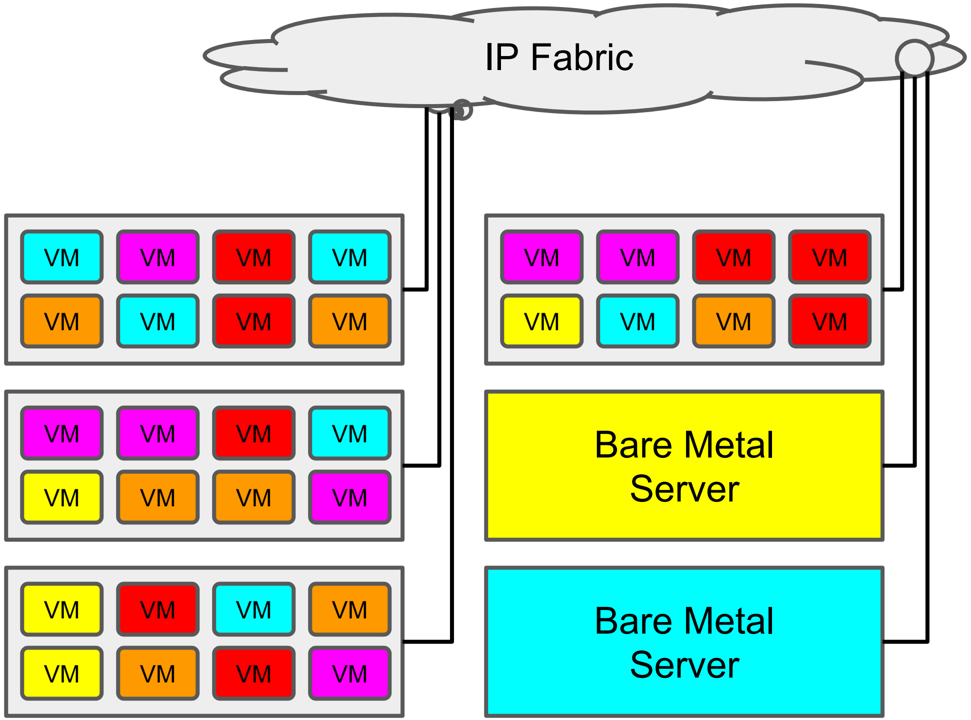
Challenges in Distributed SDN

Duarte Nunes
duarte@midokura.com
@duarte_nunes

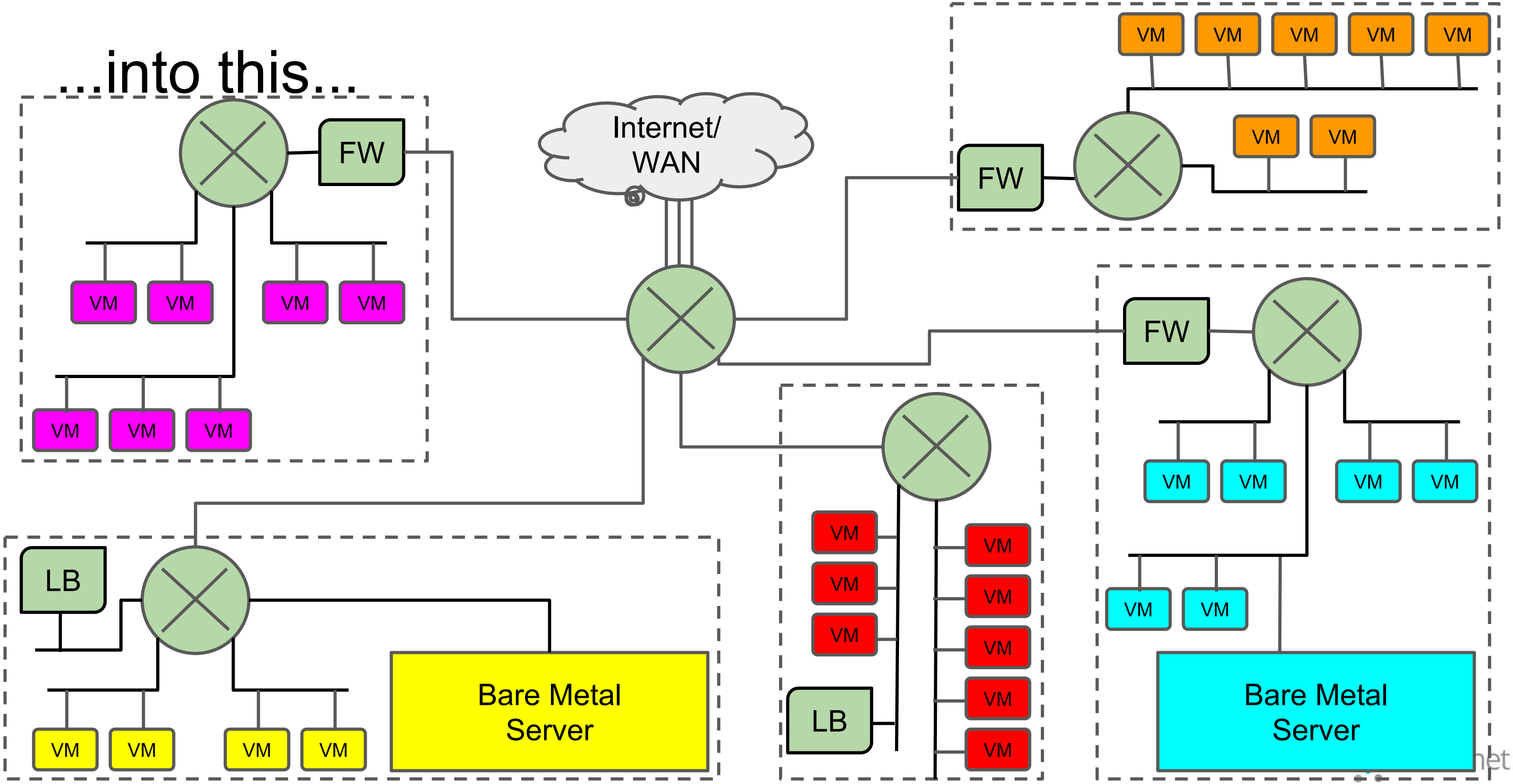
midonet

MidoNet transform this...



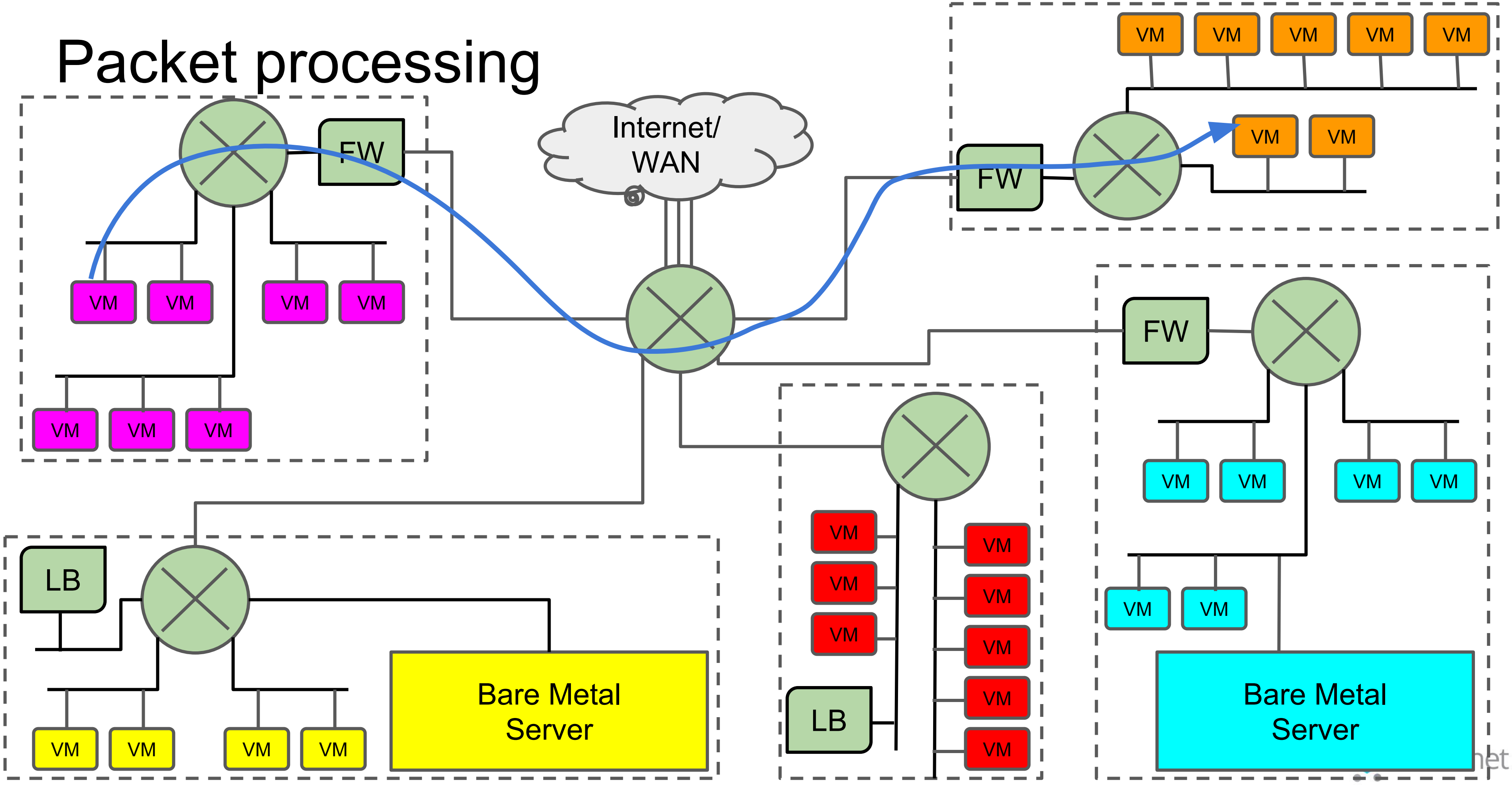
midonet

...into this...

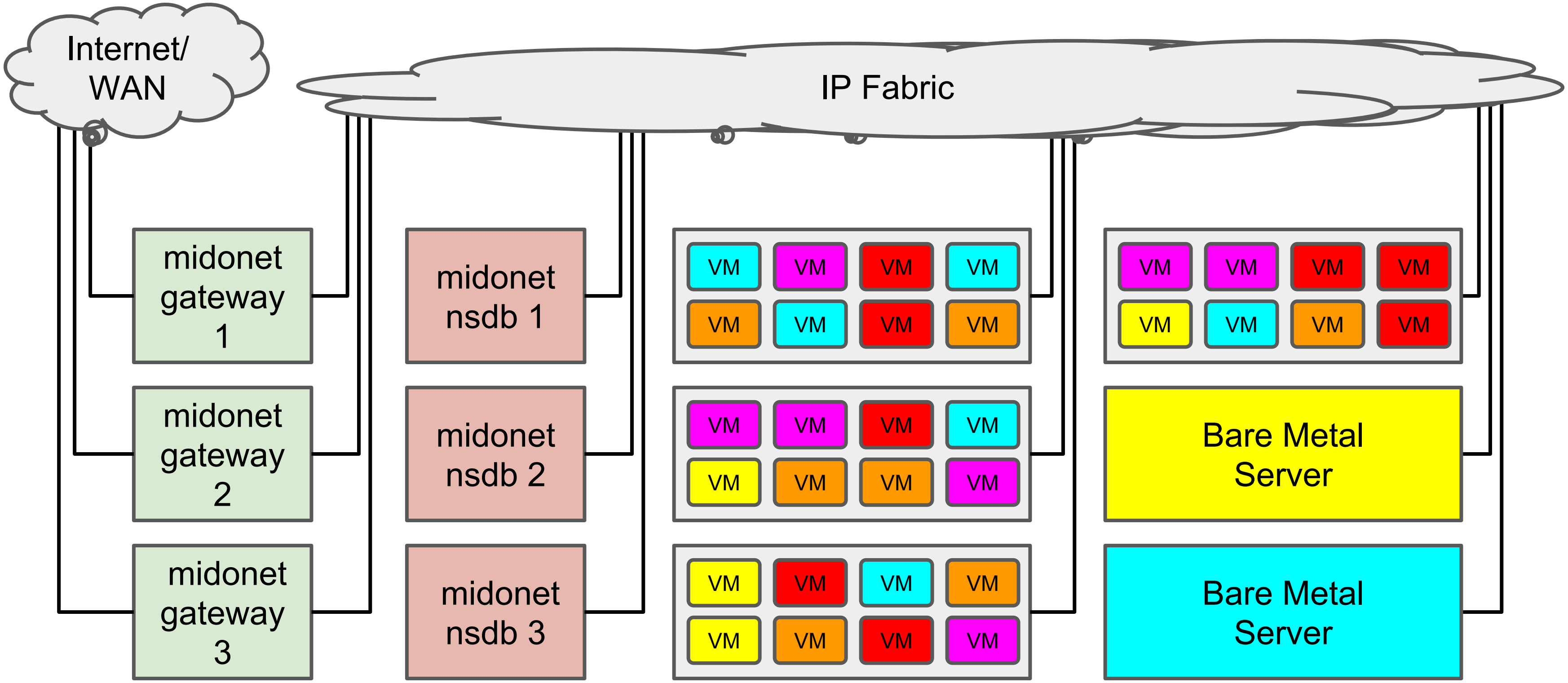


Internet/
WAN
FW
FW
FW
LB
LB
VM
Bare Metal
Server
Bare Metal
Server

Packet processing



Physical view



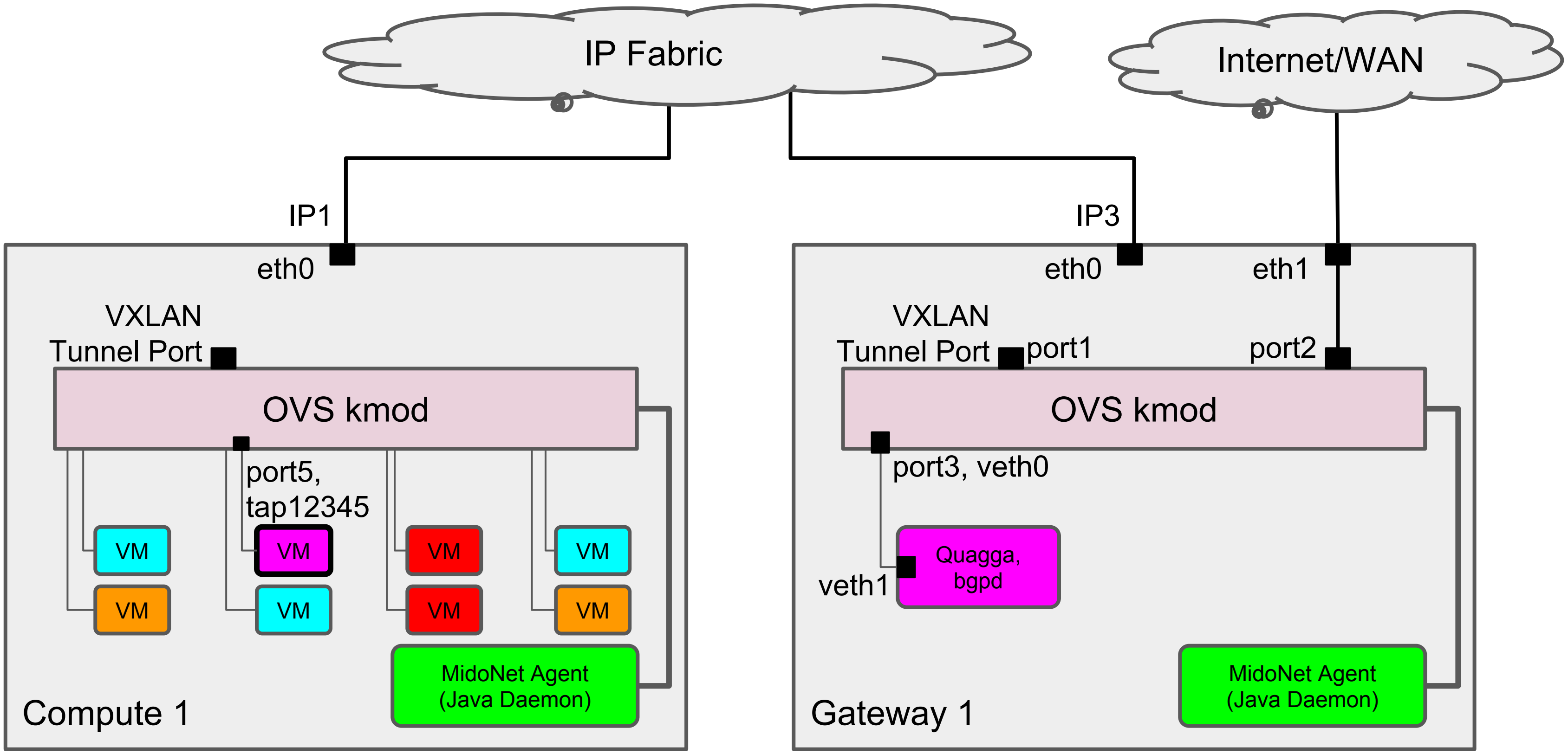
midonet

MidoNet

- Fully distributed architecture
- All traffic processed at the edges, i.e., where it ingresses the physical network
  - virtual devices become distributed
  - a packet can traverse a particular virtual device at any host in the cloud
  - distributed virtual bridges, routers, NATs, FWs, LBs, etc.
- No SPOF
- No middle boxes
- Horizontally scalable L2 and L3 Gateways

midonet

MidoNet Hosts



midonet

Flow computation and tunneling

- Flows are computed at the ingress host
  - by simulating a packet's path through the virtual topology
  - without fetching any information off-box (~99% of the time)
- Just-in-time flow computation
- If the egress port is on a different host, then the packet is tunneled
  - the tunnel key encodes the egress port
  - no computation is needed at the egress

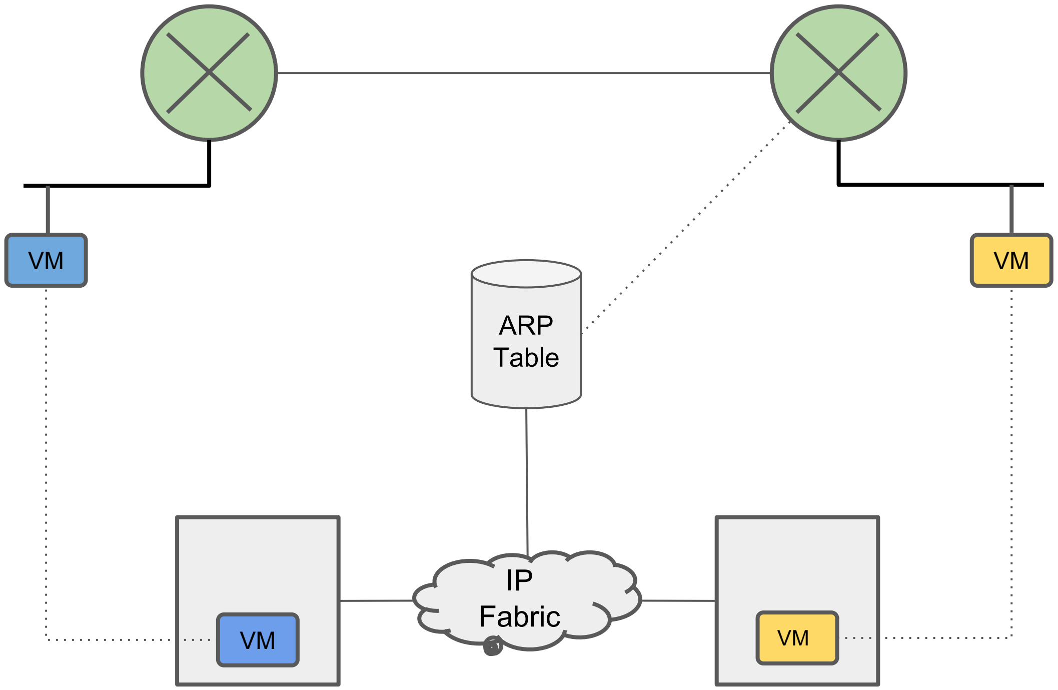
Virtual Devices

Device state

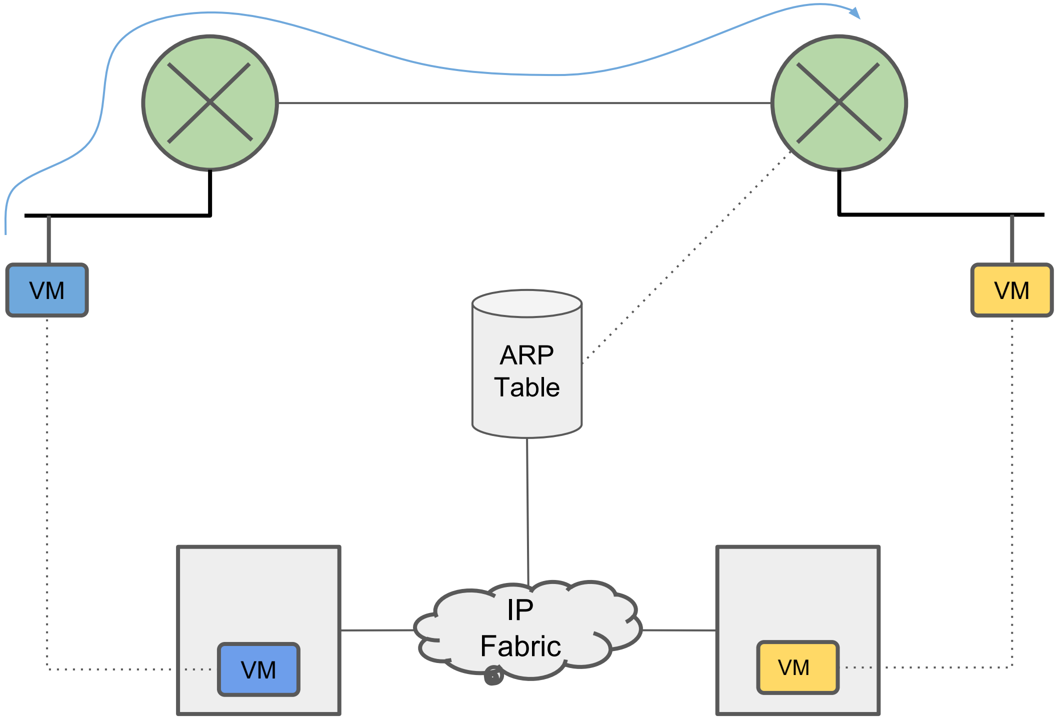
- ZooKeeper serves the virtual network topology
  - reliable subscription to topology changes
- Agents fetch, cache, and “watch” virtual devices on-demand to process packets
- Packets naturally traverse the same virtual device at *different* hosts
- This affects device state:
  - a virtual bridge learns a MAC-port mapping a host and needs to read it in other hosts
  - a virtual router emits an ARP request out of one host and receives the reply on another host
- Store device state tables (ARP, MAC-learning, routes) in ZooKeeper
  - interested agents subscribe to tables to get updates
  - the owner of an entry manages its lifecycle
  - use ZK Ephemeral nodes so entries go away if a host fails

midonet

ARP Table

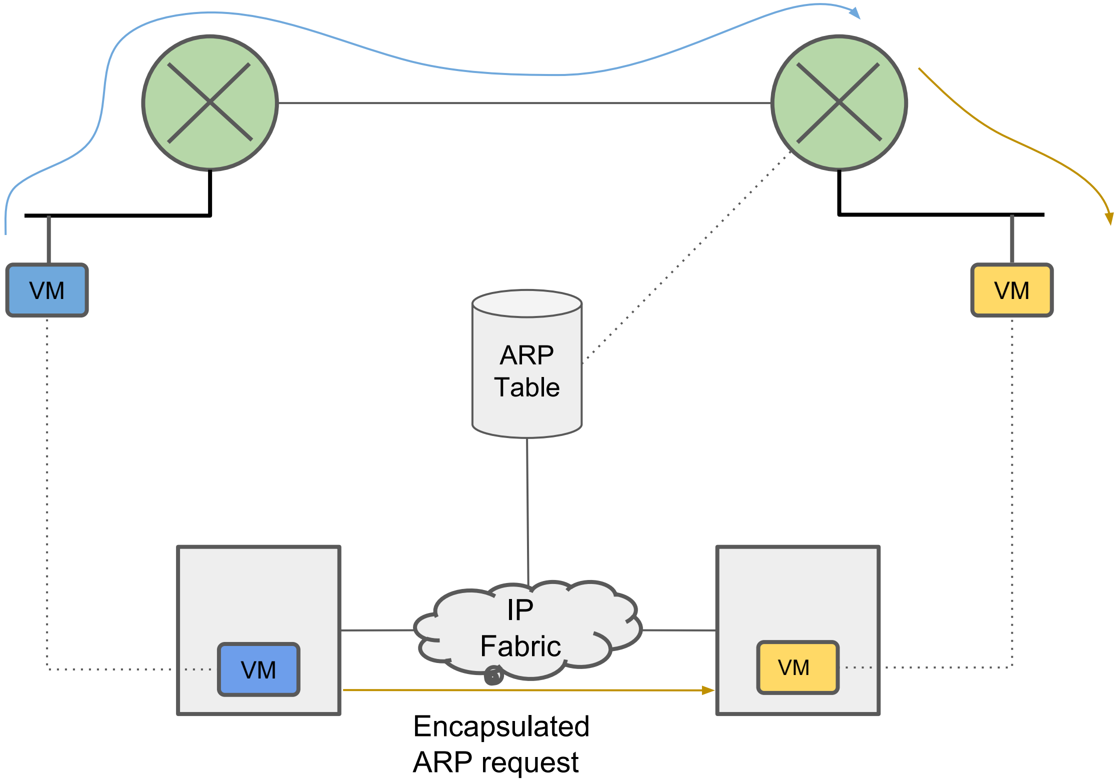


ARP Table



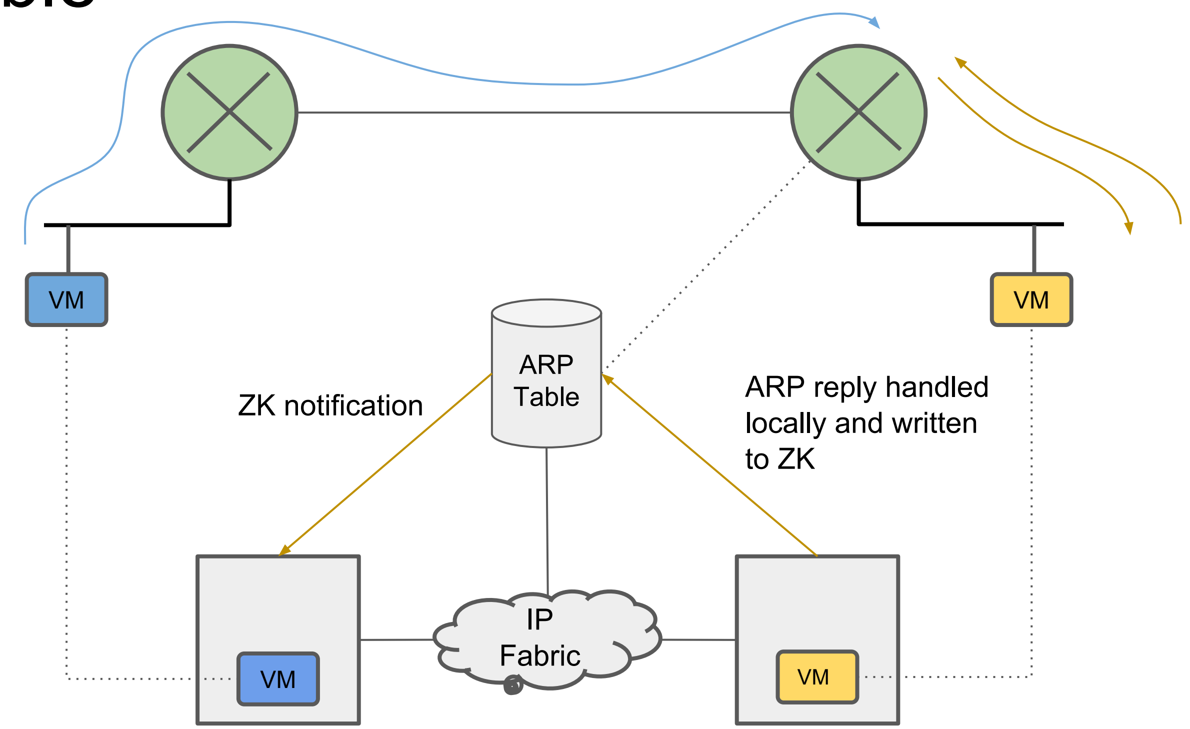
midonet

ARP Table



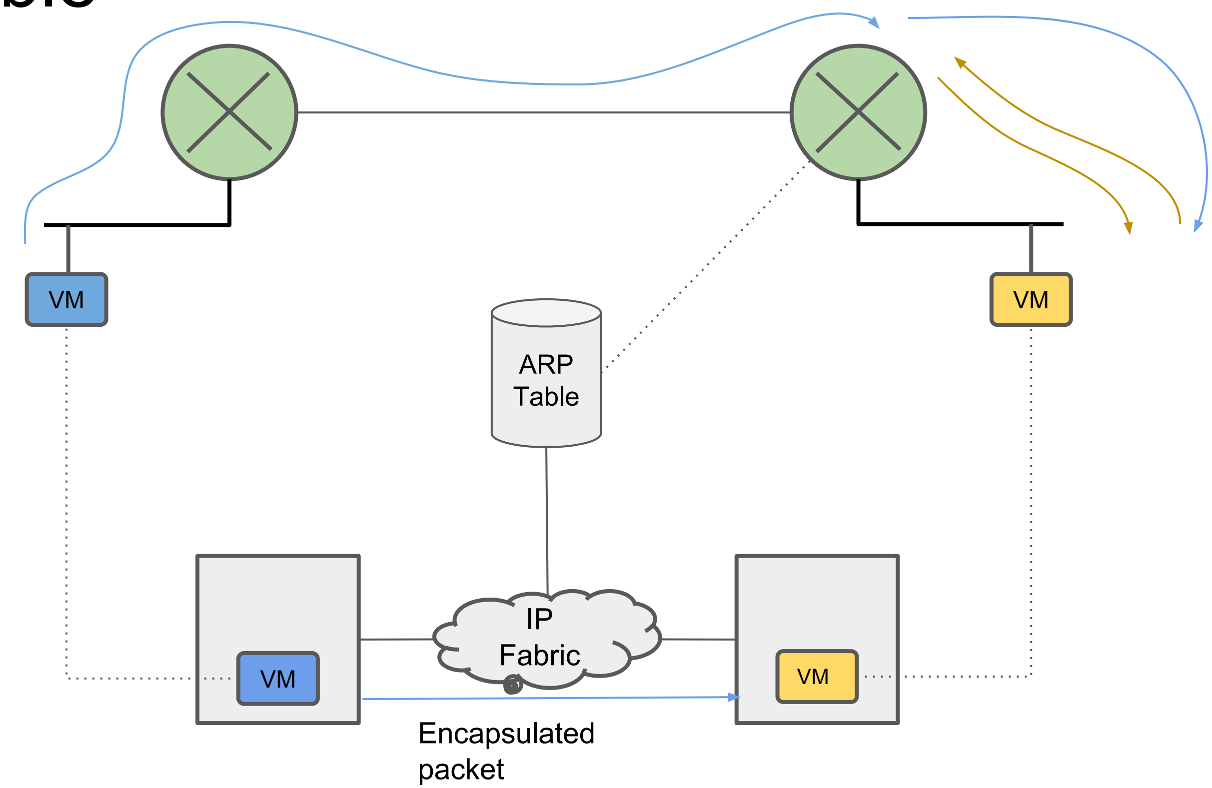
midonet

ARP Table



midonet

ARP Table



midonet

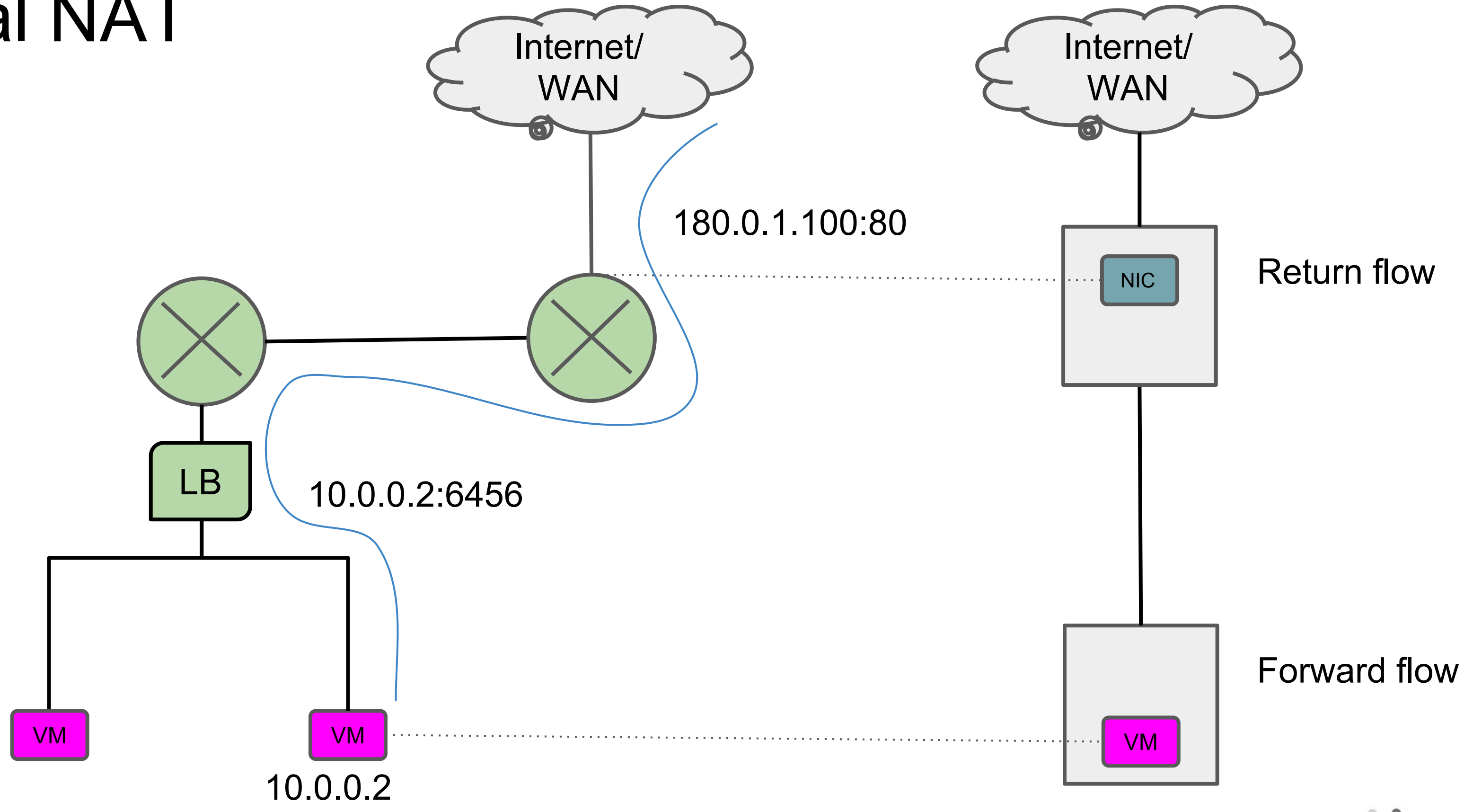
Flow State

Flow state

- Per-flow L4 state, e.g. connection tracking or NAT
- Forward and return flows are typically handled by different hosts
  - thus, they need to share state

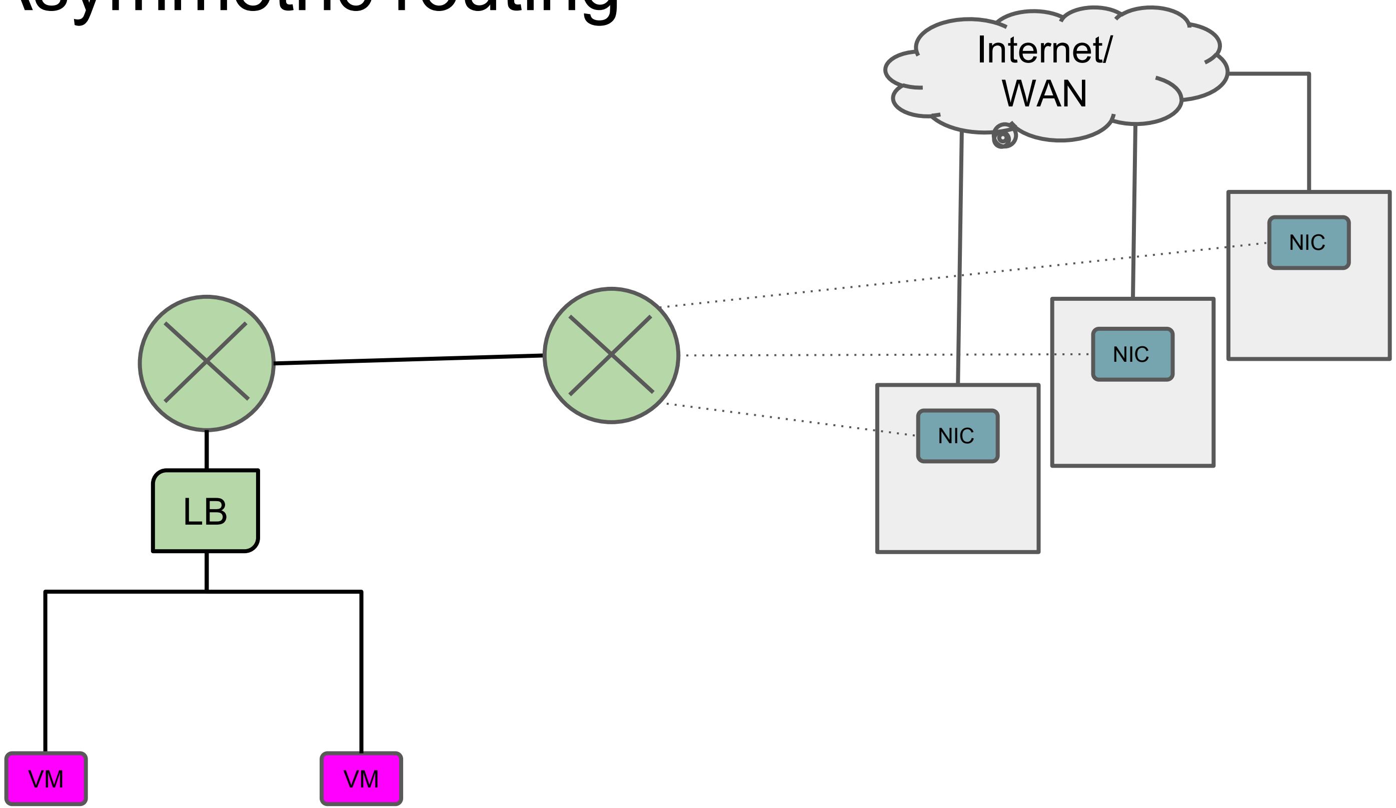
midonet

Virtual NAT

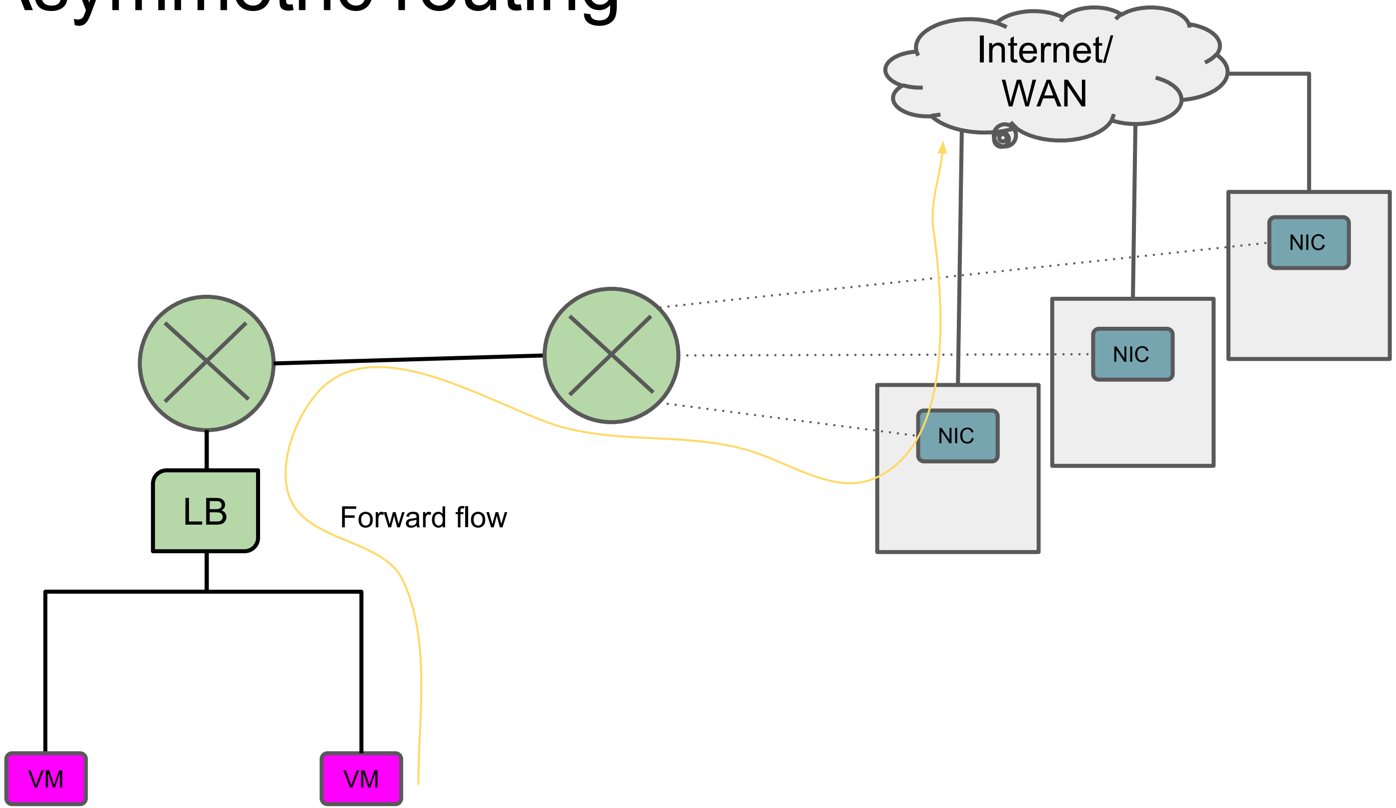


midonet

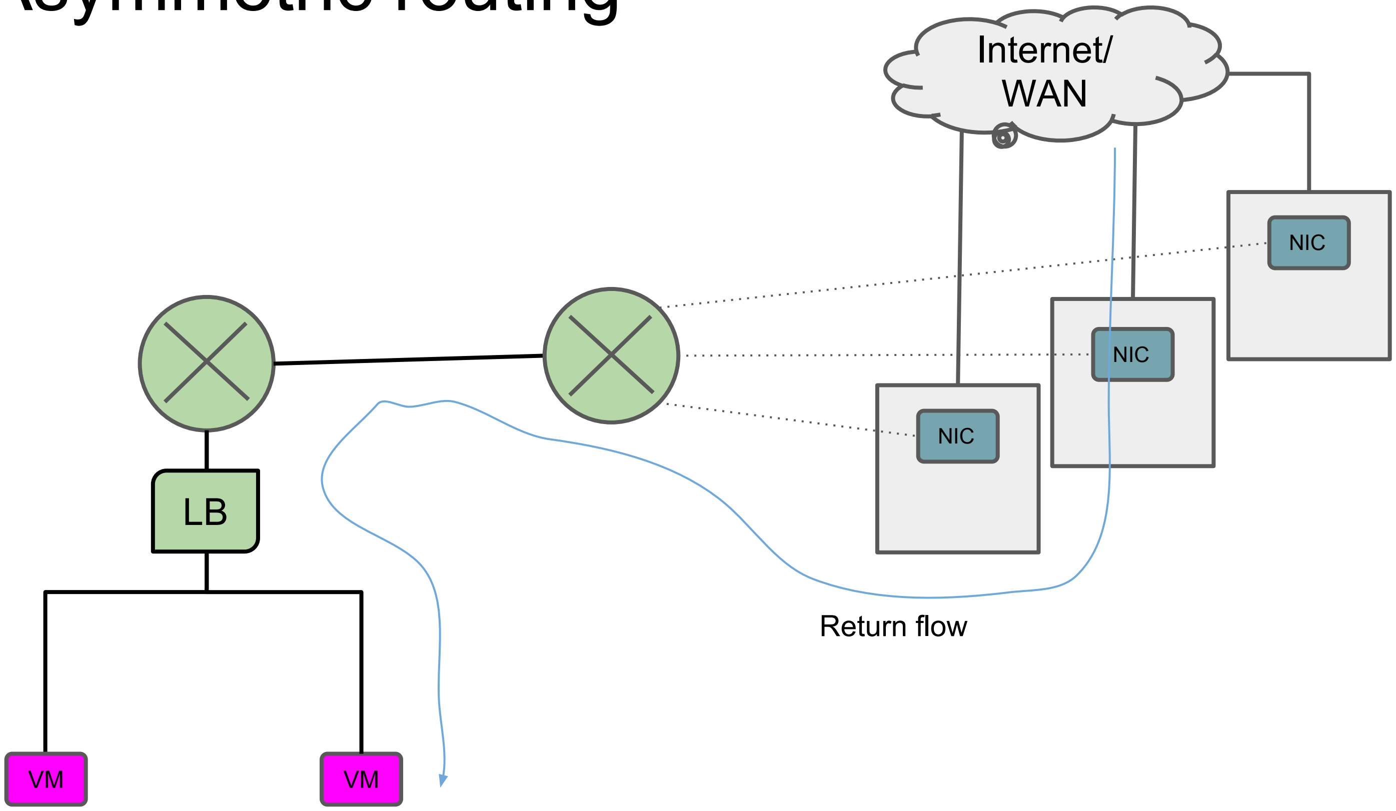
Asymmetric routing



Asymmetric routing

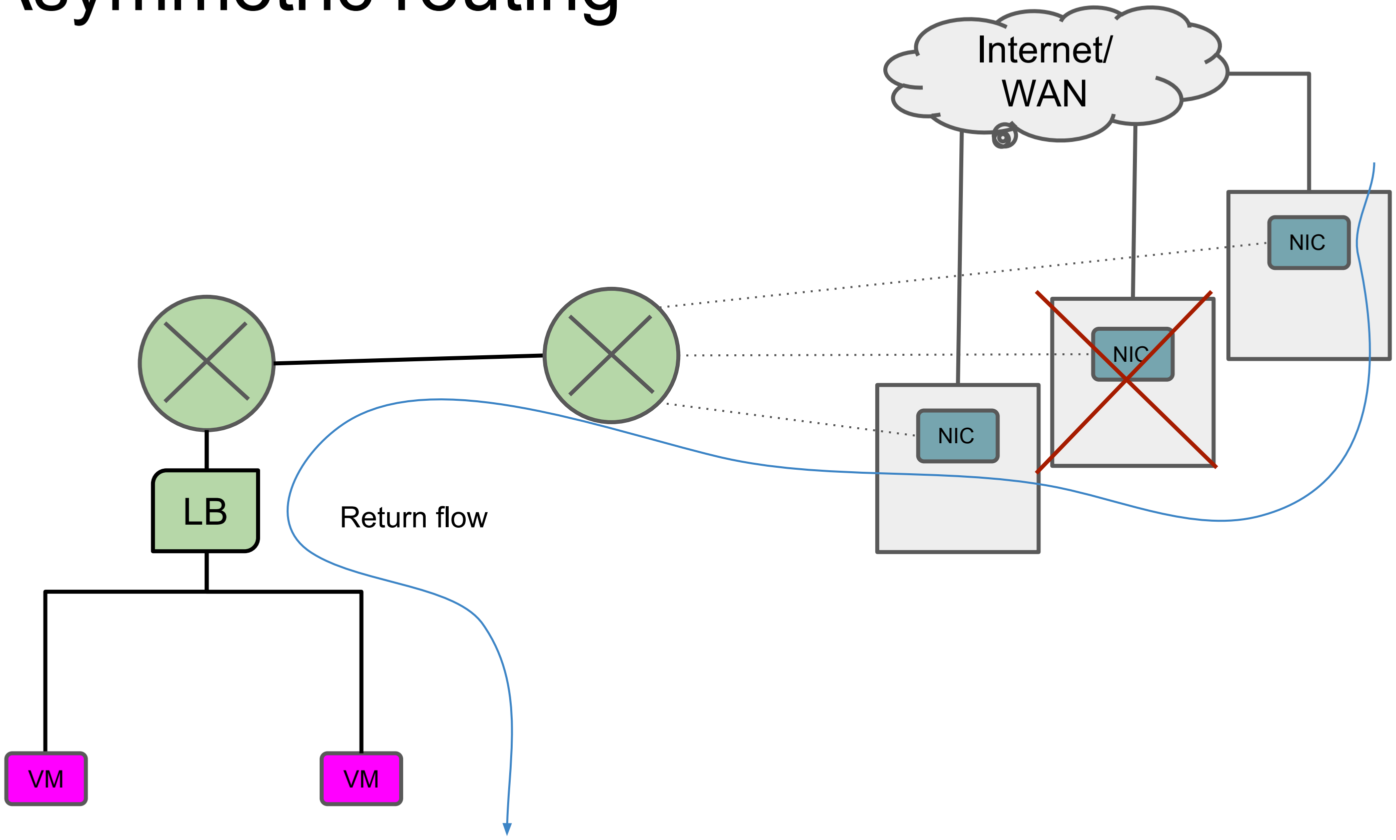


Asymmetric routing



midonet

Asymmetric routing

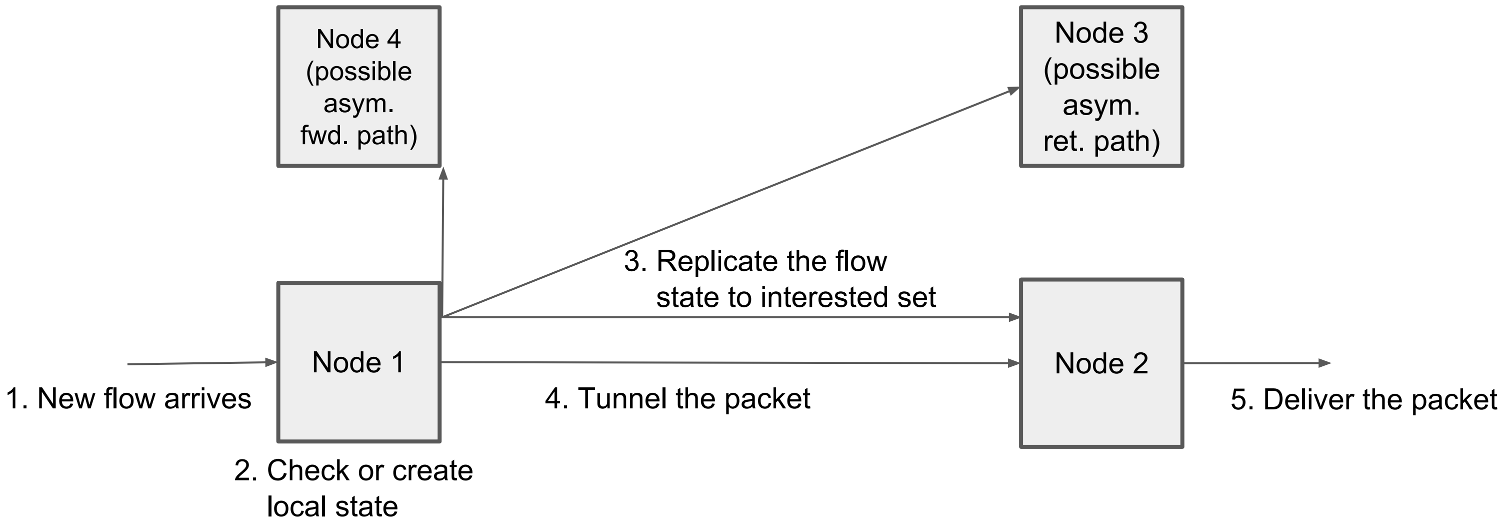


midonet

Flow state

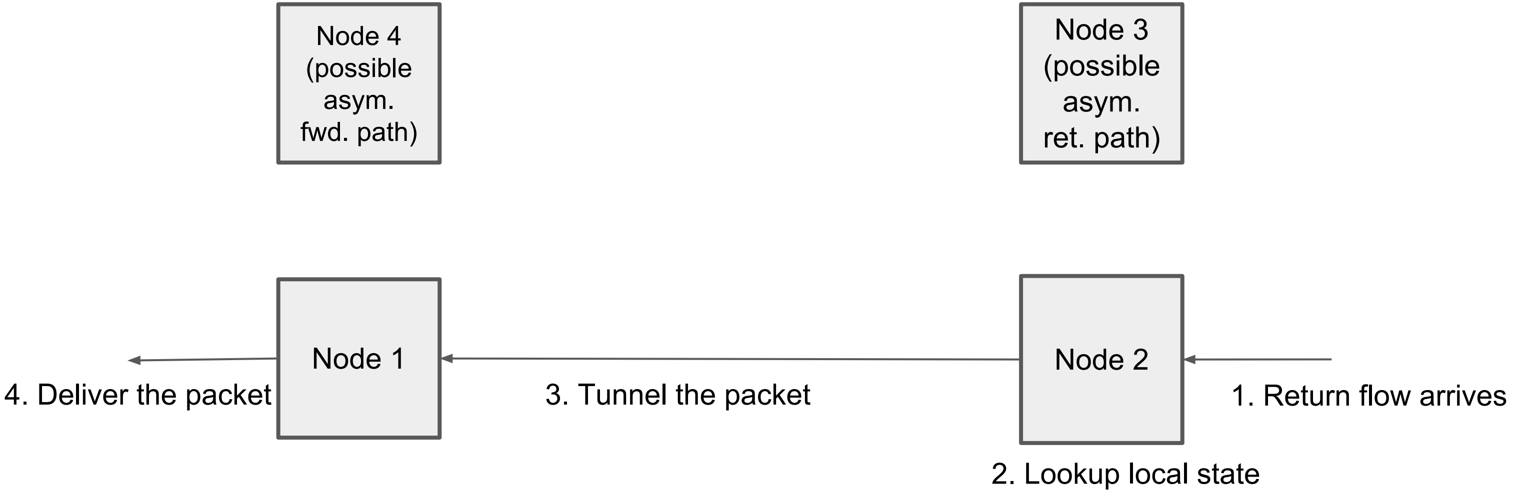
- Connection tracking
  - Key: 5 tuple + ingress device UUID
  - Value: NA
  - Forward state not needed
  - One flow state entry per flow
- NAT
  - Key: 5 tuple + device UUID under which NAT was performed
  - Value: (IP, port) binding
  - Possibly multiple flow state entries per flow
- Key must always be derivable from the packet

Sharing state - Peer-to-peer handoff



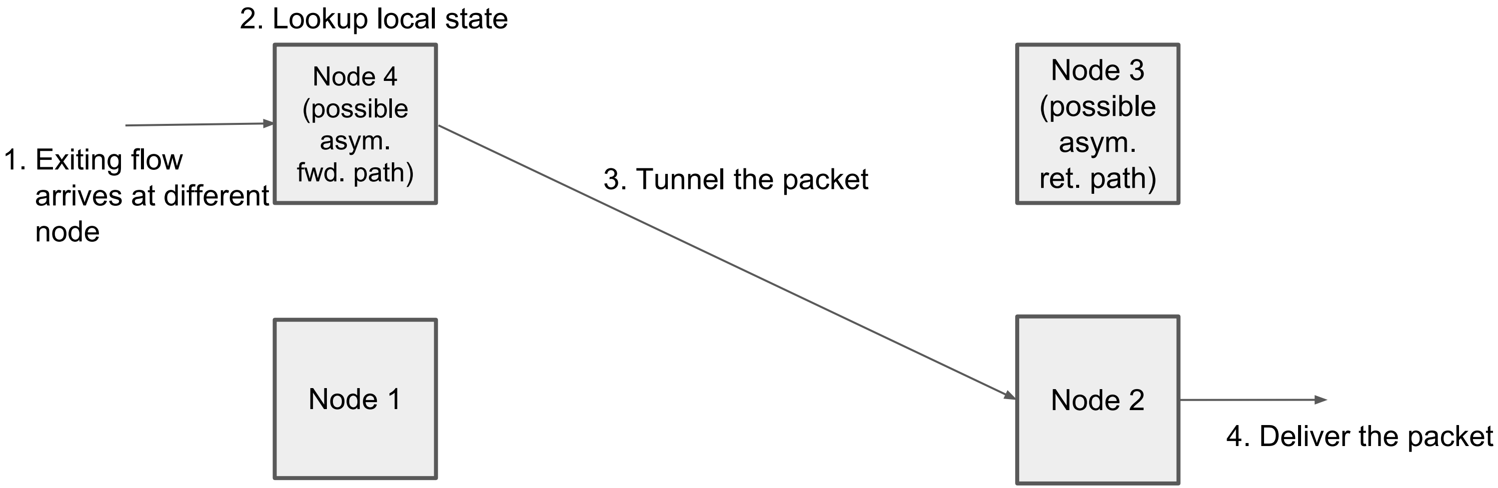
midonet

Sharing state - Peer-to-peer handoff



midonet

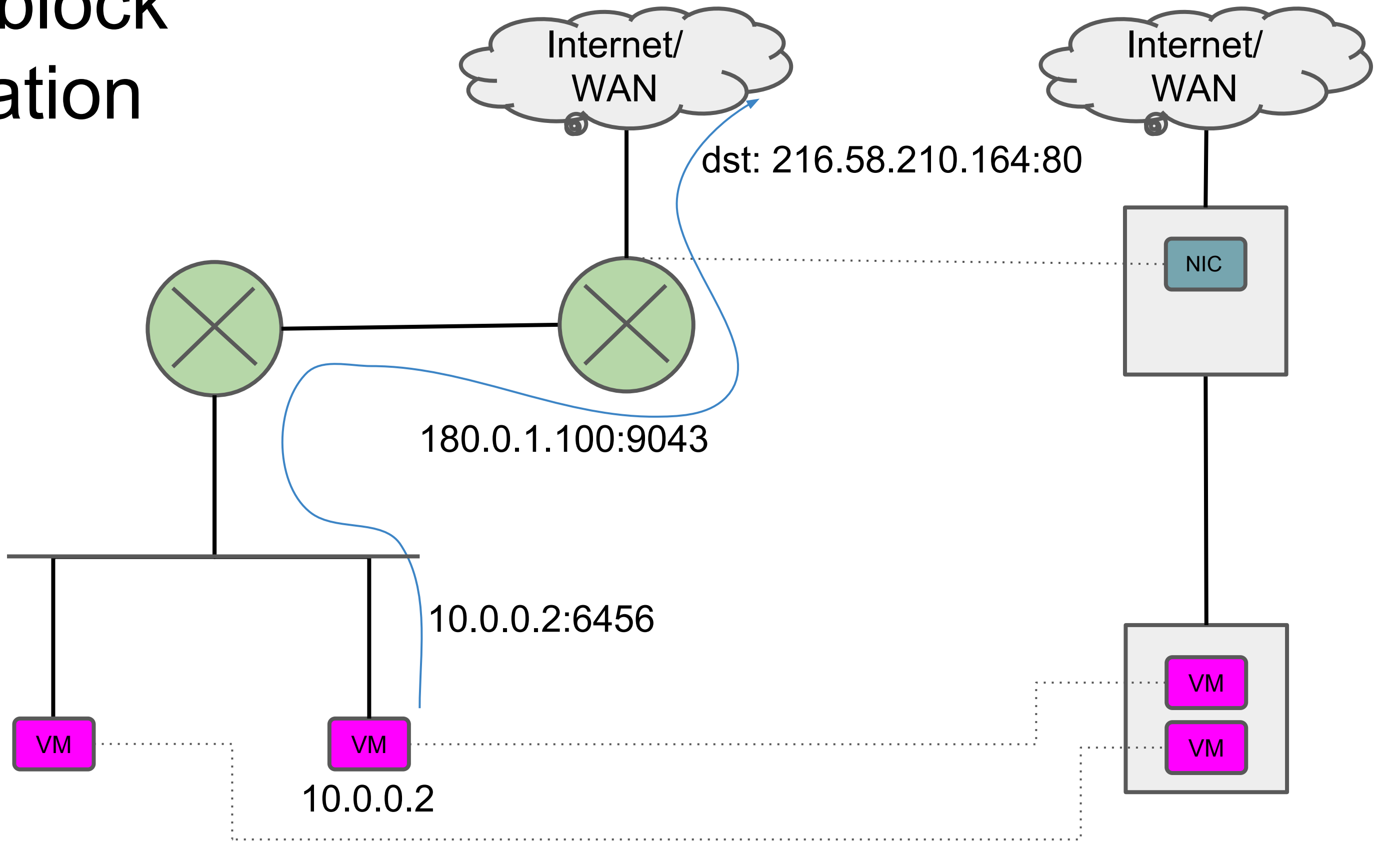
Sharing state - Peer-to-peer handoff



Sharing state - Peer-to-peer handoff

- No added latency
- Fire-and-forget or reliable?
- How often to retry?
- Delay tunneling the packets until the flow state has propagated or accept the risk of the return flow being computed without the flow state?

SNAT block reservation



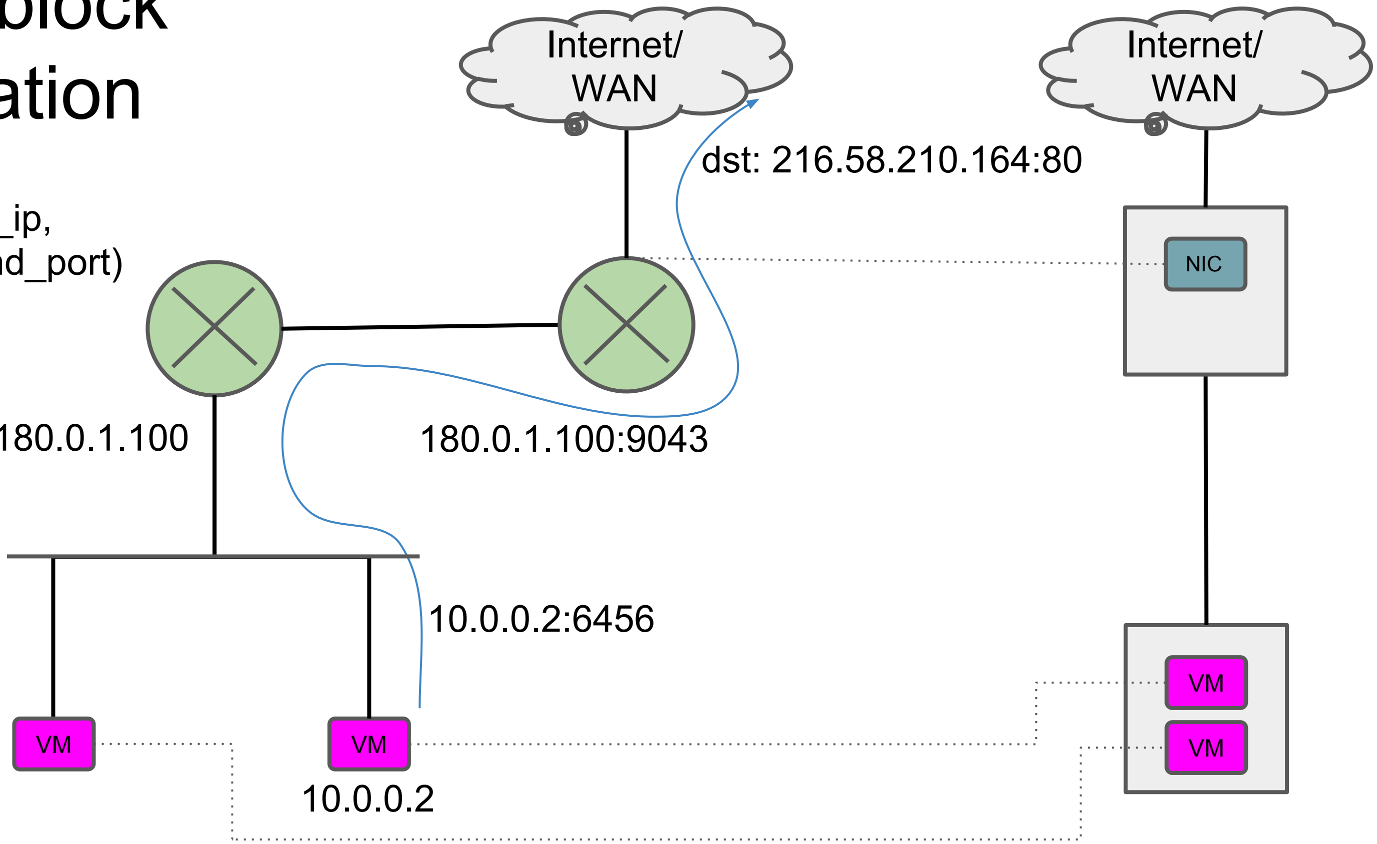
midonet

SNAT block reservation

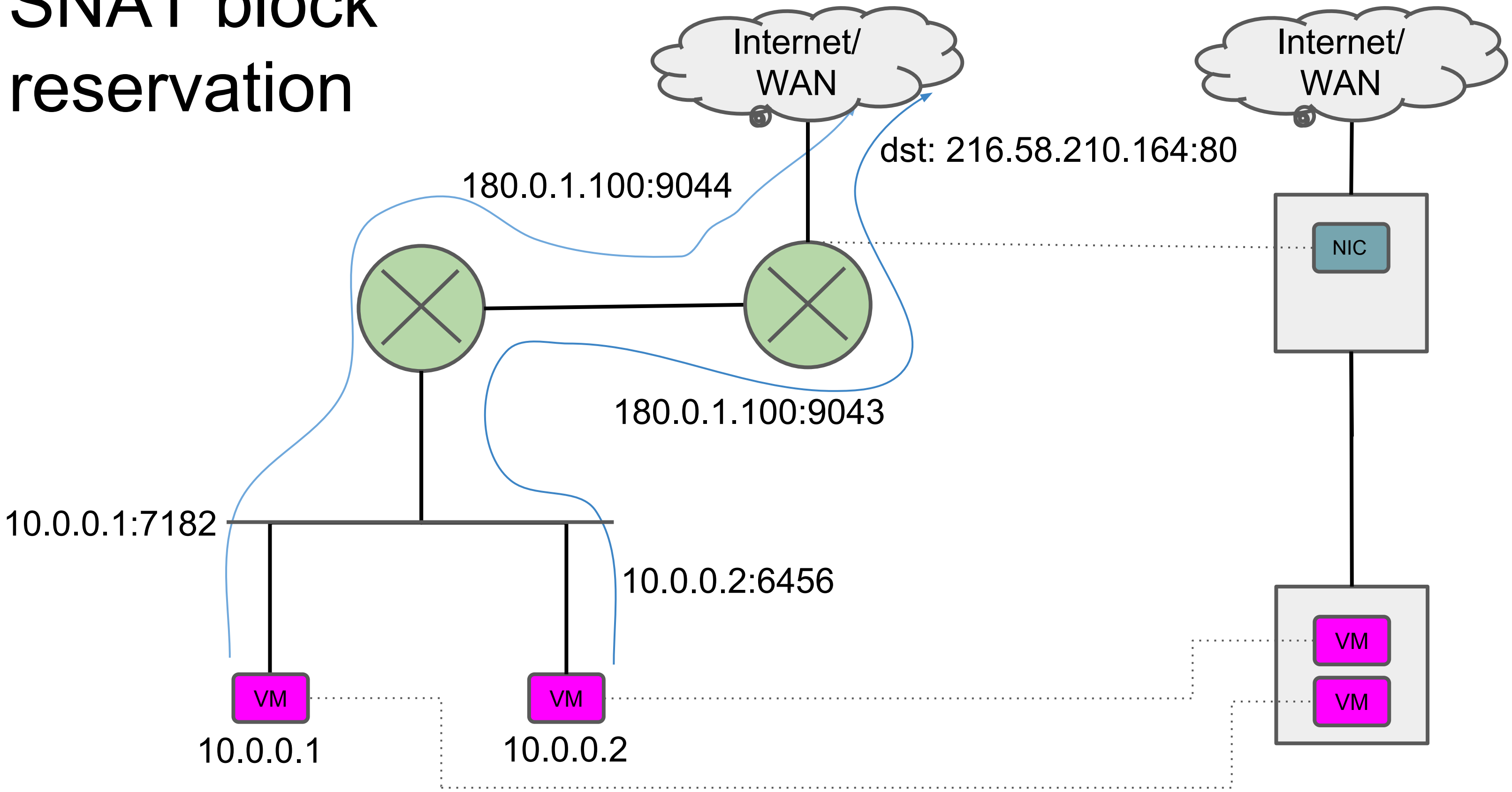
NAT Target:
(start_ip..end_ip,
start_port..end_port)

e.g.

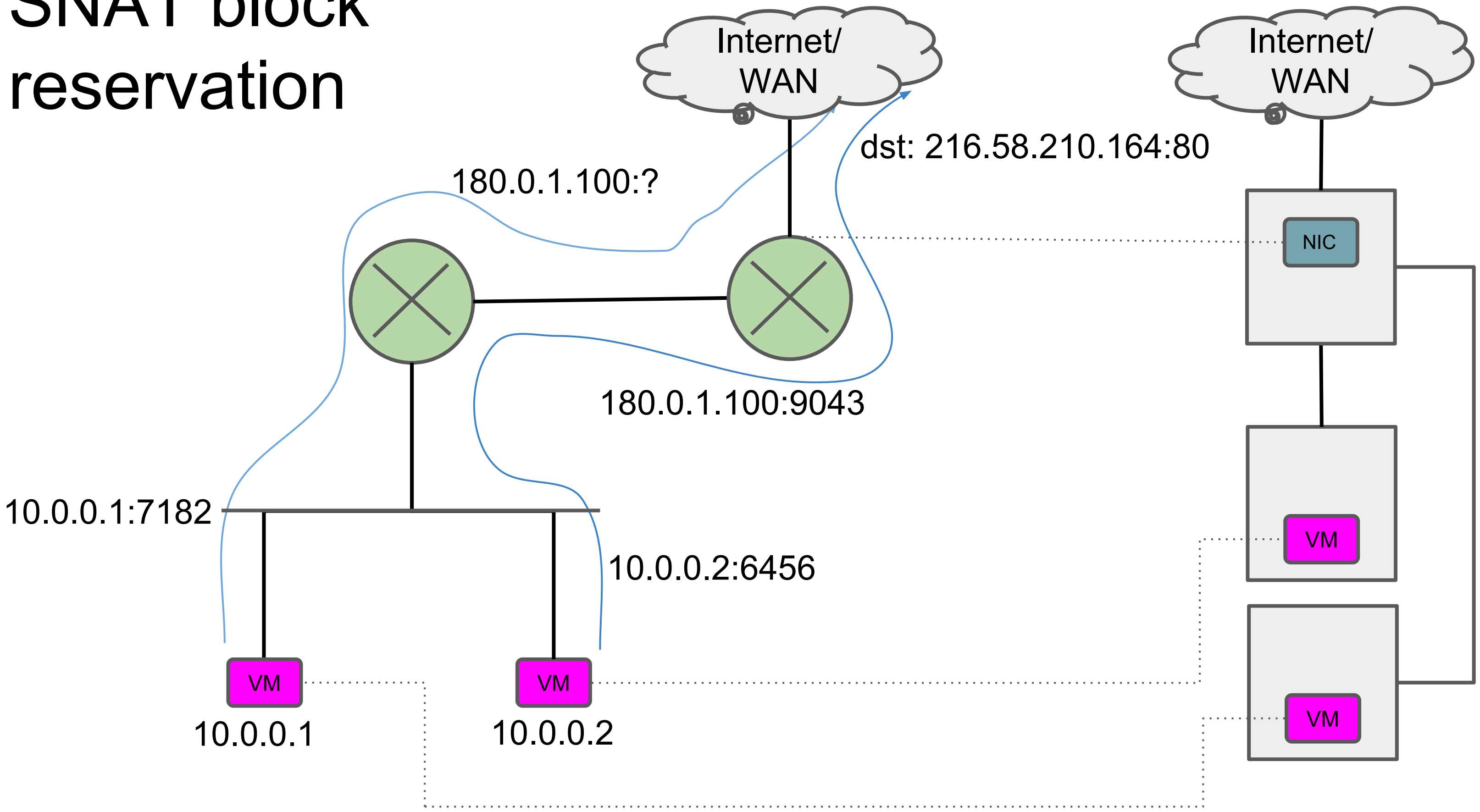
180.0.1.100..180.0.1.100
5000..65535



SNAT block reservation



SNAT block reservation



midonet

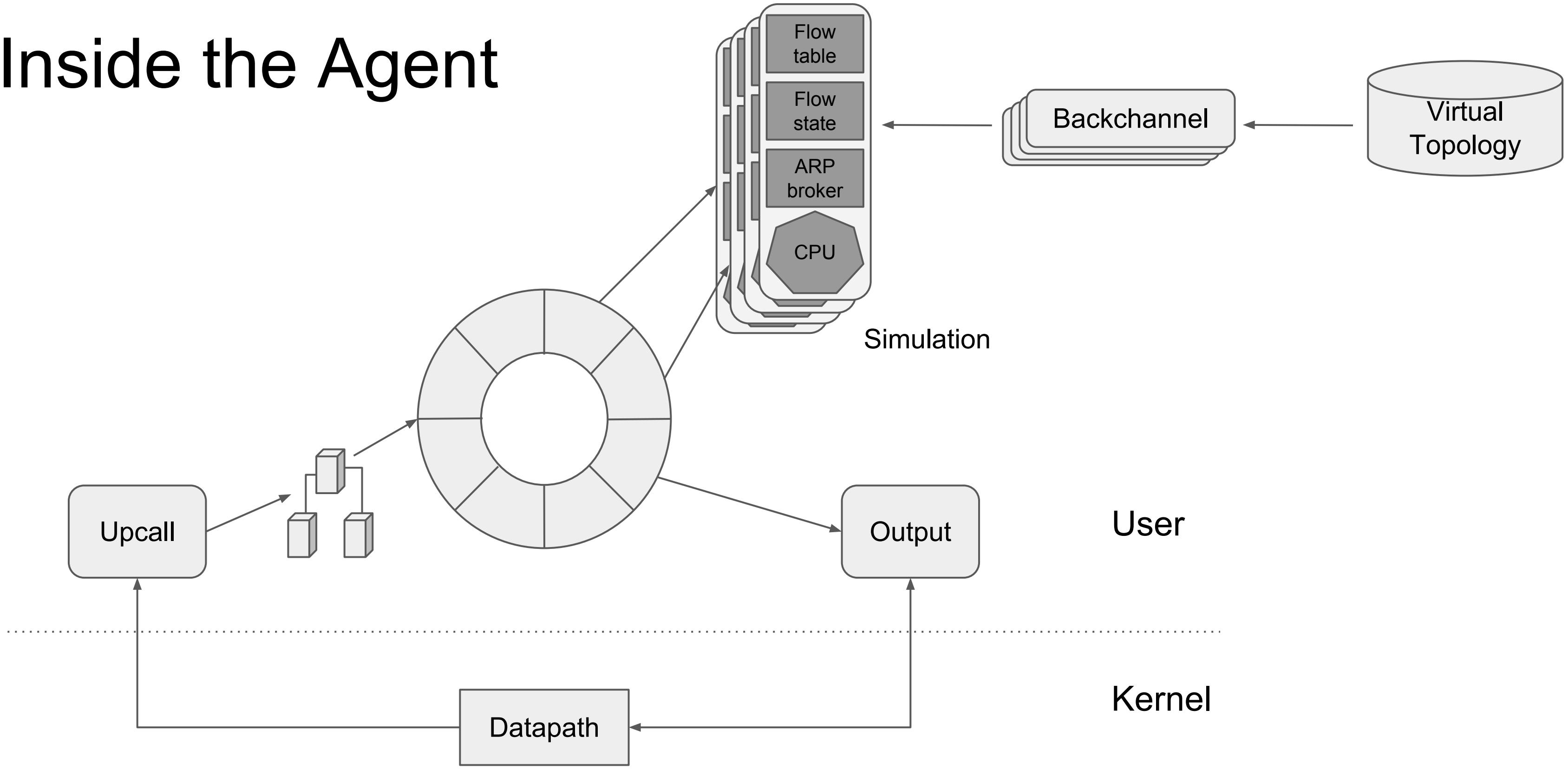
SNAT block reservation

- Performed through ZooKeeper
- /nat/{device_id}/{ip}/{block_idx}
- 64 ports per block, 1024 total blocks
- LRU based allocation
- Blocks are referenced by flow state

Thank you!
Q&A

Low-level

Inside the Agent



midonet

Performance

- Sharding
  - Share nothing model
  - Each simulation thread is responsible for a subset of the installed flows
  - Each simulation thread is responsible for a subset of the flow state
  - Each thread ARPs individually
  - Communication by message passing through “backchannels”
- Run to completion model
  - When a piece of the virtual topology is needed, simulations are parked
- Lock-free algorithms where sharding is not possible