OPEN NETWORKING
FOUNDATION

The Benefits of Multiple Flow Tables and TTPs

Version Number 1.0

February 2, 2015

ONF TR-510

OpenFlow

ONF Document Type: TR (Technical Recommendation)
ONF Document Name: TR_Benefits of Multiple Flow Tables and TTPs_v.1.0

Disclaimer

Table of Contents

1 Introduction

One of the most significant enhancements of OpenFlow 1.3 vs. OpenFlow 1.0 is the capability to handle Multiple Flow Tables (MFTs). MFTs add power and flexibility. However, this power and flexibility come with a price: added complexity. This complexity makes it particularly challenging to support on ASIC-based products. Taming this complexity prompted the Forwarding Abstractions Working Group (FAWG) to develop the "OpenFlow Table Type Patterns" (OF-TTP) specification. TTPs provide a specific framework for dealing with MFTs with sufficient specificity to enable ASIC implementation among other benefits described below.

This paper describes the benefits of using MFTs and gives specific examples of how TTPs can accelerate the adoption of MFT-based SDN solutions.

2 Background: The Single Flow Table

In order to understand the impact of MFTs, we need to spend a moment looking at their predecessor, the single flow table of OF1.0. In OF1.0, packet forwarding within an OpenFlow logical switch is controlled by a single flow table that contains a set of flow entries installed by the controller. The flow entries contain match fields, actions and a priority assigned by the controller. The switch checks incoming packets against the flow entry match fields. The highest priority matching flow entry defines (via its actions) how to handle matching packets. The set of packets that match with an entry (and no higher priority entry) is, by definition, a flow in OpenFlow terms. Thus the controller, by populating the single flow table with flow entries, defines all the relevant flows and how they should be handled. All matching and processing is described in this single table. That’s very simple and straightforward, but the single table model is too restrictive to address many interesting networking situations in a practical, scalable way.

2.1 The Limitations of a Single Flow Table

This paper considers two use case categories for which a single table is too constraining. The first category involves performing independent actions based on matching different fields in a packet, which effectively requires a separate look up. The second category involves a natural two-stage processing model. In the two stage model, packets are first tagged (or meta-tagged) based on some packet characteristic, then more matching and processing occurs. Each of the above categories can, in theory, be handled in a single table, but the handling is generally awkward and the need to combine matching fields forces an explosion of the number flow entries. Both categories will be discussed in more details below. The key point is that such use cases can be handled in a straightforward way by adding more flow tables.

2.2 The Independent Action Use Case Category

There are at least two common networking situations where independent lookups and actions are needed: “MAC Address Learning” and “Reverse Path Forwarding Check”.

2.2.1 MAC Address Learning

Normal switch unicast MAC forwarding involves looking up the packet's destination MAC address in the switch's MAC Address table. If the destination MAC is found, then the packet is forwarded out the port in the MAC Address table entry. If the destination address is not found, the packet is typically flooded to all ports (excluding the ingress port). MAC address learning is commonly used to populate the MAC Address table. MAC address learning involves looking up the packet's source MAC Address (typically with the VLAN ID) in the MAC Address table. When the source MAC is not found, then the source MAC address can be "learned", either through a local process, or by sending the packet header to the controller for processing.

Although in theory both the forwarding function and the learning function could be done with a single two-address lookup, this approach would result in an explosion of entries. To support N MAC addresses would require $O(N^2)$ entries, so 1,000 MAC addresses would require 1,000,000 entries (plus all the extra messaging to manage those entries). Obviously this is unwieldy. Using two OpenFlow Flow Tables, only 2N entries would be required (with table synchronization features in newer OpenFlow versions, only N entries would be required).

2.2.2 Reverse Path Forwarding Check

Reverse Path Forwarding Check (RPFC) involves checking a packet's source IP address to validate that the packet's ingress interface is consistent with its source address, which can cover either unicast or multicast traffic. Just as with MAC Address Learning, it's theoretically possible to use a single table look up to do the source address check and the destination address forwarding, but in practice the single look up approach is unusable. In part the impracticality is because of the "table entry explosion" problem (combined lookups require N^2 instead of 2N entries). Numerically, 1,000 source IP address ranges and 1,000 destination IP address ranges can be handled with 2,000 flow entries using the MFT approach, but would require 1,000,000 entries (and associated extra messaging) if a single table approach were used.

The need to match on two IP addresses (with potentially different mask lengths) in a single lookup would also make the single-table approach more complicated than the MFT approach. As with MAC learning, using two OpenFlow Flow Tables greatly simplifies the implementation of Reverse Path Forwarding Check.

2.3 The Two-Stage Processing Use Case Category

It is often desired to perform some "pre-processing" operation on packets before performing some other function. Two examples of pre-classification are described here: Port-based VLAN IDs, and Virtual Routing and Forwarding

2.3.1 Port-based VLAN IDs

Most Ethernet switches support 802.1Q Virtual LANs (VLANs). The standard specifies how switches can isolate and forward packet traffic in separate VLANs using a system of VLAN tagging. That is, packets are tagged with their associated VLAN IDs, and switches make forwarding decisions based on the combination of VLAN ID and destination MAC address. Often VLAN tagging is only used within the network; packets on the last hop from the network to the end node are not tagged. Packets arriving untagged at a switch port must be assigned the appropriate VLAN tag before the MAC forwarding decision is made. In theory, a single compound lookup can be used to perform the dual function of tagging and forwarding, but this is complicated and approaches N_{Port} x N_{MAC} table entries when only N_{Port} + N_{MAC} are needed if two tables are available. For a 500-port switch to support 100k MAC addresses, 50 million entries (and associated messages) would be required in a single table model, while a 2 table model would require only 100k+500, or 100,500 entries.

2.3.2 Virtual Routing and Forwarding

Virtual Routing and Forwarding (VRF) resembles the VLAN concept in some respects. Just as VLAN allows for the creation virtual networks that do isolated L2 forwarding, VRF allows for independent IP routing and forwarding tables. As the Wikipedia entry states:

> In IP-based computer networks, **Virtual Routing and Forwarding** (**VRF**) is a technology that allows multiple instances of a routing table to co-exist within the same router at the same time. Because the routing instances are independent, the same or overlapping IP addresses can be used without conflicting with each other.

When VRF is implemented in a device, the device has some scheme for recognizing which VRF instance to invoke for each packet. For example, a device may map sets of destination MACs (and VLAN IDs) to each VRF instance. (Often a different per-instance MAC+VID is used for each device interface.) This mapping is similar to the Port-based association with VLAN IDs. That is, just as many ports may map to one VLAN ID, many MAC-VID combinations may map to a given VRF instance. As with Port-based VLANs, multiple VRFs could theoretically be supported in a single flow table. But doing so would be cumbersome and would result (again) in the "explosion" of flow entries. Every IP address entry for a VRF instance would need to be combined with each MAC+VID associated with the VRF. The resulting number of entries grows by N_{Port} x N_{VRF} x $N_{IPEntry}$. The multiple table approach uses a first table to "tag" packets with VRF identifier metadata and a second table to match on VRF identifier and IP address. This approach requires a number of entries on the order of N_{Port} + (N_{VRF} x $N_{IPEntry}$), a compelling reduction.

As a numerical example, if we consider a 500-port device with 50 VRFs and 10k IP entries each, a single table approach would require 250 million entries. The MFT approach would require only 500k + 500, or 500,500 entries, a large but manageable set.

2.4 The Introduction of Multiple Flow Tables

As outlined above, there are many desired functions for which the constraint of a single table is impractical. Further, SDN architects and users can be expected to want multiple such functions in a given network, each of which could benefit from MFTs. Awareness of these pressures prompted the introduction of MFTs in OF1.1 in February 2011.

OpenFlow application developers would like to benefit from the arrival of OF1.3-capable switches, eager to take advantage of multiple flow tables. But not all products that support OF1.3 will necessarily support MFTs. Why is that? Well, although OF1.3 allows for MFTs, it is still "legal" for an OF1.3-enabled switch to support just a single flow table. Because MFTs present a challenge to ASIC-based switch implementation, some switch vendors have introduced single table implementation as a way to simplify their implementations. In addition, among those vendors that have introduced MFTs, there is no common set of table attributes across the offerings. The nature of available support makes it difficult for an app developer to take full advantage of OpenFlow 1.3 on ASIC-based devices.

2.5 The Challenge of Multiple Flow Tables

The challenge of supporting Multiple Flow Tables in some switches arises because MFTs effectively introduce a new concept: a non-trivial OpenFlow switch pipeline. OF1.0 described a trivial "one-stage" switch pipeline. Switch platforms have their own underlying pipelines that, by and large, are much more complex than just one stage. Still, the trivial OF1.0 pipeline was relatively easy for switch providers to map into the stages of the underlying pipeline. Because the single-table model fits all flow handling into a single "flow-mod" message, the "pipeline mapping" task is nicely bounded. Switch coders can anticipate which flow-mod messages their switch will support, and so no additional information is needed for switches to handle the flow-mod messages as they arrive.

By contrast, when MFTs are available, the implicit OpenFlow pipeline (255 tables) exceeds what all available fixed-function (ASIC, merchant silicon and FPGA) switches can support. Pipeline mapping is no longer trivial for such platforms. Overall forwarding behavior no longer fits in a single flow-mod message. The SDN application may desire more than the switch can support. Or, looking the other direction, the switch may lack what the application needs. Yikes.

Messages have been defined to convey current and desired table properties, but the interplay between different features is very difficult for code to handle in the absence of additional information. In the end, application writers may express similar functionality in highly variable ways. Switch implementers must consider approaches to support among those an application might use. Controller coders might have it worse if they are trying to mediate between demanding applications and diverse switch hardware. The long and short of it is that the initial OpenFlow framework of "map the pipelines on the fly" does not work for MFTs. And this is part of the reason for single table implementations of OF1.3.

Does this mean MFTs cannot be supported on today's fixed function silicon? Will we be dependent on NPUs and servers to develop and deploy richly featured, scalable SDN applications? If true, this might impede adoption, because of a chicken-and-egg problem. (SDN developers may wait for broader deployment of NPUs, but the benefits of NPU deployment are unclear in the absences of SDN applications.) The good news is that there's a path forward that makes MFTs practical on familiar fixed function (e.g. ASIC) platforms. The approach is built on the idea of resolving the "pipeline mapping" problem in advance of run-time. This is not all bad.

It turns out that network operators want their networks, whether traditional or SDN-based, to behave in deterministic ways. They expect the devices and software to be tested and predictable. This means the necessary forwarding behavior, which can be described as a precise pipeline model, can be identified before the network goes live. That is, before run time. And this means that we have additional options for figuring out the pipeline mapping.

3 Table Type Patterns: Taming the Complexity of OpenFlow

The ONF's Forwarding Abstractions Working Group (FAWG) was established in August of 2012 to define an OpenFlow framework for supporting MFTs. Since then, the FAWG members have developed a framework for the controller and the switch to agree on a pipeline model. They call the pipeline models "Negotiable Datapath Models" (NDMs) and the first generation of these models, which are compatible with the 1.x versions of OpenFlow Switch, are called "Table Type Patterns" (TTPs).

3.1 The Details of TTPs

Given the diversity of fixed-function hardware, and the variety of SDN application requirements, we can anticipate that several TTPs will be needed to serve the market. To avoid a "TTP definition bottleneck", the planned approach is to enable any ONF community member to develop their own TTPs, either on their own, with partners, or with the FAWG's support. To enable that vision, the FAWG team has written a document describing the TTP development lifecycle and language, as well as some initial tools for working with TTPs. As this document is going through final revision, one vendor has already produced a TTP that describes how to control the vendor's complex ASIC pipeline via OpenFlow. This TTP and other examples are available on ONF's github-based TTP Repository.

3.2 Benefits of TTPs

TTPs were first conceived as a mechanism to make multi-table OpenFlow practical on fixed-function silicon. But along the way, important added benefits were identified.

- Flexible platforms, based on servers and NPUs, can leverage TTPs to provide higher performance and greater scaling.
    - Although soft platforms are highly flexible, lack of pipeline information forces even flexible platforms to allocate resources inefficiently and use algorithms that are potentially more flexible (and slower) than necessary. TTPs enable such platforms to optimize their support for specific pipeline models.
- TTPs can be "test profiles," enhancing interoperability, including with OpenFlow's optional functions.
    - TTPs describe abstract forwarding models. Apps and controllers that are designed around TTPs will constrain their OpenFlow messaging to the boundaries of the models. Switches that support TTPs promise to deliver forwarding services according to those models. When both sides are "TTP-conformant", interoperability is predictable.
- TTPs simplify the compatibility matrix problem
    - Independent of technology details, TTPs will provide an easy way for buyers and partners to determine what products interoperate with each other. Suppliers can document their TTP support, making it straightforward to mix-and-match products.

4 Summary

The availability of Multiple Flow Tables expands the power and applicability of OpenFlow for the near term and the long term. The Table Type Patterns specification provides the mechanisms for delivering that power in an interoperable way. Support for TTPs is expanding within the ONF as well as more broadly in the industry, including from network silicon vendors, open source SDN controllers and tool providers.

SDN industry participants interested in the power and interoperability offered by TTPs should familiarize themselves with the Table Type Patterns specification and related resources

5 TTP Resources

Table Type Patterns specification:

https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/OpenFlow%20Table%20Type%20Patterns%20v1.0.pdf

Table Type Patterns specification introductory blog post:

https://www.opennetworking.org/?p=1373&option=com_wordpress&Itemid=471

Interview with OpenDaylight TSC Chair Colin Dixon on the ODL TTP project:

https://www.sdxcentral.com/articles/contributed/openflow-table-type-patterns-opendaylight-next-release-colin-dixon/2014/08/

ONF’s Table Type Patterns Repository:

https://github.com/OpenNetworkingFoundation/TTP_Repository

Information on Broadcom’s OpenFlow Data Plane Abstraction (OF-DPA), the first vendor-provided TTP:

http://www.broadcom.com/press/release.php?id=s886762

As well as the OF-DPA Repository

https://github.com/Broadcom-Switch/of-dpa