Lessons from Working on BGP YANG

IETF-116, Yokohama

Jeffrey Haas <jhaas@juniper.net>
Mahesh Jethanandani <mjethanandani@gmail.com>

Core Lessons

- Structure and Content
- Features
- Maintainability
- Versioning extensible types and some places we've already gotten it wrong.
- Example, examples, examples
- Interwork between models
- Collaboration

Structure and Content

- IDR was donated the initial BGP YANG work by the individuals that eventually started OpenConfig. (2014)
- BGP is a protocol with many extensions. It took several years just to get the node content in for the various features.
  - Original goal was the “core subset of modern BGP” and wasn’t intended to be comprehensive.
  - There was feature creep.
  - Multiple reviews, iterations, “did we model it right”. YANG doctors not able to be helpful here since they’re not protocol experts.
- As content was added, a continual re-examination of “where does this belong”.

Features

- Features permit YANG models to contain optional stuff. Unfortunately, they complicate the model for the server and for clients.
- Even though a core set of features was decided upon, some of the feature creep was needed to accommodate optional features that had impact upon list keying (e.g., add-paths). Others was for common but “not core” features.
- Augmentation with new modules will handle most of this for the future, but this means the model had to be structured in such a way that the work could be done in the future.

Maintainability

- A continuous check for “does this enable future work” is required.
- Tests were done to see if we could augment the model with new modules. These tests showed we sometimes needed to restructure things, or re-do groupings.
  - Many “bess” groups will have need to eventually augment BGP state: Path Attributes, Address Families.
  - New core IDR features happen regularly.
  - Some work involves incremental extensions, such as new BGP Extended Community types for things like route-targets.

Type Versioning

- Extensible work needed to be delegated to IANA: Capabilities, Community Types and Formats, Notification (Sub-)Codes, Features, Address Families (pre-existing).
- When possible, avoid enums and use identities. Identities can be added in augmentation modules while enums will require an update to the defining module.
- Extended Communities (string format) are extremely painful and will be needing regular maintenance.

Type Versioning (2)

- IETF previously defined types for some BGP primitives like route-targets, route-origins, route-distinguisher. (RFC 8294)
  - The types are not in IANA maintained modules and will require RFC updates to the whole module.
  - While using common implementation formats for the types, string ambiguity means those types couldn't be used in the BGP module; mostly, lack of qualifying keywords.
  - We'll likely see a need to update these types as part of working on VPN modules. If we don't, we'll have types with overlapping purpose and different presentation depending on what type is used.

Type Versioning (3)

- Protocols, including BGP, have bit fields. YANG does fine representing these a few ways *when they are defined:*
  - Identities can work ok, but give no hint about what position they fill on the wire unless it's in the description.
  - The "bits" type does a good job for on positioning as well as name.
- Trouble is, when they're not defined in the module, what would you display for operational purposes if you get something unknown?
  - There's no "default" bit name in bits.
  - You're not allowed to change the name of a bit once assigned.
  - Motivation for draft-haas-netmod-unknown-bits.

Examples - please

- More is not less, when it comes to examples
- Real world examples help
- Take a complete example, not just a snippet
  - See example A.5 in the BGP draft
- Examples validate changes, as we found when making changes in the model

Module Interwork

- Not only does the model have to work well with future extensions and be maintainable, but it has to work with other models:
  - key-chain to manage connection layer security.
  - policy, which provides the core framework
  - tcp, supporting internal typedefs for the connection and hooks for tcp-ao
  - bfd
- Testing integration with the above models found bugs in the BGP model, but also those models as well. Issues were found in bfd, tcp, and policy and (mostly) addressed prior to those modules getting to RFC.
  - ipsec module found to have issues similar to those found in tcp model, but it's already shipped. New rev needed.
- "Does this work" isn't getting done enough during review!

Collaboration

- github meant that individual pieces of work could progress via Issues when problems were found, and reviewed by the team as part of pull requests.
  - Not enough eyes on the work. The project suffered from contribution cycles being spread out too thinly over too much time.
- Containerized build environment radically simplified working on a large module.
  - Everything needed to build the models, validate/lint them, and build the resultant draft was in one place.

Conclusions

- Doing the work in shorter timeframe would have reduced some pain.
- Even just getting all of the content initially into the model is hard.
  - Once you have it added, it's hard to review what you might be missing.
  - If you don't get the structure right first round, it's hard to notice in later rounds.
- Maintainability needs to be evaluated and *tested.*
  - Delegate things that can change frequently to IANA.
- Interwork needs to be *tested*.

Some github stats

- 200+ Issues
- 135+ pull requests
- Multiple comparisons vs. vendor features and OpenConfig model.