OPEN NETWORKING FOUNDATION

OpenFlow Time Extension

Version 0.7

December 23, 2014

Disclaimer

Contents

1 Introduction

This document describes an ONF extension for OpenFlow version 1.3.X that enables time-based updates. This new capability is added as an extension to the Bundle feature, which is defined in EXT-230 [1].

2 How it works

As specified in EXT-230, a bundle is a sequence of (one or more) OpenFlow modification requests from the controller that is applied as a single OpenFlow operation. The controller uses a commit message to apply the set of requests in the bundle. Consequently, the switch applies all messages in the bundle as a single operation or returns an error.

This extension defines *scheduled bundles*; a bundle commit request may include an *execution time*, specifying *when* the bundle should be committed. A switch that receives a scheduled bundle, commits the bundle as close as possible to the execution time that was specified in the commit message.

This document also defines the bundle features message, allowing the controller to retrieve information about the switch's bundle support, and specifically about its scheduled bundle support.

2.1 The Scheduled Bundle Procedure

The extension defined in this document allows a bundle operation to be invoked at a scheduled time that is determined by the controller.

The time-based bundle procedure is illustrated in Figure 1:

1. The controller starts the bundle procedure by sending an `ONF_BCT_OPEN_REQUEST`, and receives a reply from the switch.
2. The controller then sends a set of N `ONF_ET_BUNDLE_ADD_MESSAGE` messages, for some $N \geq 1$.
3. The controller may then send an `ONF_BCT_CLOSE_REQUEST`. The close request is optional, and thus the controller may skip this step.
4. The controller sends an `ONF_BCT_COMMIT_REQUEST`. The `ONF_BCT_COMMIT_REQUEST` includes two time-related fields: the time flag and optionally the time property. When the time flag is set, it indicates that this is a *scheduled commit*. A scheduled commit request includes the time property field, which contains the scheduled time at which the switch is expected to apply the bundle.
5. After receiving the commit message, the switch applies the bundle at the scheduled time, T_s, and sends a `ONF_BCT_COMMIT_REPLY` to the controller.

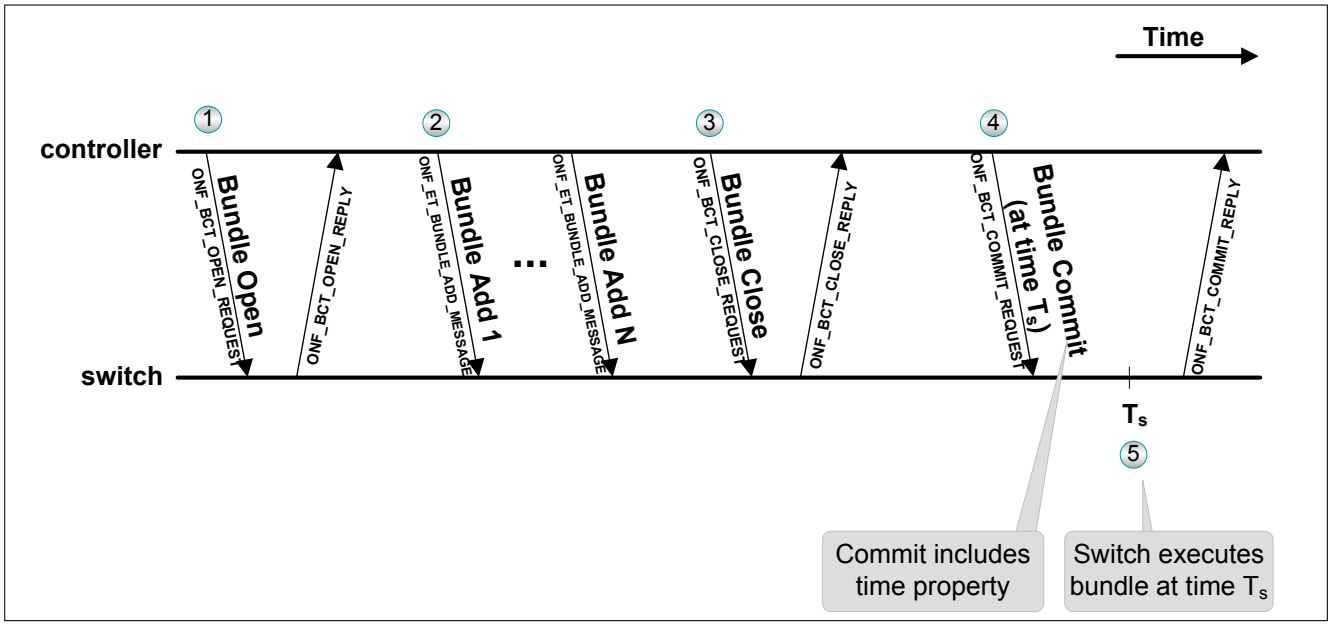

Figure 1: Scheduled Bundle Procedure

2.2 Discarding Scheduled Bundles

The controller may cancel a scheduled commit by sending an `ONF_ET_BUNDLE_CONTROL` message with type `ONF_BCT_DISCARD_REQUEST`. An example is shown in Figure 2; if the switch is not able to schedule the operation after receiving the commit message, it responds to the controller with an error message

(see 7). This indication may be used for implementing a coordinated update where either all the switches successfully schedule the operation, or the bundle is discarded; when a controller receives a scheduling error message from one of the switches it can send a discard message (step 5' in in Figure 2) to other switches that need to commit a bundle at the same time, and abort the bundle.

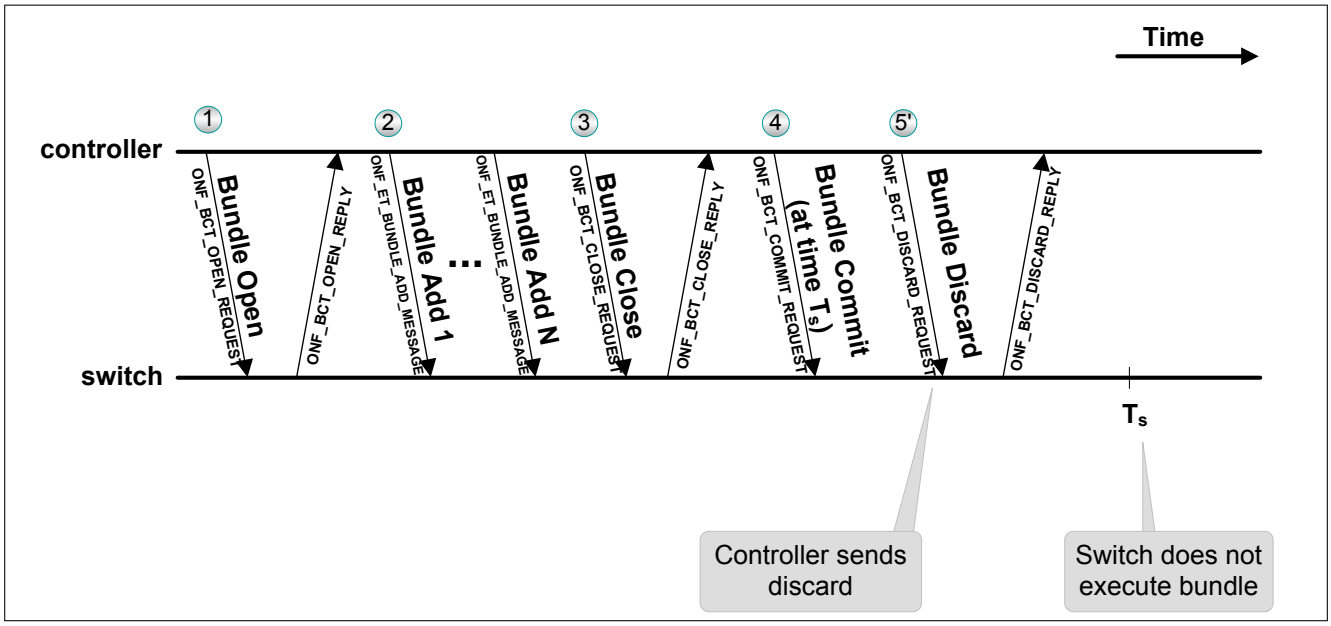

Figure 2: Discarding a Scheduled Commit

2.3 Timekeeping and Synchronization

Every switch that supports scheduled bundles must maintain a clock. It is assumed that clocks are synchronized by a method that is outside the scope of this document, e.g., the Network Time Protocol (NTP) or the Precision Time Protocol (PTP).

Two factors affect how accurately a switch can commit a scheduled bundle; one factor is the accuracy of the clock synchronization method used to synchronize the switches' clocks, and the second factor is the switch's ability to execute real-time operations, which greatly depends on how it is implemented.

This document does not define any requirements pertaining to the degree of accuracy of performing scheduled operations. However, every switch that supports the time extension is able to report its estimated scheduling accuracy to the controller. The controller can retrieve this information from the switch using the bundle features message, defined in Section 6.1.

Since a switch does not perform configuration changes instantaneously, the processing time of required operations should not be overlooked; in the context of the extension described in this paper the scheduled time and execution time always refer to the start time of the relevant operation.

2.4 Scheduling Tolerance

When a switch receives a scheduled commit message, it verifies that the scheduled time, T_s, is not too far in the past or in the future. As illustrated in Figure 3, the switch verifies that T_s is within the *scheduling tolerance* range.

The lower bound on T_s verifies the freshness of the packet so as to avoid acting upon old and possibly irrelevant messages. Similarly, the upper bound on T_s guarantees that the switch does not take a long-term commitment to execute an action that may become obsolete by the time it is scheduled to be invoked.

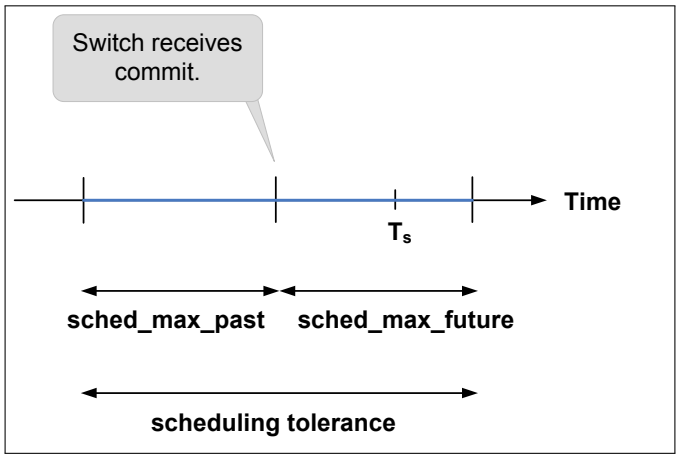

Figure 3: Scheduling Tolerance

The scheduling tolerance is determined by two parameters, `sched_max_future` and `sched_max_past`. The default value of these two parameters is 1 second. The controller may set these fields to a different value using the bundle features request, as described in Section 6.1.

If the scheduled time, T_s is within the scheduling tolerance range, the scheduled commit is performed; if T_s occurs in the past and within the scheduling tolerance, the switch applies the bundle as soon as possible. If T_s is a future time, the switch applies the bundle at T_s. If T_s is not within the scheduling tolerance range, the switch responds to the controller with an error message.

3 Copy-Field Experimenter ID

The Experimenter ID of this extension is:

`ONF_EXPERIMENTER_ID` = 0x4F4E4600

4 Time Format

This extension defines a time format that is used in various other structures. The time format uses the following structure:


```
/* Time format */
struct onf_time {
    uint64_t seconds;
    uint32_t nanoseconds;
    uint8_t pad[4];
};
OFP_ASSERT(sizeof(struct onf_time) == 16);
```

The time format defined in this extension is based on the one defined in [2]. It consists of two sub-fields; a `seconds` field, representing the integer portion of time in seconds[1], and a `nanoseconds` field, representing the fractional portion of time in nanoseconds, i.e., $0 \leq nanoseconds \leq (10^9 - 1)$.

As defined in [2], time is measured according to the International Atomic Time (TAI) timescale. The epoch is defined as 1 January 1970 00:00:00 TAI.

5 Time Bundle property

This extension defines the following bundle property type:

```
/* Bundle property types. */
enum onf_bundle_prop_exp_type {
  ONFBUP_ET_TIME = 3400,         /* Time bundle property. */
};
```

The action `ONFBUP_ET_TIME` uses the following structure:

```
/* Bundle property structure for ONF_ET_BPROP_TIME. */
struct onf_bundle_prop_time {
    uint16_t         type;          /* ONF_ET_BPT_EXPERIMENTER. */
    uint16_t         length;        /* Length in bytes = 32. */
    uint32_t         experimenter;  /* ONF_EXPERIMENTER_ID. */
    uint32_t         exp_type;      /* ONFBUP_ET_TIME. */
    uint8_t          pad[4];        /* Align to 64 bits. */

    struct onf_time scheduled_time; /* The scheduled time at which the switch
                                       should apply the bundle. */
};
OFP_ASSERT(sizeof(struct onf_bundle_prop_time) == 32);
```

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

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

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

The `scheduled_time` field is the time at which the bundle should be executed.

[1] The seconds field in IEEE 1588 is 48 bits long. The seconds field used in this extension is a 64-bit field, but it has the same semantics as the seconds field in the IEEE 1588 time format.

The time extension allows bundle commit messages to include a time property, defining when the bundle should be executed. If a ONF_ET_BUNDLE_CONTROL message of type ONF_BCT_COMMIT_REQUEST contains a ONFBUP_ET_TIME property, the bundle must be scheduled to be applied at the time specified in the time property. This property should not be included in other bundle messages.

6 Bundle Feature multipart

This extension defines the following multipart type:

```
/* Bundle property types. */
enum onfmul_exp_type {
  ONFMUL_ET_BUNDLE_FEAT = 3400,   /* Bundle feature request/reply multipart. */
};
```

6.1 Bundle Features request

The multipart request ONFMUL_ET_BUNDLE_FEAT uses the following structure:

```
/* Body for ofp_multipart_request for ONFMUL_ET_BUNDLE_FEAT.
 * Multipart type is OFPMP_EXPERIMENTER. */
struct onf_multipart_bundle_feat_request {
    uint32_t experimenter;    /* ONF_EXPERIMENTER_ID. */
    uint32_t exp_type;        /* ONFMUL_ET_BUNDLE_FEAT. */
    uint32_t feature_request_flags; /* Bitmap of "onf_bundle_feature_flags". */
    uint8_t pad[4];

    /* Bundle features property list - 0 or more. */
    struct onf_bundle_features_prop_header properties[0];
};
OFP_ASSERT(sizeof(struct onf_multipart_bundle_feat_request) == 16);
```

The type of the multipart request must be set to OFPMP_EXPERIMENTER.

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

The exp_type field is set to ONFMUL_ET_BUNDLE_FEAT.

The feature_request_flags field is a bitmap that defines which properties are included int eh request. It may include a combination of the following flags:

```
/* Flags used in a bundle features request. */
enum onf_bundle_feature_flags {
    ONF_BF_TIMESTAMP = 1 << 0,      /* When enabled, the current request
                                     * includes a timestamp, using
                                     * the time property. */
    ONF_BF_TIME_SET_SCHED = 1 << 1, /* When enabled, the current request
                                     * includes the sched_max_future
                                     * and sched_max_past parameters, using
                                     * the time property. */
};
```


If at least one of the flags ONF_BF_TIMESTAMP or ONF_BF_TIME_SET_SCHED is set, the bundle features request includes a time property.

The properties field is a list of bundle feature properties. The list of properties defined by this extension is specified in Section 6.3.

6.2 Bundle Features reply

When the switch successfully process a bundle features request, it sends a bundle feature reply to the controller. The multipart reply ONFMUL_ET_BUNDLE_FEAT uses the following structure:

```
/* Body for ofp_multipart_reply for ONFMUL_ET_BUNDLE_FEAT.
 * Multipart type is OFPMP_EXPERIMENTER. */
struct onf_multipart_bundle_feat_reply {
    uint32_t experimenter;    /* ONF_EXPERIMENTER_ID. */
    uint32_t exp_type;        /* ONFMUL_ET_BUNDLE_FEAT. */
    uint16_t capabilities;    /* Bitmap of "onf_bundle_flags". */
    uint8_t pad[6];

    /* Bundle features property list - 0 or more. */
    struct onf_bundle_features_prop_header properties[0];
};
OFP_ASSERT(sizeof(struct onf_multipart_bundle_feat_reply) == 16);
```

The type of the multipart request must be set to OFPMP_EXPERIMENTER.

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

The exp_type field is set to ONFMUL_ET_BUNDLE_FEAT.

The capabilities field is a bitmap that defines which capabilities are supported by bundles. It may include a combination of the flags defined by onf_bundle_flags.

The properties field is a list of bundle feature properties. The list of properties defined by this extension is specified in Section 6.3.

6.3 Bundle Features Properties

The list of bundle property types that are currently defined are:

```
/* Bundle features property types. */
enum onf_bundle_features_prop_type {
    OFPTMPBF_TIME_CAPABILITY = 0x1, /* Time feature property. */
    OFPTMPBF_EXPERIMENTER = 0xFFFF, /* Experimenter property. */
};
```

A property definition contains the property type, length, and any associated data:


```
/* Common header for all bundle feature Properties */
struct onf_bundle_features_prop_header {
    uint16_t type;   /* One of OFPTMPBF_*. */
    uint16_t length; /* Length in bytes of this property. */
};
OFP_ASSERT(sizeof(struct onf_bundle_features_prop_header) == 4);
```

6.4 Bundle Features time property

The `OFPTMPBF_TIME_CAPABILITY` property uses the following structure and fields:

```
struct onf_bundle_features_prop_time {
    uint16_t type;   /* OFPTMPBF_TIME_CAPABILITY. */
    uint16_t length; /* Length in bytes of this property. */
    uint8_t pad[4];

    struct onf_time sched_accuracy;   /* The scheduling accuracy, i.e., how
                                       * accurately the switch can perform
                                       * a scheduled commit. This field is used
                                       * only in bundle features replies, and is
                                       * ignored in bundle features requests. */
    struct onf_time sched_max_future; /* The maximal difference between the
                                       * scheduling time and the current
                                       * time. */
    struct onf_time sched_max_past;   /* If the scheduling time occurs in the
                                       * past, defines the maximal difference
                                       * between the current time and the
                                       * scheduling time. */
    struct onf_time timestamp;        /* Indicates the time during the
                                       * transmission of this message. */
};
OFP_ASSERT(sizeof(struct onf_bundle_features_prop_time) == 72);
```

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

In a bundle features request, the fields of the property are defined as such:

- The `sched_accuracy` field is relevant only to bundle features replies, and the switch must ignore this field in a bundle features request.
- The `sched_max_future` and `sched_max_past` fields define the range of acceptable schedules. A switch that receives a bundle features request with `ONF_BF_TIME_SET_SCHED` set should attempt to change its scheduling tolerance values according to the `sched_max_future` and `sched_max_past` values from the time property. If the switch does not successfully update its scheduling tolerance values, it replies with an error message.
- The `timestamp` fiels indicates the controller's time during the transmission of this message. A switch that receives a bundle features request with `ONF_BF_TIMESTAMP` set, may use the received timestamp to roughly estimate the offset between its clock and the controller's clock.

In a bundle features reply, the fields of the property are defined as such:

- The `sched_accuracy` field indicates the estimated scheduling accuracy of the switch. For example, if the value of `sched_accuracy` is 1000000 nanoseconds (1 ms), it means that when the switch receives a bundle commit scheduled to time T_s, the commit will in practice be invoked at $T_s \pm 1\ ms$. The factors that affect the scheduling accuracy are discussed in Section 2.3.
- The `sched_max_future` and `sched_max_past` fields indicates the scheduling tolerance values of the switch. If the corresponding bundle features request has the `ONF_BF_TIME_SET_SCHED` flag enabled, these two fields are identical to the ones sent be the controller in the request.
- The `timestamp` field indicates the switch's time during the transmission of this feature reply. Every bundle feature reply that includes the time property also includes a timestamp. The timestamp may be used by the controller to get a rough estimate of whether the switch's clock is synchronized to the controller's.

6.5 Bundle Features experimenter property

The `OFPTMPBF_EXPERIMENTER` property uses the following structure and fields:

```
/* Experimenter bundle features property */
struct onf_bundle_features_prop_experimenter {
    uint16_t         type;    /* OFPTMPBF_EXPERIMENTER. */
    uint16_t         length;  /* Length in bytes of this property. */
    uint32_t         experimenter;  /* Experimenter ID which takes the same
                                       form as in struct
                                       ofp_experimenter_header. */
    uint32_t         exp_type;      /* Experimenter defined. */
    /* Followed by:
     *   - Exactly (length - 12) bytes containing the experimenter data, then
     *   - Exactly (length + 7)/8*8 - (length) (between 0 and 7)
     *     bytes of all-zero bytes */
    uint32_t         experimenter_data[0];
};
OFP_ASSERT(sizeof(struct onf_bundle_features_prop_experimenter) == 12);
```

The `experimenter` field is the Experimenter ID, which takes the same form as in struct `ofp_experimenter`.

7 Time Bundle error message

The following errors are defined by this extension:

```
/* Error codes */
enum onf_error_exp_type {
  ONFERR_ET_SCHED_NOT_SUPPORTED = 3400, /* Scheduled commit was received and
                                         * scheduling is not supported. */
  ONFERR_ET_SCHED_FUTURE        = 3401, /* Scheduled commit time exceeds upper
                                         * bound. */
  ONFERR_ET_SCHED_PAST          = 3402, /* Scheduled commit time exceeds lower
                                         * bound.  */
```



```
  ONFERR_ET_MULTIPART_BAD_SCHED = 3403, /* Switch received a bundle features
                                         * request and failed to update
                                         * the scheduling tolerance. */
};
```

The error ONFERR_ET_SCHED_NOT_SUPPORTED, ONFERR_ET_SCHED_FUTURE, ONFERR_ET_SCHED_PAST and ONFERR_ET_MULTIPART_BAD_SCHED use the following structure:

```
/* Message structure for all errors. */
struct onf_error_msg {
  struct ofp_header header;
  uint16_t type;            /* OFPET_EXPERIMENTER. */
  uint16_t exp_type;        /* One of ONFERR_ET_* above. */
  uint32_t experimenter;    /* ONF_EXPERIMENTER_ID. */
  uint8_t data[0];          /* Up to 64 bytes of failed request. */
};
OFP_ASSERT(sizeof(struct onf_error_header) == sizeof(struct ofp_error_experimenter_msg));
```

The type field must be set to OFPET_EXPERIMENTER.

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

The data fields contains a copy of the failed request message, truncated to 64 bytes.

The exp_type field is set to ONFERR_ET_SCHED_NOT_SUPPORTED, ONFERR_ET_SCHED_FUTURE, ONFERR_ET_SCHED_PAST and ONFERR_ET_MULTIPART_BAD_SCHED.

When the switch has an error related to the time bundle operation, or related to a bundle features request, the switch may generate the following errors:

The ONFERR_ET_SCHED_NOT_SUPPORTED code is used when the switch does not support scheduled bundle execution and receives a commit message with the ONF_BF_TIME flag set.

The ONFERR_ET_SCHED_FUTURE code is used when the switch receives a scheduled commit message and the scheduling time exceeds the sched_max_future (see Section 2.4).

The ONFERR_ET_SCHED_PAST code is used when the switch receives a scheduled commit message and the scheduling time exceeds the sched_max_past (see Section 2.4).

The ONFERR_ET_MULTIPART_BAD_SCHED code is used when switch receives a bundle features request with the ONF_BF_TIME_SET_SCHED flag enabled, and the switch failed to update the scheduling tolerance values.

References

[1] Open Networking Foundation, "EXT-230 — Bundle Extension," *Version 0.1*, 2013.

[2] IEEE TC 9, "1588 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems Version 2," 2008.

