Table Type Patterns Proposal

Name

Table Type Patterns

Status

This project proposal was approved for incubation on April 17, 2014: https://meetings.opendaylight.org/opendaylight-meeting/2014/opendaylight-meeting.2014-04-17-16.56.html

Subsequently, the ONF chose to retitle is soon-to-be-published specification from "Negotiable Datapath Models" to "Table Type Patterns" (TTPs are the first instance of NDMs). Related to that title change, the project team has requested that the OpenDaylight project name change to " Table Type Patterns" as well. The meeting notes above include reference to the TTP as well, and we can leave other breadcrumbs. Formal approval of this name change will be requested in upcoming TSC meeting(s).

For now, the project page will be created using TTP, with a redirect at Negotiable Datapath Models and Negotiable Datapath Models:Main. (My boldness here derives from Be bold. Steer me right if I'm out of line.)

Repo Name

ndm

Description

Negotiable Datapath Models (NDMs) are the (upcoming) first tangible output from the ONF's Forwarding Abstractions Working Group (FAWG). The goal is to allow for an OpenFlow controller and OpenFlow switch to agree on a set of functionality to help manage the increased diversity made possible with OpenFlow versions 1.1+.

While the current focus of NDMs inside the ONF is on OpenFlow, these abstractions are likely to be more broadly useful and inside OpenDaylight they could expand to be used for other protocols.

Table Type Patterns

The first incarnation of NDMs is Table Type Patterns which are, at a high level, an extension of OpenFlow table features messages that are used to describe subsets of OpenFlow 1.3+ functionality.

Plans

In the long run, we'd like for a TTP (or other NDM) to be negotiated as part of connecting to the switch or out of band, e.g., via OFCONFIG or OVSDB. The TTP would then either be self-describing or provide a link (either explicitly via a URI or implicitly by looking up the name in a naming-authority database) to a description of what features the TTP provides. For example something saying it had 3 tables: one flexible OF1.0-style table, one L2 forwarding table and one L3 forwarding table.

Until vendors start to support this, the most useful thing seems to be supporting something we'd like to call “empirical TTPs”. The general idea is that as OpenFlow 1.3 hardware starts to be available, we will need a way to understand what capabilities it supports and what capabilities it doesn't. There are just too many variables and optional features to be able to discover this at runtime and do anything with the information.

To do this in a sane manner, we will need to maintain a (static, offline) database of devices and their features. For example, knowing that Huawei 8500 series support only a single table with OF1.0-style features. (I'm making the vendor and model number up, not basing it on any real system.) I also want to note the the description of supported features need not be complete or static over time. We can start off by describing some of the features we've found that work and then move on from there adding more “known good” features as we discover/need them.

Fortunately, TTPs provide a relatively straightforward way to do this. They allow for an unambiguous description of the features a switch supports. If we have tooling in OpenDaylight to read and parse these TTPs we can have semi-progammatic support for OpenFlow 1.3 hardware in a way that makes it actually usable. First we start to store the features of switches in the offline database and then we fake the negotiation by reading the vendor ID and whatever else we need to identify the switch (Do we have to fall back to just Vendor + DPID or is there other info we can use?) and then just inferring the right TTP.

Along these lines, the OpenDaylight NDM project/effort would like to start by working on basic tooling to read, parse and write TTPs as well as maintain a database of the TTPs keyed off of vendors and device models (with hopefully enough information to suss that out during OF1.3 switch connection). Once we have this, hopefully we can start to build up some information around current OpenFlow 1.3 devices and make those much more useful.

Also, as we do this, we can start to make tools that let us get real things done with OpenFlow 1.3 hardware in a non-device specific way. For example, writing test suites that profile OF1.3 switches and produce a TTP by testing their features directly. Also, we'd like to be able to do things like discover isomorphisms between TTPs and/or find the commonality between multiple TTPs.

Really, this is all just in the spirit of enabling developers to take as much advantage of OpenFlow 1.3 hardware as we can in the least device-specific manner we can as fast as we can.

Hardware Targets

- **Broadcom OF-DPA:** Provides true OpenFlow 1.3 multiple table support with binaries and source code to work on most recent Broadcom-based switches assuming firmware access.
- **Intel Flexpipe:** ???
- **IBM Switches:** Have support for L2 forwarding in OF1.0 and adds MPLS support in OF1.3.

Software Targets

Glue code that shouldn't have to do much to put software switches into the appropriate mode. Mostly, this involves finding currently unused table numbers and allocating them to supporting a profile. Also preventing existing apps from violating the notion of how those tables should work established with NDMs.

References

- Curt and Colin's OpenDaylight Summit Talk
- Slides from the OpenDaylight Summit Talk
- Curt's slides from US Ignite
- Source, binaries and documentation for Broadcom's OF-DPA

Scope

This proposal provides a way (in the short run) to specify subsets of the functionality provided by OpenFlow 1.3

Resources Committed (developers committed to working)

- Colin Dixon
- Curt Beckmann
- Joe Tardo
- Abhijit Kumbhare

Initial Committers

- Colin Dixon
- Curt Beckmann

Vendor Neutral

- Since this is a vendor-neutral implementation of tools for using a vendor-neutral specification, this should not be an issue.

Meets Board Policy (including IPR)

TBD. Minimal code currently exists which should ease IPR.