[ovs-dev] [PATCH] ofproto: Return error codes for Rule insertions

Aravind Prasad S raja.avi at gmail.com
Thu Jul 12 18:04:47 UTC 2018

- Previous message: [ovs-dev] [PATCH v3 1/3] rhel: rename openvswitch kmod rhel6 spec file
- Next message: [ovs-dev] [PATCH] ofproto: Return error codes for Rule insertions
- **Messages sorted by:** [date] [thread] [subject] [author]

```
Currently, rule_insert() API does not have return value. There are some possible
scenarios where rule insertions can fail at run-time even though the static
checks during rule_construct() had passed previously.
Some possible scenarios for failure of rule insertions:
**) Rule insertions can fail dynamically in Hybrid mode (both Openflow and
Normal switch functioning coexist) where the CAM space could get suddenly
filled up by Normal switch functioning and Openflow gets devoid of
available space.
**) Some deployments could have separate independent layers for HW rule
insertions and application layer to interact with OVS. HW layer
could face any dynamic issue during rule handling which application could
not have predicted/captured in rule-construction phase.
Rule-insert errors for bundles are handled too in this pull-request.

Signed-off-by: Aravind Prasad S <raja.avi at gmail.com>
---
 ofproto/ofproto-dpif.c     |   4 +-
 ofproto/ofproto-provider.h |   6 +--
 ofproto/ofproto.c          | 105 ++++++++++++++++++++++++++++++-----------
 3 files changed, 85 insertions(+), 30 deletions(-)

diff --git a/ofproto/ofproto-dpif.c b/ofproto/ofproto-dpif.c
index ad1e8af..d1678ed 100644
--- a/ofproto/ofproto-dpif.c
+++ b/ofproto/ofproto-dpif.c
@@ -4443,7 +4443,7 @@ rule_construct(struct rule *rule_)
     return 0;
 }

-static void
+static enum ofperr
 rule_insert(struct rule *rule_, struct rule *old_rule_, bool forward_counts)
     OVS_REQUIRES(ofproto_mutex)
 {
@@ -4473,6 +4473,8 @@ rule_insert(struct rule *rule_, struct rule *old_rule_, bool forward_counts)
         ovs_mutex_unlock(&rule->stats_mutex);
         ovs_mutex_unlock(&old_rule->stats_mutex);
     }
+
+    return 0;
 }

 static void
diff --git a/ofproto/ofproto-provider.h b/ofproto/ofproto-provider.h
index 2b77b89..3f3d110 100644
--- a/ofproto/ofproto-provider.h
+++ b/ofproto/ofproto-provider.h
@@ -1297,8 +1297,8 @@ struct ofproto_class {
     struct rule *(*rule_alloc)(void);
```

```
      enum ofperr (*rule_construct)(struct rule *rule)
          /* OVS_REQUIRES(ofproto_mutex) */;
 -    void (*rule_insert)(struct rule *rule, struct rule *old_rule,
 -                        bool forward_counts)
 +    enum ofperr (*rule_insert)(struct rule *rule, struct rule *old_rule,
 +                               bool forward_counts)
          /* OVS_REQUIRES(ofproto_mutex) */;
      void (*rule_delete)(struct rule *rule) /* OVS_REQUIRES(ofproto_mutex) */;
      void (*rule_destruct)(struct rule *rule);
 @@ -1952,7 +1952,7 @@ enum ofperr ofproto_flow_mod_learn_start(struct ofproto_flow_mod *ofm)
      OVS_REQUIRES(ofproto_mutex);
  void ofproto_flow_mod_learn_revert(struct ofproto_flow_mod *ofm)
      OVS_REQUIRES(ofproto_mutex);
 -void ofproto_flow_mod_learn_finish(struct ofproto_flow_mod *ofm,
 +enum ofperr ofproto_flow_mod_learn_finish(struct ofproto_flow_mod *ofm,
                                           struct ofproto *orig_ofproto)
      OVS_REQUIRES(ofproto_mutex);
  void ofproto_add_flow(struct ofproto *, const struct match *, int priority,
 diff --git a/ofproto/ofproto.c b/ofproto/ofproto.c
 index f946e27..cb09ee6 100644
 --- a/ofproto/ofproto.c
 +++ b/ofproto/ofproto.c
 @@ -245,10 +245,12 @@ static void replace_rule_revert(struct ofproto *, struct rule *old_rule,
                                  struct rule *new_rule)
      OVS_REQUIRES(ofproto_mutex);

 -static void replace_rule_finish(struct ofproto *, struct ofproto_flow_mod *,
 -                                const struct openflow_mod_requester *,
 -                                struct rule *old_rule, struct rule *new_rule,
 -                                struct ovs_list *dead_cookies)
 +static enum ofperr replace_rule_finish(struct ofproto *,
 +                                       struct ofproto_flow_mod *,
 +                                       const struct openflow_mod_requester *,
 +                                       struct rule *old_rule,
 +                                       struct rule *new_rule,
 +                                       struct ovs_list *dead_cookies)
      OVS_REQUIRES(ofproto_mutex);
  static void delete_flows__(struct rule_collection *,
                             enum ofp_flow_removed_reason,
 @@ -270,7 +272,7 @@ static enum ofperr ofproto_flow_mod_start(struct ofproto *,
  static void ofproto_flow_mod_revert(struct ofproto *,
                                      struct ofproto_flow_mod *)
      OVS_REQUIRES(ofproto_mutex);
 -static void ofproto_flow_mod_finish(struct ofproto *,
 +static enum ofperr ofproto_flow_mod_finish(struct ofproto *,
                                      struct ofproto_flow_mod *,
                                      const struct openflow_mod_requester *)
      OVS_REQUIRES(ofproto_mutex);
 @@ -4855,7 +4857,7 @@ add_flow_revert(struct ofproto *ofproto, struct ofproto_flow_mod *ofm)
  }

  /* To be called after version bump. */
 -static void
 +static enum ofperr
  add_flow_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
                  const struct openflow_mod_requester *req)
      OVS_REQUIRES(ofproto_mutex)
 @@ -4864,8 +4866,14 @@ add_flow_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
          ? rule_collection_rules(&ofm->old_rules)[0] : NULL;
      struct rule *new_rule = rule_collection_rules(&ofm->new_rules)[0];
      struct ovs_list dead_cookies = OVS_LIST_INITIALIZER(&dead_cookies);
 +    enum ofperr error = 0;
 +
 +    error = replace_rule_finish(ofproto, ofm, req, old_rule, new_rule,
 +                                &dead_cookies);
```

```
+    if (error) {
+        return error;
+    }

-    replace_rule_finish(ofproto, ofm, req, old_rule, new_rule, &dead_cookies);
     learned_cookies_flush(ofproto, &dead_cookies);

     if (old_rule) {
@@ -4878,6 +4886,8 @@ add_flow_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
         /* Send Vacancy Events for OF1.4+. */
         send_table_status(ofproto, new_rule->table_id);
     }
+
+    return error;
 }

 /* OFPFC_MODIFY and OFPFC_MODIFY_STRICT. */
@@ -5074,22 +5084,25 @@ ofproto_flow_mod_learn_revert(struct ofproto_flow_mod *ofm)
     ofproto_flow_mod_revert(rule->ofproto, ofm);
 }

-void
+enum ofperr
 ofproto_flow_mod_learn_finish(struct ofproto_flow_mod *ofm,
                               struct ofproto *orig_ofproto)
     OVS_REQUIRES(ofproto_mutex)
 {
     struct rule *rule = rule_collection_rules(&ofm->new_rules)[0];
+    enum ofperr error = 0;

     /* If learning on a different bridge, must bump its version
      * number and flush connmgr afterwards. */
     if (rule->ofproto != orig_ofproto) {
         ofproto_bump_tables_version(rule->ofproto);
     }
-    ofproto_flow_mod_finish(rule->ofproto, ofm, NULL);
+    error = ofproto_flow_mod_finish(rule->ofproto, ofm, NULL);
     if (rule->ofproto != orig_ofproto) {
         ofmonitor_flush(rule->ofproto->connmgr);
     }
+
+    return error;
 }

 /* Refresh 'ofm->temp_rule', for which the caller holds a reference, if already
@@ -5144,7 +5157,7 @@ ofproto_flow_mod_learn(struct ofproto_flow_mod *ofm, bool keep_ref,

             error = ofproto_flow_mod_learn_start(ofm);
             if (!error) {
-                ofproto_flow_mod_learn_finish(ofm, NULL);
+                error = ofproto_flow_mod_learn_finish(ofm, NULL);
             }
         } else {
             static struct vlog_rate_limit rll = VLOG_RATE_LIMIT_INIT(1, 5);
@@ -5244,7 +5257,7 @@ replace_rule_revert(struct ofproto *ofproto,
 }

 /* Adds the 'new_rule', replacing the 'old_rule'. */
-static void
+static enum ofperr
 replace_rule_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
                     const struct openflow_mod_requester *req,
                     struct rule *old_rule, struct rule *new_rule,
@@ -5252,6 +5265,8 @@ replace_rule_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
     OVS_REQUIRES(ofproto_mutex)
```

```
 {
     struct rule *replaced_rule;
+    enum ofperr error = 0;
+    struct oftable *table = &ofproto->tables[new_rule->table_id];

     replaced_rule = (old_rule && old_rule->removed_reason != OFPRR_EVICTION)
         ? old_rule : NULL;
@@ -5261,8 +5276,15 @@ replace_rule_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
      * link the packet and byte counts from the old rule to the new one if
      * 'modify_keep_counts' is 'true'.  The 'replaced_rule' will be deleted
      * right after this call. */
-    ofproto->ofproto_class->rule_insert(new_rule, replaced_rule,
-                                        ofm->modify_keep_counts);
+    error = ofproto->ofproto_class->rule_insert(new_rule, replaced_rule,
+                                                ofm->modify_keep_counts);
+    if (error) {
+        if (classifier_remove(&table->cls, &new_rule->cr)) {
+            ofproto_rule_destroy__(new_rule);
+        }
+        return error;
+    }
+
     learned_cookies_inc(ofproto, rule_get_actions(new_rule));

     if (old_rule) {
@@ -5298,6 +5320,8 @@ replace_rule_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
                             req ? req->request->xid : 0, NULL);
         }
     }
+
+    return error;
 }

 /* ofm->temp_rule is consumed only in the successful case. */
@@ -5448,17 +5472,18 @@ modify_flows_revert(struct ofproto *ofproto, struct ofproto_flow_mod *ofm)
     }
 }

-static void
+static enum ofperr
 modify_flows_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
                     const struct openflow_mod_requester *req)
     OVS_REQUIRES(ofproto_mutex)
 {
     struct rule_collection *old_rules = &ofm->old_rules;
     struct rule_collection *new_rules = &ofm->new_rules;
+    enum ofperr error = 0;

     if (rule_collection_n(old_rules) == 0
         && rule_collection_n(new_rules) == 1) {
-        add_flow_finish(ofproto, ofm, req);
+        error = add_flow_finish(ofproto, ofm, req);
     } else if (rule_collection_n(old_rules) > 0) {
         struct ovs_list dead_cookies = OVS_LIST_INITIALIZER(&dead_cookies);

@@ -5467,12 +5492,17 @@ modify_flows_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
         struct rule *old_rule, *new_rule;
         RULE_COLLECTIONS_FOR_EACH (old_rule, new_rule, old_rules, new_rules) {
-            replace_rule_finish(ofproto, ofm, req, old_rule, new_rule,
-                                &dead_cookies);
+            error = replace_rule_finish(ofproto, ofm, req, old_rule, new_rule,
+                                        &dead_cookies);
+            if (error) {
+                return error;
```

```
+            }
         }
         learned_cookies_flush(ofproto, &dead_cookies);
         remove_rules_postponed(old_rules);
     }
+
+    return error;
 }

 static enum ofperr
@@ -5838,7 +5868,7 @@ handle_flow_mod__(struct ofproto *ofproto, const struct ofputil_flow_mod *fm,
     error = ofproto_flow_mod_start(ofproto, &ofm);
     if (!error) {
         ofproto_bump_tables_version(ofproto);
-        ofproto_flow_mod_finish(ofproto, &ofm, req);
+        error = ofproto_flow_mod_finish(ofproto, &ofm, req);
         ofmonitor_flush(ofproto->connmgr);
     }
     ovs_mutex_unlock(&ofproto_mutex);
@@ -7668,19 +7698,21 @@ ofproto_flow_mod_revert(struct ofproto *ofproto, struct ofproto_flow_mod
*ofm)
     rule_collection_destroy(&ofm->new_rules);
 }

-static void
+static enum ofperr
 ofproto_flow_mod_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
                         const struct openflow_mod_requester *req)
     OVS_REQUIRES(ofproto_mutex)
 {
+    enum ofperr error = 0;
+
     switch (ofm->command) {
     case OFPFC_ADD:
-        add_flow_finish(ofproto, ofm, req);
+        error = add_flow_finish(ofproto, ofm, req);
         break;

     case OFPFC_MODIFY:
     case OFPFC_MODIFY_STRICT:
-        modify_flows_finish(ofproto, ofm, req);
+        error = modify_flows_finish(ofproto, ofm, req);
         break;

     case OFPFC_DELETE:
@@ -7698,6 +7730,8 @@ ofproto_flow_mod_finish(struct ofproto *ofproto, struct ofproto_flow_mod *ofm,
     if (req) {
         ofconn_report_flow_mod(req->ofconn, ofm->command);
     }
+
+    return error;
 }

 /* Commit phases (all while locking ofproto_mutex):
@@ -7781,10 +7815,8 @@ do_bundle_commit(struct ofconn *ofconn, uint32_t id, uint16_t flags)

         if (error) {
             /* Send error referring to the original message. */
-            if (error) {
-                ofconn_send_error(ofconn, be->msg, error);
-                error = OFPERR_OFPBFC_MSG_FAILED;
-            }
+            ofconn_send_error(ofconn, be->msg, error);
+            error = OFPERR_OFPBFC_MSG_FAILED;
```

```
             /* 2. Revert.  Undo all the changes made above. */
             LIST_FOR_EACH_REVERSE_CONTINUE(be, node, &bundle->msg_list) {
@@ -7827,13 +7859,34 @@ do_bundle_commit(struct ofconn *ofconn, uint32_t id, uint16_t flags)
                     struct openflow_mod_requester req = { ofconn, be->msg };

                     if (be->type == OFPTYPE_FLOW_MOD) {
-                        ofproto_flow_mod_finish(ofproto, &be->ofm, &req);
+                        error = ofproto_flow_mod_finish(ofproto, &be->ofm,
+                                                        &req);
                     } else if (be->type == OFPTYPE_GROUP_MOD) {
                         ofproto_group_mod_finish(ofproto, &be->ogm, &req);
                     } else if (be->type == OFPTYPE_PACKET_OUT) {
                         ofproto_packet_out_finish(ofproto, &be->opo);
                     }
                 }
+                if (error) {
+                    break;
+                }
             }
+            if (error) {
+                /* Send error referring to the original message. */
+                ofconn_send_error(ofconn, be->msg, error);
+                error = OFPERR_OFPBFC_MSG_FAILED;
+
+                /* Revert.  Undo all the changes made above. */
+                LIST_FOR_EACH_REVERSE_CONTINUE (be, node, &bundle->msg_list) {
+                    if (be->type == OFPTYPE_FLOW_MOD) {
+                        ofproto_flow_mod_revert(ofproto, &be->ofm);
+                    } else if (be->type == OFPTYPE_GROUP_MOD) {
+                        ofproto_group_mod_revert(ofproto, &be->ogm);
+                    } else if (be->type == OFPTYPE_PACKET_OUT) {
+                        ofproto_packet_out_revert(ofproto, &be->opo);
+                    }
+                    /* Nothing needs to be reverted for a port mod. */
+                }
+            }
         }

--
1.9.1
```

- Previous message: [ovs-dev] [PATCH v3 1/3] rhel: rename openvswitch kmod rhel6 spec file
- Next message: [ovs-dev] [PATCH] ofproto: Return error codes for Rule insertions
- **Messages sorted by:** [date] [thread] [subject] [author]

More information about the dev mailing list