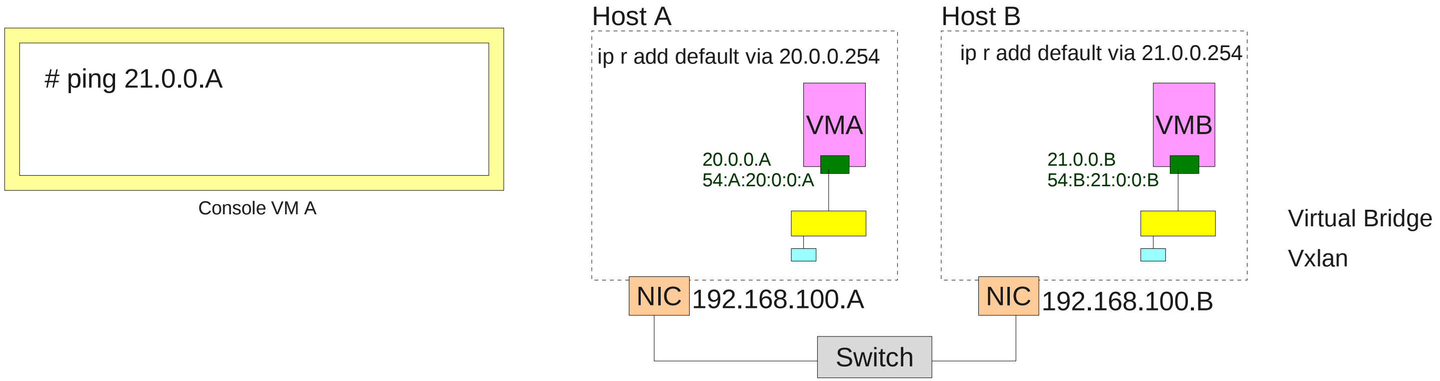
VNI: VXLAN Network Identifier or VXLAN Segment Identifier

VTEP: Virtual Tunnel End Point

VXLAN: Virtual extensible Local Area Network

BACKUP

Route Short Circuit (RSC)



```
# ping 21.0.0.A
```

Console VM A

```
ip r add default via 20.0.0.254
```

```
ip r add default via 21.0.0.254
```

Neighbor & FDB Host A:

ARP: 20.0.0.254 → 54:A:20:0:0:FE (1)
21.0.0.B → 54:B:21:0:0:B (2b)
FDB: 54:B:21:0:0:FE → 1.2.3.4 router (2a)
54:B:21:0:0:B → 192.168.100.B (4)

1) Look up router IP to MAC mapping in neighbor table

2) Router flag set
   - a) Remote IP address in FDB entry ignored
   - b) Look up destination IP address to MAC mapping in neighbor table

3) Replace destination MAC in inner header 54:A:20:0:0:FE → 54:B:21:0:0:1

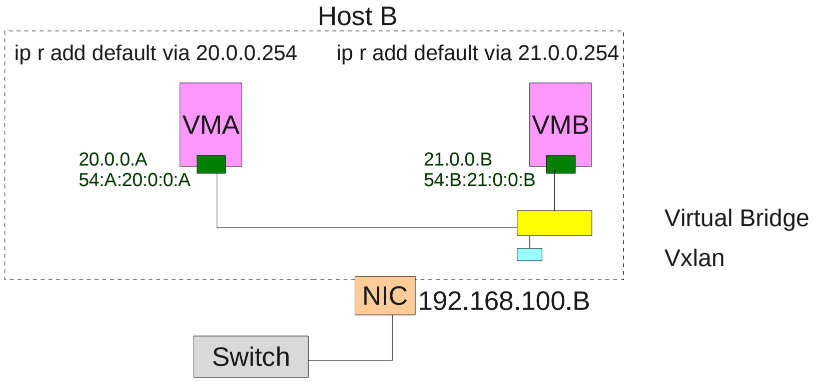
4) Look up destination MAC in FDB and transmit to VTEP

Traffic flow between VM A ↔ VM B

Route Short Circuit 2 (Migration VM A to Host B)

```
# ping 21.0.0.A
```

Console VM A



Neighbor:

20.0.0.254 →	54:0:1:2:3:4 (1)
21.0.0.254 →	54:0:1:2:3:4
20.0.0.A →	54:A:20:0:0:A
21.0.0.B →	54:B:21:0:0:B (3)

FDB:

54:0:1:2:3:4 →	1.2.3.4 router (2)
54:B:21:0:0:B →	0.0.0.0 (4)
54:A:20:0.0:A →	0.0.0.0

1) Look up router IP to MAC mapping in neighbor table

2) Router flag set
   - a) Remote IP address in FDB entry ignored
   - b) Look up destination IP address to MAC mapping in neighbor table

3) Replace destination MAC in inner header 54:A:20:0:0:FE → 54:B:21:0:0:1

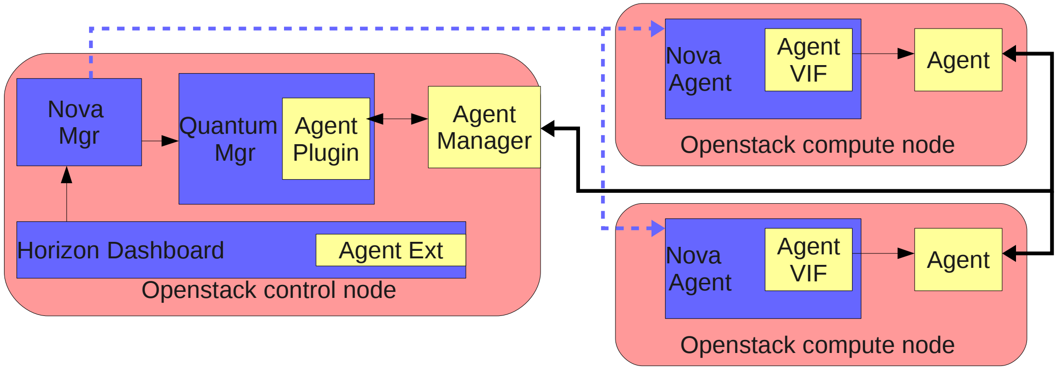
4) Look up destination MAC in FDB and feed back to local bridge (destination IP 0.0.0.0)

Traffic flow between VM A ↔ VM B

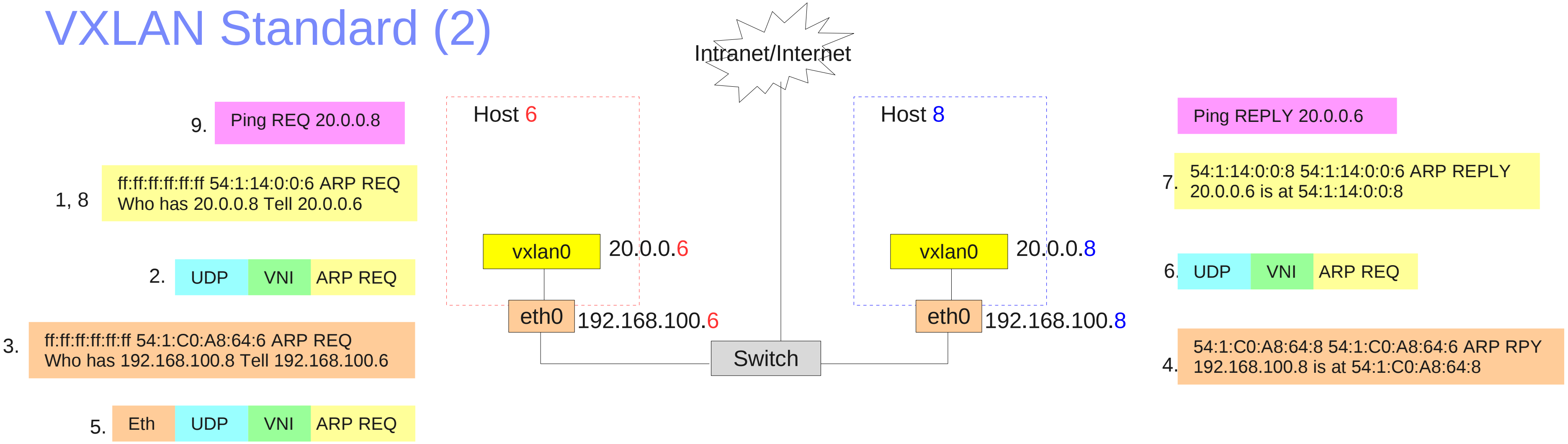
Open Stack Integration

See paper R. Cohen (References Nr 4)

– Map bridge name to VNI



VXLAN Standard (2)

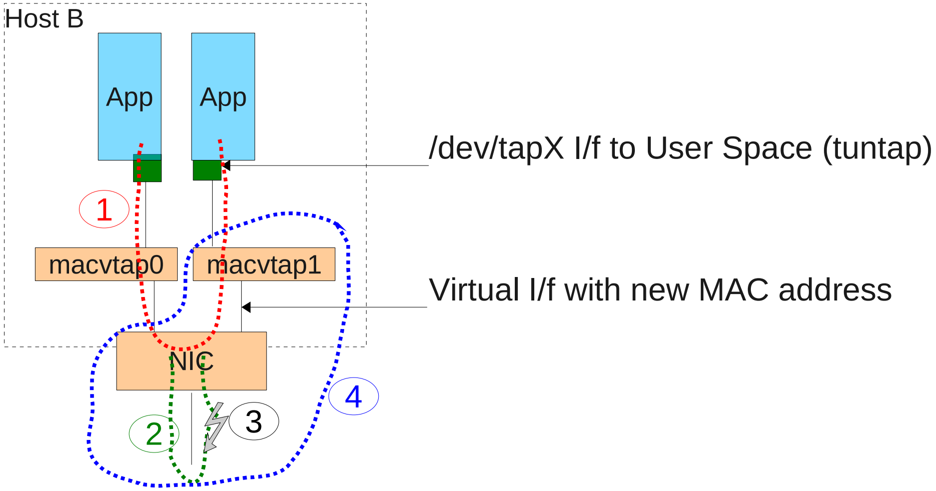

Intranet/Internet
Host 6
Host 8
9.
Ping REQ 20.0.0.8
1, 8
ff:ff:ff:ff:ff:ff 54:1:14:0:0:6 ARP REQ
Who has 20.0.0.8 Tell 20.0.0.6
2.
UDP
VNI
ARP REQ
3.
ff:ff:ff:ff:ff:ff 54:1:C0:A8:64:6 ARP REQ
Who has 192.168.100.8 Tell 192.168.100.6
5.
Eth
UDP
VNI
ARP REQ
vxlan0
20.0.0.6
eth0
192.168.100.6
Switch
vxlan0
20.0.0.8
eth0
192.168.100.8
Ping REPLY 20.0.0.6
7.
54:1:14:0:0:8 54:1:14:0:0:6 ARP REPLY
20.0.0.6 is at 54:1:14:0:0:8
6.
UDP
VNI
ARP REQ
4.
54:1:C0:A8:64:8 54:1:C0:A8:64:6 ARP RPY
192.168.100.8 is at 54:1:C0:A8:64:8

Ping 20.0.0.8 (on host 6)

1) Host 6 ARP request for 20.0.0.8 on vxlan0 device (broadcast)
2) Host 6 vxlan0 device prepends VXLAN header and forwards via UDP to eth0 device
3) Host 6 sends ARP request for 192.168.100.8 on eth0 device (broadcast)
4) Host 8 sends ARP reply for 192.168.100.8 to host 6 (unicast)
5) Host 6 sends encapsulated vxlan0 arp request to host 8 via eth0
6) Host 8 strips off eth and udp header and forwards to vxlan0 device
7) Host 8 vxlan0 device responds to arp request from host 6 vxlan0 device
8) Host 6 now knows destination MAC address of host 8 vxlan0 device
9) Host 6 now sends ICMP ping request to correct host 8 vxlan mac address

??? Step 3: How to find out vxlan IF 20.0.0.8 hosted by host 8 (reachable via 192.168.100.8)

VM Attachment and Macvtap Device Options


Host B
App
App
/dev/tapX I/f to User Space (tuntap)
1
macvtap0
macvtap1
Virtual I/f with new MAC address
NIC
4
2
3

Macvtap

- Combines tun/tap and macvlan devices
- Modes:

(1)Bridged: destination MAC address lookup on all macvtap devices defined on NIC

(2)Vepa:Traffic forwarded to external switch

(3)Private: Same as vepa, but ingress traffic blocked

(4)Passthrough: Only 1 macvtap device allowed per NIC (“exclusive” use)