Simple Sample TTPs with Brief Discussion

In the examples attached to this PDF file, TTPs (see TS_OpenFlow_Table_Type_Patterns_v.1.0_052014.pdf) are represented using JSON syntax. JSON was chosen in part because it is both human readable and machine consumable. Machine consumability imposes structure and reduces ambiguity. It also enables creation of TTP-related tools such as syntax and consistency checkers. JSON is broadly supported by existing libraries and tools, and is supported by an active community. One useful tool for viewing JSON files (including the attached TTPs) is the online editor at http://jsoneditoronline.org/. (It's also available as an offline Chrome extension.)

Each of the attached "Simple Sample" TTPs describes a 2-table logical switch model in order to highlight the purpose of TTPs. Multiple flow tables represent a tough challenge for software developers writing code to translate OpenFlow messages into API calls on ASIC-based switches. A close look at these two TTPs makes clear that a controller might use, say, Table 1, in a variety of ways that the coder can't predict. That's problematic when coding a message handler because handling of messages in Table 0 or later tables can be impacted by what happens in Table 1. TTPs provide all the table linkages and other context the switch-side programmer needs to write the message handling code (assuming, of course, that the ASIC is able to do the required functions). In addition, when the controller and the switch are both configured to use the same TTP, the controller-side coding is easier because the switch functionality is clear. We can think of a TTP as a "Function Level Agreement". If both sides sign up to the "FLA", then the results are more predictable.

Both of the TTPs attached to this PDF represent some basic device functionality. Both use two tables, but in different ways. In each case, the same functionality might be doable in a single table; but for the VID-MAC TTP it would require more flow entries (and more messages) than the two-table approach. For the ACL-IPv4 TTP a single table approach would hide (by combining) different table capacities for ACLs and IP forwarding. The reduced messaging and more precise table capacity are two advantages of using multiple tables. The diversity of table properties in these sample TTPs is intended to highlight the need (from a switch coder's perspective) for more context to clarify the appropriate handling of a given OpenFlow Switch message.

There is a lot that these sample TTPs do NOT do. They were not designed to handle some relevant kinds of traffic that one might expect. For example, the VID-MAC TTP handles port-based VLANs, L2 multicast and VID translation. But it does not have explicit handling of control frame detection (LLDP, LACP, STP, etc.), VLAN flooding, or support for MAC learning. The ACL-IPv4 TTP handles simple ACLs, router MACs and unicast and multicast routing. But it doesn't include features for some ARP cases, multicast RPFC, ECMP or other control traffic. Both TTPs could be enhanced feature by feature to some degree. But if those enhancements were added, the TTPs would not be so simple any more.

The complexity of a full featured TTP is not caused by the TTP framework. Instead, the complexity of networking is the normal state of affairs which is usually masked by the beauty of network layers. But because SDN "externalizes" network control (instead of keeping it all in the box), SDN exposes all that layer-hidden magic. During the early adoption phase of SDN, each SDN network will have to connect to a legacy network at some point. At that point of connection, the design of the SDN edge device will require consideration of lots of corner cases. What corner case traffic needs special handling? What can be dropped? What special packets will the controller need to send, and how will the responses be handled? To make sure the device will work, the detailed SDN architecture work must be done, whether or not TTPs are used. Once the required device behavior is understood, we use a TTP to capture it.

The attached TTPs help illustrate how TTPs work. Check them out to begin getting familiar with TTPs. These samples can be starting points for more powerful TTPs. A deeper guide to TTP development is in the works, so stay tuned.