Using VXLAN

… to network virtual machines, jails, and other fun things on FreeBSD

John Nielsen, john@jnielsen.net
BSDCan, 6/10/2016

Overview

- Introduction
- VXLAN Compared to VLAN and Network Tunnels
- VXLAN More in Depth
- Tips and Tricks
- A Few Use Cases
- Demos!

Introduction

- About Me
- VXLAN in Brief
- Quick Review of Network Protocol Layers
- Anatomy of a VXLAN Packet

About Me

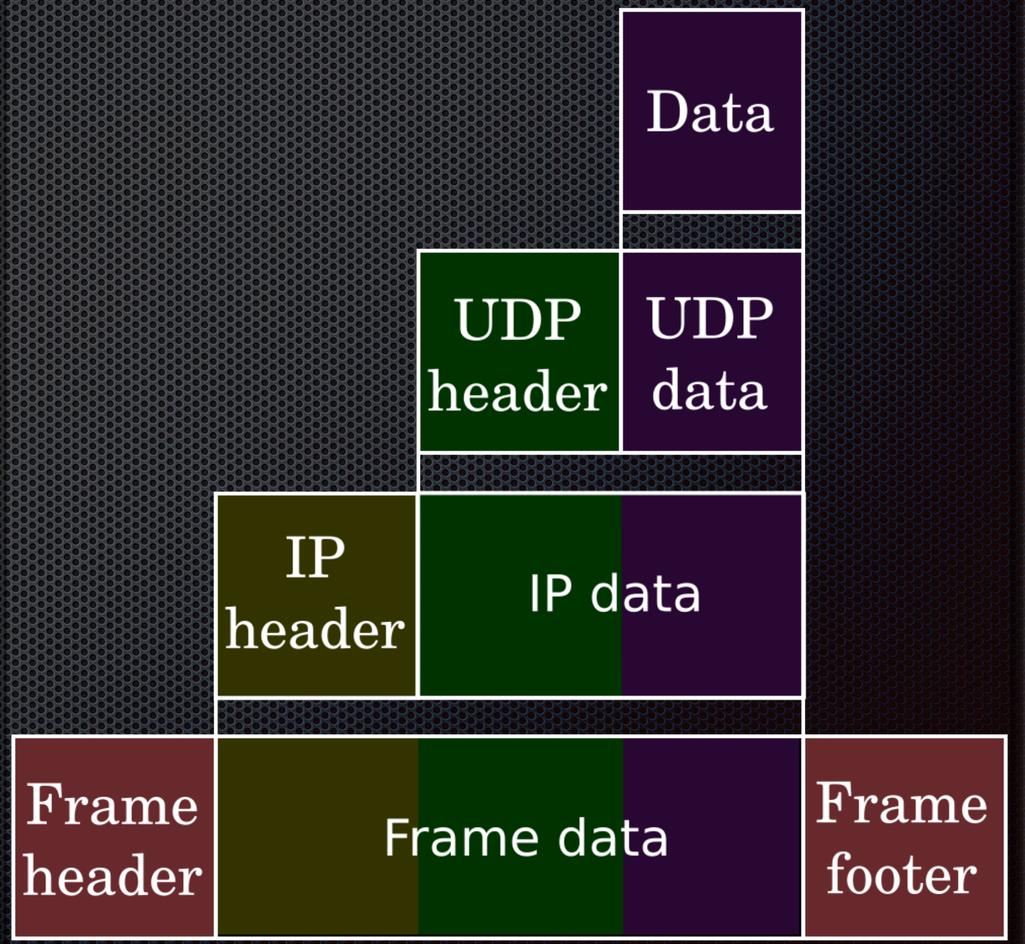
- First Computer: TI-99/4A (circa 1983)
- FreeBSD user since 1999 (FreeBSD 3.4)
- BS CS from BYU in 2005, MS CS from UNC Charlotte in 2009
- Systems Administrator/Engineer since 2000
- Currently employed at Domo
- Especially interested in virtualization, host-side networking and storage

VXLAN in Brief

- VXLAN stands for Virtual eXtensible Local Area Network
- Creates overlay networks by encapsulating Ethernet frames in UDP/IP with an associated 24-bit Virtual Network Identifier (VNI)
- "Virtual" tunnels are created in a one-to-many fashion between end hosts
- Hosts are called "Virtual Tunnel End Points" or VTEPs.
- VTEPs learn about each other as they exchange traffic
- Broadcast, Unknown destination and Multicast (a.k.a BUM) traffic is sent to all participating VTEPs via multicast

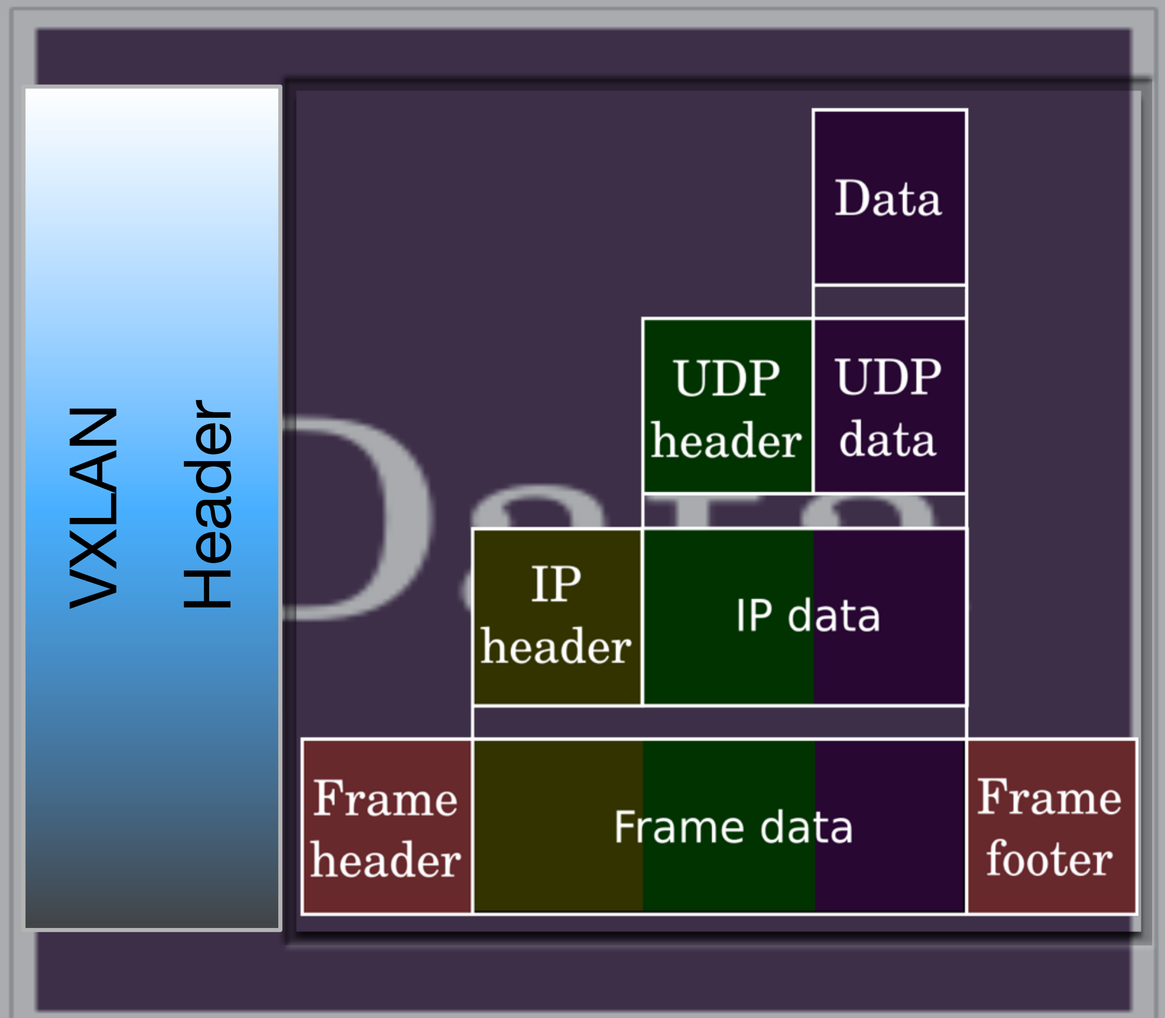
Quick Review of IP Protocol Layers

- “Application” (OSI 5-7): HTTP, SSH, SMTP, … (data payload)
- “Transport” (OSI 4): TCP, UDP, SCTP, …
- “Network” (OSI 3): IPv4, IPv6, …
- “Data Link” (OSI 2): Ethernet, InfiniBand, …



Anatomy of a VXLAN Packet

- The original (“inner”) Ethernet frame is prepended with a VXLAN header
- Together they form the data payload of a new (“outer”) encapsulating packet with its own UDP and IP headers



VXLAN Compared to VLAN and Network Tunnels

- VLAN
  - Properties
  - Limitations
- Network Tunnels
  - Tunnel Types and Properties
  - Limitations
- VXLAN Benefits and Limitations

VLAN Properties

- Virtual Local Area Network
- Layer 2 multiplexing enhancement allowing up to 4094 "virtual" Layer 2 networks on a single set of connected switches
- Typically requires switch support and configuration
- Generally well-understood and well-supported

VLAN Limitations

- The underlying network *must* be Ethernet
- Further, the *whole network* has to be a *single* Layer 2 segment (i.e. VLANs don't cross routers)
- At most 4094 VLANs per segment (12-bit ID minus 2 reserved values)
- Some switches impose their own (lower) limits on number of VLANs

Network Tunnels Overview

- Point-to-point (1-to-1) connections between two end points
- Typically encapsulate Ethernet (L2) or IP (L3) traffic within IP or TCP/UDP
- L2TP, GRE, PPP, GIF, EtherIP, various VPNs, etc
- Great for links between two hosts (like a static VPN)
- Good for multiple links in a star topology (like a multi-user VPN where the users just want to talk to the office, not each other)

Tunnel Limitations

- They don't scale well, especially when direct communication between multiple end points is desired (full mesh topology)
  - For N hosts, the number of tunnels needed is $\frac{N^2-N}{2}$
  - In other words, the number of tunnels grows *exponentially* with the number of endpoints
  - 2 endpoints: 1 tunnel. 3 endpoints: 3 tunnels. 4 endpoints: 6 tunnels. 10 endpoints: 45 tunnels. 100 endpoints: 4950 tunnels, etc
- Typically only one "inner" network is carried over each tunnel
- Each tunnel needs to be configured on both ends
- A routing protocol or something similar is required to decide which tunnel to use for which traffic

VXLAN Benefits

- One-to-many "virtual" tunnels only need to be configured once per host, even if more hosts will be added in the future
- Each VTEP automatically creates and maintains its own forwarding table, so most communication happens directly with the correct peer VTEP automatically
- No switch support required
- In fact, the underlying networks don't even have to be Ethernet. InfiniBand, wireless, PPP, ATM or any combination can work
- 24-bit VNIs with no reserved values allow for 16,777,216 independent VXLAN networks per multicast domain

VXLAN Limitations

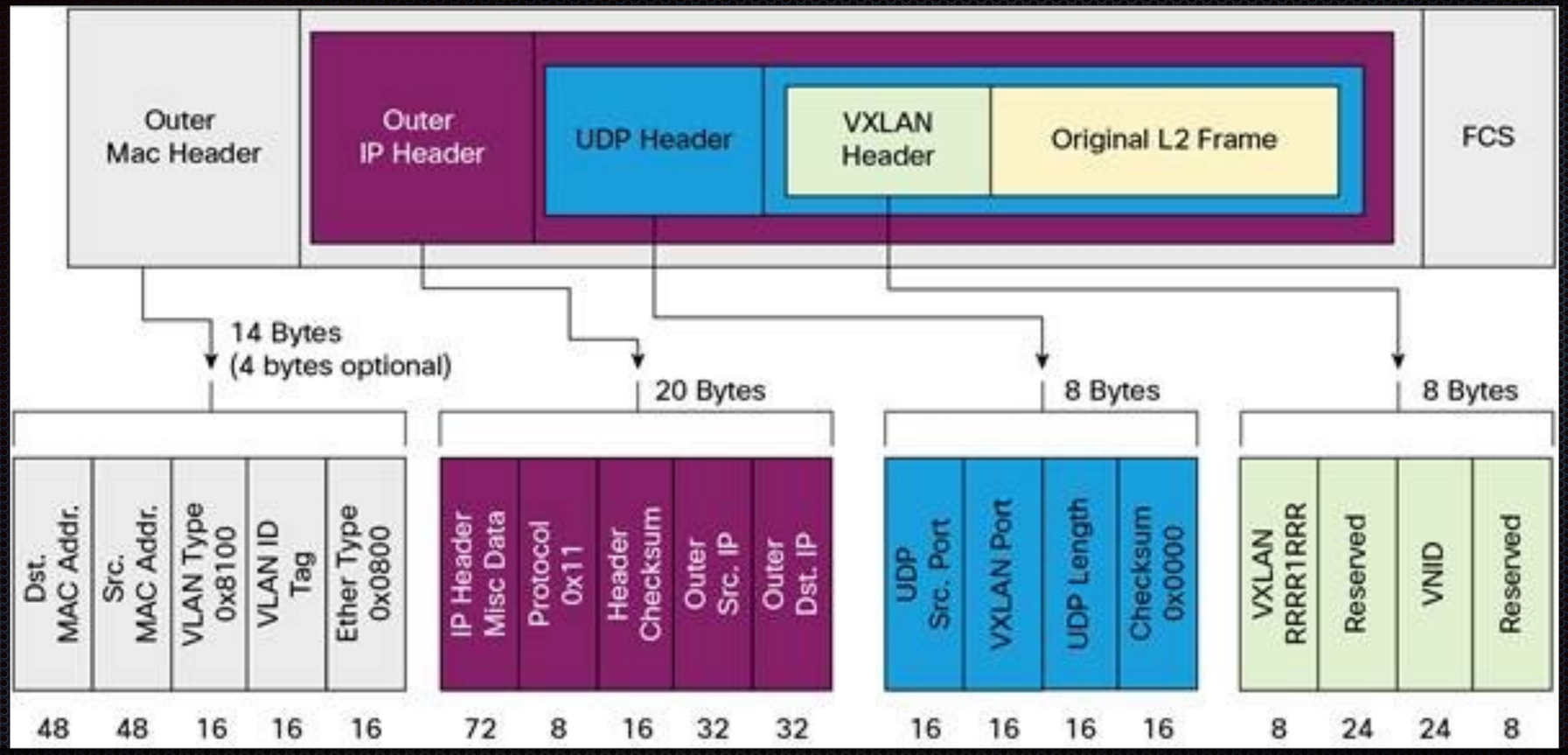
- Each network segment (and the routers between them, if more than one) needs to support multicast.
  - For a single organization this isn't usually too hard
  - Some implementations can use out-of-band discovery instead of multicast
- Encapsulation adds 50 bytes to each packet
  - (This is also true of most 1-to-1 tunnels)
  - Need either a larger MTU on the outer network(s) or a smaller MTU on the inner networks
  - My suggestion: use decent networking gear. If you can set an MTU of 9216 on the outer network(s), the end user can choose an MTU of either 1500 (normal) or 9000 (“jumbo”) for each inner network
- Some CPU overhead (at least until VXLAN-offloading NICs and drivers are more common)

VXLAN More in Depth

- How Does It Work?
  - Sending VXLAN packets
  - Receiving VXLAN packets
- Where Is It Available?

How Does VXLAN Work?

- On each host, a virtual interface is created for each VXLAN network to be used on that host
  - The interface has a name, VNI, and multicast address
- The interfaces can be used directly (by e.g. assigning IP addresses to them) or bridged together with other real or virtual Ethernet interfaces
- The host maintains a forwarding table for each interface, mapping (inner) MAC addresses to (outer) remote VTEP IP addresses. Much like in a switch, the forwarding table is updated by snooping traffic to and from the vxlan interface.

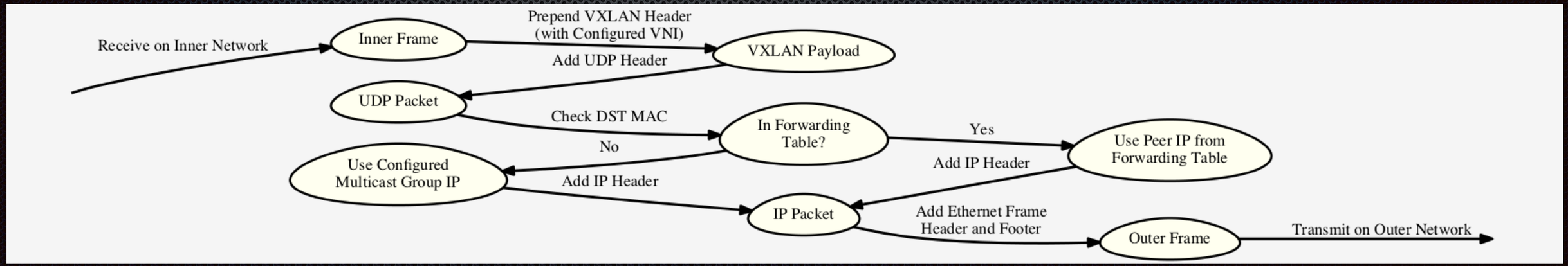


How Does VXLAN Work? (Cont)

Packets sent over the (outer) transport network have a VXLAN header prepended and are then encapsulated with outer IP and UDP headers

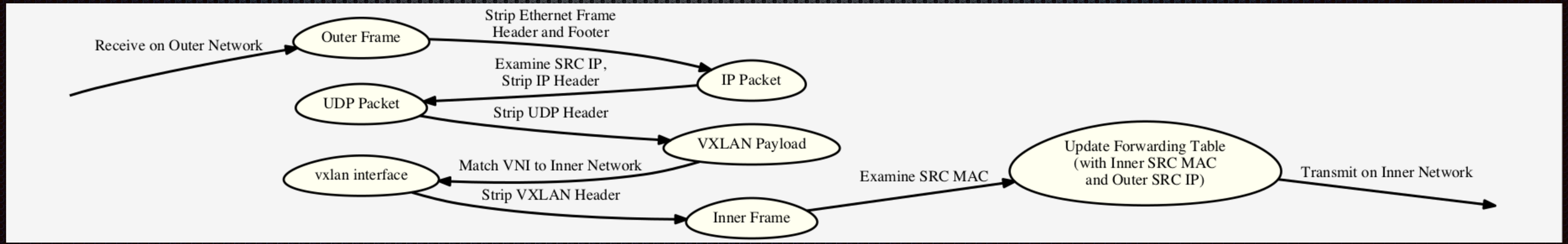
Sending VXLAN packets

- To send a packet "out" the vxlan interface, its destination MAC address is looked up in the forwarding table.
  - If a match is found then the packet is encapsulated and sent to the remote VTEP directly (unicast)
  - If no match is found (or if the (inner) destination is broadcast or multicast), the packet is encapsulated and sent to the multicast address configured for the interface



Receiving VXLAN Packets

- To receive a packet "in" the vxlan interface, the encapsulated packet is first received by the outer network interface and matched to a vxlan interface by the VNI in the VXLAN header. The (inner) source MAC address and (outer) source IP address are used to update the forwarding table for the interface.
  - The packet is then un-encapsulated and the original (inner) L2 frame is sent out the appropriate vxlan interface



Where Is It Available?

- OpenBSD since 5.5
- FreeBSD since 10.2
- Linux kernel since 3.7 (wth corresponding iproute[2])
  - Kernel 3.10 or newer recommended
- VMware ESX since 5.1(?)
- Some switches and routers from Cisco, Arista, Juniper and others
- Docker’s “overlay” network driver uses a userspace VXLAN implementation for inter-host communication

Tips and Tricks

- Use a Unique Multicast Address for Each VNI
- Use the Official UDP Port (4789)
- Remember to Adjust the Firewall on your VTEPs

Use a Unique Multicast Address for Each VNI

- The entire IPv4 multicast range is 224.0.0.0 - 239.255.255.255
- The upper end of the range, 239.0.0.0 - 239.255.255.255, is classified by the IANA as "organization-local scope"
- In other words, 239.0.0.0/8 is 24 bits and is likely available for your VXLANs
- Take the VNI, convert it to hex, convert each octet back to decimal and stick 239 in front. That's your one-to-one mapping from VNI to multicast.
- Example: VNI: 12345678, or in hex: 0xbc614e. 0xbc=188, 0x61=97, 0x4e=78. Multicast address = 239.188.97.78

vni_to_mcast.sh

```
#!/bin/sh

hex=`printf "%06x" ${1}`
printf "239"
for pos in 1-2 3-4 5-6; do
  printf ".%d" "0x`echo "${hex}" | cut -c ${pos}`"
done
printf "\n"
```

Use the Official UDP Port (4789)

- The IANA-assigned UDP port number for VXLAN is 4789
- The Linux implementation predates the IANA assignment and has its own default UDP port of 8472
- Unless you need to be compatible with other hosts using the Linux port number, use the official one.
- FreeBSD and OpenBSD use the official port by default. On Linux, add "dstport 4789" to your "ip link create" command

Remember to Adjust the Firewall on your VTEPs

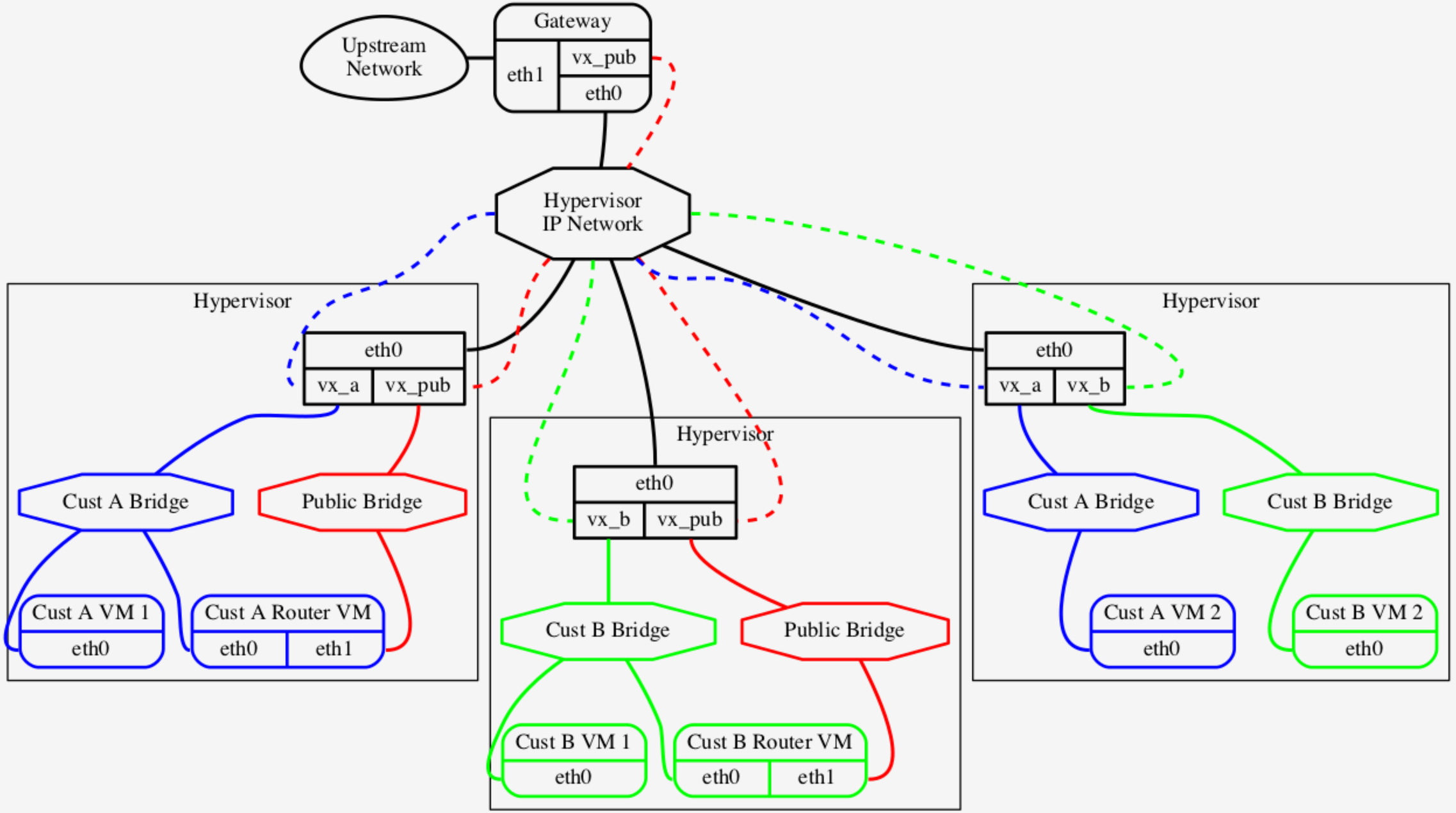
- And Don’t forget multicast! For the outer network:
  - allow UDP from {other hosts} to {me} port 4789
  - allow UDP from {other hosts} to 239.0.0.0/8 port 4789
- For the inner network:
  - whatever rules you'd normally apply to an interface

A Few Use Cases

- Virtual Networks for Bhyve (or other) VMs
- Bridging / Extending Ethernet Networks Across Non-Ethernet Segments
- Networking Between VNET Jails

Virtual Networks for Bhyve VMs

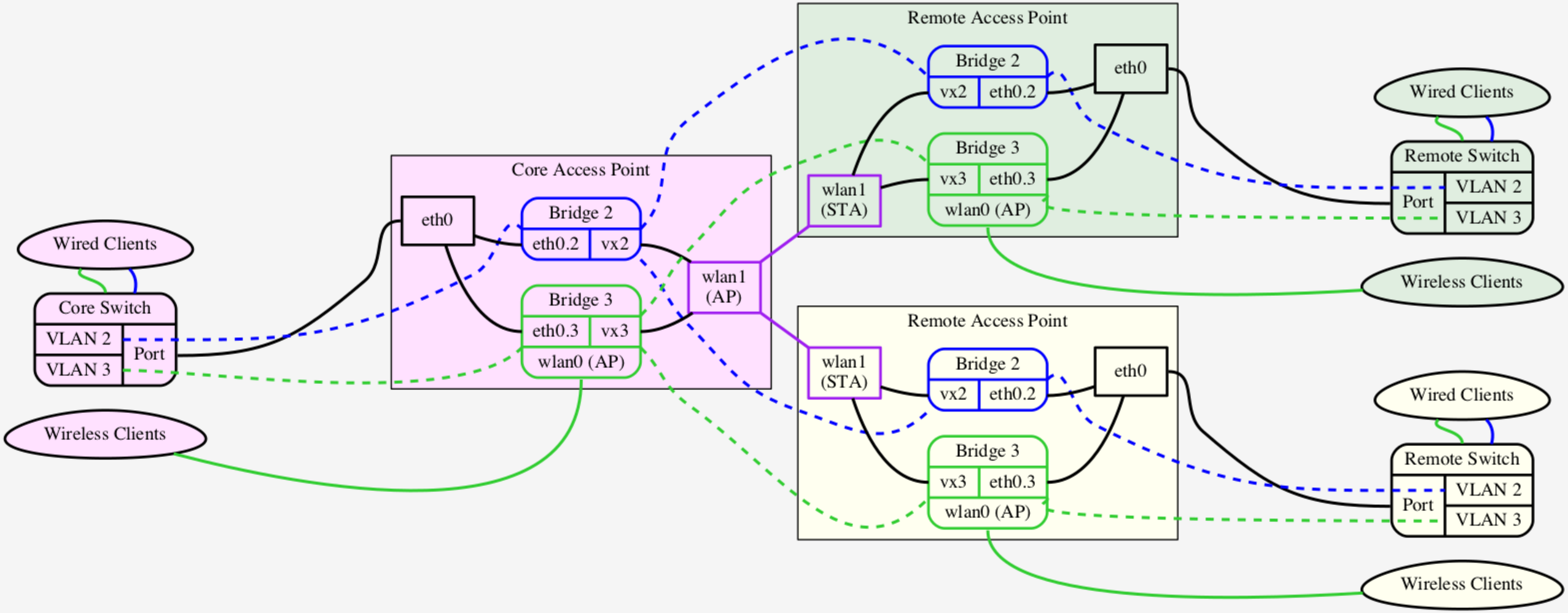
- Millions of isolated virtual networks on one (set of) outer network(s)
- Hosts can be on different IP subnets
- No switch configuration needed to add or remove networks
- Virtual interfaces can be bridged with vxlan interfaces as needed
- Each VTEP (host) only gets traffic for the VNIs it actually has VMs on



Networking Virtual Machines with VXLAN

Bridging / Extending Ethernets

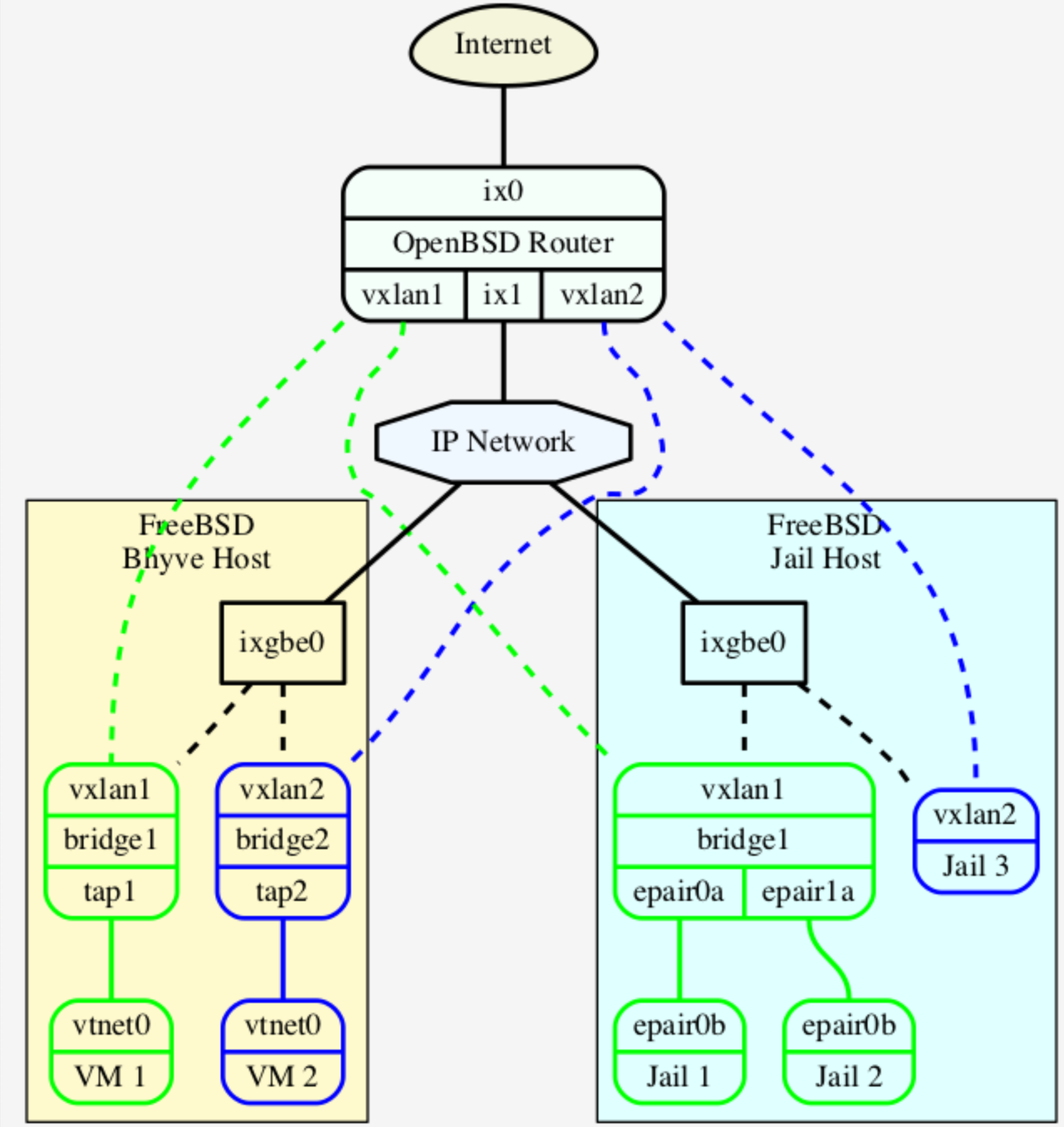
- Interfaces have to be Ethernet-like in order to be added to an Ethernet bridge
- But other Layer 2 technologies are cool too! Like InfiniBand
- Wireless access points can be bridged easily, but wireless stations cannot
- But they all support IP and multicast! VXLAN makes it easy to create as many Ethernet-like overlay networks as you need on any IP network



Extending VLANs Across Wireless Backhaul Links with VXLAN

Networking Between VNET Jails

- VNET jails on FreeBSD have their own network stacks isolated from the host's
  - Similar to the VM case, the host side of an epair interface can be added to a bridge while the other side is enslaved into a jail
  - Alternatively, if the jail is the only client on the host for a given network, the vxlan interface can be enslaved directly into the jail (while the host still manages the outer network)
- The jail can communicate with other jails (or VMs, etc) on the same VXLAN



Networking VMs and Jails with VXLAN

Demos and How-Tos

- Setup Example (FreeBSD)
  - Startup Config (FreeBSD)
- Setup Example (OpenBSD)
  - Startup Config (FreeBSD)
- Setup Example (Linux)
- Jail Networking Demo

Setup Example (FreeBSD)

```
# ifconfig vxlan1234 create vxlanid 1234 vxlanlocal 192.168.2.1
vxlandev vtnet0 vxlangroup 239.0.4.210
# ifconfig vxlan1234 inet 192.168.248.1/24 up mtu 1450
# ifconfig vxlan1234
vxlan1234: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST>
metric 0 mtu 1450
  ether ca:70:0e:dd:c0:78
  inet 192.168.248.3 netmask 0xffffff00 broadcast 192.168.248.255
  nd6 options=29<PERFORMNUD,IFDISABLED,AUTO_LINKLOCAL>
  vxlan vni 1234 local 192.168.2.1:4789 group 239.0.4.210:4789
```

Startup Config (FreeBSD)

```
### /etc/rc.conf ###
cloned_interfaces="vxlan1234" # or add to existing
create_args_vxlan1234="vxlanid 1234 \
                       vxlanlocal 192.168.2.1 \
                       vxlandev vtnet0 \
                       vxlangroup 239.0.4.210"
ifconfig_vxlan1234="inet 192.168.248.3/24 up mtu 1450"
```

Setup Example (OpenBSD)

```
# ifconfig vxlan1234 tunnel 192.168.1.100 239.0.4.210 vnetid 1234
# ifconfig vxlan1234 192.168.248.2/24 mtu 1450
# ifconfig vxlan1234
vxlan1234: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1450
  lladdr fe:e1:0e:dd:c0:78
  priority: 0
  groups: vxlan
  media: Ethernet autoselect
  status: active
  tunnel: inet 192.168.100.24 -> 239.0.4.210 vnetid 1234
  int 192.168.248.2 netmask 0xffffff00 broadcast 192.168.248.255
```

Startup Config (OpenBSD)

```
### /etc/hostname.vxlan1234 ###
tunnel 192.168.1.100 239.0.4.210 vnetid 1234
inet 192.168.248.2 255.255.255.0 NONE mtg 1450
```

Setup Example (Linux)

```
# ip link add vx1234 type vxlan id 1234 group 239.0.4.210 dev eth0 ttl
64 dstport 4789
# ip addr add 192.168.248.3/24 dev vx1234
# ip link set up mtg 1450 dev vx1234
# ip a show dev vx1234
5: vx1234: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1450 qdisc noqueue
state UNKNOWN
    link/ether 7e:a3:ea:f3:5a:49 brd ff:ff:ff:ff:ff:ff
    inet 192.168.248.3/24 scope global vx1234
       valid_lft forever preferred_lft forever
    inet6 fe80::7ca3:eaff:fef3:5a49/64 scope link
       valid_lft forever preferred_lft forever
```

Jail and VM Network Demo

Questions?

- Slides will be posted soon
- Thank you!