

Can 'ofproto/trace' go live?

Adrián Moreno

ofproto/trace

- ✓ It's super useful!
- ✓ Helps connect datapath flows with OVN
- ✗ It's offline
 - Conntrack information is not reliable
- ✗ Typing the flow / packet is a bit cumbersome

Live of proto/trace?



ofproto/trace internals

```
static void
ofproto_trace__(struct ofproto_dpif *ofproto, const struct flow *flow,
                const struct dp_packet *packet, struct ovs_list *recirc_queue,
                const struct ofpact ofpacts[], size_t ofpacts_len,
                struct ds *output, bool names)
{
    struct ofpbuf odp_actions;
    ofpbuf_init(&odp_actions, 0);

    struct xlate_in xin;
    struct flow_wildcards wc;
    struct ovs_list trace = OVS_LIST_INITIALIZER(&trace);
    struct ofputil_port_map map = OFPUTIL_PORT_MAP_INITIALIZER(&map);
    struct hmap *portno_names = NULL;
    xlate_in_init(&xin, ofproto,
                 ofproto_dpif_get_tables_version(ofproto), flow,
                 flow->in_port.ofp_port, NULL, ntohs(flow->tcp_flags),
                 packet, &wc, &odp_actions);
    xin.ofpacts = ofpacts;
    xin.ofpacts_len = ofpacts_len;
    xin.trace = &trace;
    xin.recirc_queue = recirc_queue;
    xin.names = names;

    struct xlate_out xout;
    enum xlate_error error = xlate_actions(&xin, &xout);
}
```

ofproto/trace internals

```
static void
ofproto_trace__(struct ofproto_dpif *ofproto, const struct flow *flow,
                const struct dp_packet *packet, struct ovs_list *recirc_queue,
                const struct ofpact ofpacts[], size_t ofpacts_len,
                struct ds *output, bool names)
{
    struct ofpbuf odp_actions;
    ofpbuf_init(&odp_actions, 0);

    struct xlate_in xin;
    struct flow_wildcards wc;
    struct ovs_list trace = OVS_LIST_INITIALIZER(&trace);
    struct ofputil_port_map map = OFPUTIL_PORT_MAP_INITIALIZER(&map);
    struct hmap *portno_names = NULL;
    xlate_in_init(&xin, ofproto,
                 ofproto_dpif_get_tables_version(ofproto), flow,
                 flow->in_port.ofp_port, NULL, ntohs(flow->tcp_flags),
                 packet, &wc, &odp_actions);
    xin.ofpacts = ofpacts;
    xin.ofpacts_len = ofpacts_len;
    xin.trace = &trace;
    xin.recirc_queue = recirc_queue;
    xin.names = names;

    struct xlate_out xout;
    enum xlate_error error = xlate_actions(&xin, &xout);
}
```

Exploration mode: on

Hacking mode: on

How to “trace” an upcall?

```
@@ -233,7 +236,6 @@ struct upcall {
|
|     enum upcall_type type;          /* Type of the upcall. */
|     const struct nlattr *actions;   /* Flow actions in DPIF_UC_ACTION Upcalls. */
-
|     bool xout_initialized;          /* True if 'xout' must be uninitialized. */
|     struct xlate_out xout;          /* Result of xlate_actions(). */
|     struct ofpbuf odp_actions;      /* Datapath actions from xlate_actions(). */
@@ -257,6 +259,8 @@ struct upcall {
|     struct user_action_cookie cookie;
|
|     uint64_t odp_actions_stub[1024 / 8]; /* Stub for odp_actions. */
+
+     struct dpif_tracer *tracer;      /* Upcall dpif tracer. */
| };
```


We don't want to "trace" everything right?

we could use a filter ...

- ▶ Hmmm... How about a flow / match?
 - `"ovs-appctl ofproto/trace-add br-int "ipv4,nw_src=1.1.1.1"`

We don't want to "trace" everything right?

we could use a filter ...

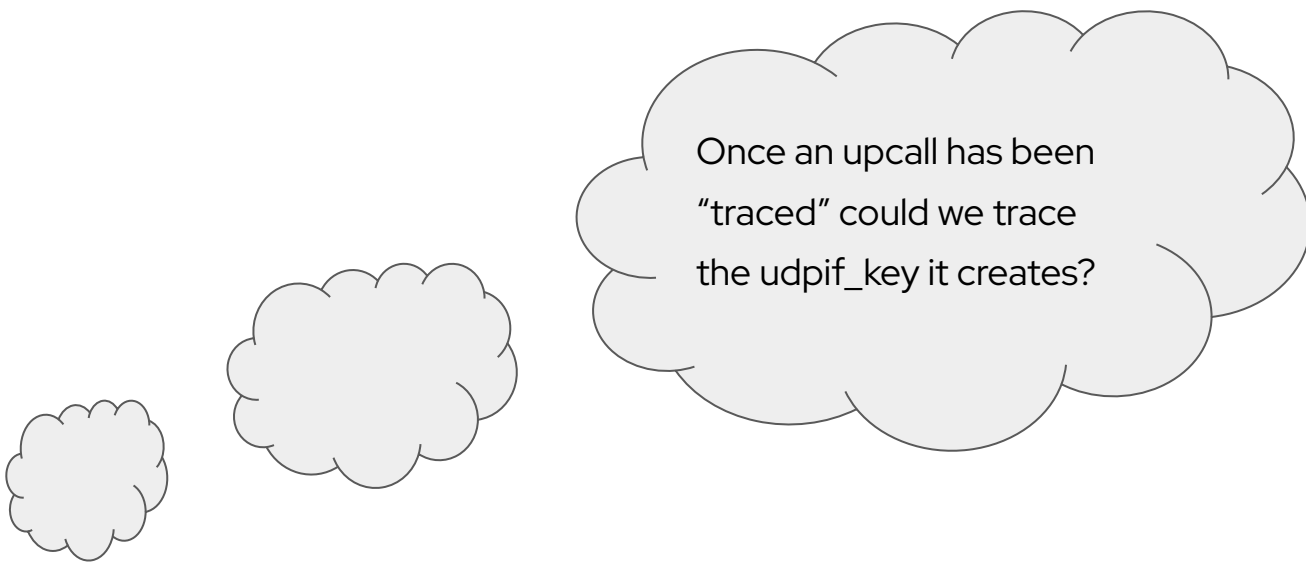
- ▶ Hmmm... How about a flow / match?
 - "ovs-appctl ofproto/trace-add br-int "ipv4,nw_src=1.1.1.1"



How do we return the string back to the user?

- ▶ Log?
 - Problems: log interleave, dirty
- ▶ Store it and use another unixctl command to retrieve it?
 - “ovs-appctl dpif/trace-show br-int <id>”
 - Problems: object lifetime
- ▶ USDT probes
 - Each probe is unconnected to others and we don't have global ids to bind them
 - If we send strings, we pay the cost of generating them
 - Else, we add binary dependency and external formatting

That's it? ...



Once an upcall has been
"traced" could we trace
the `udpif_key` it creates?

How to trace a ukey?

```
@@ -352,6 +382,8 @@ struct udpif_key {
    long long int flow_time;      /* last pps update time */
    uint64_t flow_packets;       /* #pkts seen in interval */
    uint64_t flow_backlog_packets; /* prev-mode #pkts (offl or kernel) */
+
+    struct dpif_tracer *tracer;  /* Optional tracer */
};
```

- ▶ *struct udpif_key* are created from an upcall. If we were tracing the upcall, trace the *udpif_key*.

Additional challenges of udpif_ukey tracking

- ▶ More complex lifetime management
- ▶ Connection between udpif_ukey and the upcall that created it has to be done manually,
 - Do we need an upcall_id?

Alternative approaches

- ▶ Just emit actions being translated via USDT
 - Plus add probes before and after running **xlate_actions**
 - Pros: simple
 - Cons:
 - No udpif_key tracking
 - Need to format actions / keys externally (e.g: usdt script or retis)

Demo

<https://asciinema.org/a/690480>

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat