Challenges in Distributed SDN

Duarte Nunes
duarte@midokura.com
@duarte_nunes
MidoNet transform this...
...into this...
Physical view

Internet/WAN

midonet gateway 1
midonet gateway 2
midonet gateway 3

midonet nsdb 1
midonet nsdb 2
midonet nsdb 3

VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM

Bare Metal Server

Bare Metal Server

IP Fabric

VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM

VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM
VM VM VM VM
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 Hosts

- **Gateway 1**
  - IP Fabric
  - Internet/WAN
  - OVS kmod
  - VXLAN Tunnel Port
  - eth0
  - port1, port2, veth0
  - Quagga, bgpd
  - MidoNet Agent (Java Daemon)

- **Compute 1**
  - IP Fabric
  - OVS kmod
  - VXLAN Tunnel Port
  - eth0
  - port5, tap12345
  - VMs
  - MidoNet Agent (Java Daemon)
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
ARP Table
ARP Table

![Diagram of ARP Table with VMs and IP Fabric]
ARP Table

Encapsulated ARP request
ARP Table

ARP reply handled locally and written to ZK

ZK notification
ARP Table

Diagram showing the interaction between VMs and an ARP Table, with encapsulated packets and an IP fabric.
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
Virtual NAT

Forward flow:
180.0.1.100:80
10.0.0.2

Return flow:
10.0.0.2:6456
10.0.0.2

Network elements:
- VM
- LB
- NIC
- Internet/WAN
Asymmetric routing
Asymmetric routing

Forward flow
Asymmetric routing
Asymmetric routing
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

1. New flow arrives
2. Check or create local state
3. Replicate the flow state to interested set
4. Tunnel the packet
5. Deliver the packet
Sharing state - Peer-to-peer handoff

1. Return flow arrives
2. Lookup local state
3. Tunnel the packet
4. Deliver the packet

Node 1
Node 2
Node 3 (possible asym. ret. path)
Node 4 (possible asym. fwd. path)
Sharing state - Peer-to-peer handoff

1. Exiting flow arrives at different node

2. Lookup local state

3. Tunnel the packet

4. Deliver the packet

Node 1

Node 2

Node 3 (possible asym. fwd. path)

Node 4 (possible asym. ret. path)
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

180.0.1.100:9043
10.0.0.2:6456

10.0.0.2
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

10.0.0.1:7182 -> 10.0.0.1:7182
10.0.0.1:7182 -> 180.0.1.100:9043
180.0.1.100:9043 -> 180.0.1.100:9044
10.0.0.1:7182 -> 10.0.0.2:6456
10.0.0.2:6456 -> 180.0.1.100:9044
SNAT block reservation

10.0.0.1:7182

10.0.0.2:6456

10.0.0.1

10.0.0.2

180.0.1.100:9043


Internet/WAN

NIC

VM

10.0.0.1

180.0.1.100:?
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

- Flow table
- Flow state
- ARP broker
- CPU
- Backchannel
- Virtual Topology
- Simulation
- User
- Kernel
- Upcall
- Output
- Datapath
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**