Conntrack + OvS

Aaron Conole <aconole@redhat.com>

November 16, 2017

What is the talk about?

Conntrack!

It's the way we track connections (clever, right?)

Connection!!

It's more than just a logical flow. It's an actual flow with state, hopes, dreams, etc.

Latest!!!

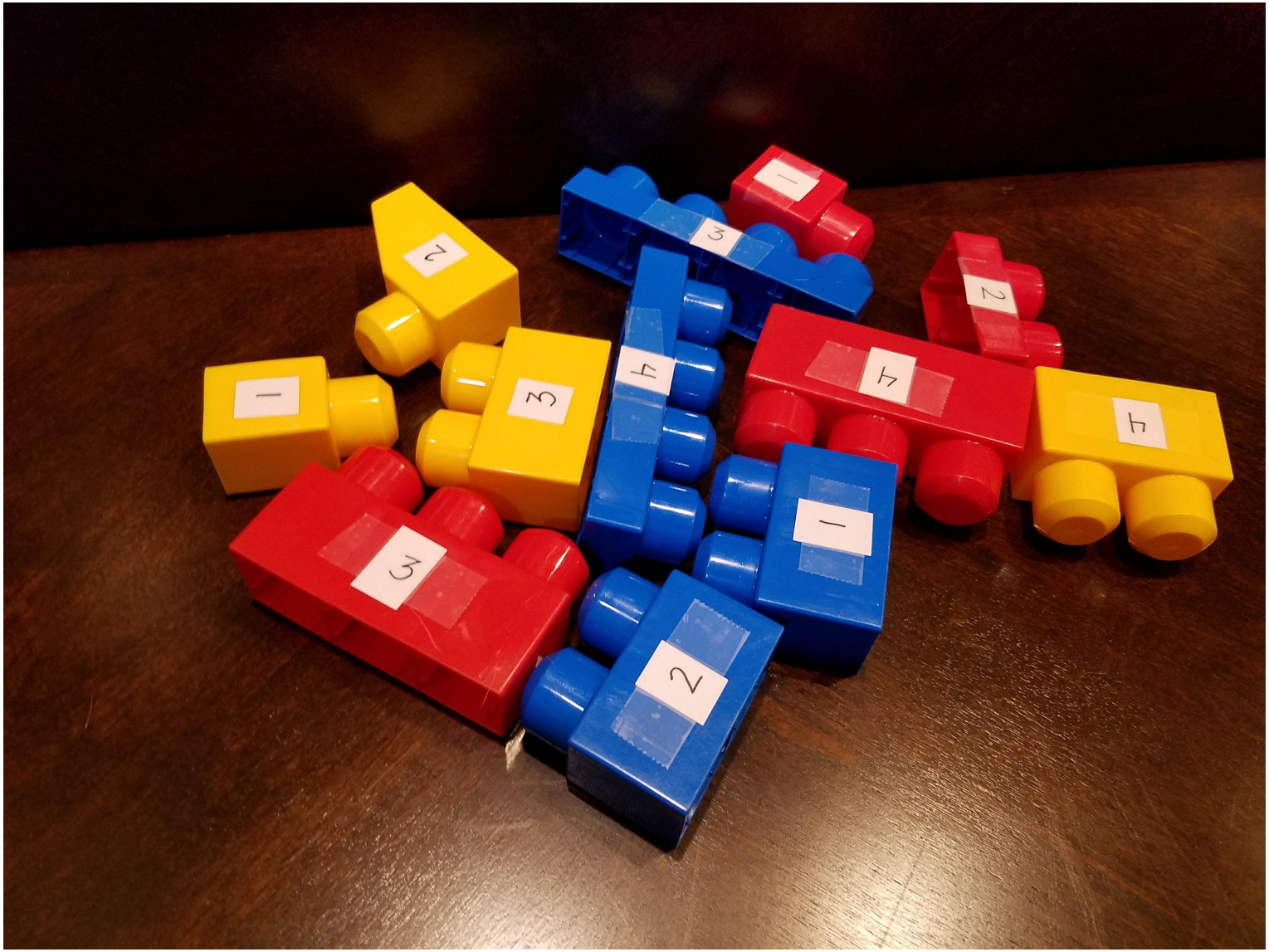
This will include conntrack features in OvS 2.8 and later.

What is a connection?

A connection is the packet-based mechanism that two software elements use to communicate information.
This differs from a flow by one important property: packets belonging to a connection have state.

redhat

Packets



redhat

Flow

Connection

Why is a connection important

Connections represent a distinct bidirectional communications channel. They are your synchronized TCP sessions, for example.

redhat

How is a connection 'stateful'? States!

NEW

A packet which is the first packet of a connection.

RELATED

Similar to NEW, but for connections which can be 'affiliated' with an existing connection. Example: ftp data connection

ESTABLISHED

A packet which is part of an existing connection.

INVALID

- A packet which is not part of any connection, and isn't the first packet of a connection.
- Additionally, an unexpected packet which is received for an existing connection (eg. duplicate SYN)

redhat

Who is using it?

OpenShift

NetworkPolicy plugin uses both xtables and ovs NAT/CT actions

OpenStack

Security Groups neutron plugin uses iptables and OvS to provide secgroup

Conntrack Helpers: The San Francisco Treat

- A conntrack helper is a bit of code that can associate packets with existing connections.
- FTP PORT command - it advertises where a new TCP connection will exist.
- Conntrack helpers make conntrack much more useful.
- Added for Userspace as of OvS 2.8.. we'll get to that (Thanks Darrell B, btw!)

Conntrack implementations

Each datapath provider has its own form of conntrack.
Windows and Linux have netlink support, so they have an 'in-kernel' conntrack.
Everything else uses the newly* added userspace conntrack. (New as of OvS 2.6)

In-kernel Conntrack for Linux (things to know)

- Well tested - every linux system is using it.
- Reusable - share the same conntrack tables between all the kernel actors (xtables, nft)
- Tunable - multiple parameters (configurable timeouts, hash table sizes, policies)
- Well supported - lots of tools, documentation, etc.
- Internals covered in lots of places (see Florian Westphal's Netdev 2.1 talk)

redhat

Userspace Conntrack

- Derived from FreeBSD's conntrack code
- Lives in lib/conntrack*.{c,h}
- Hooked into netdev datapath
- Only a few helpers at the moment (icmp, udp, tcp, and FTP)
- Not many 'tunable' parameters

Userspace Conntrack datastructures - Conntrack

Per-netdev datapath conntrack instance

```
struct conntrack {
    struct conntrack_bucket buckets[CONNTRACK_BUCKETS];
    /* ... */
    atomic_count n_conn;
    /* ... */
    atomic_uint n_conn_limit;
    /* ... */
};
```

Userspace Conntrack datastructures - Conntrack Bucket

Conntrack-buckets. Connections and sorted expiration list.

```
struct conntrack_bucket {
    /* ... */
    struct hmap connections; /* used with lock */
    /* ... */
    struct ovs_list exp_lists[N_CT_TM]
    /* ... */
}
```

redhat

Userspace Conntrack datastructures - Conn

Conn structure (the actual connection). NAT info, algorithm, tuples

```
struct conn {
    /* ... */
    struct conn_key key, rev_key;
    /* ... */
    long long expiration;
    /* ... */
    struct nat_action_info_t *nat_info;
    char *alg;
}
```

redhat

Datapath support

Support just for dumping the current conntrack connections

```
int (*ct_dump_start)(struct dpif *, \
                     struct ct_dpif_dump_state **state,
                     const uint16_t *zone, int *);
int (*ct_dump_next)(struct dpif *, \
                    struct ct_dpif_dump_state *state,
                    struct ct_dpif_entry *entry);
int (*ct_dump_done)(struct dpif *, \
                    struct ct_dpif_dump_state *state);
```

redhat

Conntrack dump command

Kernel / netlink

```
# Dump current connections
ovs-dpctl dump-conntrack
# List connections (similar to above)
conntrack -L
# Flush conntrack
conntrack -F
# Listen for events
conntrack -E
```

Userspace

```
ovs-appctl dpctl/dump-conntrack netdev@ovs-netdev
```

Simple conntrack example

Simple TCP flow example:

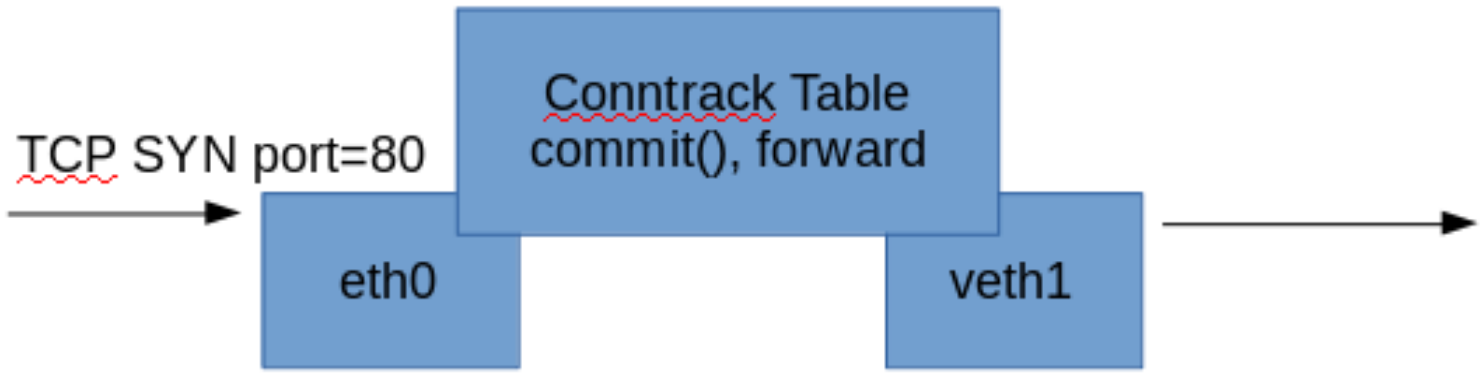
```
# First, go to the conntrack table
ovs-ofctl add-flow "table=0,priority=10,ct_state=-trk,ip,\
                    actions=ct(table=1)"
ovs-ofctl add-flow "table=0,priority=10,arp,actions=NORMAL"
ovs-ofctl add-flow "table=0,priority=1,drop"
...

# If a new input tcp packet to port 80 comes from eth0,
# track it and forward
ovs-ofctl add-flow ovsbr0 "table=1,priority=1000,
          ct_state=+new+trk,tcp,tp_dst=80,in_port=eth0,
          actions=ct(commit),output:veth1"
...
```

redhat

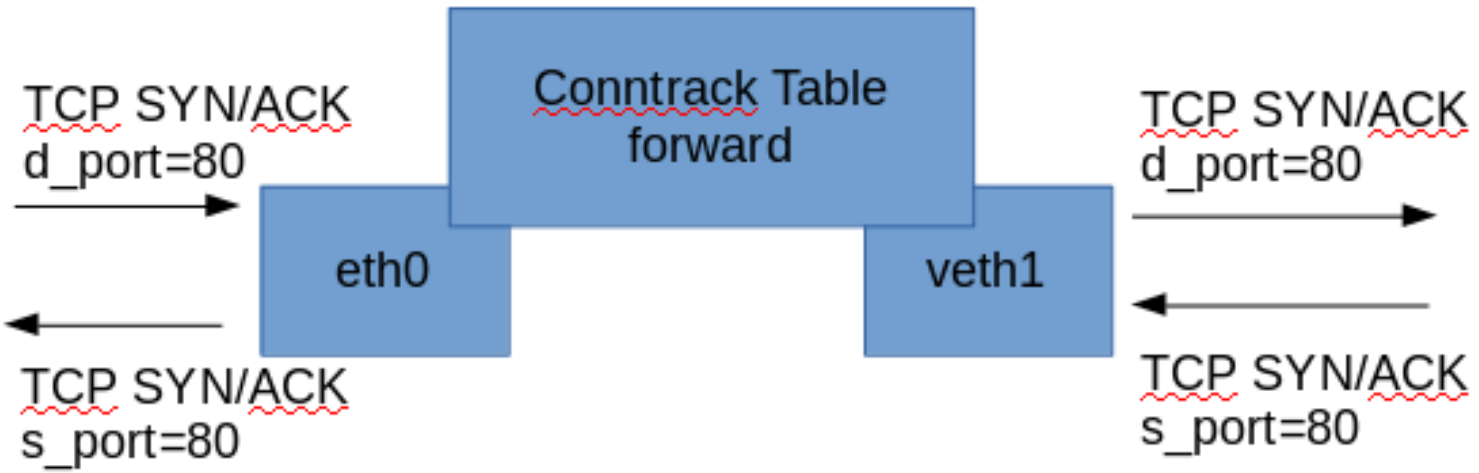
Simple conntrack example (cont'd)

```
# if an existing connection,
# forward the packets in each direction
ovs-ofctl add-flow ovsbr0 "table=1,priority=900,
          ct_state=+est+trk,in_port=eth0,
          actions=output:veth1"
ovs-ofctl add-flow ovsbr0 "table=1,priority=900,
          ct_state=+est+trk,in_port=veth1,
          actions=output:eth0"
...
```

The simple example - NEW



The simple example - ESTABLISHED



redhat

Complex conntrack example - FTP

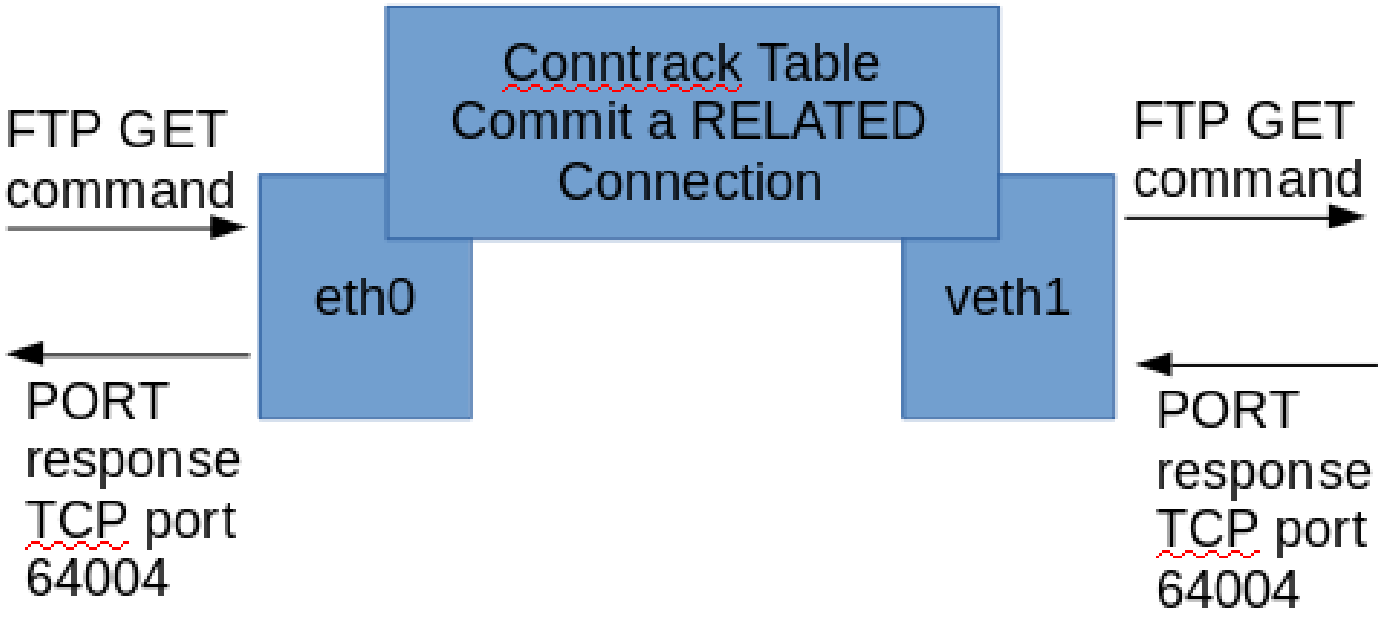
```
# First, go to the conntrack table
ovs-ofctl add-flow "table=0,priority=10,ct_state=-trk,ip,\
                    actions=ct(table=1)"
ovs-ofctl add-flow "table=0,priority=10,arp,actions=NORMAL"
ovs-ofctl add-flow "table=0,priority=1,drop"
...

# If a new input tcp packet to port 21 comes from eth0,
# track it and forward
ovs-ofctl add-flow ovsbr0 "table=1,priority=1000,
          ct_state=+new+trk,tcp,tp_dst=21,in_port=eth0,
          actions=ct(commit),output:veth1"
...
```

redhat

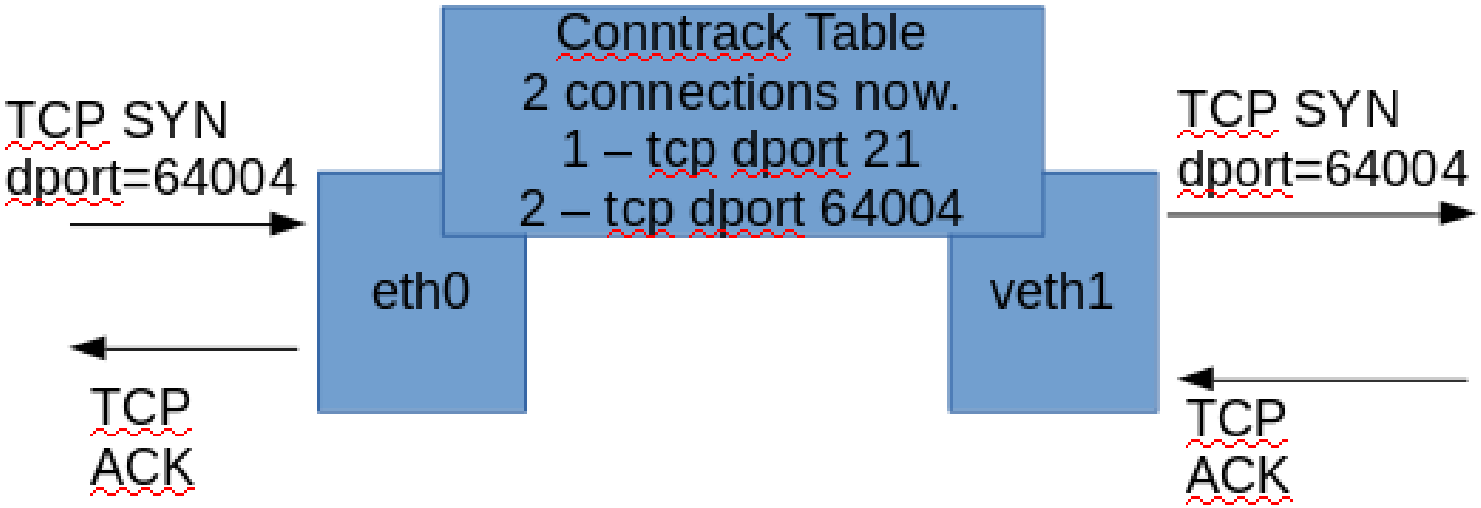
Complex conntrack example - FTP(cont'd)

```
ovs-ofctl add-flow ovsbr0 "table=1,priority=900,\
          ct_state=+new+rel+trk,\
          ip,in_port=eth0,actions=ct(commit),output:veth1"
ovs-ofctl add-flow ovsbr0 "table=1,priority=900,
          ct_state=+est+trk,in_port=eth0,
          actions=output:veth1"
ovs-ofctl add-flow ovsbr0 "table=1,priority=900,
          ct_state=+est+trk,in_port=veth1,
          actions=output:eth0"
```

The complex example - GET request



The complex example - GET request (contd)



redhat

Show the connections

```
# ovs-appctl dpctl/dump-conntrack

tcp,orig=(src=172.16.45.2,dst=172.16.45.254,sport=39082,\
dport=21),reply=(src=172.16.45.254,dst=172.16.45.2,sport=21,
dport=39082),protoinfo=(state=ESTABLISHED),helper=ftp

tcp,orig=(src=172.16.45.2,dst=172.16.45.254,sport=45344,\
dport=64004),reply=(src=172.16.45.254,dst=172.16.45.2,\
sport=64004,dport=45344),protoinfo=(state=ESTABLISHED)
```

Future work (userspace)

- IP fragmentation support
- Support for resizing the conntrack tables
- Support for tuning the connection state timeout values
- Improved logging (similar to nflog in kernel datapath)
- More extensible helper framework

FIN

- Questions?