Kent Watsen

NEMOPS Position Paper
November 17 , 2024

Dear NEMOPS PC,

The current state of network management is not bad, but it can be improved. When thinking about the Next-Era of Network Management Operations, I have three thoughts:

1. The RESTCONF with the JSON encoding should be promoted, and NETCONF and RESTCONF's XML obsoleted. Why promote RESTCONF? The reason is because it is easy to develop a RESTCONF clients. Every programming language has easy to use HTTP-client libraries. The `curl` and `wget` CLI utilities can be used for scripting. For those claiming that gRPC has a stronger ecosystem than NETCONF, I agree, but the HTTP ecosystem is even larger. Why use the JSON encoding? RESTCONF currently supports both JSON and XML encodings. In theory, RESTCONF could support a binary encoding as well (e.g., CBOR), but RFC 3535 states that text-based formats should be used. Between JSON and XML, JSON is more well supported by programming languages (e.g., a JSON file can be directly serialized into a Python object). Caveat: whilst RFC 3535 states that data should be textual, there is a strong desire for asynchronous notifications to be binary, given the volume and need for closed loop automation via telemetry analytics, Why obsolete NETCONF and RESTCONF's XML encoding? because having too many choices fractures the community.

2. The Network Management Datastore Architecture (NMDA), defined in RFC 8342, should be mandatory to implement in the next version of the RESTCONF and NETCONF protocols. Specifically, clients should be expected to understand the <factory> datastore, <system> datastore, the "template" mechanism, and the "inactive" metadata annotation (letting go of the "<running> alone must be valid" mantra). Level-upping to this basis will greatly improve manageability.

3. There should be a "library" of adaptors to transform standards-based data models (data conforming to YANG modules) to the native data model supported by devices. The reality is that devices rarely implement standard models. Many times the Orchestrator/Controller will have code to transform service-level model data to device-specific data models (adapters). Regardless where the transformation executes (on a device, on a controller, or a cloud-based service), it would be ideal if the transformation could be provided as a download. I'm envisioning something like a AppStore/Marketplace site that hosts adaptors for YANG modules, mapping to a variety of devices. The adaptors should be language-agnostic (possibly a RESTful service), as having language-specific libraries would cause fractures in the community. The adaptors could be both open source (community-developed) or commercial. It just takes some entity to create the such a space (website), which I think the IAB/IETF is ideally suited to do. [Disclaimer, data transformations may be one-way only, e.g., the Network is the Master]

4. Taking the previous idea to the next level, not only could there be *data*-adapters, but there could also be *device*-adapters. The difference being that data-adaptors assume that the target device can consume structure data e.g., via NETCONF or RESTCONF. But some devices may only support CLI (e.g., over a serial interface) or SNMP. For these devices, it is possible to reverse-engineer a YANG module for the CLI, but there is a need to send the device-specific data to the device via its CLI, which is where a device-adapter is needed. A framework could be developed to host and execute device-adapters. If the framework itself were open/free, language agnostic (e.g., micro services), and adapters could be uploaded/rated by anyone, this could enable this common Operator problem to be solved-once and shared by all. [Disclaimer: the importance of such an endeavor is questionable, given complexity and limited utility of devices. That said, to this day, some new devices only have CLI interfaces, suggesting this being an enduring problem].