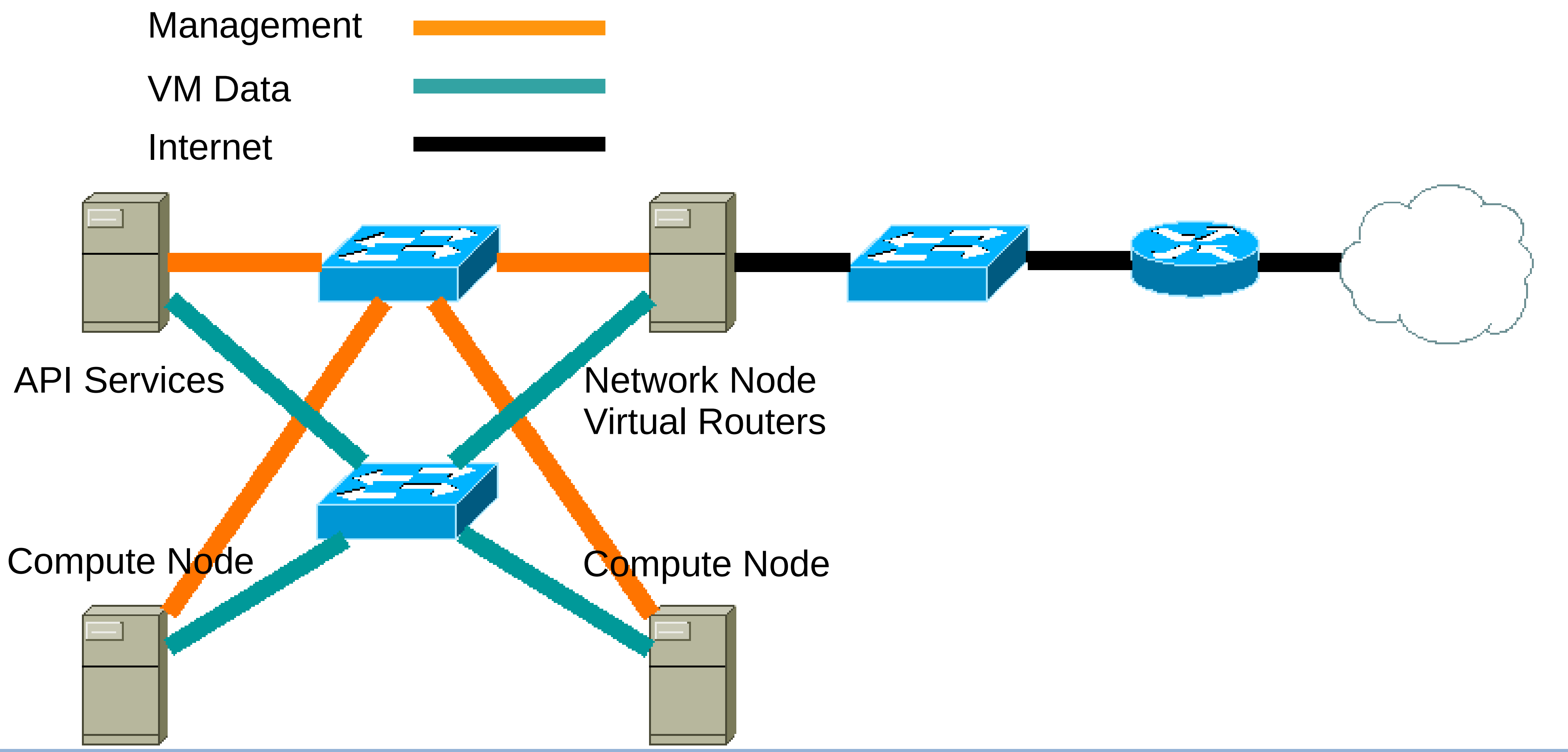
Tunnels in OpenStack

Tunnels as a Connectivity and Segregation Solution for Virtualized Networks

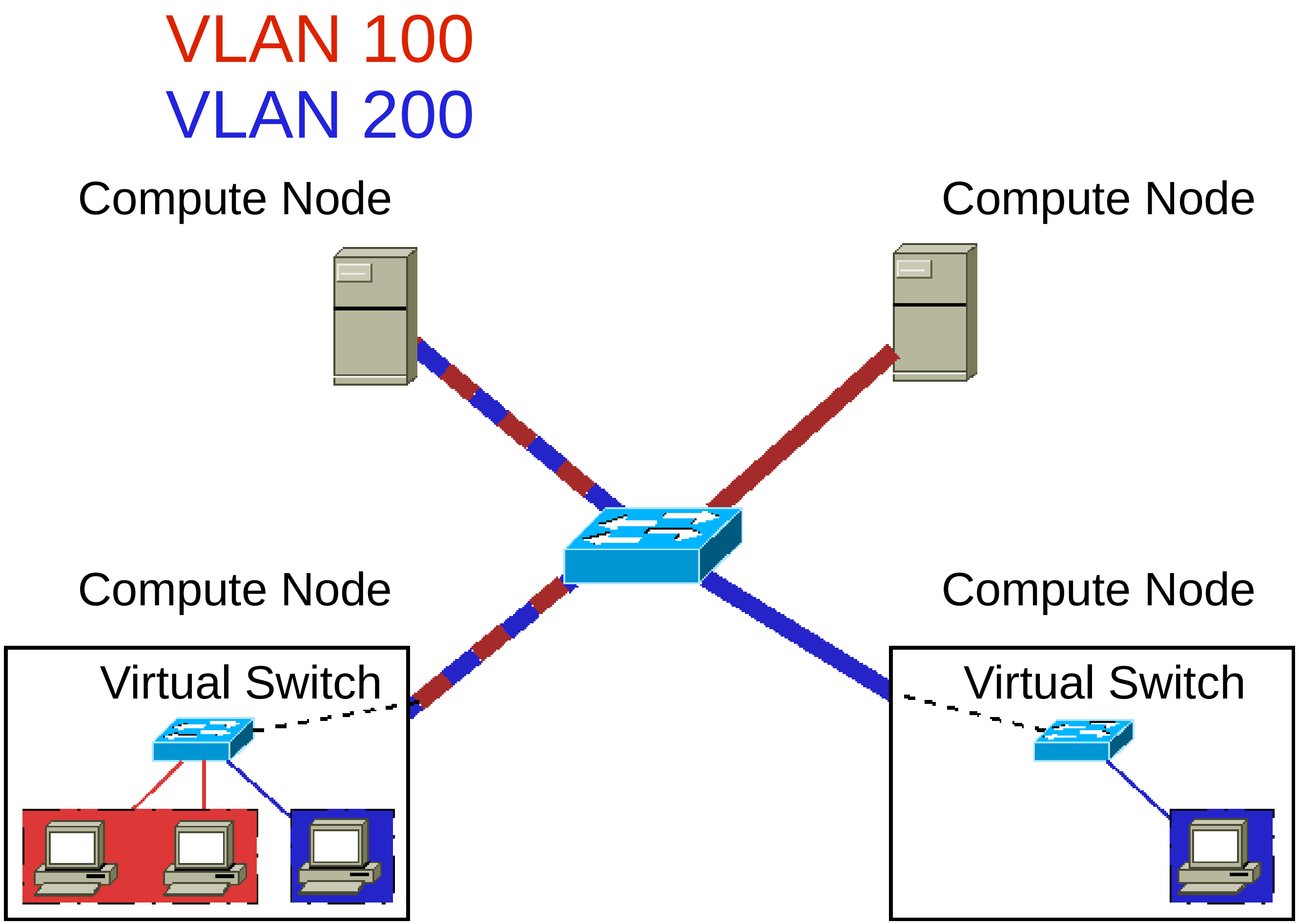
Assaf Muller, Associate Software Engineer,
Cloud Networking, Red Hat

assafmuller.wordpress.com, amuller@redhat.com, amuller on Freenode (#openstack)

Networks Topology



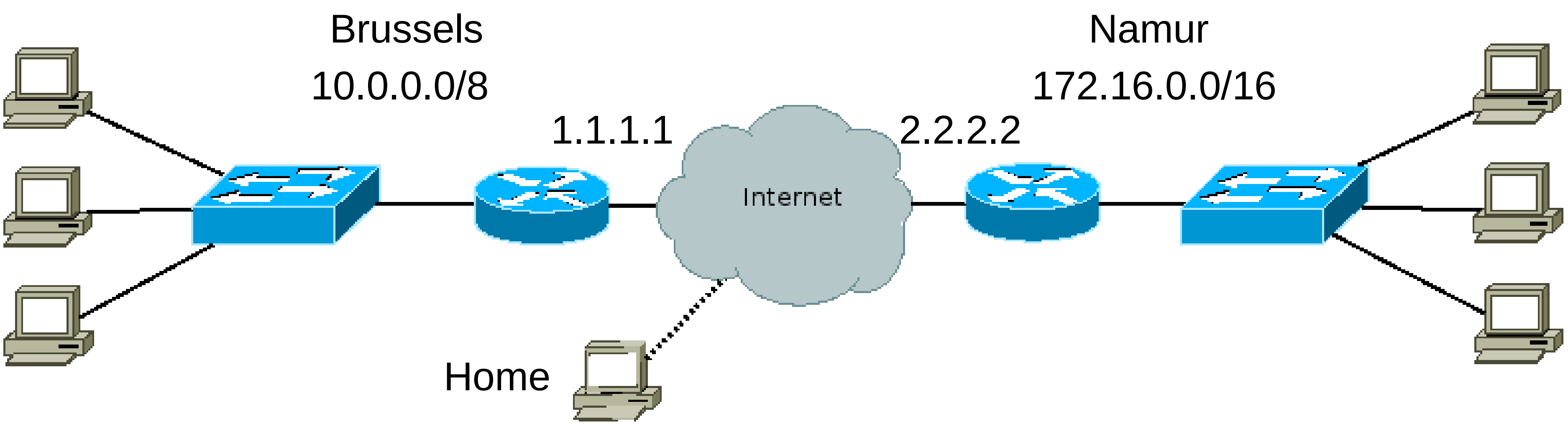
Compute Node & VLANs



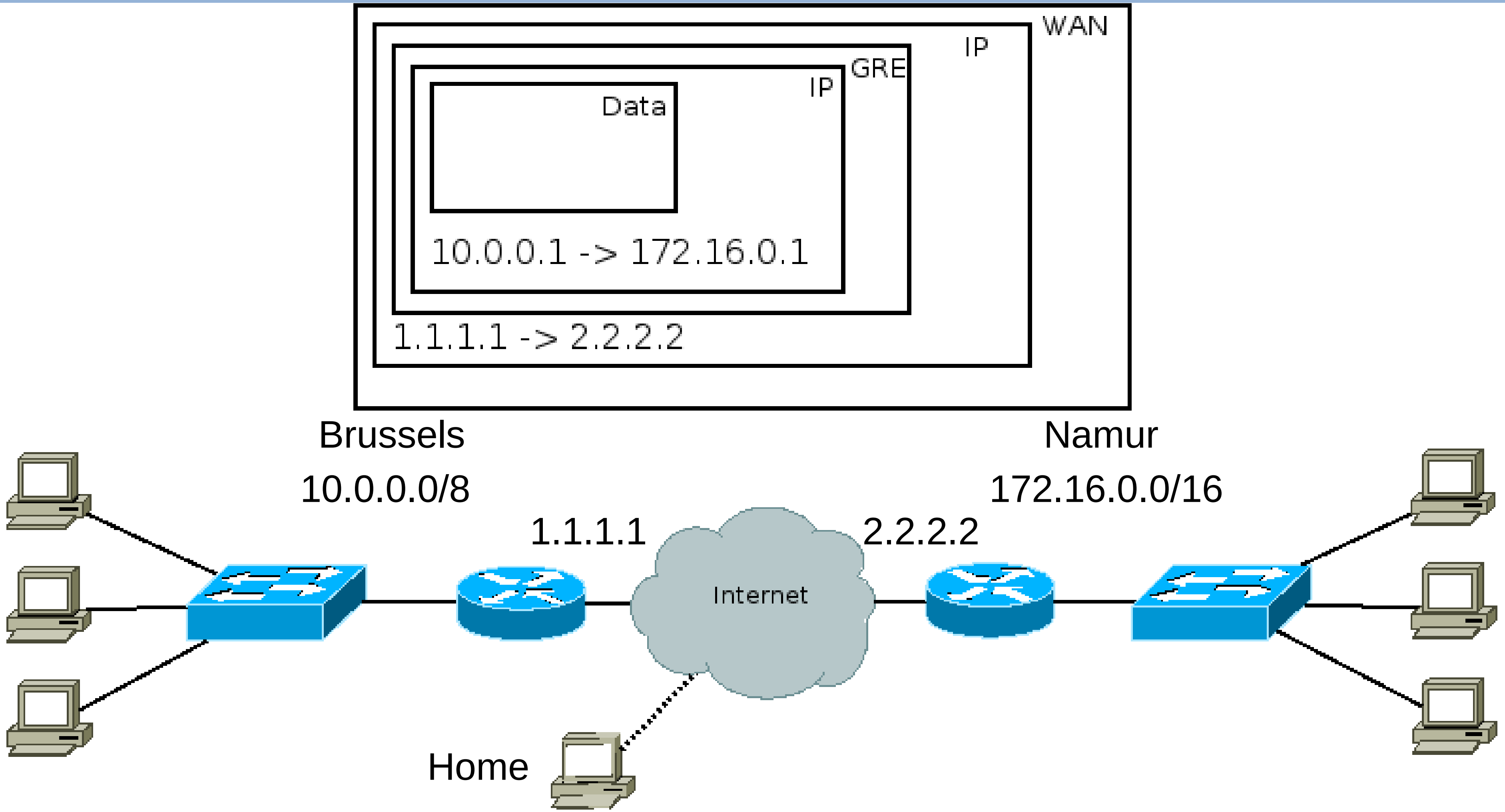
- **Manually** configure VLANs on physical switches
- VLANs where required – Tedious and rigid
- All VLANs everywhere – Simple but inefficient
- Extend VLANs into the virtual world

Tunnels in the Physical World

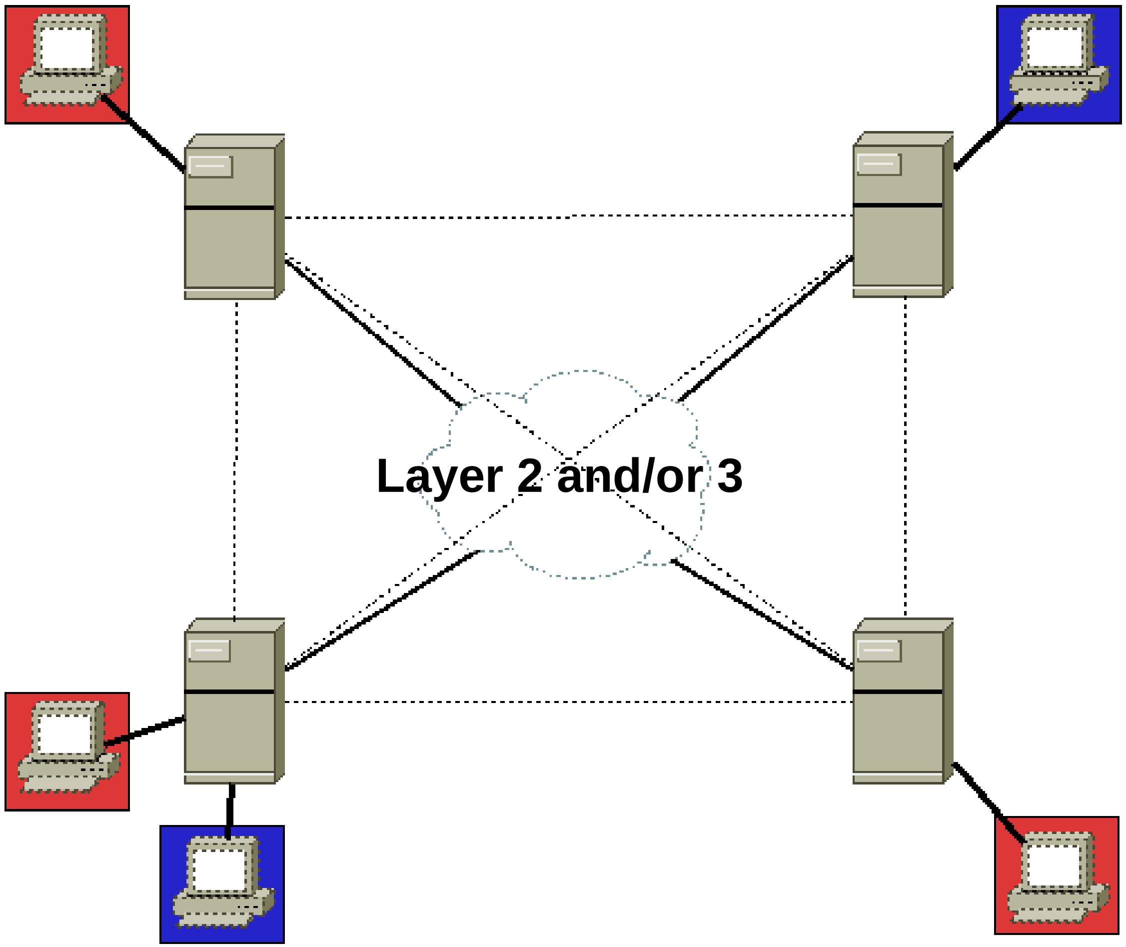
- GRE/VXLAN/Tunnel – Like a VPN, but not encrypted
- Connect two sites
- Work from home
- SSH from host (behind NAT) in Tel-Aviv to host in Sydney
- Access site resources



Encapsulation



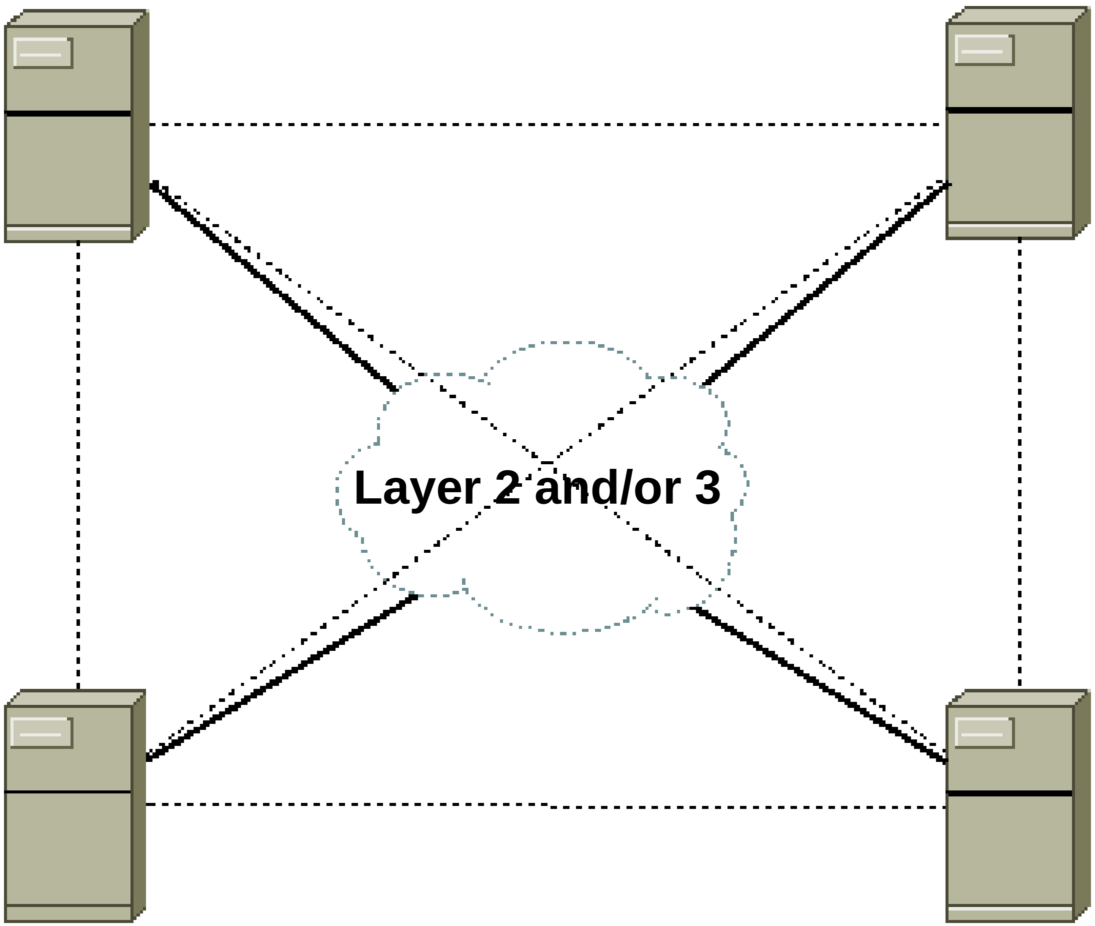
Connectivity



- Tunnels formed between all hypervisors
- VM traffic is encapsulated into traffic between hypervisors
- Hypervisors just need L3 connectivity

- Local connectivity (VMs in same hypervisor) – Same as with VLANs – One shared switch

Segregation



- Tunnel traffic is tagged, much like VLAN traffic
- Each network gets its own tunnel ID
- Incoming traffic can be identified by its tunnel ID

- Local segregation (VMs in same hypervisor) – Same as with VLANs – Locally significant VLAN tagging

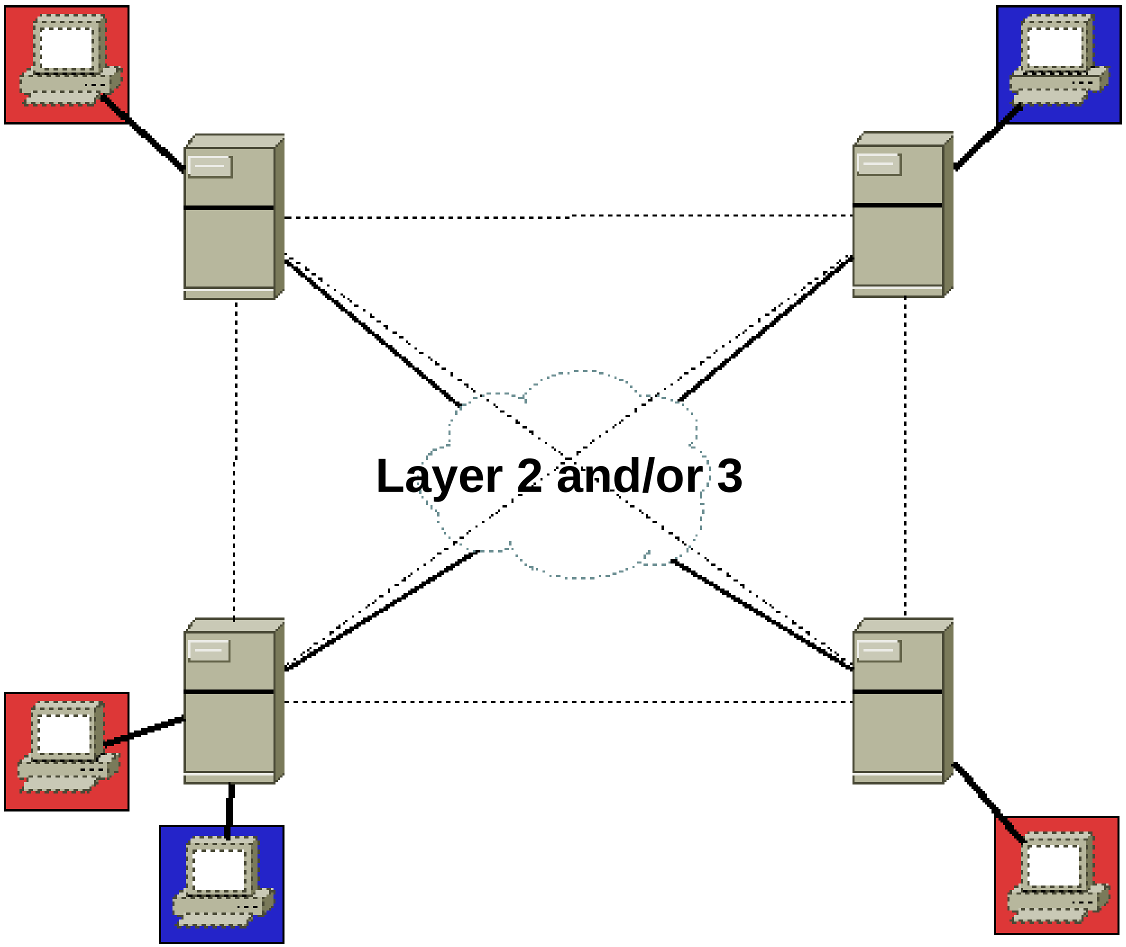
Unicast Traffic

Port	MAC
1	A
2	B
3	C, D, E

Peer & Tunnel ID	Tunnel ID & MAC
10.0.0.1, 1	1, A
10.0.0.1, 2	2, B
10.0.0.2, 3	3, (C, D, E)

- Reminder – Layer 2 learning switches map incoming port to source MAC
- Virtual switch on hypervisor maps incoming tunnel ID & peer to source MAC
- Learned unicast addresses are persisted, unknown unicast traffic is flooded

Broadcast Traffic



- Unknown unicast multicast, and broadcast traffic – Historically go out through all tunnels
- Can we do better?
  - Minimize broadcasts – hypervisors answer local ARP requests*
  - Optimize broadcasts - Forward broadcast traffic only to eligible hypervisors**

* ML2 plugin with Linux bridge mechanism driver since Havana. OVS planned for Icehouse
** ML2 plugin since Havana

Open vSwitch

- Open vSwitch bridges operate in one of two modes:
  - Normal mode is a regular layer 2 learning switch
  - Flow mode is entirely custom behavior
- Flows can be configured via:
  - Local ovs-ofctl commands
  - Remote OpenFlow calls
- neutron-openvswitch-agent configures br-tun (Tunneling bridge) via local ovs-ofctl commands, following controller RPC calls

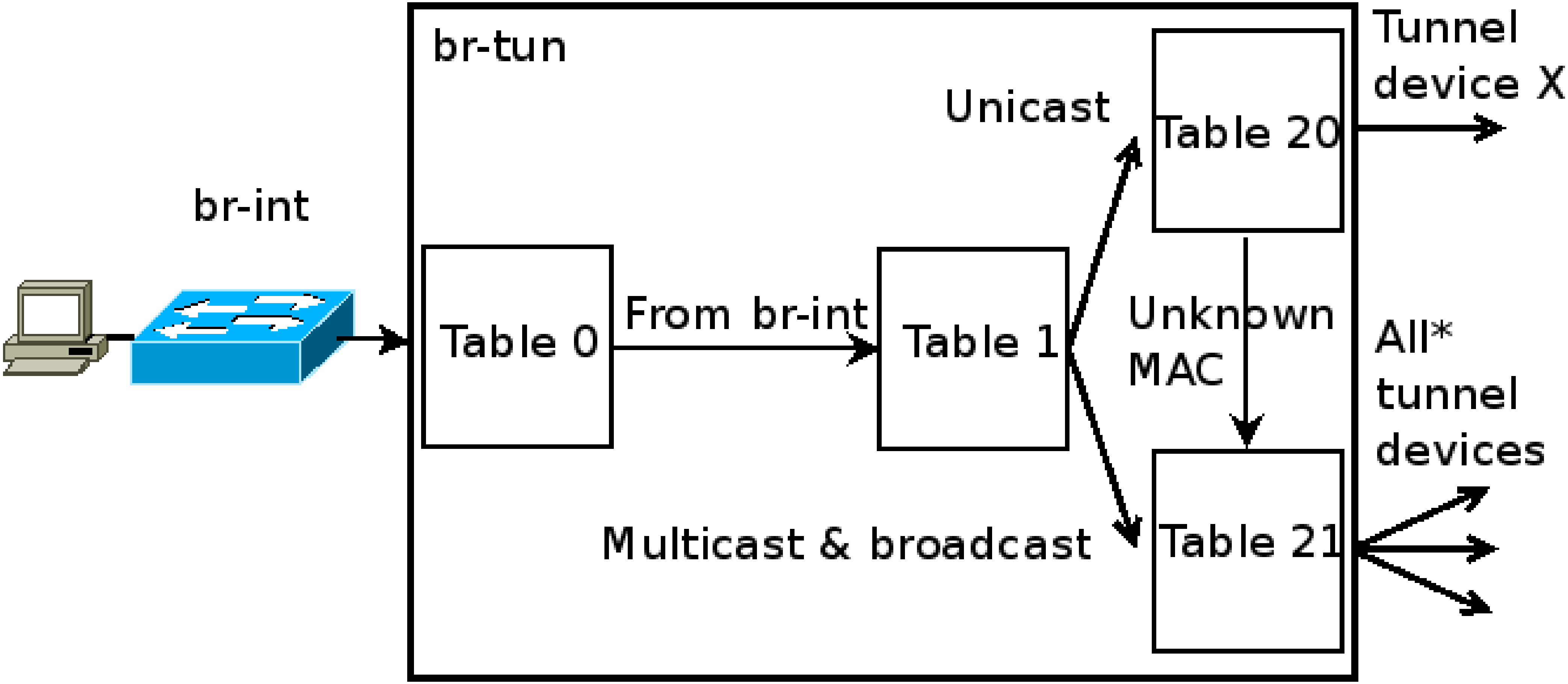
Flows

- Flows have a match and action part:
  - Should the flow process an incoming message?
    - Match against layer 2, 3, 4 headers
  - What to do:
    - Change headers
    - Forward to one or more ports
    - Broadcast
    - Drop
    - Insert new flows
    - Resubmit to another table

Tables

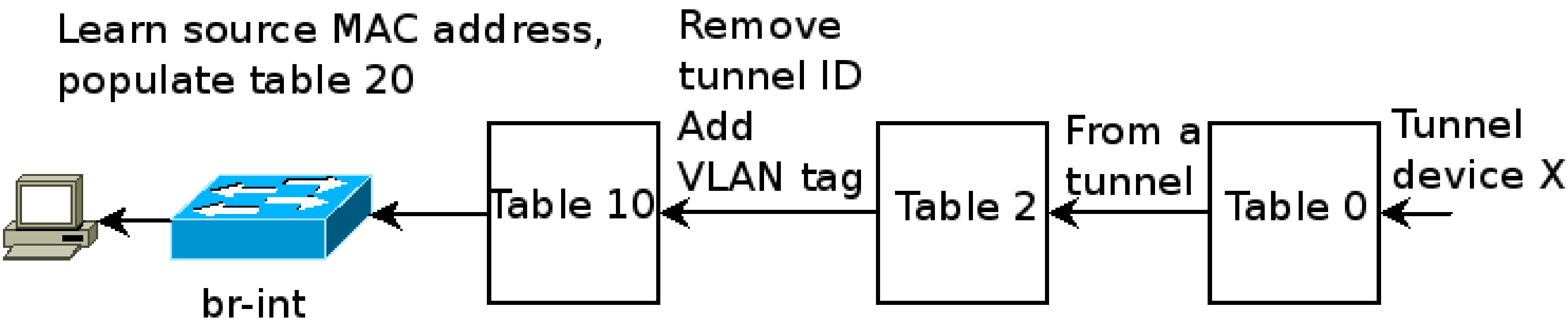
- Bridges have multiple tables:
  - Messages enter table 0
  - Messages can be resubmitted to other tables
  - Each table's flows are processed by priority, table has implicit drop at the end (Or send message to SDN controller If one is configured)

From a VM on the local node



* Depending if MAC learning mechanism is enabled

From a VM on a remote node



More Information

- Official OVS configuration tutorial
- Scott Lowe's (amazing) GRE blog posts
- ovs-vsctl show
- ovs-ofctl dump-flows br-tun
- assafmuller.wordpress.com (Shameless plug!)

Questions?

Tunnels as a Connectivity and Segregation Solution for Virtualized Networks

Assaf Muller, Associate Software Engineer,
Cloud Networking, Red Hat

assafmuller.wordpress.com, amuller@redhat.com, amuller on Freenode (#openstack)