Time-based Updates in OpenFlow: A Proposed Extension to the OpenFlow Protocol

Tal Mizrahi, Yoram Moses*
Department of Electrical Engineering
Technion — Israel Institute of Technology
dew@tx.technion.ac.il, moses@ee.technion.ac.il

July 7, 2013

Abstract

Software Defined Networking (SDN) defines a network architecture in which the control plane is managed by a logically centralized controller, and thus configuration updates occur frequently. We have recently introduced an approach that uses time-based configuration updates, allowing to simplify complex update procedures and to minimize transient effects caused by configuration changes. This paper proposes an extension to the OpenFlow Protocol that allows time-triggered configuration updates.

1 Introduction

Software Defined Networking (SDN) defines a clear distinction between the data plane and the control plane; on the data plane, forwarding decisions are taken locally at each switch in the network, while the control plane is managed by a logically-centralized controller, overcoming the need for complicated distributed control protocols and providing network operators with powerful and efficient tools to control the data plane.

The centralized approach in SDN introduces various challenges in terms of performance and consistency. The controller is required to routinely perform frequent network configuration updates. Thus, update procedures must be as simple as possible, avoiding complex and stateful processes in the controller. Moreover, the controller must take care to minimize network anomalies during update procedures, such as packet drops or misroutes caused by temporary inconsistencies.

Time-based configuration [1] can be a useful tool that enables an entire class of coordinated and scheduled configuration procedures. Time-triggered configuration allows coordinated network updates in multiple devices; a controller can invoke a coordinated configuration change by sending update messages to multiple switches with the same scheduled execution time. A controller can also invoke a time-based sequence

*The Israel Pollak academic chair at Technion; this work was supported in part by the ISF grant 1520/11.

of updates by sending n update messages with n different update times, $T_1, T_2, ..., T_n$, determining the order in which the corresponding operations are executed.

This paper introduces time-based updates in OpenFlow [2, 3]. Configuration in OpenFlow is performed on two different planes; the control plane is managed by a *controller* using the OpenFlow wire protocol [3], whereas the management plane is managed by a *configuration point* using the OpenFlow Management and Configuration Protocol (OF_CONFIG) [4]. The latter is typically used for initializing basic configuration in the network, whereas the former is used for routine and frequent configuration updates.

This paper defines an extension to the OpenFlow wire protocol that allows the use of time-triggered configuration updates. This extension is implemented using a Type/Length/Value (TLV) field that can be included in any of the existing OpenFlow messages.

Since the OF_CONFIG Protocol uses NETCONF [5] messages, the NETCONF time capability we defined in [6] can be used to provide similar functionality in OF_CONFIG.

The rest of this paper is organized as follows. Section 2 describes a few basic scenarios of time-based configuration updates. In Section 3 we present a high-level overview of our proposal for time-triggered updates in OpenFlow. In Section 4 we present a proposal for an OpenFlow extension that allows generic TLVs, and finally Section 5 introduces the OpenFlow time extension.

2 Time-based Update Scenarios

This section presents three basic scenarios of time-based updates in OpenFlow.

Example 1.

Consider a network with four switches (Figure 1). Flow A is forwarded through path 1, and we consider a scenario in which the controller updates the route from path 1 to path 2 .

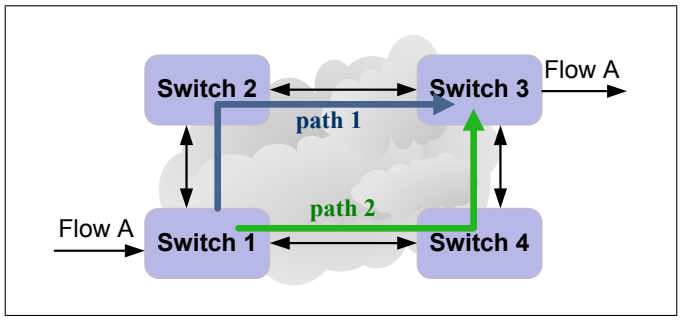

Figure 1: Rerouting a Flow

Without using time: The controller performs two steps; first the controller configures the flow table in Switch 4 to forward traffic from Flow A through path 2, and then it configures Switch 1 with the new path. Obviously if Switch 1 is updated before Switch 4, traffic may temporarily be mis-routed or dropped. Thus, the controller must wait for an acknowledgment from Switch 4 about completing the update before sending the update message to Switch 1.

Time-based Update: The controller sends two time-based update messages to Switches 1 and 4. Switch 4 is scheduled to modify its configuration at time T, and Switch 1 at $T+\Delta$. The time-based update guarantees the update order while simplifying the controller's procedure.

Example 2.

In this example two flows, A and B, are forwarded through a common path that has a bandwidth of 10 *Gbps*, and thus the combined bandwidth of the two flows cannot exceed 10 Gbps. A metering mechanism is used in the n switches to limit the rates of Flow A and Flow B to 3 *Gbps* and 7 *Gbps*, respectively. The goal in this scenario is to reconfigure the metering tables in Switches $1,2,\ldots,n$ so that the rate limits of flows A and B are 8 *Gbps* and 2 *Gbps*, respectively.

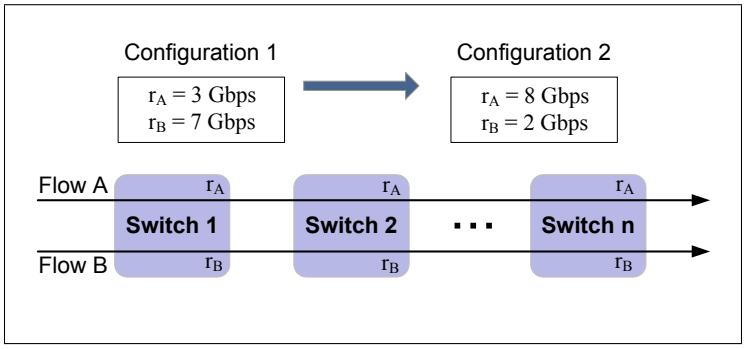

Figure 2: Metering Update

Without using time: The controller reconfigures Switches $1,2,\ldots,n$ with the new configuration. Since the n updates do not occur simultaneously, this procedure yields a transition period during which some of the switches operate according to Configuration 1, while other switches operate according to Configuration 2. During this transition period Flow A is effectively limited to 3 *Gbps* and Flow B is effectively limited to 2 *Gbps*, reducing the total bandwidth of the two flows to 5 *Gbps*.

Time-based Update: The controller configures the n switches to update the rate limits to the new values at time T. After a short transition period, the n switches are configured with Configuration 2. The length of the transition period is a function of the clock accuracy of the switches, and of the time it takes the rate limiters to adapt to the new bandwidth.

Example 3.

In this example, the controller implements the Spanning Tree Protocol (STP) by directly configuring the switches' ports.[1] The network is configured according to spanning tree 1 (Figure 3), and then the controller reconfigures the network to spanning tree 2.

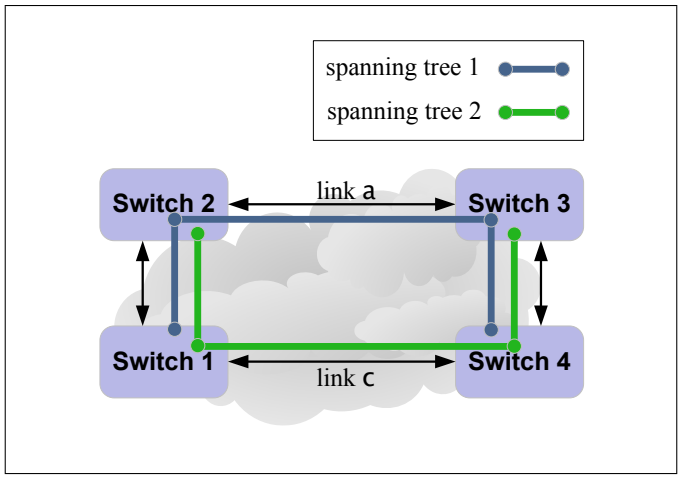

Figure 3: Spanning Tree Update

Without using time: The controller updates the network by a series of updates to the four switches. These updates essentially disable link a, and enable link c. The order in which the updates are performed is cardinal; if link c is enabled first a temporary network loop is created. On the other hand, disabling link a first temporarily breaks the connectivity between two subsets of the switches, and more importantly may also disconnect the controller from some of the switches.

Time-based Update: The controller invokes the configuration change, such that all updates are scheduled at time T. The switches update their configuration at time T, and after a short transition period the network is configured with the new spanning tree. The short transition period has two causes: the switch clocks may not be perfectly synchronized, and packets that are en-route during the configuration change may be dropped or forwarded incorrectly. The time-based update in this case reduces the transition period, and prevents the potential problem of disconnecting the controller from some of the switches.

[1]The OpenFlow spec explicitly specifies that the controller can implement the spanning tree protocol using the OpenFlow port flags. Version 1.0 of the spec [7] also includes a comment that this approach interacts poorly with in-band control, but this comment was removed in more recent versions of the spec.

3 Using Time in OpenFlow

Overview

The extension defined in this document allows to add a time field to any existing OpenFlow message.

Controller-to-switch messages: When a message includes a time field, this field indicates the time at which the switch is scheduled to perform the operation specified in the OpenFlow message. For example (see Figure 4): if an `OFPT_PORT_MOD` message is sent with a time field (T_s), it specifies the time at which the switch should perform the port attribute changes specified in the message.

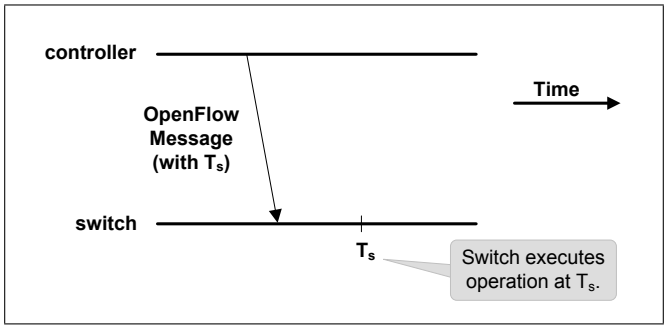

Figure 4: Controller to Switch Message

Switch-to-controller message: When a message includes a time field, it indicates the time of the event or operation that triggered this message. For example (see Figure 5), if an `OFPT_FLOW_REMOVED` message includes a time field (T_e), it specifies the time at which the corresponding flow was removed from the switch's flow table.

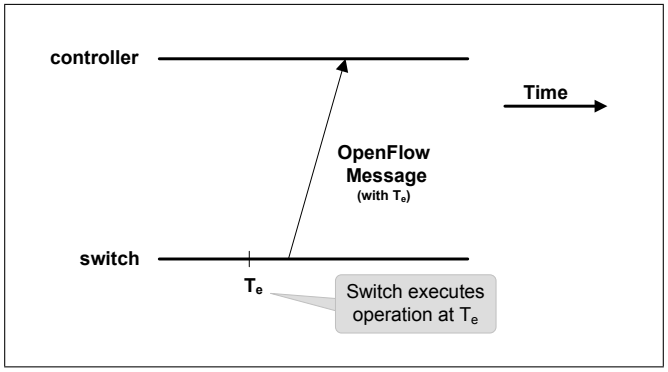

Figure 5: Switch to Controller Message

Request/reply handshakes: When the controller sends a request message, it may include the `GET_TIME` flag, indicating that the corresponding reply should include the time field. For example (see Figure 6), if the controller sends an `OFPT_MULTIPART_REQUEST` with a request for flow statistics and with the `GET_TIME` flag, the switch includes the time field (T_e) in its reply, specifying when the statistics were sampled.

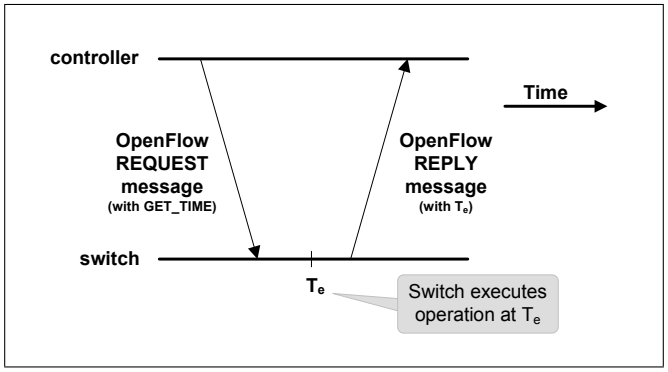

Figure 6: Controller to Switch Message with GET_TIME

The Time TLV

The proposed extension allows to add the time field to all existing and future OpenFlow messages by using a time TLV.

Type/Length/Value (TLV) fields allow a flexible method to add information with potentially variable or unknown format and length. Currently, OpenFlow [3] does not support generic TLVs. Although future versions of the protocol may support generic TLVs, the approach proposed here enables using TLVs in all message types.

Hence, we define the time TLV in two steps: first, in Section 4 we define a generic mechanism that allows to add TLVs to all existing OpenFlow messages, and second, in Section 5 we define the Time TLV.

Synchronization Aspects

The time capability defined in this document requires switches to maintain clocks. It is assumed that clocks are synchronized by a method that is outside the scope of this document.

This document does not define any requirements pertaining to the degree of accuracy of performing scheduled operations. Note that two factors affect how accurately a switch can perform a scheduled operation; one factor is the accuracy of the clock synchronization method used to synchronize the switches, and the second factor is the switch's ability to execute real-time configuration changes, which greatly depends on how it is implemented. Typical networking devices are implemented by a combination of hardware and software. While the execution time of a hardware module can

typically be predicted with a high level of accuracy, the execution time of a software module may be variable and hard to predict. A configuration update typically involves the switch's software, thus affecting how accurately the operation can be scheduled.

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 completion time of the relevant operation.

4 Generic Experimenter TLV Extension

This document describes an ONF extension to OpenFlow version 1.3.x / 1.4 that allows a generic experimenter TLV-based extension to all existing OpenFlow message types. This section is also submitted as an extension proposal to the ONF Extensibility Working Group.

How It Works

Type/Length/Value (TLV) fields allow a flexible method to add information with potentially variable or unknown format and length. Thus, TLVs can provide a useful tool for experimentation.

This extension allows to add TLV-based extensions to any existing and future OpenFlow message. This is done by encapsulating the standard OpenFlow message in the "Experimenter TLV Message" defined in this document, as shown in Figure 7.

The experimenter TLV message format is illustrated in Figure 7. The message is comprised of the Experimenter message header, followed by one or more TLV fields, and finally the encapsulated OpenFlow message, which can be any type of OpenFlow message.

The experimenter TLV approach is a generic mechanism that can add extensions to existing OpenFlow messages without changing their format.

Experimenter ID

The experimenter ID of this extension is: `TBD-TLV-Extension-ID`

Experimenter TLV Message

Experimenter TLV Header

This extension defines the following experimenter types :

```
/* Message types */
enum onf_exp_type {
  /* Experimenter TLV Message. */
  ONF_TLV_EXTENSION = 0x3210,
};
```

The experimenter TLV header is based on `ofp_experimenter_header`, as follows:

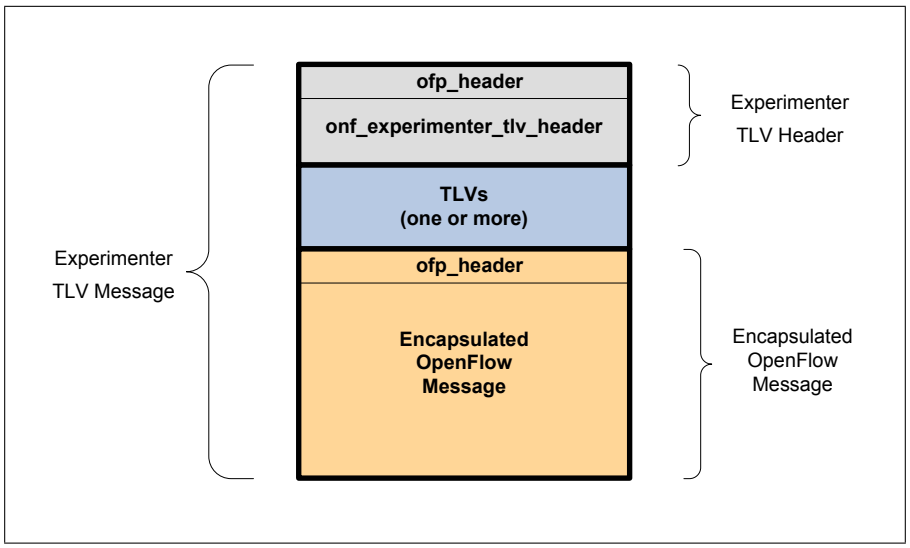

Figure 7: Experimenter TLV Message Format

```
/* Experimenter TLV header */
struct onf_experimenter_tlv_header {
  struct ofp_header header; /* Type OFPT_EXPERIMENTER. */
  uint32_t experimenter;    /* TBD-TLV-Extension-ID */
  uint32_t exp_type;        /* ONF_TLV_EXTENSION */
  uint16_t tlv_length;      /* Total length of the (one or more) TLV fields */
  uint8_t pad[6];           /* 64-bit alignment */
};
OFP_ASSERT(sizeof(struct ofp_experimenter_header) == 24);
```

The experimenter TLV header is followed by one or more TLV.

TLV Fields

The TLV fields defined in this document have the following format:

```
/* TLV Format. */
struct onf_experimenter_tlv {
  uint16_t type;    /* Defines the type of this TLV. */
  uint16_t length;  /* Length of the TLV in bytes including the type and length fields.
                       MUST be a multiple of 8 to guarantee 64-bit alignment.*/
  uint8_t value[0];  /* value. */
};
```

Experimenter TLV Parsing

The following procedure highlights the main steps in Experimenter TLV message parsing:

1. Upon receiving an OpenFlow message, it is identified as an Experimenter TLV message if it matches the following 3 criteria:
   - The type field in the OpenFlow header matches `OFPT_EXPERIMENTER`.
   - The experimenter field matches `TBD-TLV-Extension-ID`.
   - The `exp_type` field matches `ONF_TLV_EXTENSION`.
2. The parser processes each TLV according to its `type` and `length` fields. The `tlv_length` field determines the total length of the TLV fields.
3. The parser processes the encapsulated OpenFlow message.

The parser should verify the following sanity checks:

- The `length` in the OpenFlow header is equal to `tlv_length` plus the `length` in the encapsulated OpenFlow message header.
- The `tlv_length` is equal to the sum of all `length` fields in the packet's TLVs.

5 The Time TLV Extension

The time TLV is an ONF extension for OpenFlow version 1.3.X / 1.4 that enables time-based updates in OpenFlow. This extension allows controllers to invoke configuration updates at scheduled times, and allows switches to attach timestamps to the responses they send to the controller. This section is also submitted as an extension proposal to the ONF Extensibility Working Group.

Experimenter ID

The experimenter ID of this extension is:
`TBD-Time-Extension-ID`

The Time TLV — Detailed Description

When the controller sends a message to a switch it may incorporate a time TLV, as described below. A switch that sends a message to a controller may incorporate a time TLV in the message. This extension uses the Experimenter TLV message type defined in Section 4. The message format is illustrated in Figure 8, where the TLV section includes at least the time TLV and possibly other TLVs as well.

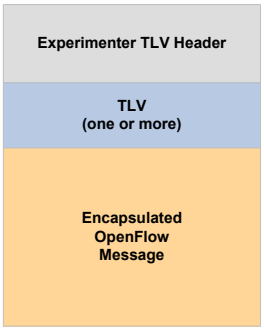

Figure 8: Experimenter TLV Message Format

This extension defines the Time TLV, as follows:

```
/* Time TLV Format. */
struct onf_time_tlv {
  uint16_t type;           /* ONF_TIME_TLV_TYPE. */
  uint16_t length;         /* 24 bytes.*/
  uint64_t seconds;
  uint32_t nanoseconds;
  uint32_t flags;
  uint8_t pad[4];          /* 64-bit alignment */
};
```

This extension defines the following TLV type:

```
/* TLV type */
enum onf_tlv_type {
  /* Time TLV type. */
  ONF_TIME_TLV_TYPE = 0xFA57,
};
```

Time Format: The time format defined in this extension is similar to the one defined in [8]. It consists of two sub-fields; a `seconds` field, representing the integer portion of time in seconds, and a `nanoseconds` field, representing the fractional portion of time in nanoseconds. As defined in [8], time is measured according to the International Atomic Time (TAI) timescale. The epoch is defined as 1 January 1970 00:00:00 TAI.

Flags: The flags field is a bitmap that has the following structure:

```
/* Time TLV flags. */
enum onf_time_flags {
  INCLUDE_TIME = 1 << 0; /* Indicates that this TLV includes a
                            valid time field. */
  GET_TIME = 1 << 1;     /* Indicates that the REPLY message
                            should include the execution time. */
};
```

`INCLUDE_TIME`: specifies whether the current time TLV includes a valid time field. When this flag is enabled, it indicates that the seconds and the nanoseconds fields include a valid time. The `INCLUDE_TIME` flag may be disabled when the `GET_TIME` flag is enabled.

`GET_TIME`: this is relevant only to REQUEST message types. When this flag is enabled it indicates that the corresponding REPLY message must incorporate a valid time, indicating the execution time of the corresponding operation.

Note: when a message includes the time TLV at least one of the two flags, `INCLUDE_TIME` and `GET_TIME`, must be enabled.

References

[1] T. Mizrahi and Y. Moses, "Time-based updates in software defined networks," in *the second workshop on hot topics in software defined networks (HotSDN), to appear*, 2013.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008.

[3] Open Networking Foundation, "Openflow switch specification," *Version 1.3.2*, 2013.

[4] ——, "OpenFlow Management and Configuration Protocol (OF-Config 1.1)," *Version 1.1*, 2012.

[5] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, "Network configuration protocol (NETCONF)," IETF, RFC 6241, 2011.

[6] T. Mizrahi and Y. Moses, "Time Capability in NETCONF," IETF, draft-mm-netconf-time-capability, work in progress, 2013.

[7] Open Networking Foundation, "Openflow switch specification," *Version 1.0.0*, 2009.

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