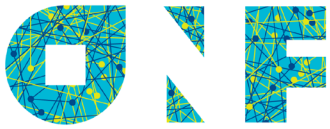

OPEN NETWORKING
FOUNDATION

Flow entry eviction Extension

Version 0.1
June 24, 2013

ONF TS-011

ONF Document Type: OpenFlow Spec
ONF Document Name: openflow-switch-extension-ext192-eviction

Disclaimer

Contents

1 Introduction **2**

2 How it works **2**

3 Flow entry eviction Experimenter ID **2**

4 Flow entry eviction messages **3**

5 Flow entry eviction instructions **4**

1 Introduction

This document describes an ONF extension for OpenFlow version 1.3.X that evict flow entries from flow tables when flow tables get full. Flow entry of lower importance are removed to make space for new flow entries of higher importance.

2 How it works

This extension add a third way by which flow entries are removed from flow tables. Flow entries are removed from flow tables in three ways, either at the request of the controller, via the switch flow expiry mechanism, or via the optional switch eviction mechanism described in this extension.

Flow entries may be **evicted** from flow tables when the switch needs to reclaim resources. Flow entry eviction occurs only on flow tables where it is explicitly enabled (see below). Flow entry eviction is an optional feature, and the mechanism used to select which flow entries to evict is switch defined and may depend on flow entry parameters, resource mappings in the switch and other internal switch constraints.

When a flow entry is evicted, the switch must check the flow entry's `OFPFF_SEND_FLOW_REM` flag. If this flag is set, the switch must send a flow removed message to the controller with reason `OFPRR_DELETE`.

3 Flow entry eviction Experimenter ID

The Experimenter ID of this extension is:

`ONF_EXPERIMENTER_ID` = 0x4F4E4600

4 Flow entry eviction messages

The following message types are defined by this extension.

```
/* Message types */
enum onf_exp_type {
  ONF_ET_SET_EVICTION         = 1925, /* Set eviction - Controller/switch msg */
  ONF_ET_GET_EVICTION_REQUEST = 1926, /* Get eviction - Controller/switch msg */
  ONF_ET_GET_EVICTION_REPLY   = 1927, /* Get eviction - Controller/switch msg */
};
```

The `ONF_ET_SET_EVICTION` message is used by the controller to configure eviction on a flow table, and uses the following message structure :

```
/* Set eviction message. */
struct onf_message_set_eviction {
  struct ofp_header     header;
  uint32_t              experimenter;      /* ONF_EXPERIMENTER_ID. */
  uint32_t              exp_type;          /* ONF_ET_SET_EVICTION. */
  uint8_t table_id;       /* ID of the table, OFPTT_ALL indicates all tables */
  uint8_t eviction_enable;                 /* Enable eviction. */
  uint8_t pad2[6];                         /* Pad to 64 bits. */
};
OFP_ASSERT(sizeof(struct onf_message_set_eviction) == sizeof(struct ofp_experimenter_header) + 8);
```

The `table_id` field is the table which threshold should be changed.

The field `eviction_enable` configure eviction on the flow table. If this is set to zero, eviction is disabled on the flow table and no flow entry can be evicted from the flow table. If it is set to non-zero, eviction is enable and the flow table may evict flow entries when it gets full.

The `ONF_ET_GET_EVICTION_REQUEST` message is used by the controller to request the current eviction configuration from a flow table, and uses the following message structure :

```
/* Get eviction request message. */
struct onf_message_get_eviction_request {
  struct ofp_header     header;
  uint32_t              experimenter;      /* ONF_EXPERIMENTER_ID. */
  uint32_t              exp_type;          /* ONF_ET_GET_EVICTION_REQUEST. */
  uint8_t table_id;                        /* ID of the table. */
  uint8_t pad[7];                          /* Pad to 64 bits. */
};
OFP_ASSERT(sizeof(struct onf_message_get_eviction_request) == sizeof(struct ofp_experimenter_header) + 8);
```

The `table_id` field is the table which threshold are requested.

The `ONF_ET_GET_EVICTION_REPLY` message is used by the switch to reply to a `ONF_ET_GET_EVICTION_REQUEST` message from the controller and report the current eviction configuration of a flow table. It uses the following message structure :


```
/* Get eviction reply message. */
struct onf_message_get_eviction_reply {
  struct ofp_header     header;
  uint32_t              experimenter;      /* ONF_EXPERIMENTER_ID. */
  uint32_t              exp_type;          /* ONF_ET_GET_EVICTION_REPLY. */
  uint8_t table_id;                        /* ID of the table. */
  uint8_t eviction_enable;                 /* Eviction enabled. */
  uint8_t pad2[6];                         /* Pad to 64 bits */
};
OFP_ASSERT(sizeof(struct onf_message_get_eviction_reply) == sizeof(struct ofp_experimenter_header) + 8);
```

The `table_id` field is the table which threshold are reported.

The field `eviction_enable` is the current eviction configuration of the table.

5 Flow entry eviction instructions

The following instructions are defined by this extension.

```
/* Instruction types */
enum onf_instruction_exp_type {
  ONFIST_ET_EVICTION_IMPORTANCE = 1920,     /* Eviction importance. */
};
```

The `ONFIST_ET_EVICTION_IMPORTANCE` instruction type uses the following instruction structure :

```
/* Instruction structure for ONFIST_ET_EVICTION_IMPORTANCE. */
struct onf_instruction_eviction_importance {
  uint16_t      type;           /* OFPIT_EXPERIMENTER. */
  uint16_t      len;            /* Length is 16. */
  uint32_t      experimenter;   /* ONF_EXPERIMENTER_ID. */
  uint16_t      exp_type;       /* ONFIST_ET_EVICTION_IMPORTANCE. */
  uint16_t      importance;     /* Eviction importance. */
  uint8_t       pad[4];
};
OFP_ASSERT(sizeof(struct onf_instruction_eviction_importance) == 16);
```

The `type` field must be set to `OFPIT_EXPERIMENTER`.

The `experimenter` field is the Experimenter ID (see 3).

The `exp_type` field is set to `ONFIST_ET_EVICTION_IMPORTANCE`.

The `importance` field is the importance of the flow entry. This field may be optionally used by the flow entry eviction mechanism (see above). The eviction mechanism may evict flow entries with lower importance first and try to preserve flow entries with the highest importance. The eviction mechanism is switch defined, therefore it is not possible to predict in which order flow entries will get evicted. Even if the eviction mechanism uses the `importance` field, flow entries of highest importance may be evicted before flow entries of lower importance due to various internal constraint on the resources.

The instruction ordering is not modified, as this instruction is not executed for each packet, and does not modify pipeline processing. This instruction is only a hint for the flow entry eviction process.