The State of Stateful Services

Joe Stringer, Jarno Rajahalme
{joe,jarno}@ovn.org
Agenda

- Connection Tracking
- Firewalling
- NAT
- Other stateful services
- Summary
Motivation

● OVN heating up
  ○ OpenStack
  ○ Kubernetes

● Expanding feature set
  ○ Firewalls
  ○ NAT
  ○ Load Balancing
Connection Tracking

- Track connections
  - Per-connection state stored in datapath
  - Expose concepts like “new connection”

- Microflow steering without matching every microflow
  - Avoid upcall when possible

- Leverage existing work

- Foundation for a variety of stateful services
Connection Tracking

OVS Flow Table

1. Lookup
2. table=N?
3. commit?
Create Connttrack Table

Populate ct_* fields, continue processing

actions=ct(...),...

userspace

datapath

Connection Tracking

OVS Flow Table

1. Lookup
2. table=N?
3. commit?
Create Connttrack Table

Populate ct_* fields, continue processing

actions=ct(...),...

userspace

datapath
# Example firewall

<table>
<thead>
<tr>
<th>Table</th>
<th>Match</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>priority=100,in_port=1,ip</td>
<td>ct(commit),2</td>
</tr>
<tr>
<td>0</td>
<td>priority=100,in_port=2,ip,ct_state=-trk</td>
<td>ct(table=1)</td>
</tr>
<tr>
<td>0</td>
<td>priority=10,arp</td>
<td>normal</td>
</tr>
<tr>
<td>0</td>
<td>priority=1</td>
<td>drop</td>
</tr>
<tr>
<td>1</td>
<td>priority=100,in_port=2,ip,ct_state=+est</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>priority=1</td>
<td>drop</td>
</tr>
</tbody>
</table>
Packet & connection states

- Packets are untracked initially*, become tracked via ct()
- Tracked (trk) packets may be:
  - Part of a new or established connection
  - Reply (rpl): Connection must be established
  - Related: Related to an established connection
  - Invalid

* Exception: internal ports in current namespace may inherit state from local network stack
Conntrack match fields

- **ct_state**
- **ct_zone**
  - Logically separate connection tracking table
  - Multi-tenancy
- **ct_mark**
  - Attach 32 bits of metadata to particular connections
- **ct_label**
  - Similar to mark, 128 bits
Conntrack action

- Transparently reassemble IP fragments (re-fragment on output)
- No args: Let the connection tracker know, ignore its results.
- zone=N: Track in logical zone N
- alg=ftp: Apply protocol-specific tracking, eg FTP detect data connections
- exec(..): Additional actions in connection tracking context
  - set_field(...->ct_mark); set_field(...->ct_label)
  - Changes matchable only on recirculated packets.
- table=N: Clone packet to send to connection tracker. When the connection tracker is finished, resume processing in table N for that packet. The original packet continues right after the ct(...) action.
- commit: Persist state about this connection
NAT & Load Balancing
Network Address Translation Use Cases

- OpenStack allows a persistent *Floating IP* to be assigned for a VM in addition to dynamically allocated *Fixed IP*
  - Both Source NAT (*SNAT*) and Destination NAT (*DNAT*) needed to map between these

- Kubernetes Services hide servers behind a *Virtual IP addresses*
  - *Load balancer* chooses the server for each connection
  - *DNAT* to map the virtual IP to the chosen server’s IP address

- The corresponding transport port can also be mapped
  - Without an explicit port (range) the port is mapped only in case of a collision
NAT Action Extends The CT Action

- Always executes in the context of the current connection
  - CT(..., NAT(...), ...)
  - Typically NAT can be added to CT actions already used for ACLs.

- **New connections** need a source or destination address (range) and optionally a port (range) + a CT commit and possibly the zone argument
  - ct(commit,nat(src=10.0.0.240),alg=ftp)
  - ct(commit,zone=1,nat(src=10.0.0.240:32768-65535,random))
  - ct(commit,nat(dst=10.0.0.128-10.0.0.254,hash))
  - ct(commit,nat(dst=10.0.0.240-10.0.0.254:32768-65535,persistent))

- **NAT without arguments** only NATs committed, established, or related uncommitted connections
NAT for OpenStack Floating IPs

VM
Fixed IP: 10.0.0.1
Floating IP: 176.1.2.3

E.g., Public Internet

10.0.0.1
new
ct(nat,table=1)

output
est
c(t(nat(src=176.1.2.3),commit),output)

176.1.2.3
new, nw_dst=176.1.2.3
ct(nat,table=2)

output

c(t(nat(dst=10.0.0.1),commit),output)

13
DNAT Load Balancing

Virtual IP
ip_dst=10.1.1.64

Select Group

NAT(dst=10.1.1.2)
NAT(dst=10.1.1.3)
NAT(dst=10.1.1.4)

Client₁
Client₂

Server₁
Server₂
Server₃
DNAT Load Balancing (cont.)

- Controller needs to balance traffic by (re-)specifying group weights
  - Based on server feedback or group stats
- Bucket selection currently happens on Ethernet + 5-tuple hash
  - $\text{recirc}_\text{id}(0), \text{in}_\text{port}(2), \text{eth}(\text{src}=80:88:88:88:88:11, \text{dst}=80:88:88:88:88:88), \text{eth}_\text{type}(0x0800), \text{ipv4}(\text{src}=10.1.1.1, \text{dst}=10.1.1.64, \text{proto}=6, \text{frag}=\text{no}), \text{tcp}(\text{src}=60754, \text{dst}=80), ...$, actions: $\text{ct}(\text{commit, nat(\text{dst}=10.1.1.4)}), \text{recirc}(0x1)$
- Every connection goes to userspace as a miss upcall
- More work needed to avoid unnecessary upcalls
Connection Tracking Status

- **Conntrack kernel patches merged and part of Linux-4.3**
- **Open vSwitch conntrack patches:**
  - Userspace (ofproto) support in master
  - System-traffic testsuite in master
  - Kernel datapath backport under review
  - DPDK/Userspace datapath series posted
- **NAT: RFC series posted on net-next and ovs-dev**
  - Non-RFC when net-next window opens
  - DPDK/Userspace datapath future work
- **Load-balancing: Investigation phase**
  - Plausible with NAT functionality
  - May need further extension for a full implementation
Q&A