Challenges in Distributed SDN

Duarte Nunes
duarte@midokura.com
@duarte_nunes

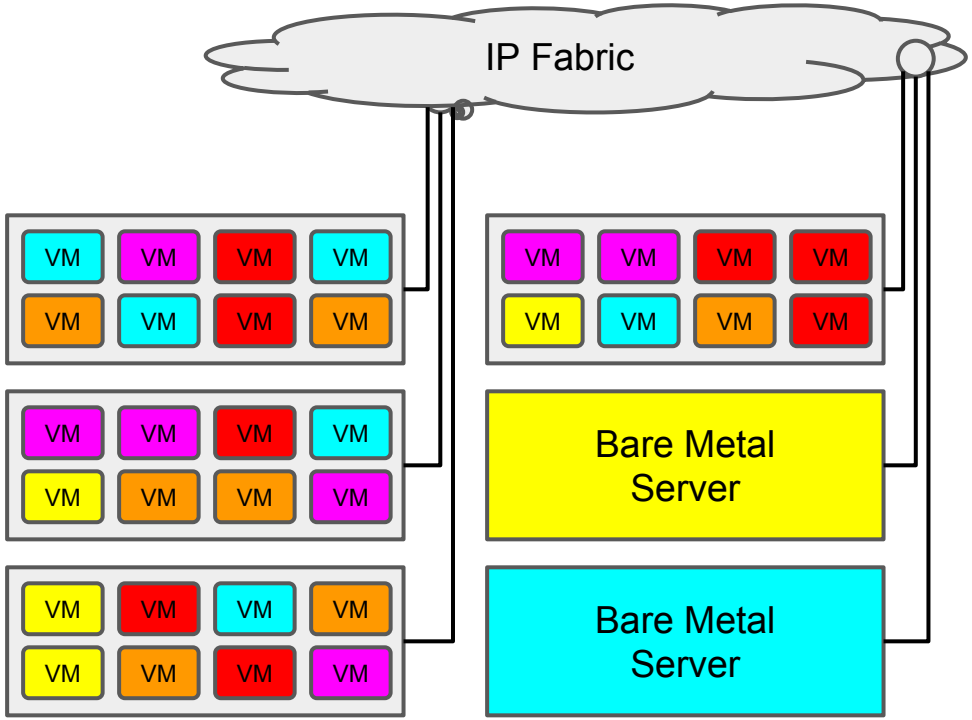
Guillermo Ontañón
guillermo@midokura.com
@gontanon

midonet

Agenda

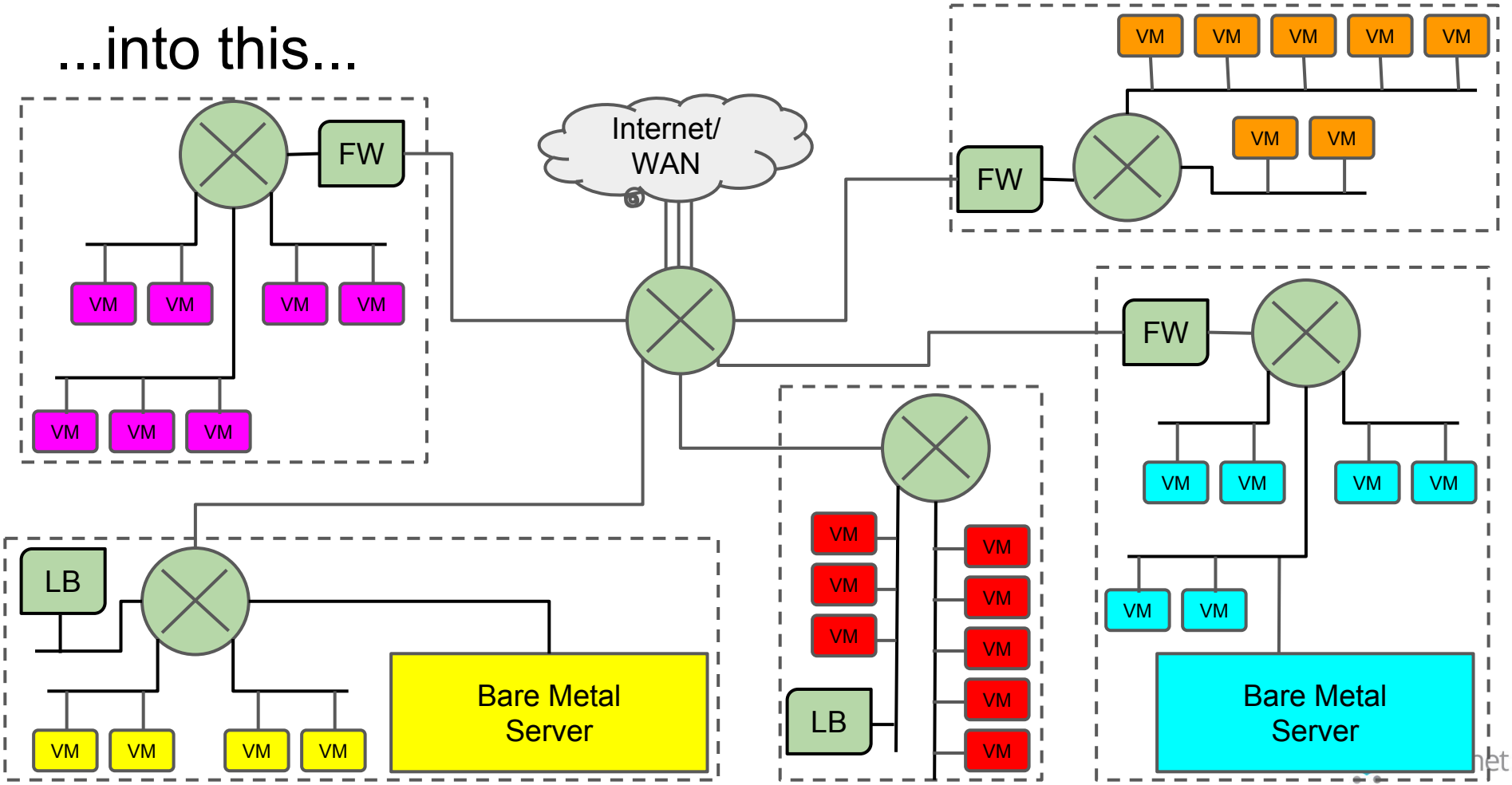
- Network virtualization overlays
  - MidoNet
- Distributed devices
  - Managing the virtual topology
  - Replicating device state
- Distributed flow state
  - Use cases
  - State replication
  - SNAT block reservation
- A low-level view
- Scaling

NVOs transform this...



midonet

...into this...


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

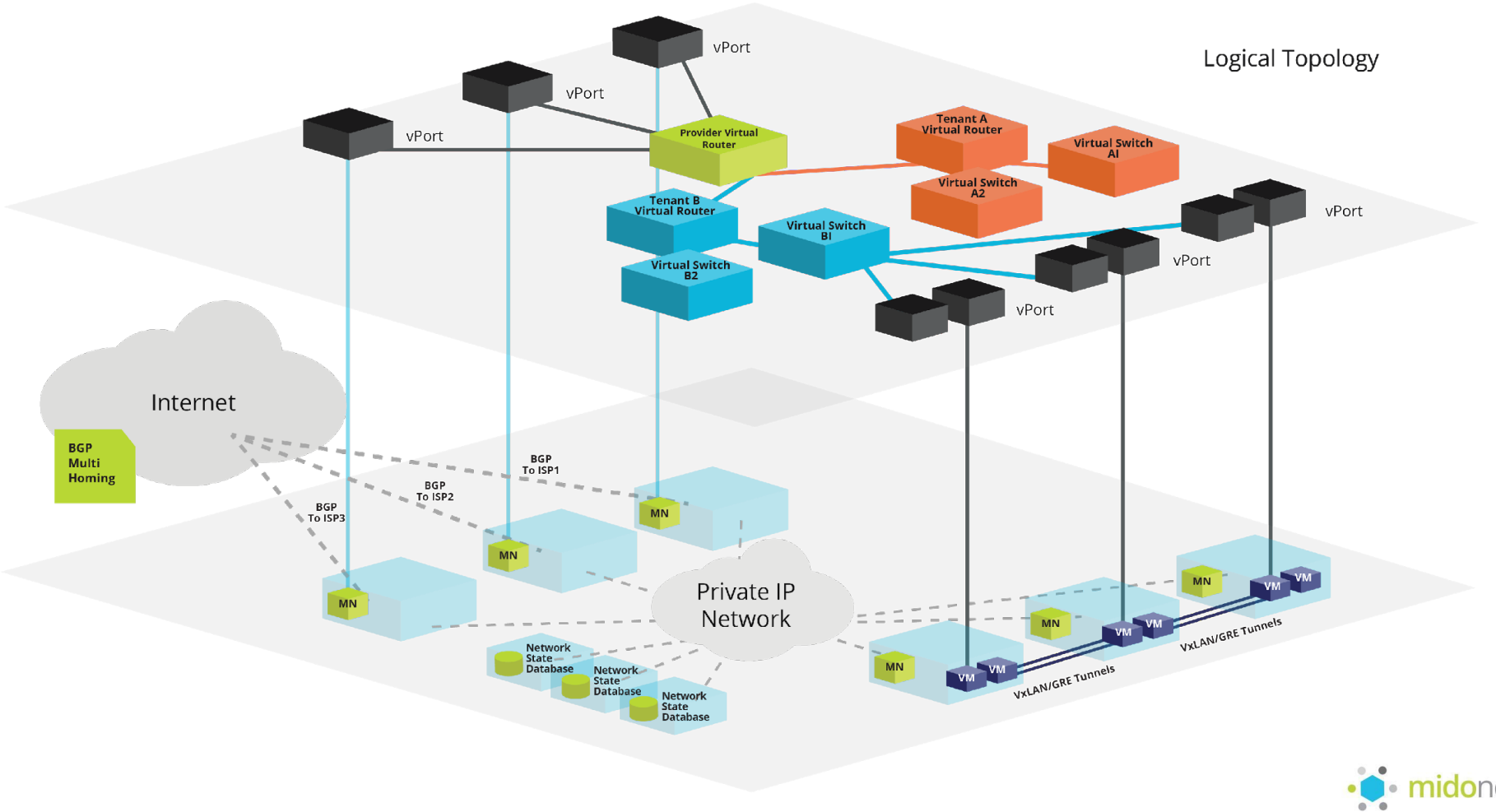
Network virtualization overlays

- Decouples logical and physical configuration and address space
- Isolation between tenants
  - Tenant traffic is encapsulated at its local edge and carried by a tunnel over an IP network to another edge where the packet is decapsulated and delivered to the target
  - Tenant address space can overlap
- Allows virtual machine mobility
- Optimal Forwarding
  - One single physical hop

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
- No SPOF
- No middle boxes
- Horizontally scalable L2 and L3 Gateways

midonet


Logical Topology
vPort
vPort
vPort
Provider Virtual Router
Tenant A Virtual Router
Virtual Switch A1
Virtual Switch A2
Tenant B Virtual Router
Virtual Switch B1
Virtual Switch B2
vPort
vPort
vPort
Internet
BGP Multi Homing
BGP To ISP1
BGP To ISP2
BGP To ISP3
MN
MN
MN
Private IP Network
Network State Database
Network State Database
Network State Database
MN
MN
MN
VM
VM
VM
VM
VM
VM
VxLAN/GRE Tunnels
VxLAN/GRE Tunnels
midonet

MidoNet Hosts

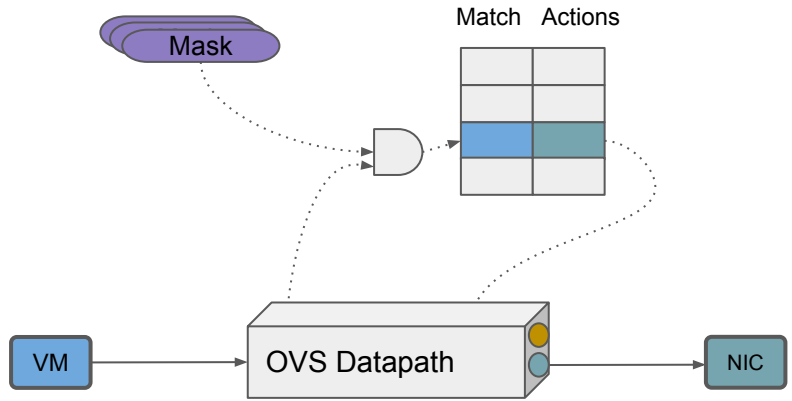
- Gateway
  - On-ramp and off-ramp into the cloud
- Compute
  - Where virtual machines or containers are hosted
- NSDB
  - A distributed database containing the virtual topology

MidoNet Agent

- Flow programmable switch at the bottom
  - Open vSwitch kernel module
- Flow simulator at the top
- Agents perform just-in-time flow computation
  - Each flow is the result of simulation a packet's trip through the virtual network topology

Datapath - Flow programmable switch

- Port bindings
- Flow match
  - Input port
  - Tunnel header
  - Ethernet header
  - IP header
  - Transport header
  - …
- Flow actions
  - Output
  - Encapsulate
  - Transform
  - ...

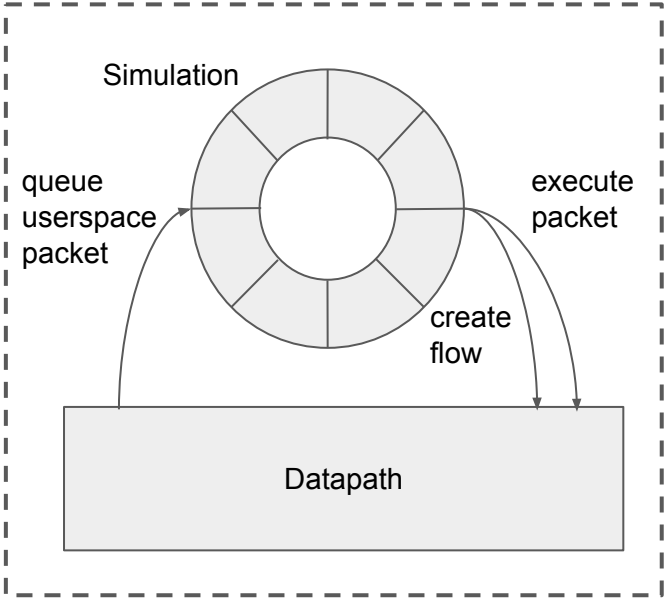


Eth src: 9e:d9:b3:74:0f:c7
Eth dst: 16:ed:98:8d:8b:1a
IP src: 10.0.3.1
IP dst: 10.0.3.2

Tunnel src: 119.15.120.139
Tunnel dst: 119.15.168.162
Tunnel key: 12
Eth src: f6:73:84:a4:26:54
Eth dst: 26:eb:ac:ea:1e:d4
IP src: 10.0.3.1
IP dst: 10.0.3.2

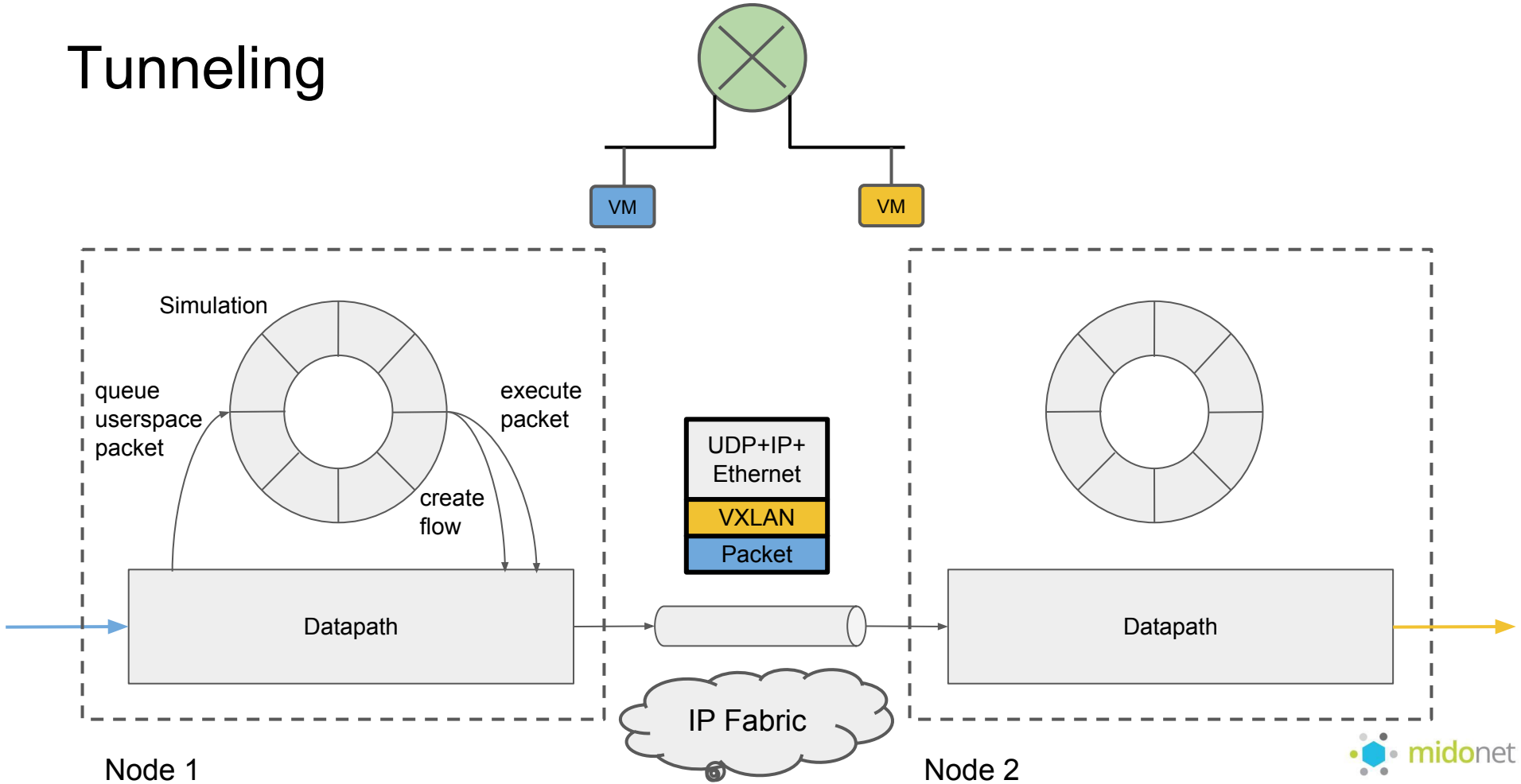
midonet

Just-in-time flow computation



midonet

Tunneling



midonet

Virtual Devices

Managing topology changes

- ZooKeeper serves the virtual network topology
  - Reliable subscription to topology changes
- Agents fetch and “watch” virtual devices as they need them to process packets they see
- Every traversed device applies a tag on a computed flow
- Agents react to topology changes by invalidating the corresponding flows by tag
  - For example, the ID of a device that was removed

Replicating device state

- Remember: we process each packet locally at the physical host where it ingresses the virtual network:

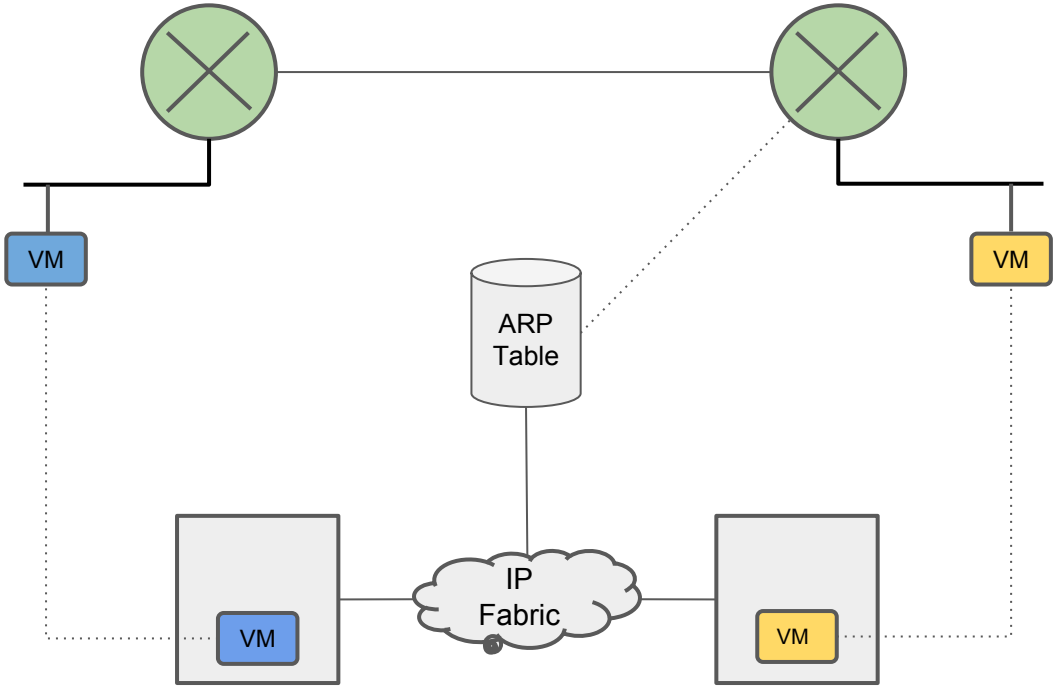
  Different packets traverse the same virtual device at *different* hosts

- This affects virtual network devices:
  - 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

Replicating device state

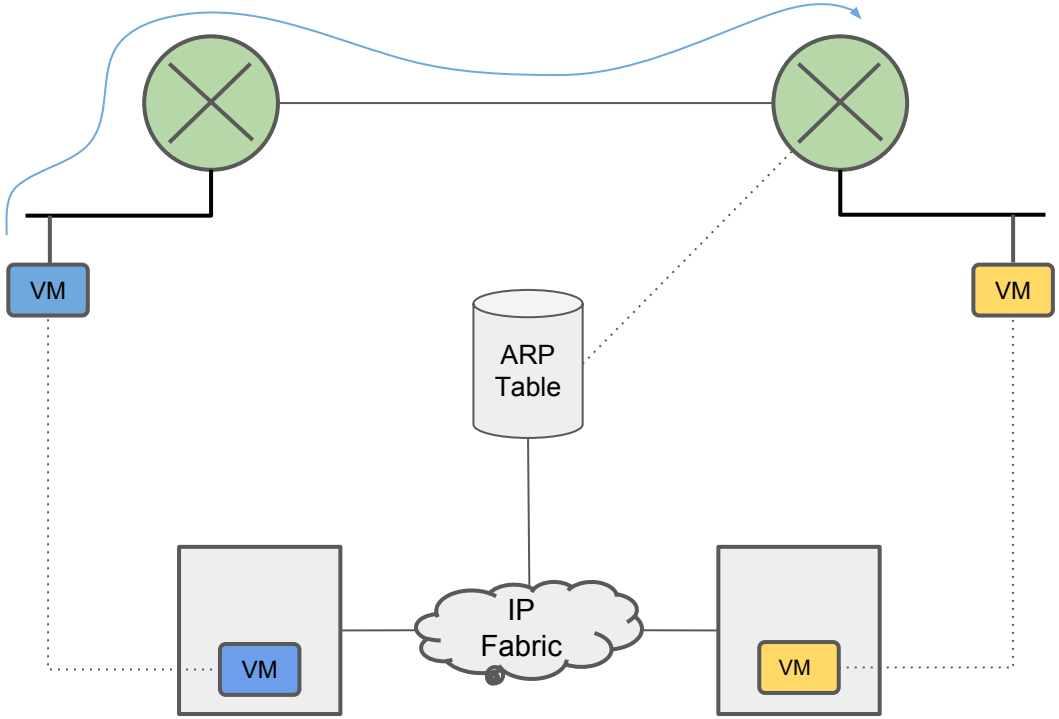
- Store device state tables (ARP, MAC-learning, routes) in ZooKeeper
- Interested agents subscribe to tables to get updates
- The owner of an entry manages its lifecycle:
  - Refresh ARP entries
  - Reference count which flows use a MAC-learning entry
- Use ZK Ephemeral nodes so entries go away if a host dies

ARP Table



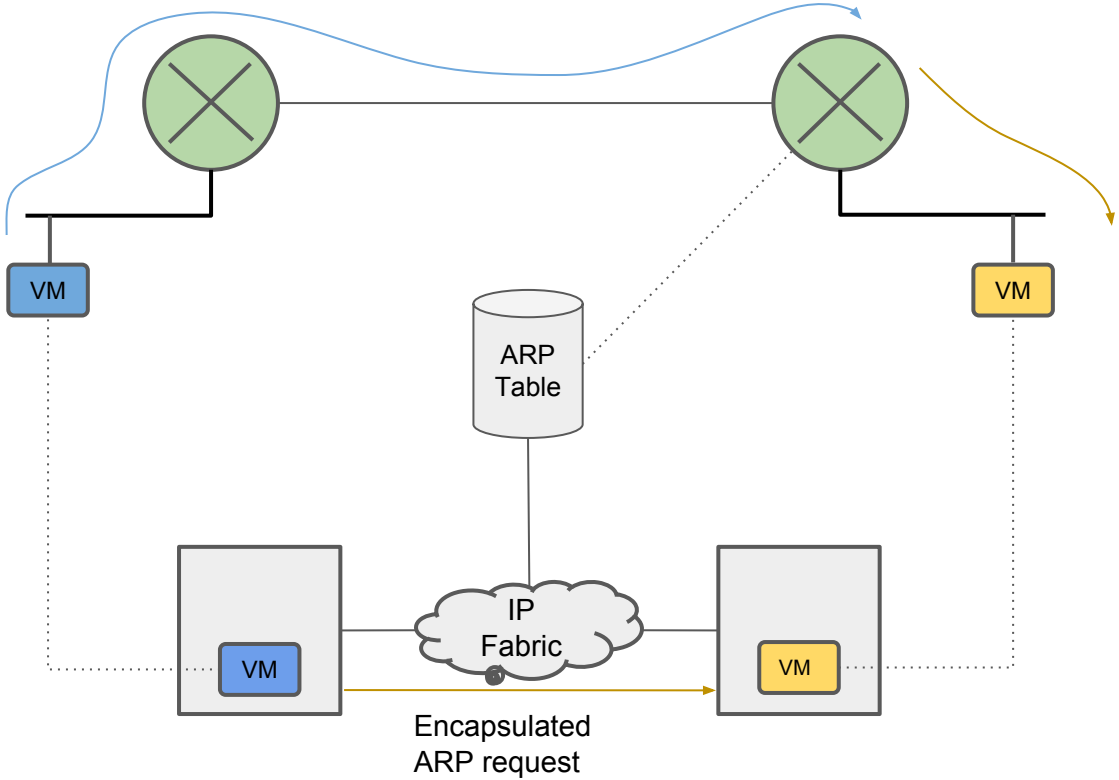
midonet

ARP Table



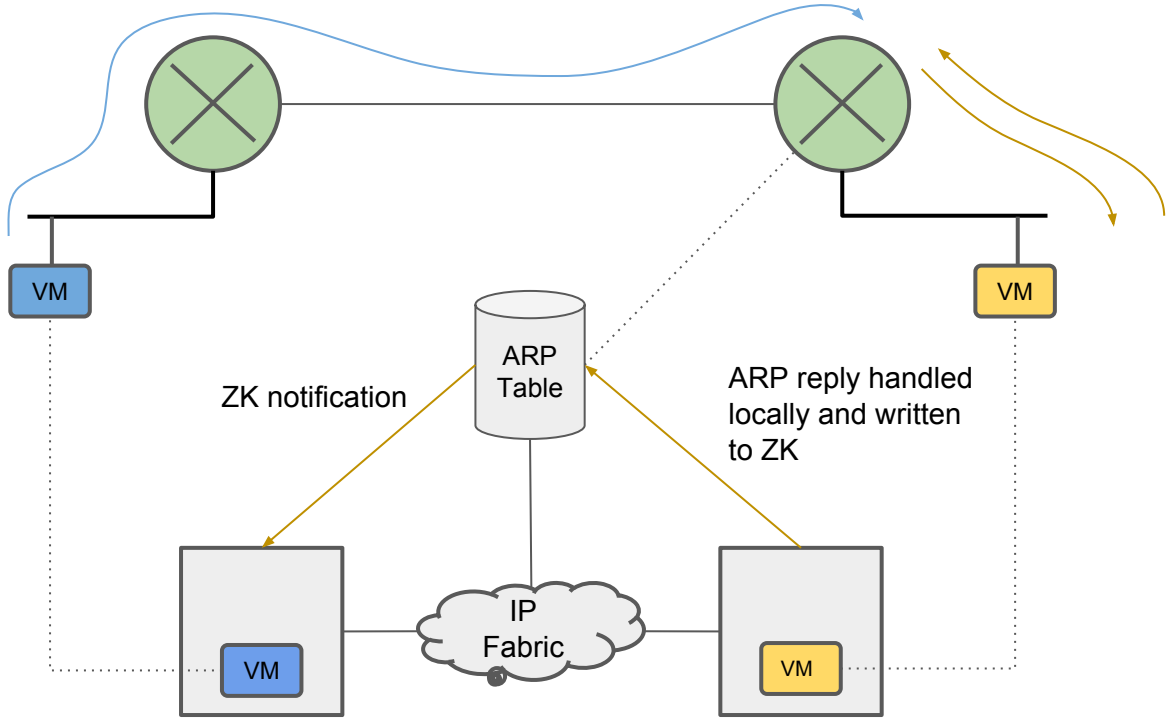
midonet

ARP Table



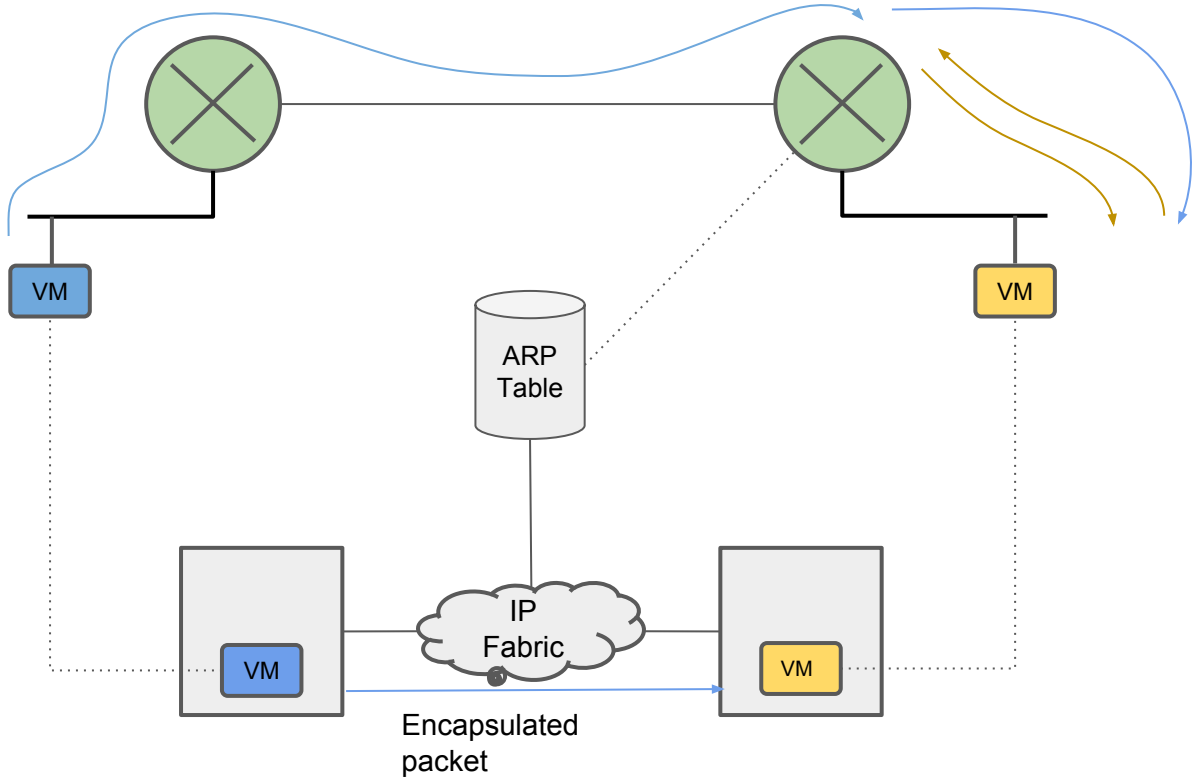
midonet

ARP Table



midonet

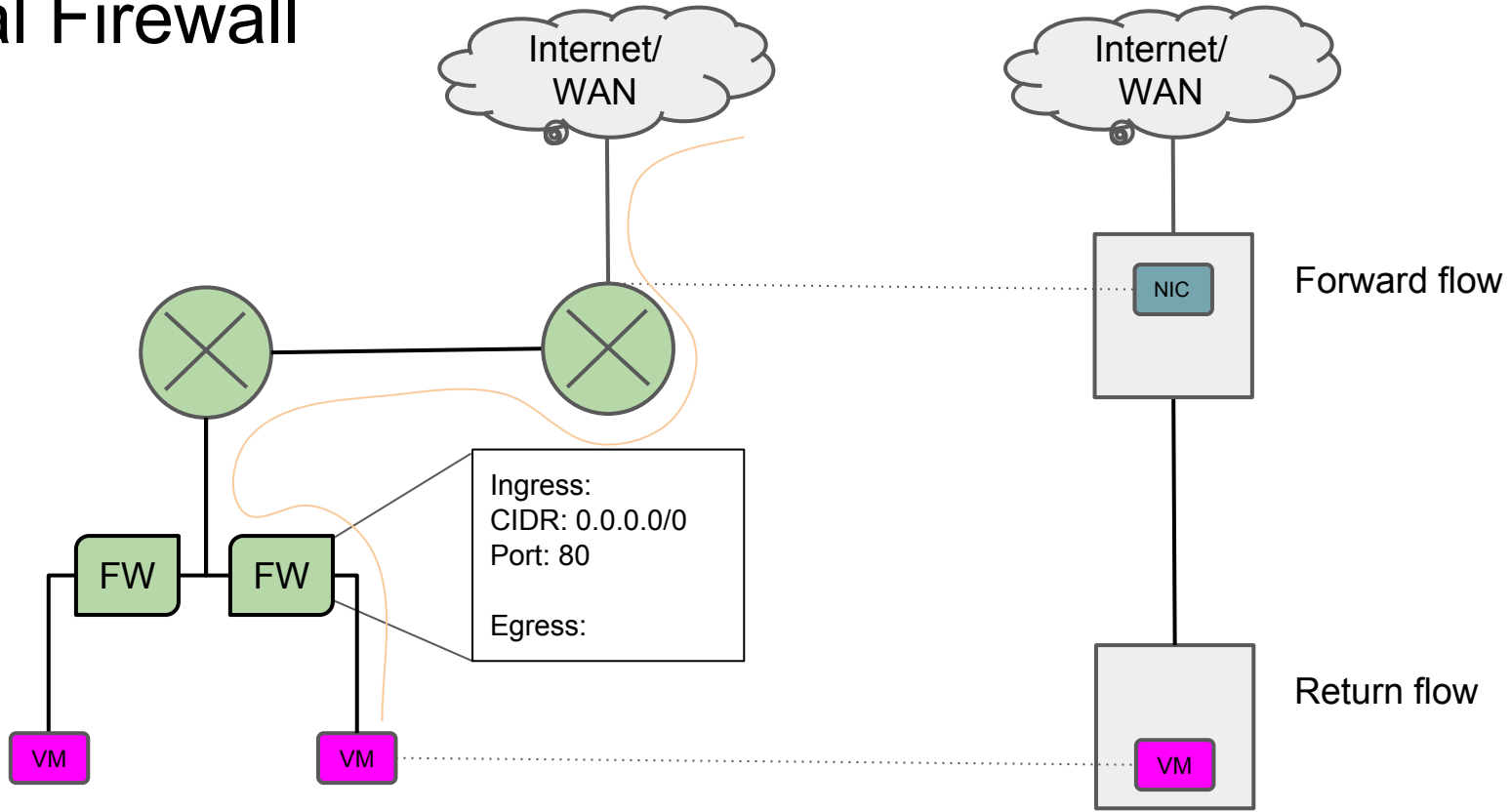
ARP Table



midonet

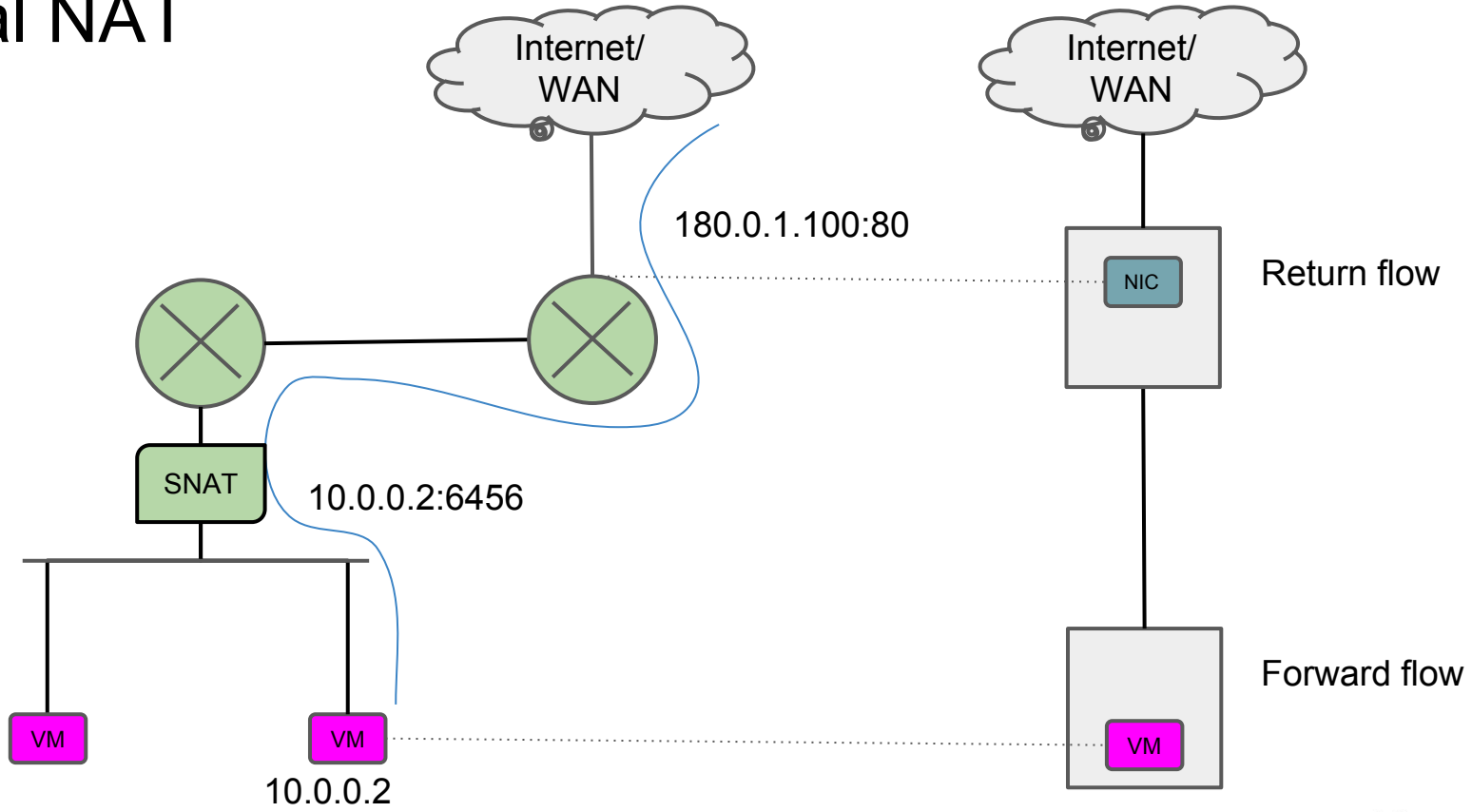
Distributed Flow State

Virtual Firewall



midonet

Virtual NAT

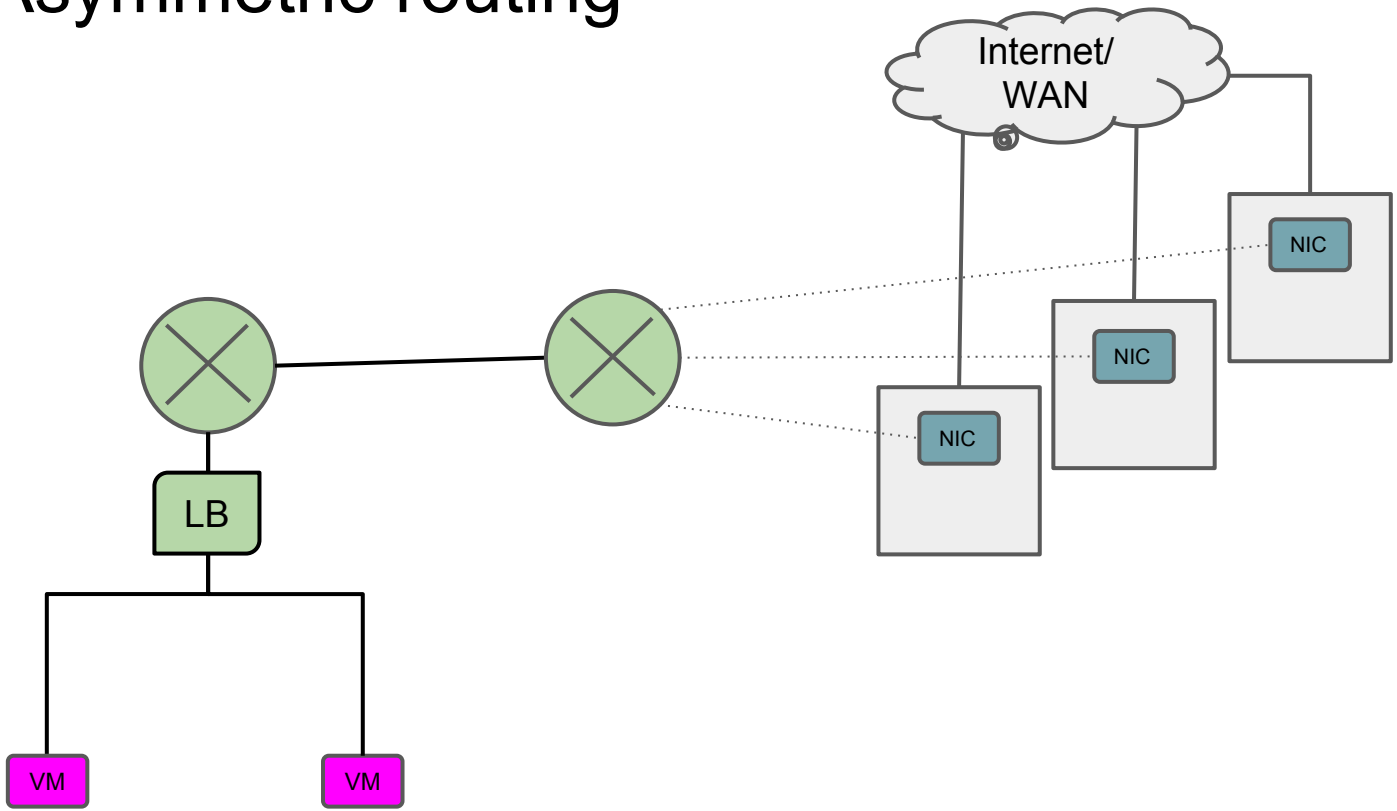


midonet

Flow state

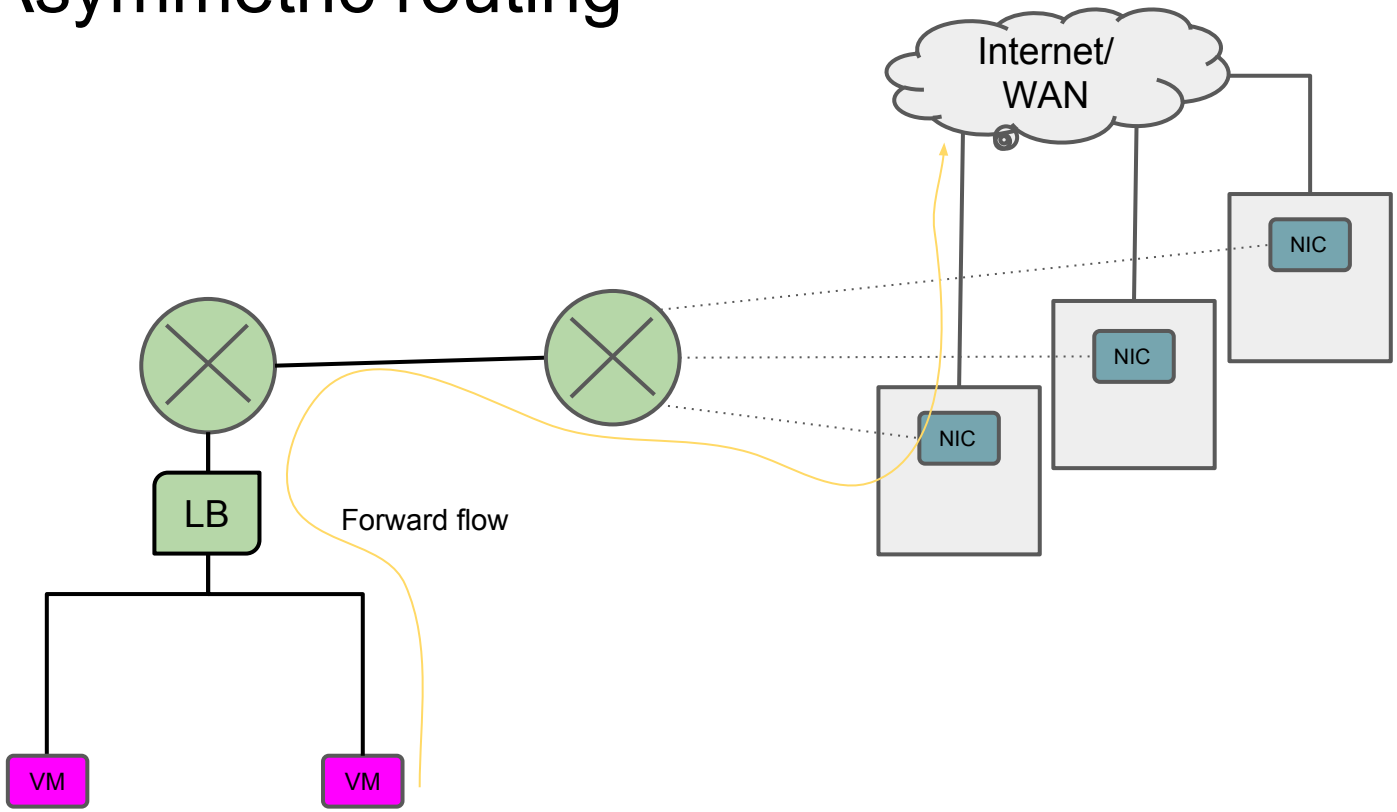
- Connection tracking
  - Key: 5 tuple + ingress device UUID
  - Value: NA
  - Forward state not needed
  - One flow state entry per flow
- NAT
  - Key: 5 tuple + device UUID under which NAT was performed
  - Value: (IP, port) binding
  - Possibly multiple flow state entries per flow

Asymmetric routing



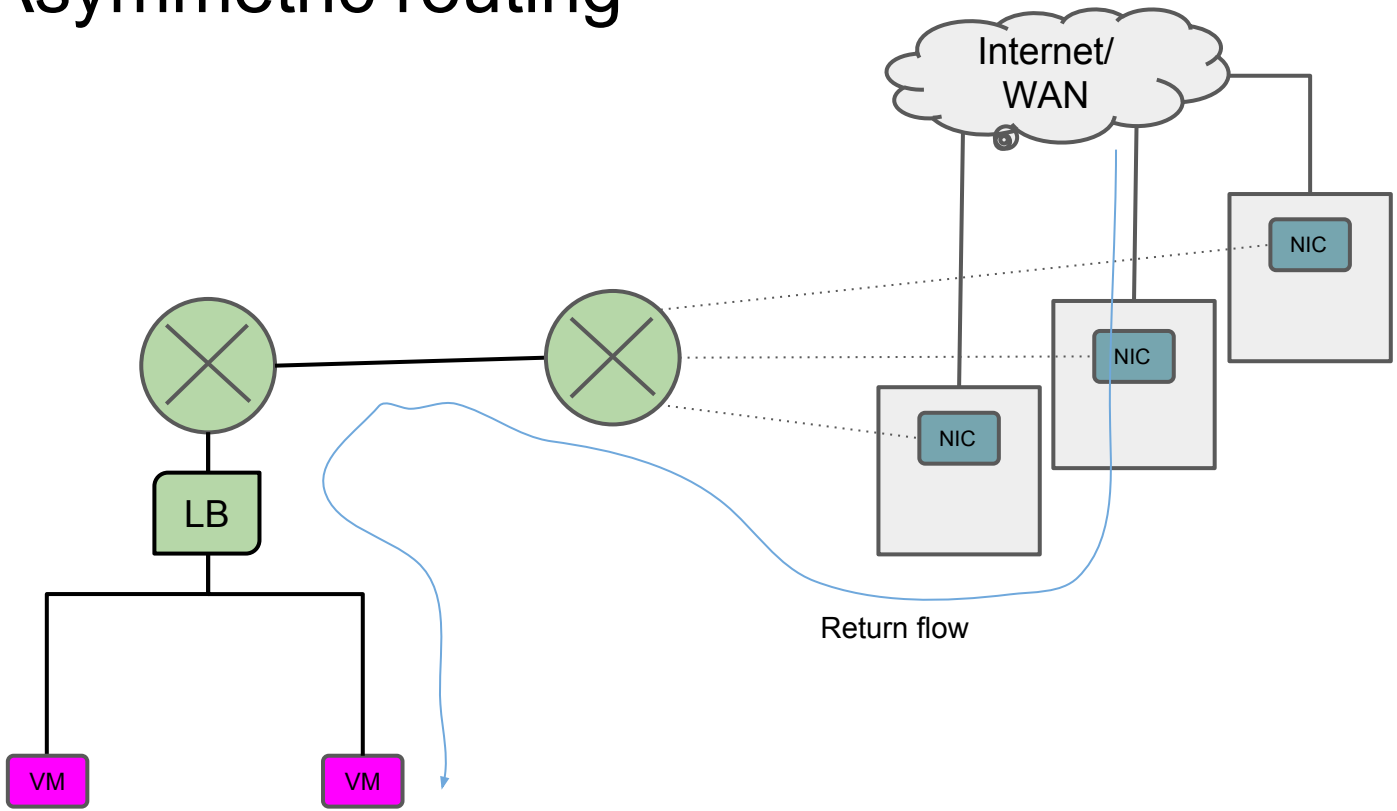
midonet

Asymmetric routing



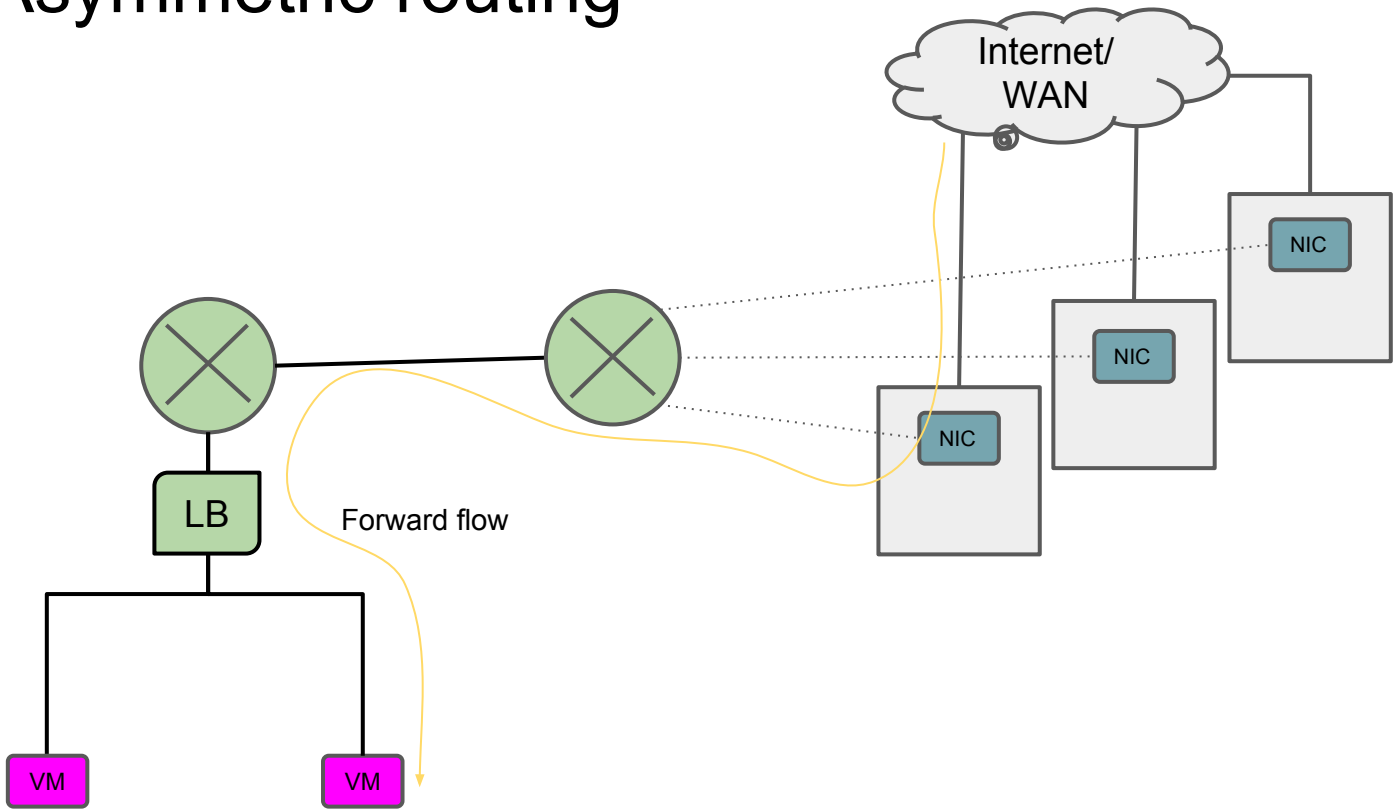
midonet

Asymmetric routing



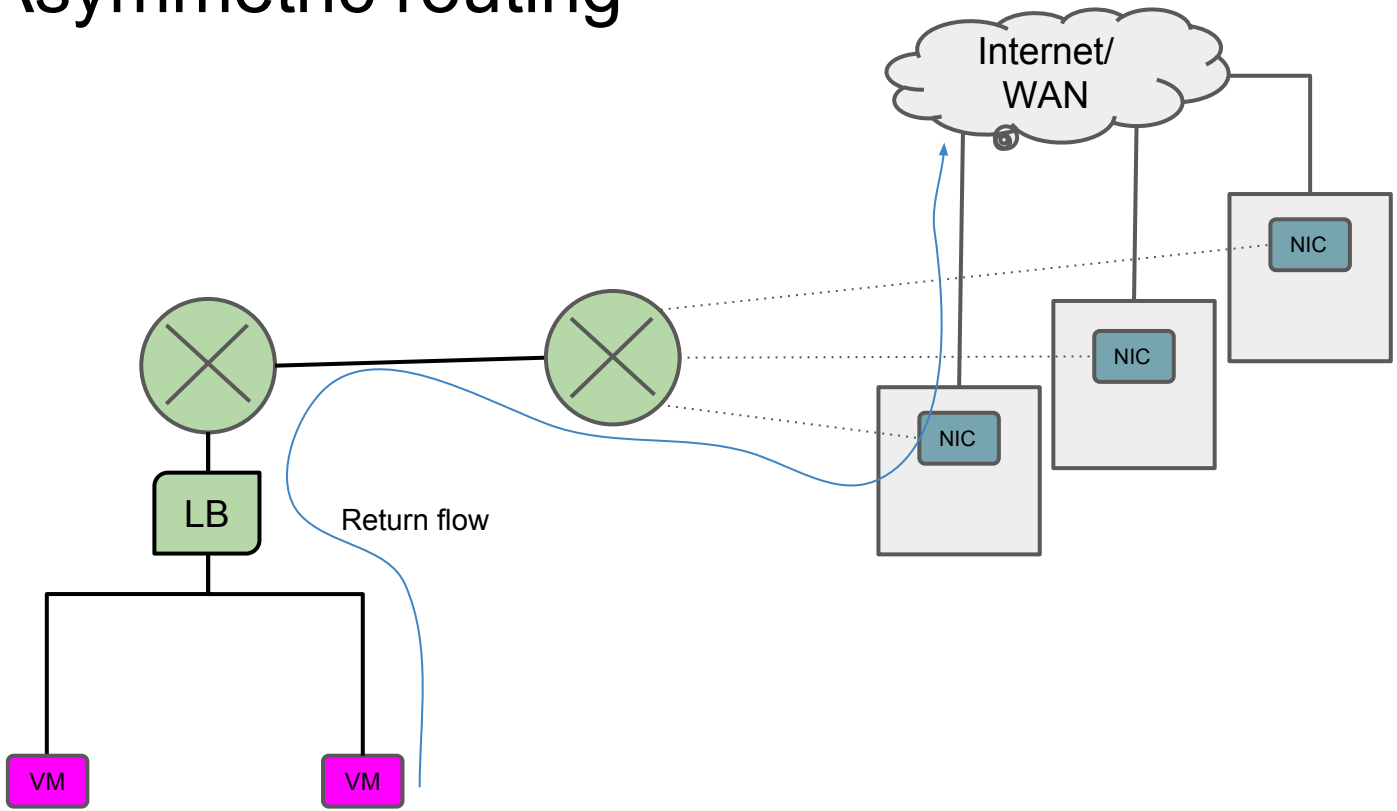
midonet

Asymmetric routing



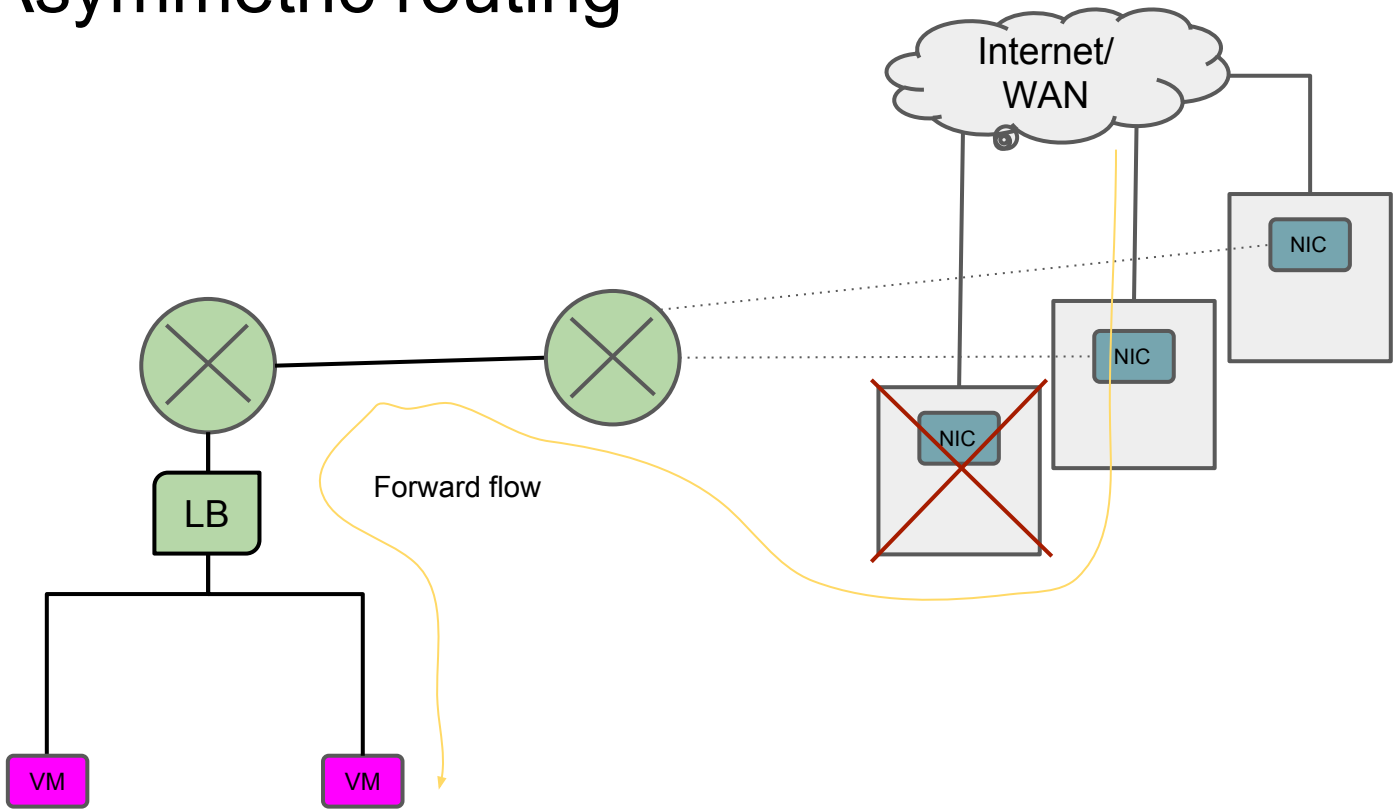
midonet

Asymmetric routing



midonet

Asymmetric routing

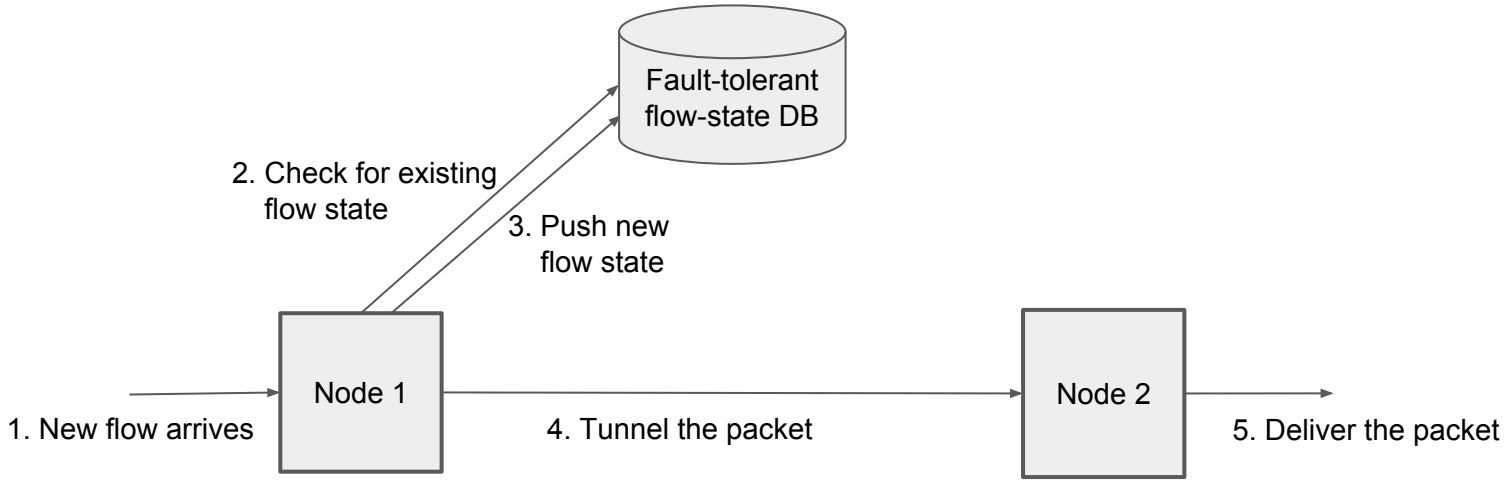


midonet

Interested set

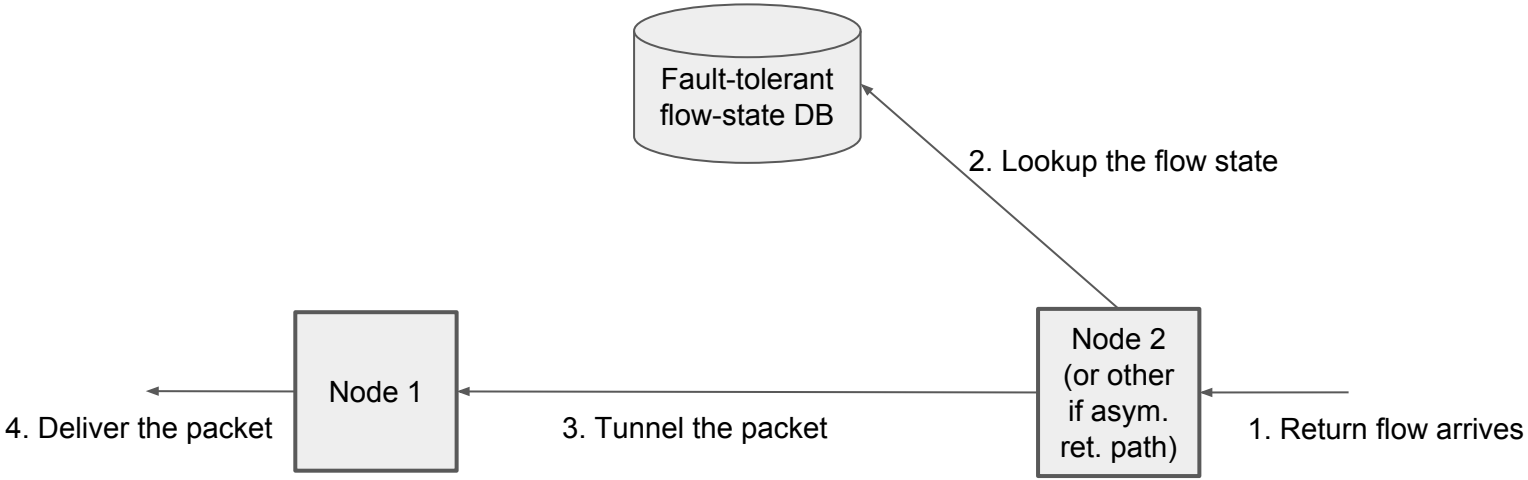
- Egress host
  - Which will (probably) simulate the return flow
- Hinting via port groups (ingress and egress)
  - Depending on upstream routes, any of the static or dynamic uplinks of the Provider Router may receive North-to-South flows (an L3 Gateway)
  - Similarly, a tenant may have a redundant L3 VPN from their office network to their Tenant Router, which may ingress MidoNet at more than one node (on-ramp)
  - Also a VLAN L2 Gateway, which allows a 802.1Q virtual bridge with 2 physical links; traffic from the physical workloads ingresses either of the bridge's uplink ports depending on STP
- VXLAN L2 Gateway
  - To support bare metal servers behind a VTEP sending traffic to MidoNet, which ingresses a Flooding Proxy

Sharing state - Distributed database



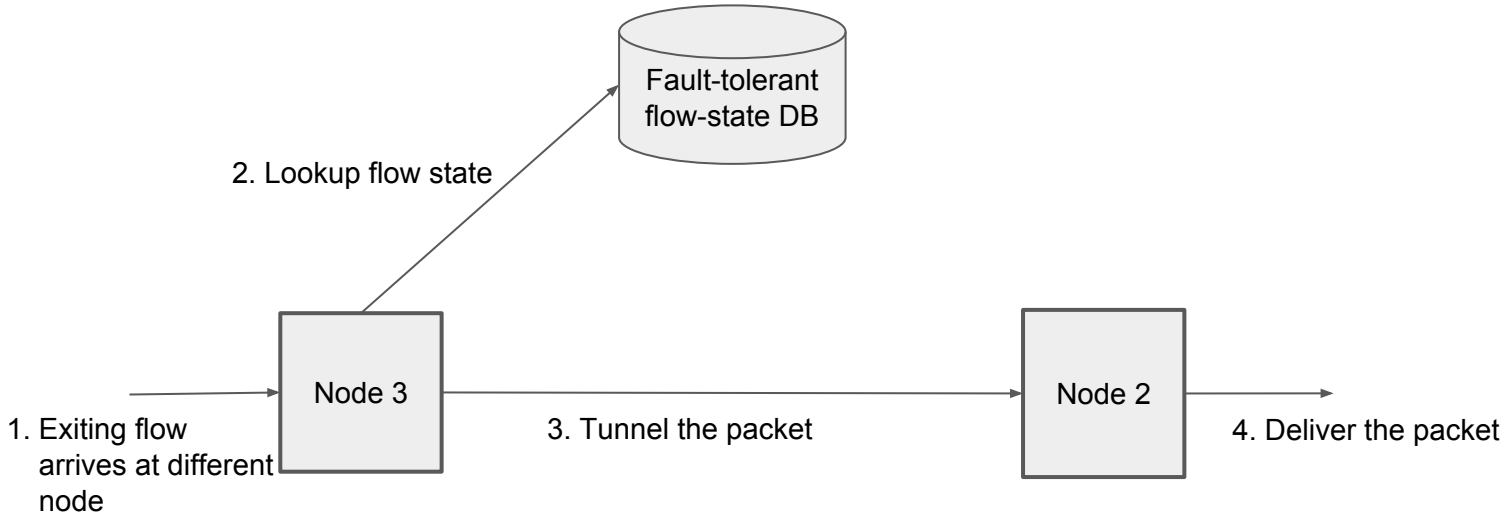
midonet

Sharing state - Distributed database



midonet

Sharing state - Distributed database

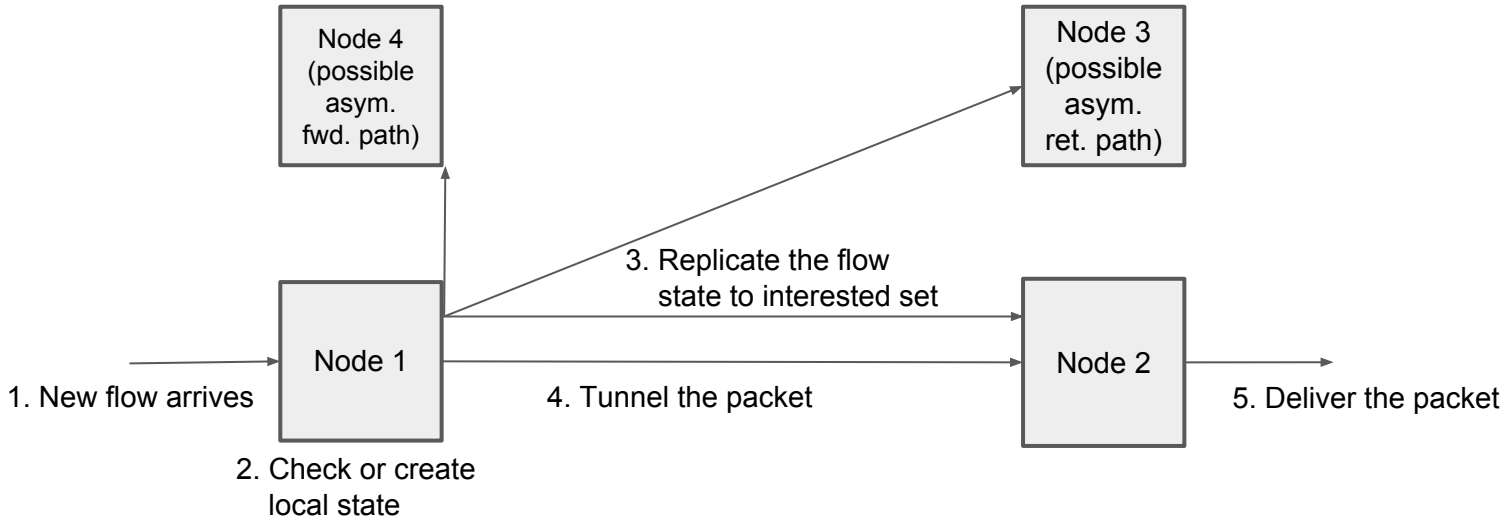


midonet

Sharing state - Distributed database

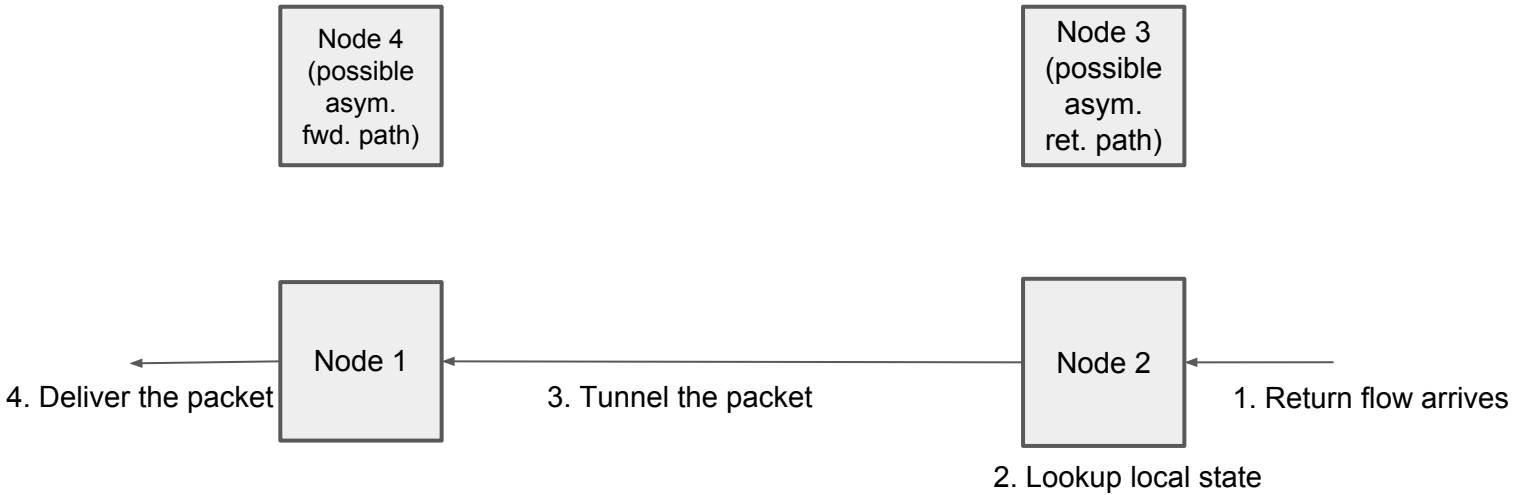
- How many replicas receive the flow state?
- How many replicas must acknowledge before finishing the simulation?
- How many replicas to read the flow state from in order to simulate a return flow or re-simulate a forward flow?
- Adds significant latency to simulation

Sharing state - Peer-to-peer handoff

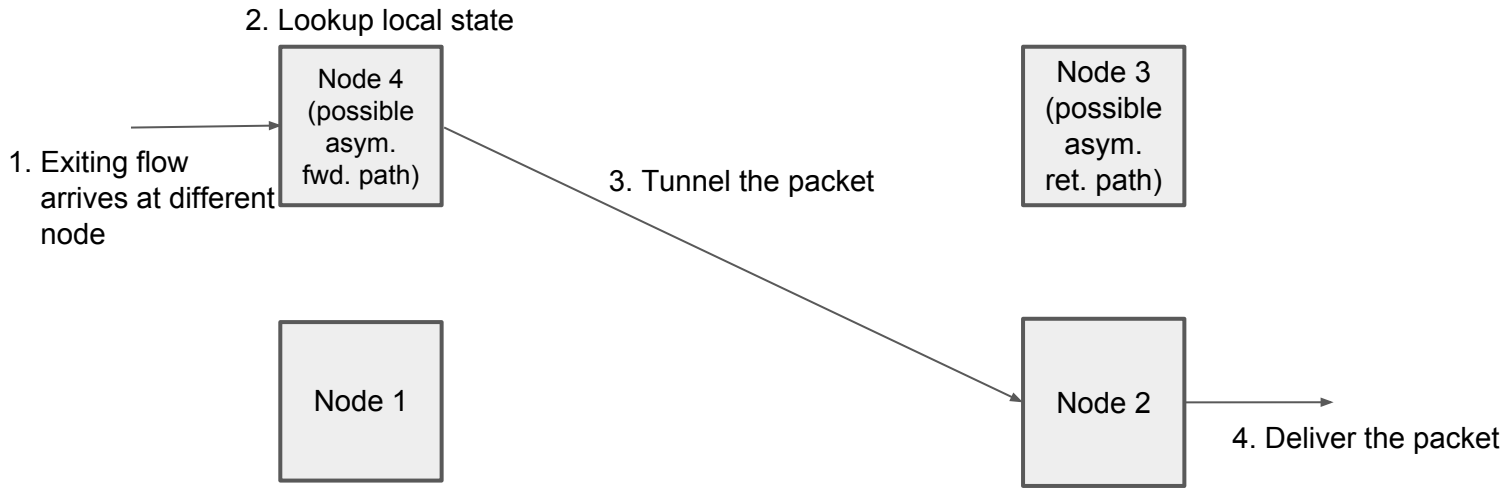


midonet

Sharing state - Peer-to-peer handoff



Sharing state - Peer-to-peer handoff



midonet

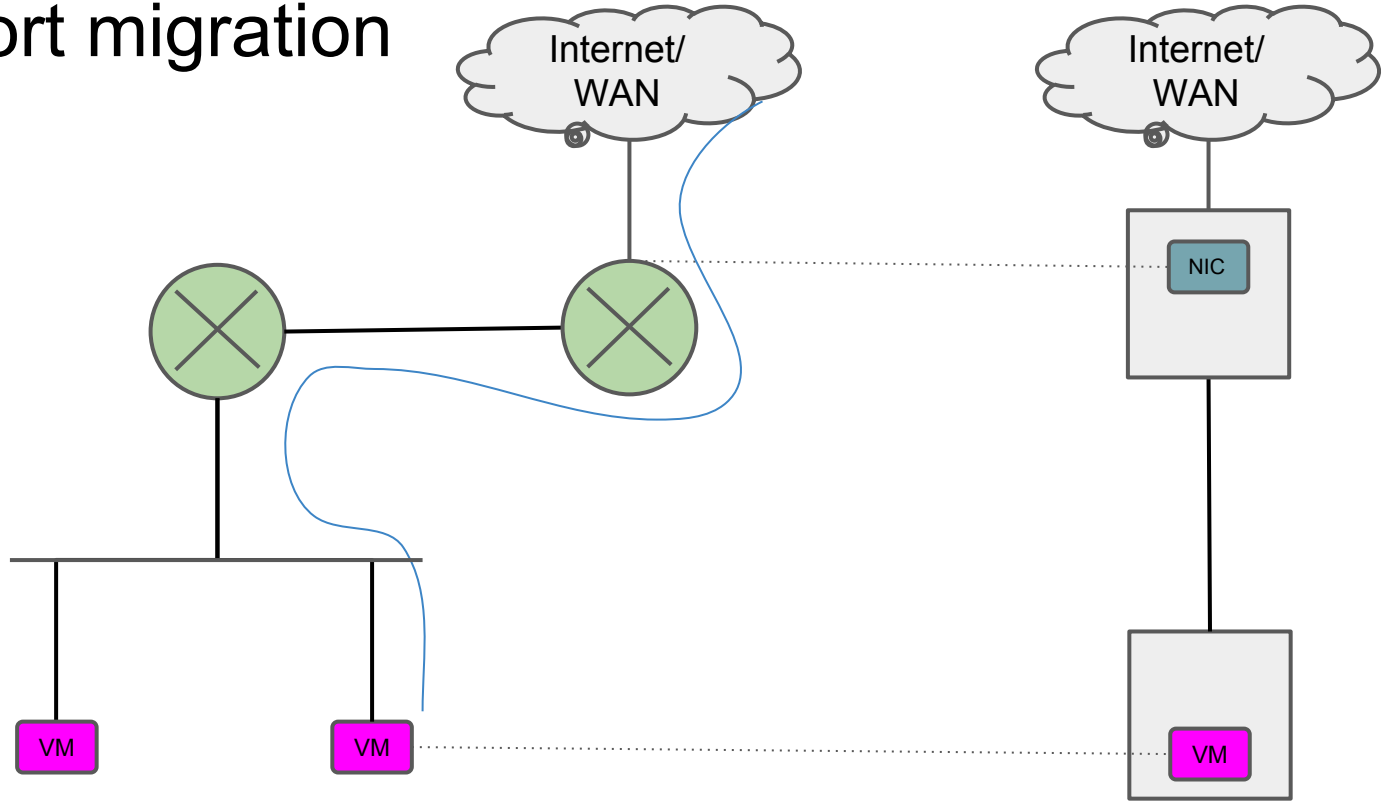
Sharing state - Peer-to-peer handoff

- 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?

Lifecyle management

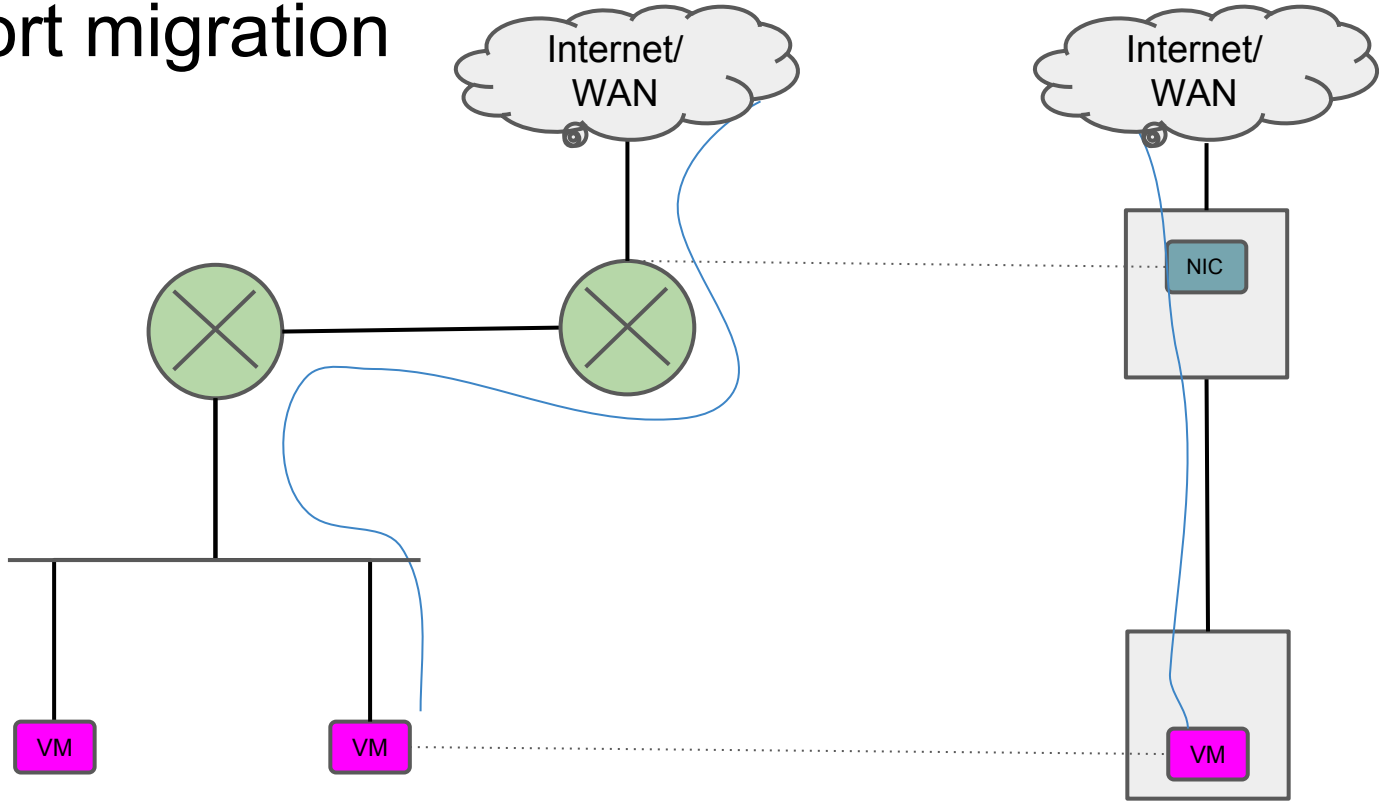
- Local reference counting on ingress hosts
- Flows have a hard timeout, triggers expiration of associated flow state
  - Flow state is expired independently at each host
  - Tries to minimize coordination between hosts
- Flow state refreshed by new, simulated packets
  - At any ingress host
  - Means that connections die in the absence of forward packets

Port migration



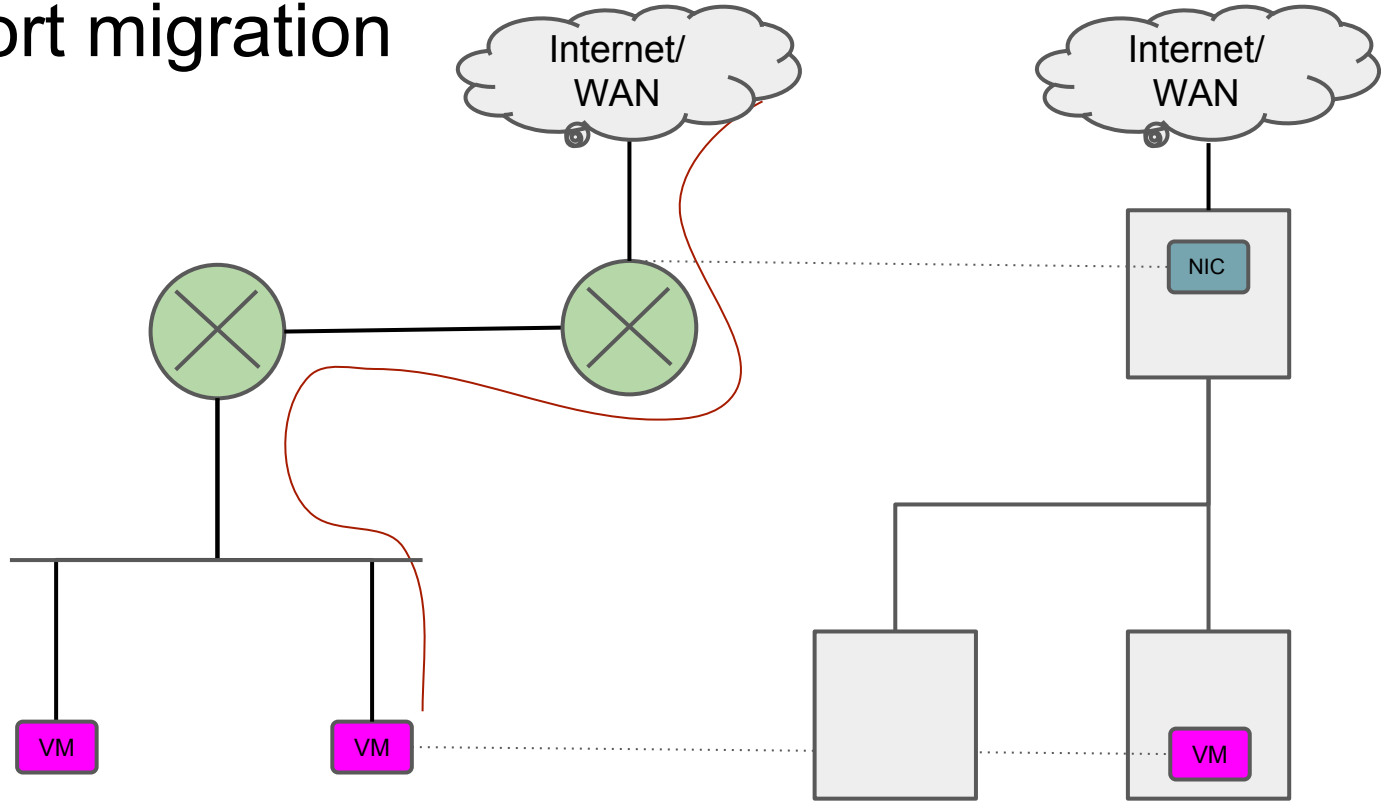
midonet

Port migration



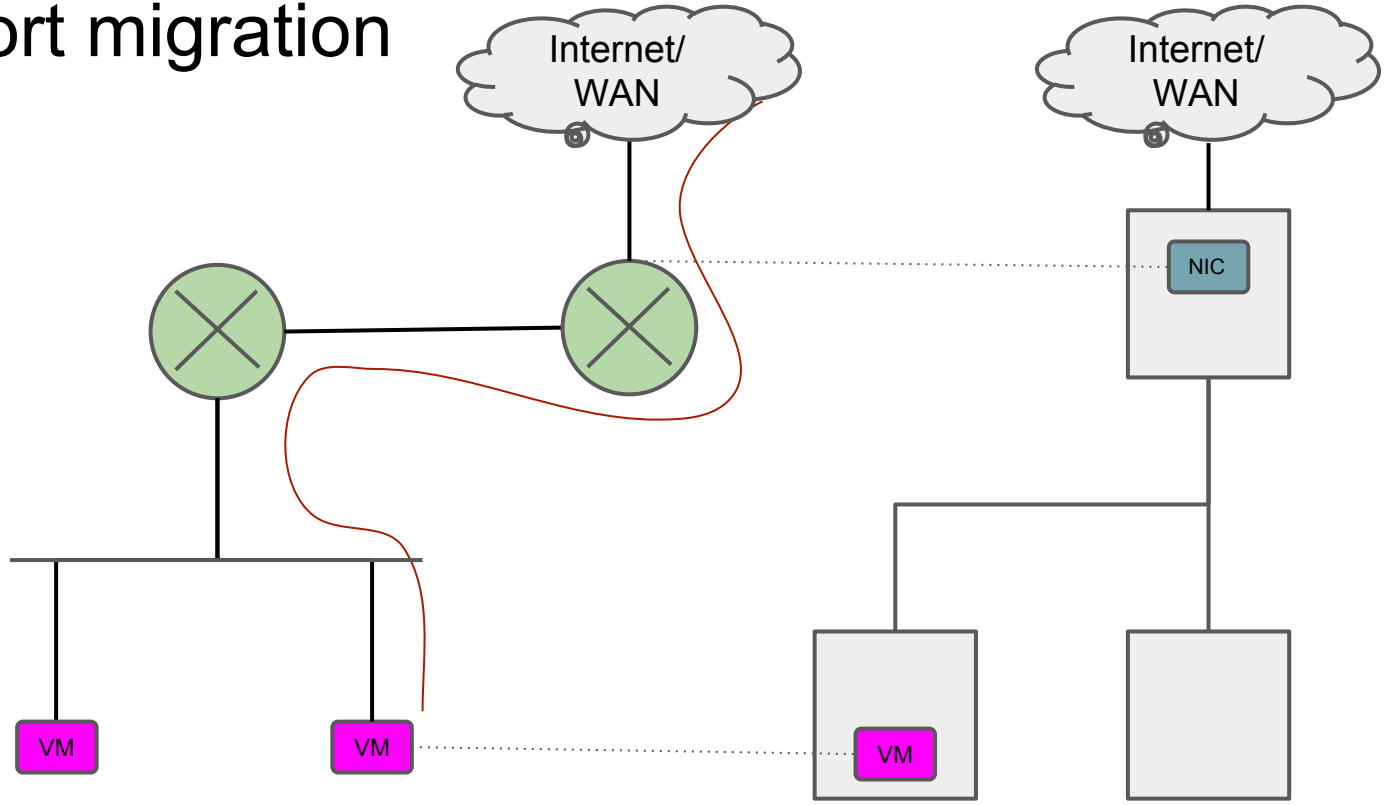
midonet

Port migration



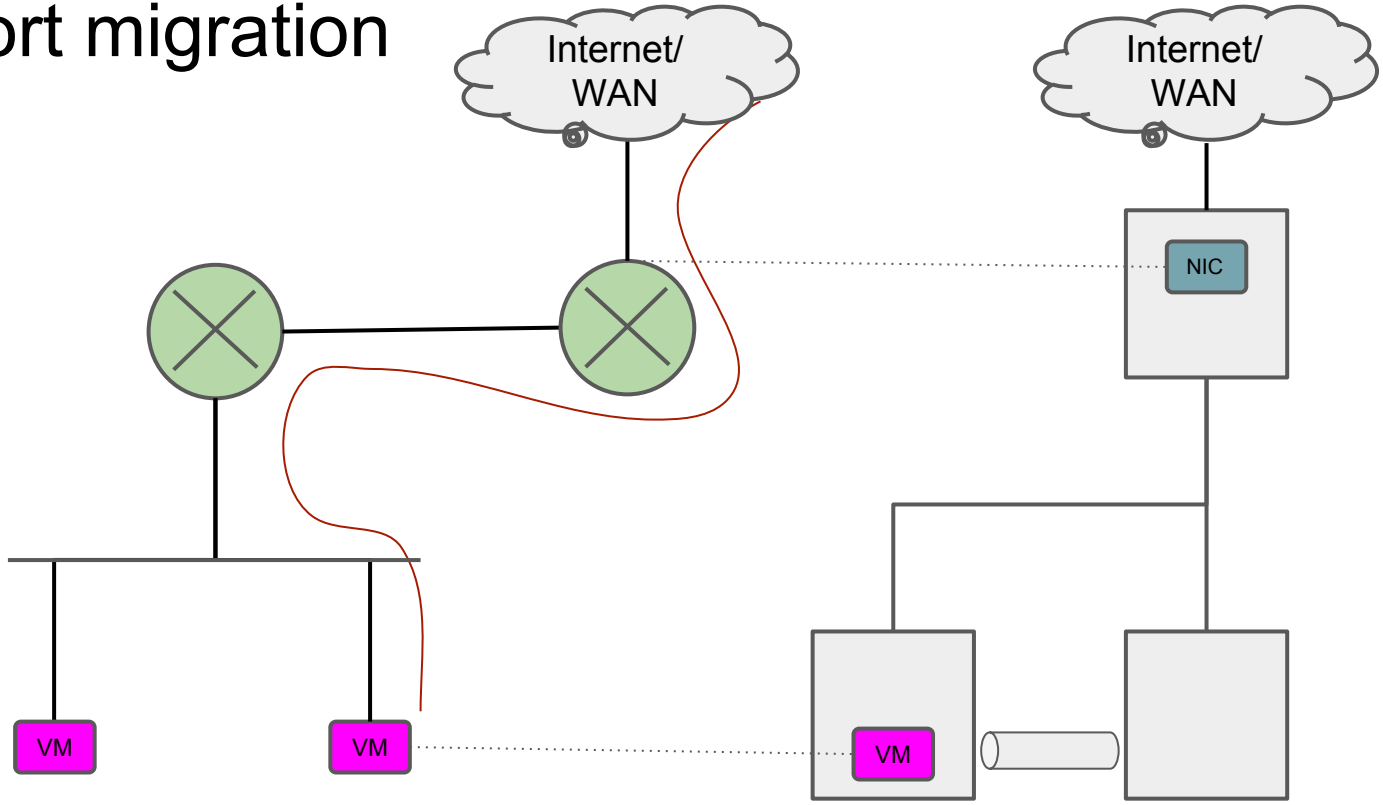
midonet

Port migration



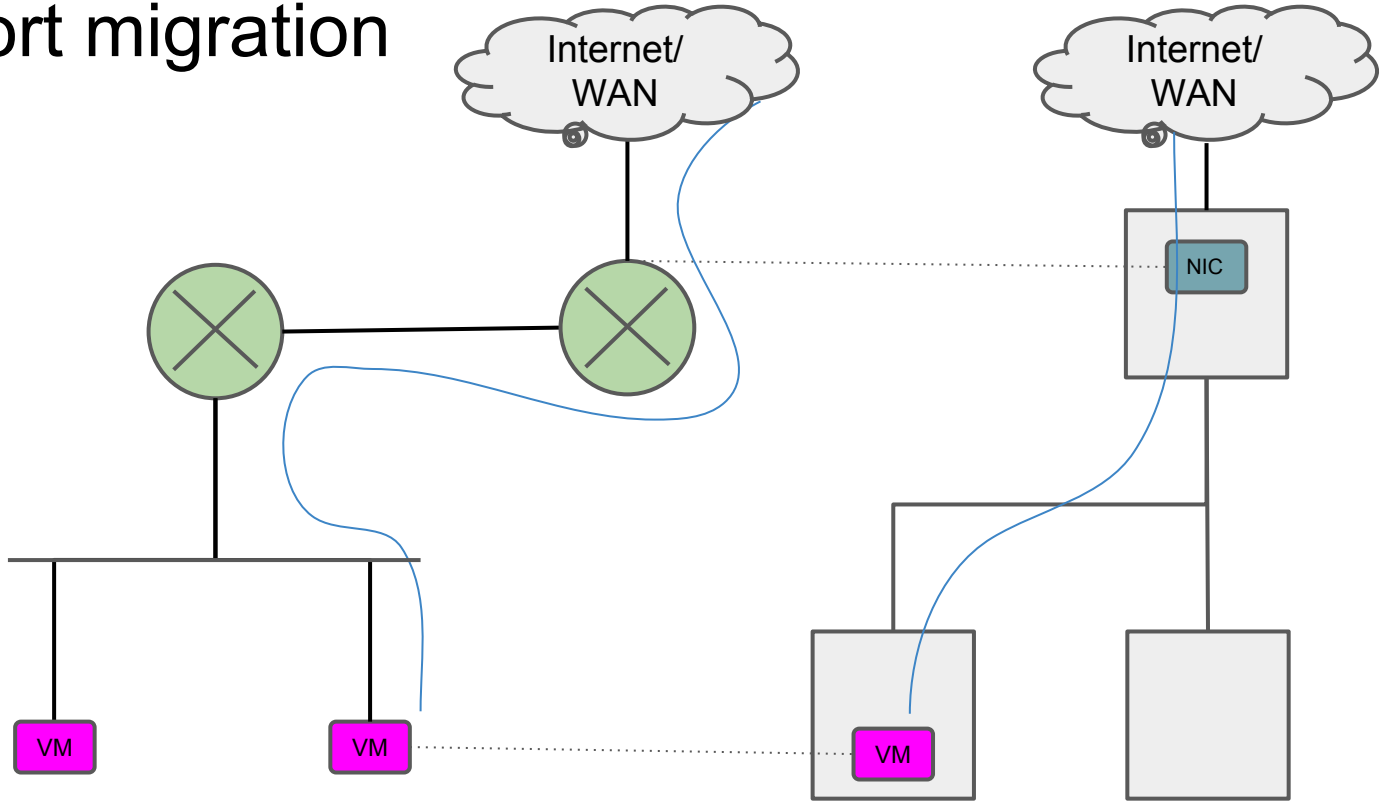
midonet

Port migration



midonet

Port migration

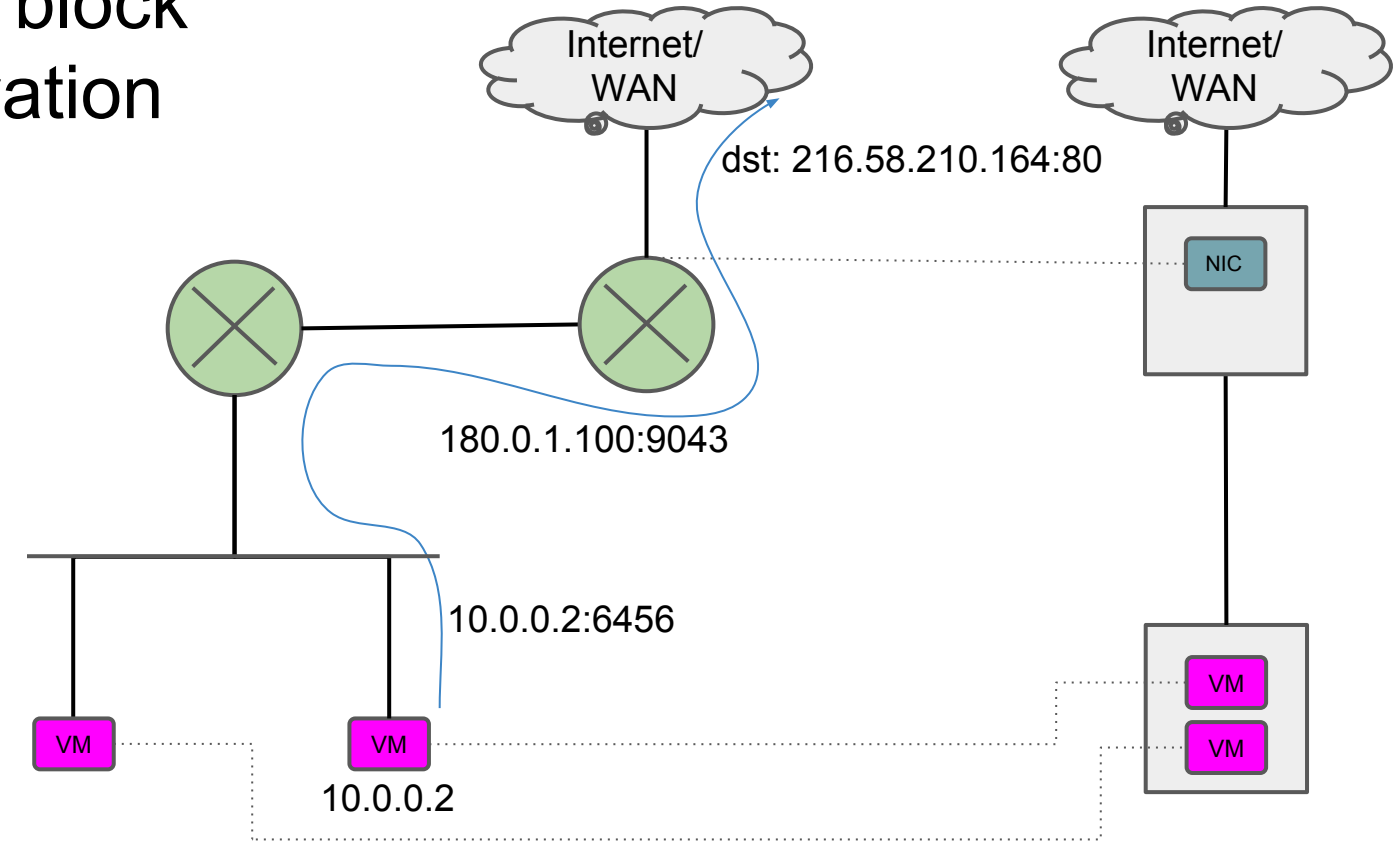


midonet

State transfer

- Directly between hosts
  - Requires hinting from the integration layer
- Through a data store
  - Can be considered part of the interested set
  - Requires state to be replicated to it (asynchronously, fire-and-forget)
  - Also solves restart problem

SNAT block reservation



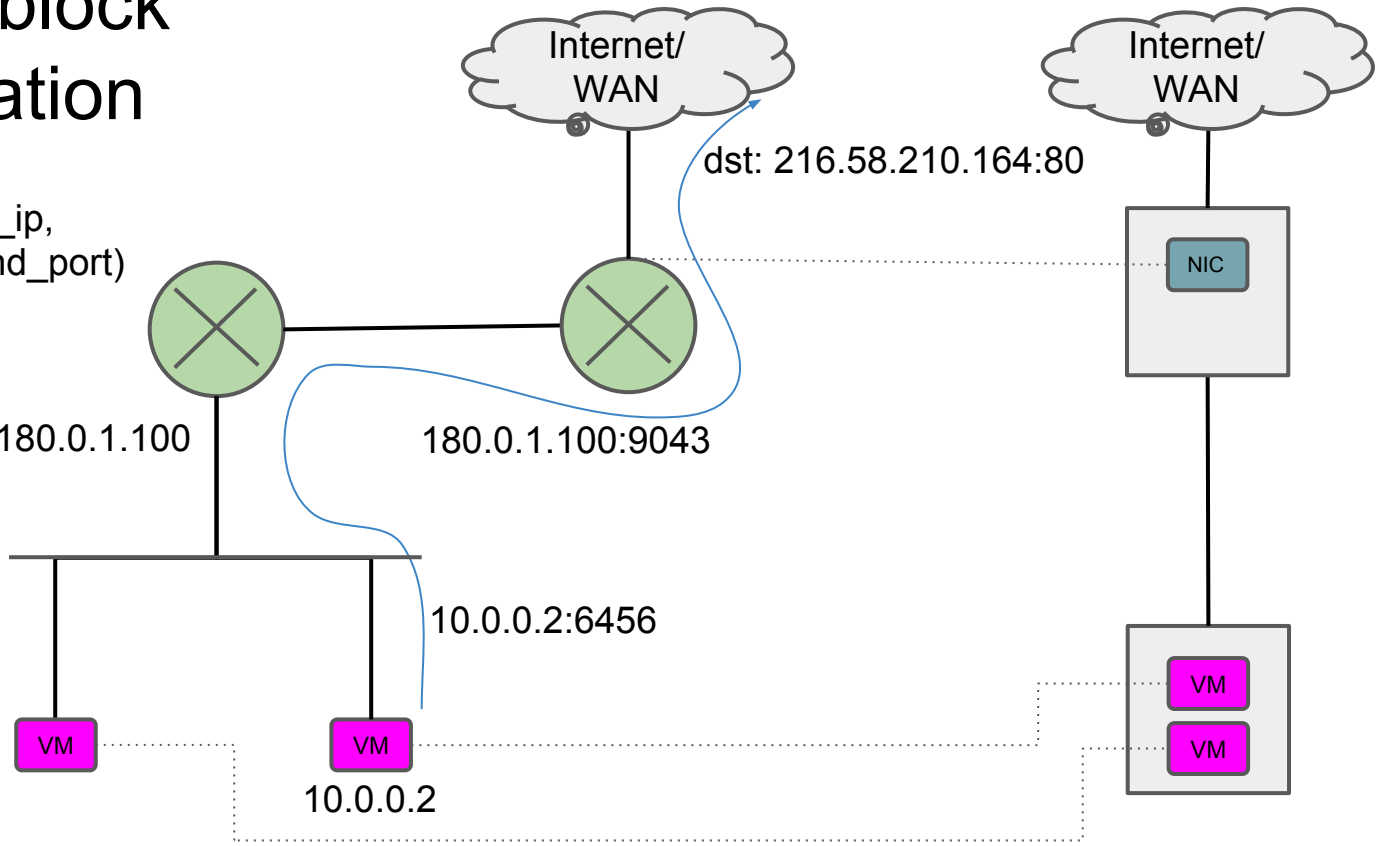
midonet

SNAT block reservation

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

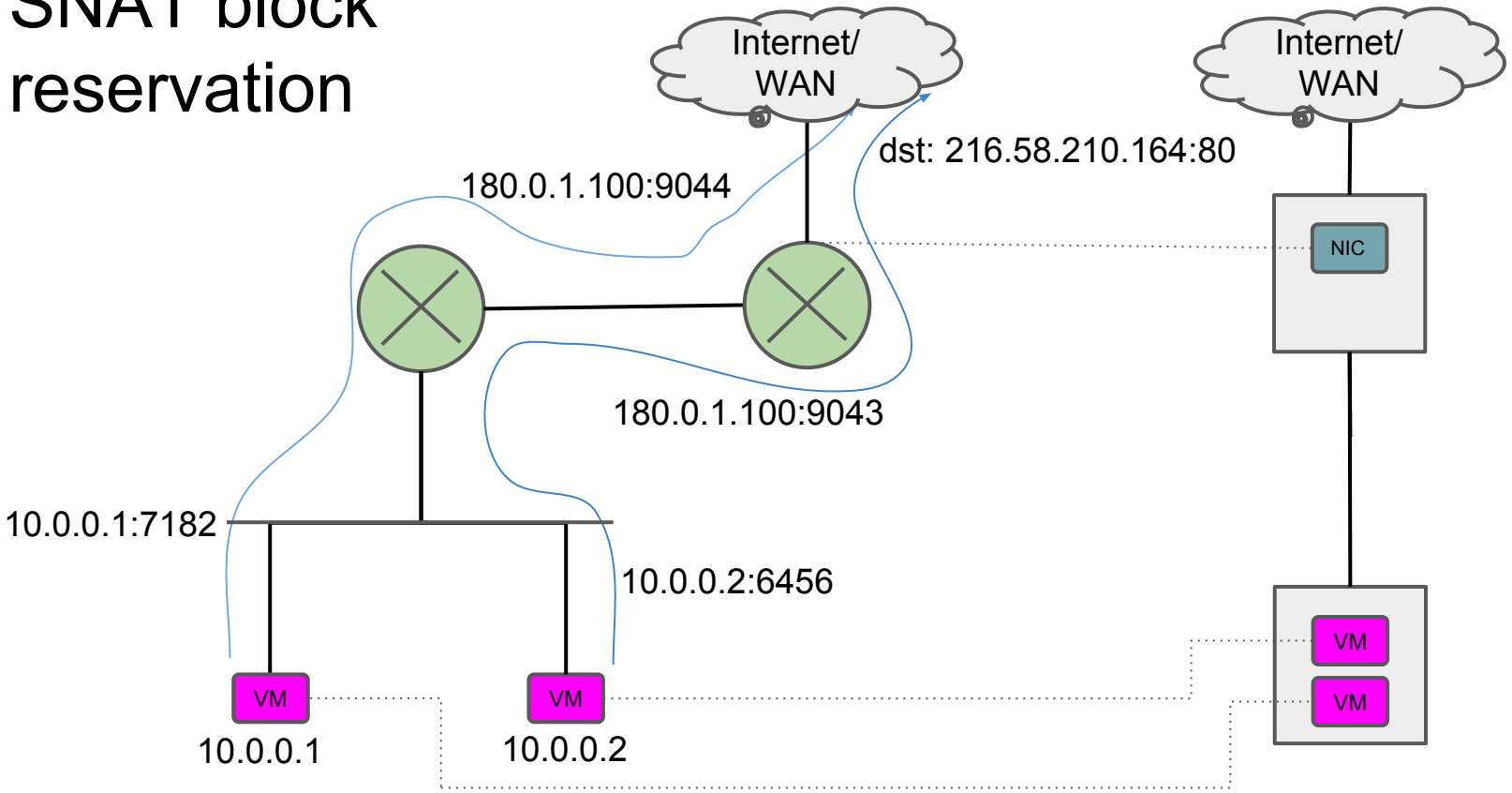
e.g.

180.0.1.100..180.0.1.100
5000..65535



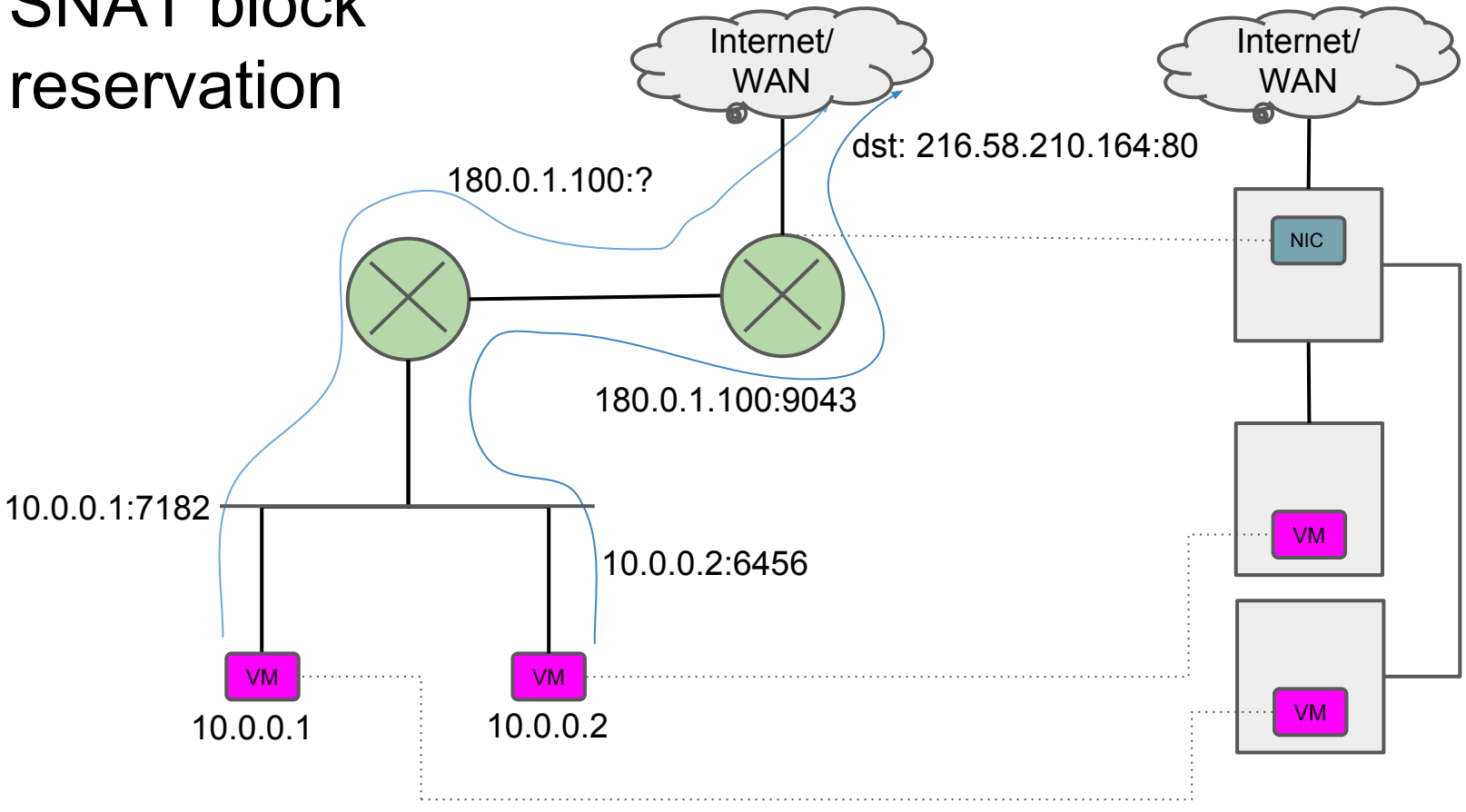
midonet

SNAT block reservation



midonet

SNAT block reservation

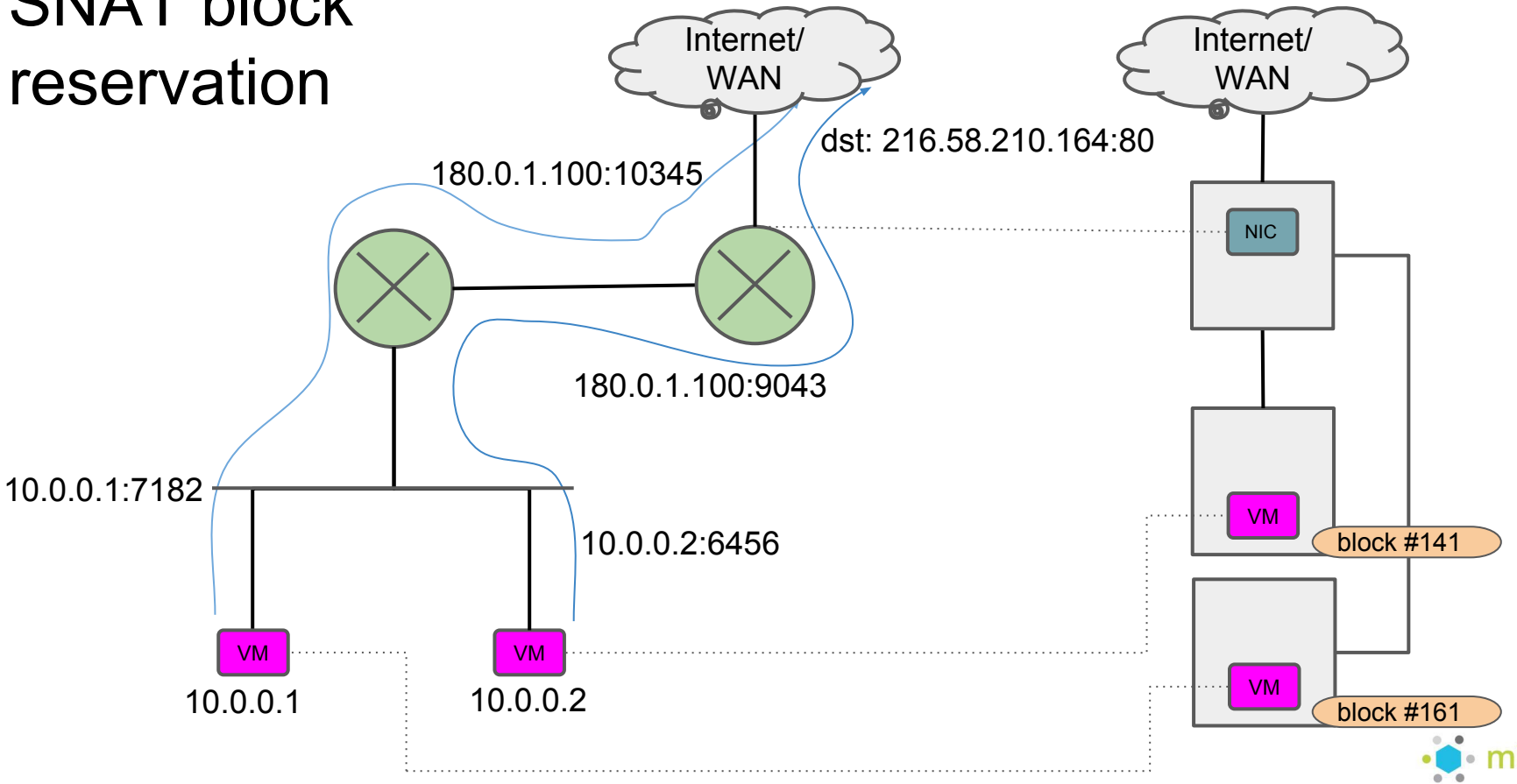


midonet

SNAT block reservation

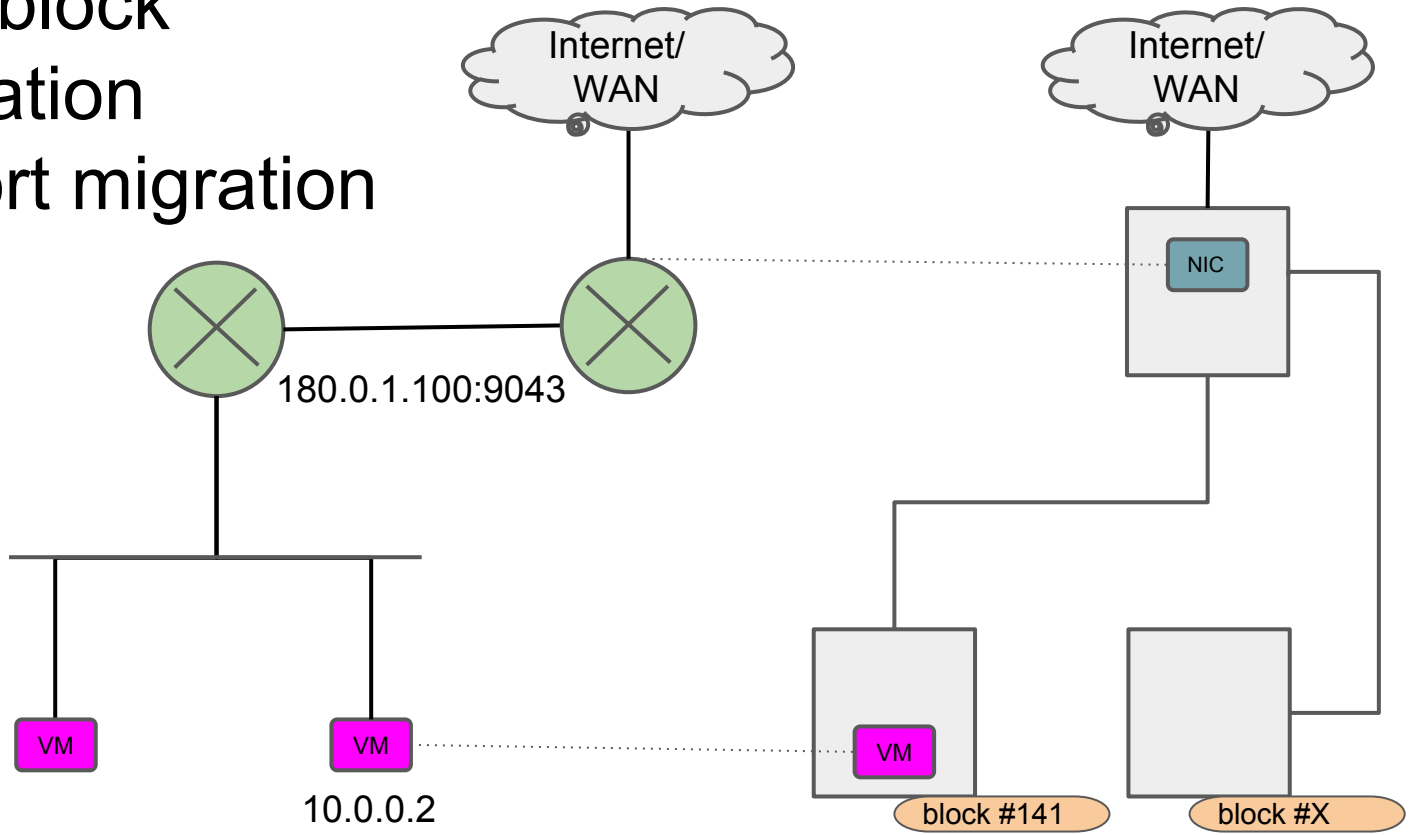
- Performed through ZooKeeper
- /nat/{device_id}/{ip}/{block_idx}
- 64 ports per block, 1024 total blocks
- LRU based allocation
- Blocks are referenced by flow state

SNAT block reservation



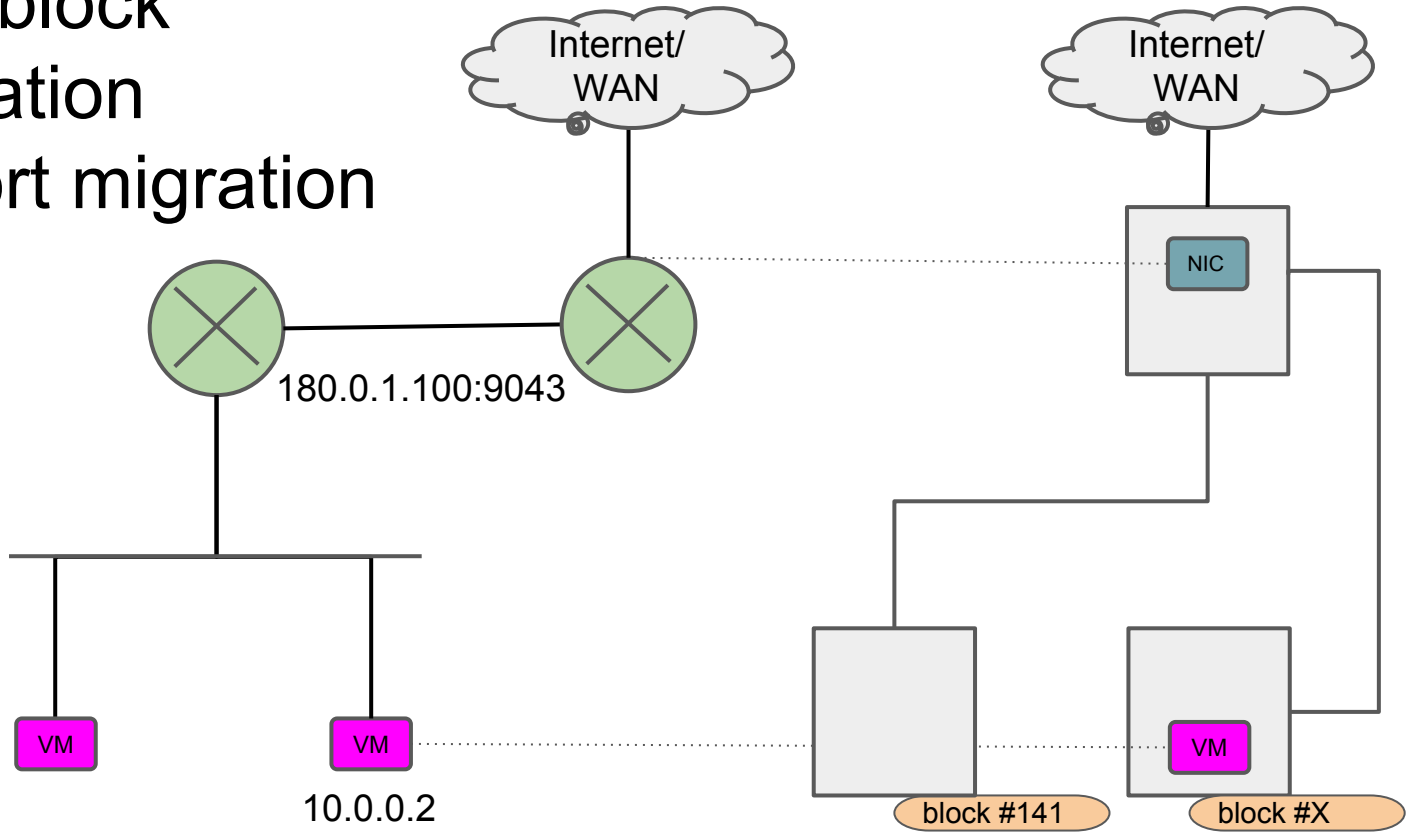
midonet

SNAT block reservation and port migration



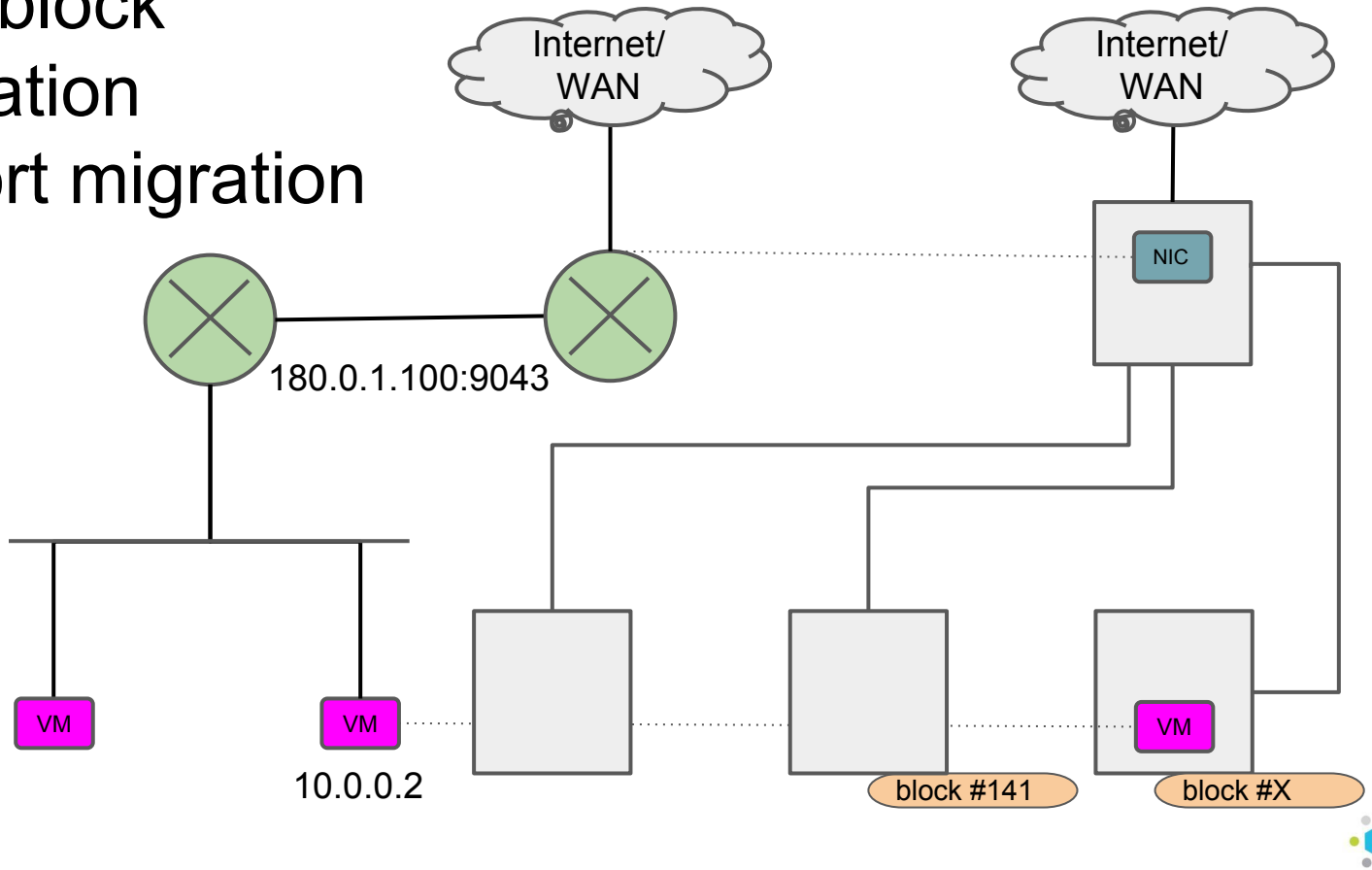
midonet

SNAT block reservation and port migration



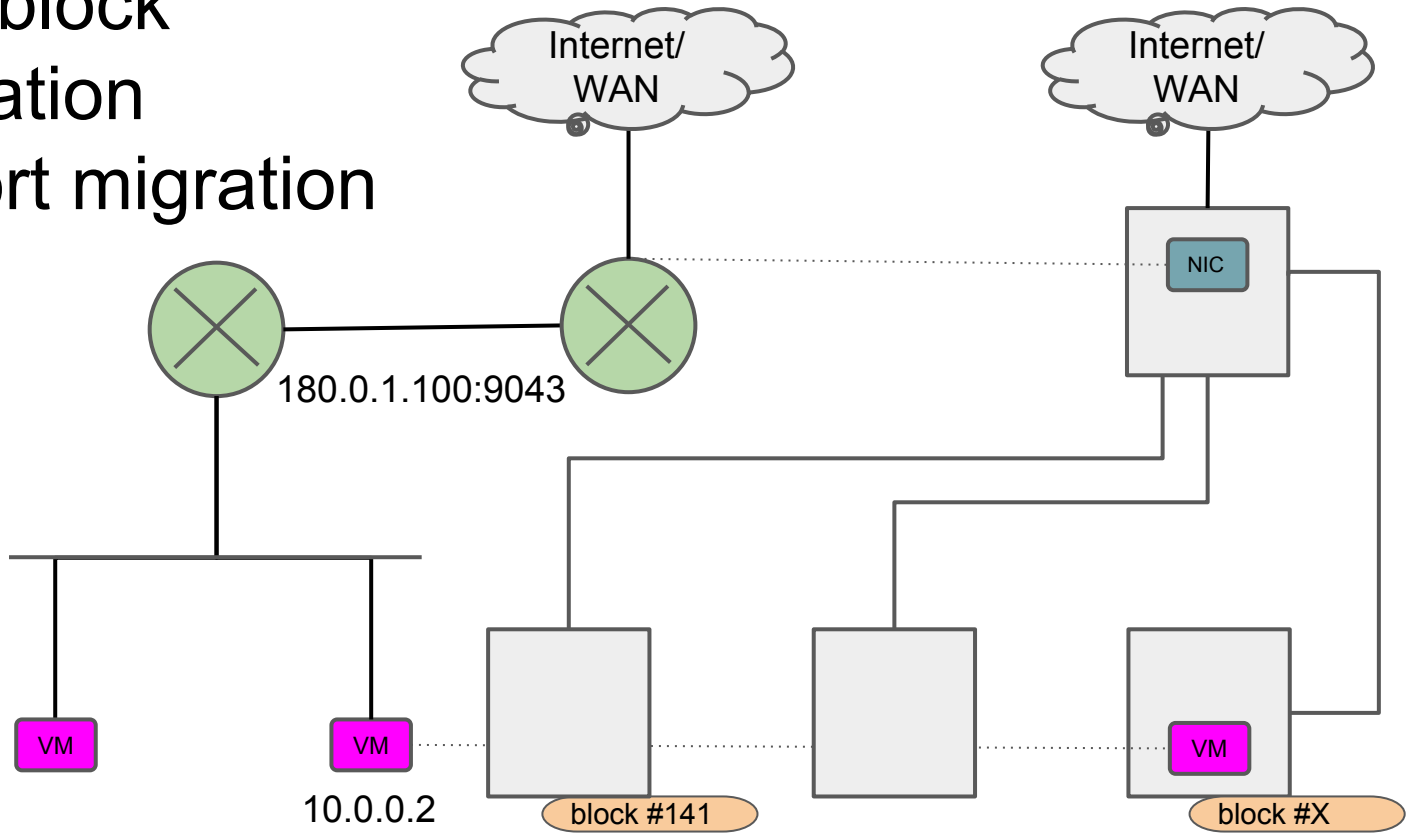
midonet

SNAT block reservation and port migration



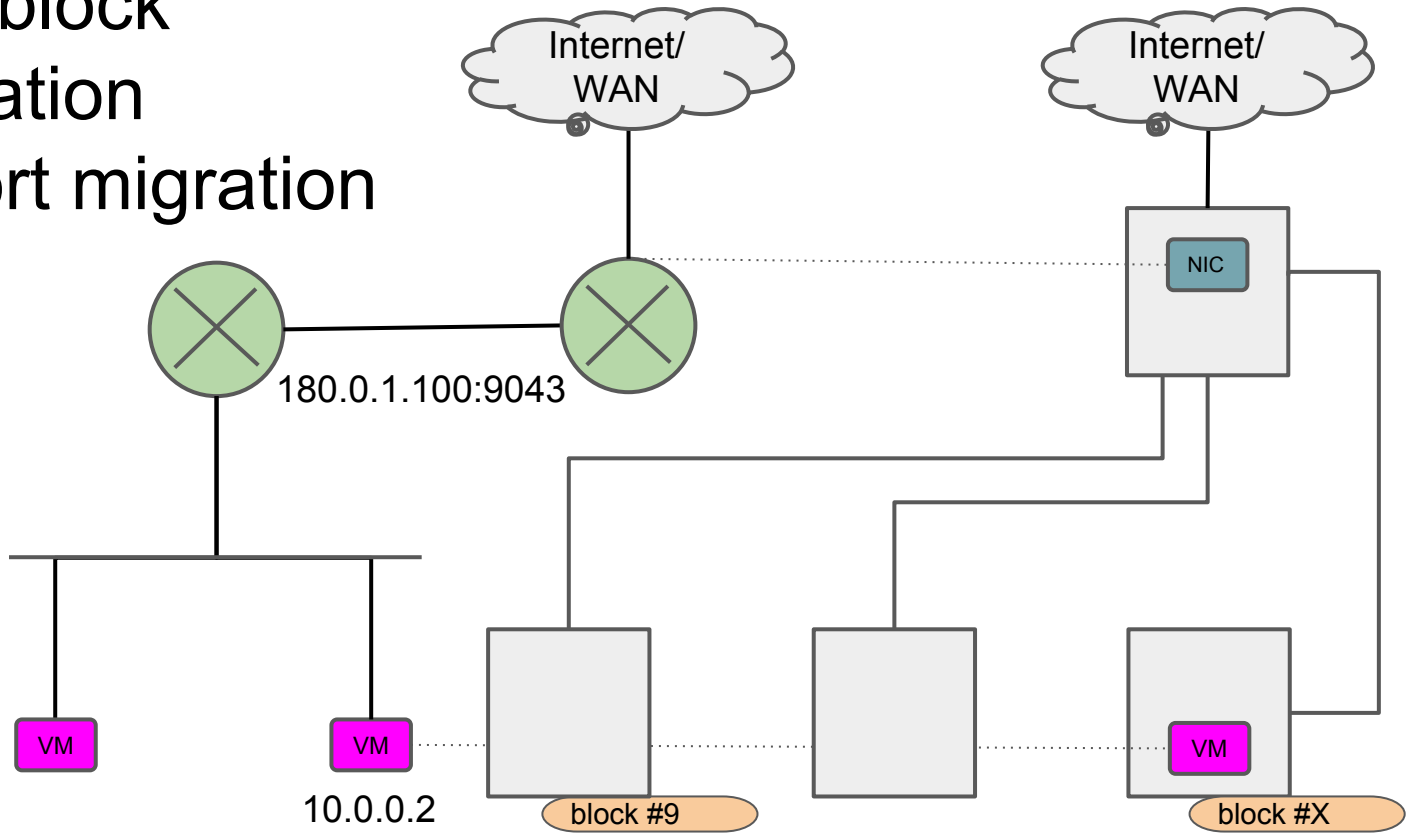
midonet

SNAT block reservation and port migration



midonet

SNAT block reservation and port migration



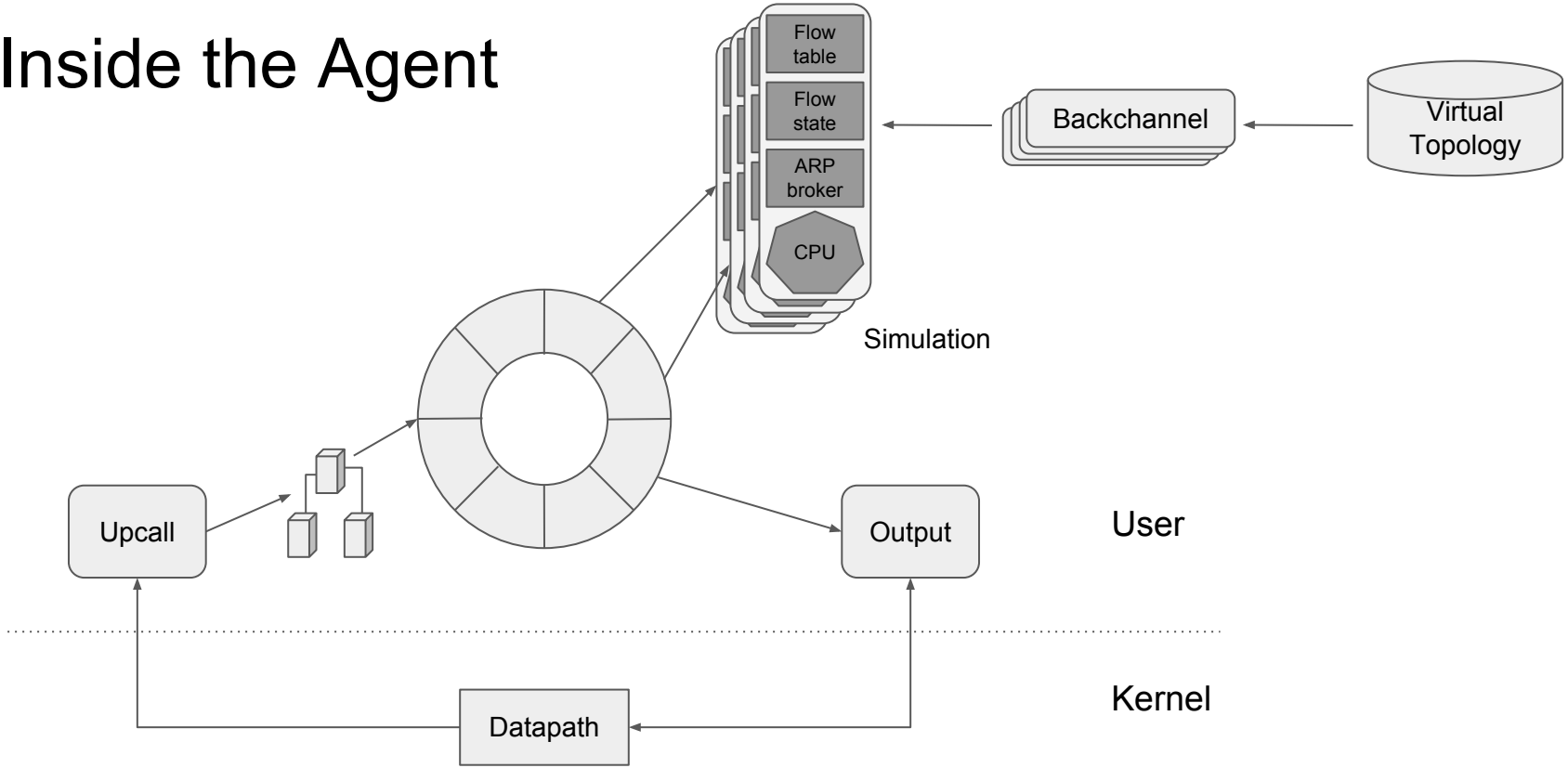
midonet

SNAT block overloading

- Overload source port for same source IP, based on
  - Destination IP
  - Destination port
- Resiliency against port scanning a given destination IP

Low-level

Inside the Agent



midonet

Performance

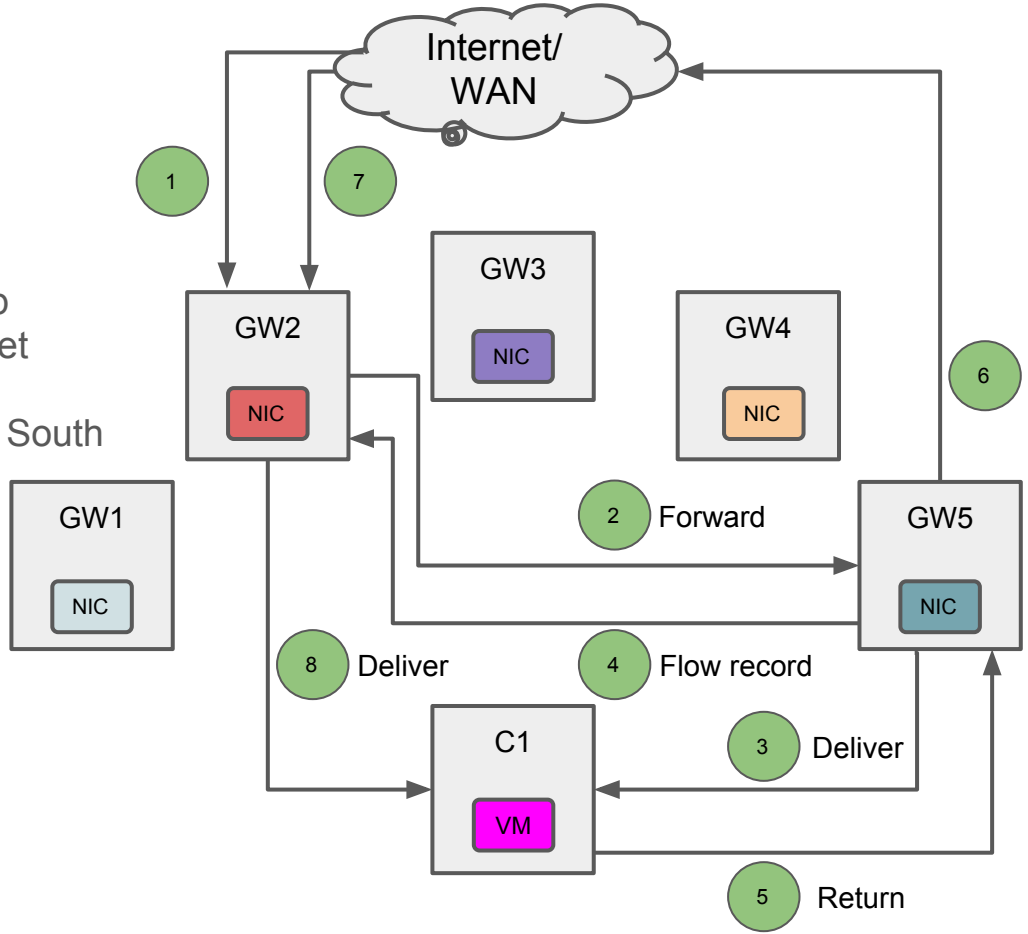
- Sharding
  - Share nothing model
  - Each simulation thread is responsible for a subset of the installed flows
  - Each simulation thread is responsible for a subset of the flow state
  - Each thread ARPs individually
  - Communication by message passing through “backchannels”
- Run to completion model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

Scaling

- Horizontally scale gateways
- Consider
  - 6 GWs doing 100kfps, 100 Computes doing 25kfps
  - Retention time: 30min
  - Global flow state dataset size: 177GB
- Can’t hold everything in memory
  - Need partitioning
  - Static, by partitioning announced routes
  - Dynamically
- Have to send a lot of flow state messages
  - Can use IP multicast

Dynamic partitioning

- Consistent hashing
- One extra inter-DC hop
  - For the first packet
- Special case TCP
  - Only for North -> South
- eBPf in the kernel



midonet

Questions?