OpenState Specification

March 16, 2015

Contents

1 Introduction

This document is intended to be an extension to the OpenFlow specification v1.x (???)

A packet entering an OpenFlow switch is processed through a set of linked flow tables that provide matching, forwarding, and packet modification.

We indicate with the term stateless block the processing operated by a single flow table. Conversely, we define as stateful block a logical block comprising a state table followed by a flw table, and implementing the OpenState abstraction.

A new switch capability has been defined in order to support all the OpenState functionalities, namely flow-states and global-states. By default all the flow tables in the switch are intended as stateless blocks, the controller can then enable stateful processing for one or more blocks by sending a special control message to the switch and by configuring the key extractors (lookup-scope and update-scope) associated with the state table. A new state modify message called `OFP_STATE_MOD` has been defined to allow the controller to configure the state entries and key extractors. Finally two new actions `OFPAT_SET_STATE` and `OFPAT_SET_FLAGS` have been defined in order to respectively implement and configure the XFSM state transitions in the flow table and set the global states.

2 Switch and table configuration

2.1 Openstate capability

A new OFPC_OPENSTATE capability has been introduced. The basic flow table data structure has been extended with a support data structure implementing the state table (a hash map indexed by the flow key), the lookup and update key extractor (two ordered lists of flow match TLV field indexes) and the global states. By retrieving all the capabilities from the switch, the controller is able to properly configure the switch. If a switch is OpenState aware, OFPC_OPENSTATE capability is defined enabling the controller to configure the statefulness of each stage by sending table feature message [2.2].

2.2 Stateful stage configuration

If OFPTC_TABLE_STATEFUL flag is set, right after the packet headers are parsed, the flow state is retrieved and written in the state field, otherwise the packet directly jumps to the flow table.

3 Flow states

A flow state is an information that persists for lifetime of a flow and it is shared among all the flow's packets. In case of table-miss (the key is not matched) then a DEFAULT state will be appended to the packet headers. If the header fields specified by the lookup-scope are not found (e.g. extracting the IP source address when the Ethernet type is not IP) or are only partially found, a special state value NULL is returned and no information about state is appended to the packet.

3.1 Set state action

The OFPAT_SET_STATE action allows to set flow states in a particulare stage of the pipeline.

Structures in ofsoftswitch13:

```
OFPAT_SET_STATE = 28

/* Action structure for OFPAT_SET_STATE */
struct ofp_action_set_state {
    uint16_t type;  /* OFPAT_SET_STATE */
    uint16_t len;   /* Length is 8. */
    uint32_t state; /* State instance. */
    uint8_t stage_id; /*Stage destination*/
    uint8_t pad[7];   /* Align to 64-bits. */
};
OFP_ASSERT(sizeof(struct ofp_action_set_state) == 16);
```

In the flow modification message, the controller defines the action with the following parameters:

- state: the new flow state
- stage_id: it identifies the target stage of the set state action.

Atomicity

The usage of a set-state action not always ensures the state transactions atomicity, because the Openflow's default actions are carried on at the end of the pipeline. Not ensuring state transactions atomicity can bring to consistency issues for some applications. The only way to ensure that is to include the set-state action into the apply-action instruction (instead of the write-actions instruction) in order to execute the action in that specific stage of the pipeline.

Checks and errors

- Set state action must be performed onto a stateful stage. This check is performed at action execution time (maybe the flow-mod message with a set-state action is received by the switch before configuring a stage as stateful. The important thing is that the stage is stateful at action execution time). Should we inform the controller about this error?
- Set state action must be performed onto a stage with stage_id less or equal than the number of pipeline's tables. This check is performed at msg unpack time (the number of table is fixed, so installing a flow with a wrong action does not make sense...) =¿ **TODO!!!**

Set state priority

The new OFPAT_SET_STATE action has been set with an higher priority with respect to the OFPAT_SET_FIELD action. Given an action set containing both a set field and a set state action, with this setting it is avoided that the set field modifies header fields used by the set state's update scope before the set state execution.

3.2 State match field

The OXM_OF_STATE field is used to match the state on the packet header. It is a 32 bit field. =¿ **Does OXM_OF_STATE need to have a mask?**

Definition in ofsoftswitch13 (oxm-match.h):

```
/* Flow State */
#define OXM_OF_STATE OXM_HEADER     (0x8000, 41, 4)
```

3.3 State modification messages

There are four different `OFPT_STATE_MOD` messages:

- Set lookup extractor command
- Set update extractor command
- Add flow state command
- Delete flow state command

Structures in ofsoftswitch13 (openflow.h)

```
OFPT_STATE_MOD = 30, /* Controller/switch message */

/*OFPT_STATE_MOD*/

#define OFPSC_MAX_FIELD_COUNT 6 /* number of fields composing the key */
#define OFPSC_MAX_KEY_LEN 48 /* number of bytes composing the key */

struct ofp_state_mod {
    struct ofp_header header;
    uint64_t cookie;
    uint64_t cookie_mask;
    uint8_t table_id;
    uint8_t command;
    uint8_t payload[];
};

struct ofp_state_entry {
    uint32_t key_len;
    uint32_t state;
    uint8_t key[OFPSC_MAX_KEY_LEN];
};

struct ofp_extraction {
    uint32_t field_count;
    uint32_t fields[OFPSC_MAX_FIELD_COUNT];
};

enum ofp_state_mod_command {
OFPSC_SET_L_EXTRACTOR = 0,
OFPSC_SET_U_EXTRACTOR,
OFPSC_ADD_FLOW_STATE,
OFPSC_DEL_FLOW_STATE
 };
```

=¿ who decides OFPSC_MAX_FIELD_COUNT and OFPSC_MAX_KEY_LEN? Should they be configurable?

3.3.1 Set lookup extractor command

The `OFPSC_SET_L_EXTRACTOR` command gives the opportunity to set the lookup scope used in the state table. The controller sends a `OFPT_STATE_MOD` message with the following parameters:

- command = 0 (`OFPSC_SET_L_EXTRACTOR`)
- field_count = number of specified field
- fields = key's fields
- table_id = ID of the stage to be setup

3.3.2 Set update extractor command

The OFPSC_SET_U_EXTRACTOR command gives the opportunity to set the update scope used in the state table. The controller sends a `OFPT_STATE_MOD` message with the following parameters:

- command = 1 (OFPSC_SET_U_EXTRACTOR)
- field_count = number of specified field
- fields = key's fields
- table_id = ID of the stage to be setup

Checks

- field_count must be consistent with the number of fields provided in fields, otherwise an error (OFPET_BAD_ACTION, OFPBAC_BAD_LEN) is returned at msg unpack time
- OpenState says "lookup-scope" and update-scope must provide same length keys". This check must be performed switch side only (**=¿ TODO**: when we receive a set extractor message we check if we have already set the other extractor with the same length. The check is performed at msg unpack time or at msg execution time).

3.3.3 Add flow state command

The OFPSC_ADD_FLOW_STATE command gives the opportunity to insert (or to overwrite) an entry in the state table. The controller sends a `OFPT_STATE_MOD` message with the following parameters:

- command = 2 (OFPSC_ADD_FLOW_STATE)
- key_count = it is the key size in byte
- state = it is the state to insert in the state table
- keys = key splitted in bytes (e.g: ip 10.0.0.1 is stored as [10,0,0,1])
- table_id = ID of the stage to be modified

3.3.4 Delete flow state command

With the OFPSC_DEL_FLOW_STATE command is possible to delete a state table's entry. The controller sends a `OFPT_STATE_MOD` message with the following parameters:

- command = 3 (OFPSC_DEL_FLOW_STATE)
- key_count = it is the key size in byte
- state = ANY
- keys = key splitted in bytes (e.g: ip 10.0.0.1 is stored as [10,0,0,1])
- table_id = ID of the stage to be modified

The state value is not taken in consideration because the state table simply uses as key the lookup-scope's fields to delete the entry with key keys.

Checks

- key_count must be consistent with the number of fields provided in key, otherwise an error (OFPET_BAD_ACTION, OFPBAC_BAD_LEN) is returned at msg unpack time.
- Set state message must be executed onto a stage with stage_id less or equal than the number of pipeline's tables, otherwise the switch returns an error (OFPET_BAD_REQUEST, OFPBRC_BAD_TABLE_ID). This check is performed at msg unpack time (the number of table is fixed).
- key_count must be consistent with the number of fields of the update-scope (defined before with a OFPSC_SET_U_EXTRACTOR)

4 Global states

Global states (or flags) are defined at datapath level and are not related to a single flow of a particular stage. Flags are 32 boolean registers and are grouped in an uint32_t variable global_states in the datapath struct.

4.1 Flags match field

If a switch supports OpenState (flag OFPTC_TABLE_STATEFUL set), right after the packet headers are parsed, the global states are retrieved and written in the flags field. OXM_OF_FLAGS is a field with mask, so it is possible to match it either exactly or with wildcards. A 0 bit in the mask means i-th flags value is "do not care", while a 1 bit value means "exact match".

Definition in ofsoftswitch13 (oxm-match.h file)

```
/* Global States */
#define OXM_OF_FLAGS OXM_HEADER     (0x8000, 40, 4)
#define OXM_OF_FLAGS_W OXM_HEADER_W (0x8000, 40, 4)
```

Example match:

```
flags=(4,5)
```

This command allows to match over ********************************1*0 flags configuration (4 in binary is 100 and the mask 5 is 101 that is exact match on LSB 1 (0 value) and LSB 3 (1 value) and "don't care" over all the other flags. In order to perform an exact match on flags value no mask is required.
Example match:

```
flags=4
```

NB: this match is very differend from the previous one. With this command we are matching over 00000000000000000000000000000100 flags configuration, so it is an exact match.

4.2 Set flag action

The OFPAT_SET_FLAG action is used to set flags' value. In the flow modification message, the controller defines the action with the following parameters:

- flag = flags value
- mask = flags mask

Structures in ofsoftswitch13:

```
OFPAT_SET_FLAG = 29,   /* Set a single flag value of the global state */

/* Action structure for OFPAT_SET_FLAG */
struct ofp_action_set_flag {
    uint16_t type; /* OFPAT_SET_FLAG */
    uint16_t len;  /* Length is 8. */
    uint32_t value; /* flag value */
    uint32_t mask;    /*flag mask*/
    uint8_t pad[4];   /* Align to 64-bits. */
};
OFP_ASSERT(sizeof(struct ofp_action_set_flag) == 16);
```

4.3 Flag modification messages

The OFPT_FLAG_MOD message allow the controller to modify global states value.

Structures in ofsoftswitch13:

```
OFPT_FLAG_MOD = 31,  /* Controller/switch message */

struct ofp_flag_mod {
    struct ofp_header header;
    uint32_t flag;
    uint32_t flag_mask;
    uint8_t command;
    uint8_t pad[7];                  /* Pad to 64 bits. */
};

enum ofp_flag_mod_command {
    OFPSC_MODIFY_FLAGS = 0,
    OFPSC_RESET_FLAGS
};
```

There are two message types.

4.3.1 Modify flags command

The OFPSC_MODIFY_FLAGS allows to modify global states values. The controller sends a OFPSC_MODIFY_FLAGS message with the following parameters:

- flag = flags value
- flag_mask = mask value
- command = 0 (OFPSC_MODIFY_FLAGS)

4.3.2 Reset flags command

The OFPSC_MODIFY_FLAGS allows to reset global states to the default value (OFP_GLOBAL_STATE in openflow.h) The controller sends a OFPSC_MODIFY_FLAGS message with the following parameters:

- flag = ANY
- flag_mask = ANY
- command = 1 (OFPSC_RESET_FLAGS)

Authors

...