First Meeting of the Protocol Design team: 8/21/2015

Slides:

- **Finalized Requirements (by 9/1)**
- **Discuss how we will break up the work with a potential break-up:**
  - ephemeral data relating to persistent,
  - ephemeral data relating to dynamic - and how to handle validation & events when state goes away,
  - pub/sub for information flows - with different transports/encodings/destinations/etc.
  - pub/sub for "automatic" handling of events when dependent state is changed/removed,
  - Security transport selection.
- Good idea we failed to discuss – Kent's ideas on the overlay model:
  - https://mailarchive.ietf.org/arch/msg/i2rs/S5iNRWmaqn3p8TQnhD-b4ELrsUE
- **Can we show prototype in Yokohama at hack-a-thon?**
  - **I2RS RIB + I2RS FB + Topology**
  - **Extra Credit: BGP, OSPF, ISIS, MPLS-TE**

- **Are the requirements ok?**

Action items:

- Sue will set-up a mail-list for the design team and a wiki for comments on design documents,
- Sue will send to the list the examples of ephemeral to operational state, ephemeral to configuration, and ephemeral to ephemeral on the mail list and Wiki
- We will focus on two ideas as being key and other requirements being fluid
  - Feedback look for applications,
  - Being able to tie ephemeral to config

Call: 10-11:30am ET every Friday – until we get past current work.

Discussion Notes:

Kent: Should we really have the secondary identity?

Jan: I am concerned that the secondary identity and priority?

Alia: In the storing of the identity. May be we should talk about the need for secondary identity for tracing. We had some discussion about panes of glass and having a limited number of requirements.

Jan: Another concern is storing the identity for each. We are installing millions of records. Lots of meta-data with each node will explode the size of the data needed.

Alia: I assumed we would store a byte per entry. With millions, this is a lot of bytes. How much information do we see the applications passing?

Jan: Last-in-Wins works ok now. All the APPs we used with the single-head has been ok. The multi-headed is useful in some cases.

Andy: The priority is to prevent flapping. If two clients flap then you will need multi-headed control.

Jan: The use cases we've implemented do not require multi-head.

Andy: It will change the standard.

Jan: This is single-headed is working for the ODL.

Alia: This is for the Netconf-agent in the ODL.

Jan: The Netconf is under the controller. The controller handles contention. By the time the applications get to the controller, it is back to a single controller.

Alia: How do handle the situation where the client goes away? How do handle ephemeral state that goes away?

Jan: We do not have the ephemeral state use cases.

Alia: There are two nuances to storing the identity:

a) let me make sure 2 clients do not thrash,
b) the second case is something happens to the network – so ephemeral state is lot; and you must tell the client.

Jan: If you have something that goes away due to the network change (configuration state or ephemeral state), when you reconnect you have to reconcile the state. No matter what the protocol or the agent, the ephemeral state must be reconciled.

Alia: I agree that storing the identity with each of your nodes is expensive. One of the issues is the multi-headed controller, and the second is the notification of the state changes from the config. You are side-stepping the multi-header by using the controller.

Jan: The notification is there for all clients that subscribe to the controller. You can subscribe the tree, sub-tree or an element.

Alia: You are getting subscriptions at a fine grain level and not storing the state. When a client comes into a state, we could subscribe them to events on the sub-tree. This would provide the feedback loop. You are storing the identity with the subscriptions instead of the node.

Jan: Subscription to a subtree could be stored in different datastore. Each implementation is different.

Andy: Our implementation stores owners per node. I think that storing the client ID – just takes up one byte.

Alia: If you are having more 256 clients, maybe you want some controls in your network. This is good feedback. Is there a way forward with requirements?

Jan: When you actually do a change (deep in a tree) with a merge/put – we graph the data into the existing data tree. This is fast for the current data-store. Even if the data storage is not much of an issue, the performance is an issue.

Alia: How does it play for the notifications?

Jan: Granularity is an issue. I do not know if the notifications examine all the notes. I will check with the team to determine if the nodes.

Alia: There is a fine line between use cases now and 5 years from now. A couple ideas that we have been batting around:

a) If we have a small number of priorities, so that we do not have to do first in – and allow for last-in?
b) It feels like there might be a middle ground – where some implementations are storing identity in the node and others are using the notification?

Andy: In order to have multi-headed control, there is possibility of something. What Jan is talking about is the single-headed control.

Alia: The multi-headed control is happening at the controller. They are not solving the problem. Could we take first-in within a set of priority? If we do panes of glass, how does this work?

Jan: These (multi-headed control) are useful requirements for very big systems. It will be a large change into the system for the (priorities and identities). The ephemeral state is fairly easy. We do need to consider the current systems versus future systems.

Alia: Storing an extra byte does not Andy's implementation to fail.

Kent: The panes of glass approach or the bytes per node are implementation details. These would not be represented in an IETF glass. Even though I am a proponent of the multiple pane of glass, I think this might be

Alia: The panes of glass solve the priority. If you also have to handle last-in, causes you to store IDs. If you are not storing IDs, first-in is hard.

Alexander: If you multiple clients sharing the same pane, then you have the issue of first-in.

Kent: You could think of the pane – as either a) being a priority or b) being a client.

Alia: This comes back to a client.

Kent: We need to see our way clear to a client that will work.

Alia: It is useful to have implementation in mind so that we know

Jan: Make a capability that turns on or off – that person comes in.

Kent: The Netconf system has the ability to support # clients.

Jan: Let this be a capability that supports it or not supports it.

Kent: We can defer this to later.

Kent: We have: a) orchestrator with n clients, b) clients without any orchestrator, and c) hybrid. Can we have a mix of A and B.

Andy: The multi-head in NETCONF is mandate for server to implement, and optional

The clients can pass any attributes in the RPC message, and the server sends it back in the reply. There could be a message ID or other tokens. I would consider passing secondary identity as a cookie. It is useful and could be an option like Jan is stating.

Kent implementation could our implementation.

a) Yes/data as active,
b) Yes/data cache (less priority is there, but not active).

Kent: I do not have an implementation – it is juniper's application. Are you suggestion priorities so that: If you write into a pane of glass it is ok, but your data is not operational.

Alia: If your state is not the highest priority, then you get an error for writing. Otherwise, how do get something intelligent?

Andy: Two errors: a) You failed to write, b) You failed to write, but I saved the data.

Kent: Is it atomic? Does the entire update fail or just the piece?

Alia: Three modes:

a) All – or nothing
b) Go to failure and stop
c) Treat each pieces separately.

How does this turn into reality for Netconf?

Jan: Anything other than "All-or-nothing" will be mind-boggling difficult for users and clients. People's head will start to explode.

Kent: These modes were actually copied out of NETCONF. You process the edit-config until you get an error. The priority pre-emption

Keyur: I do agree with Jan that simplicity is the key. If we start the partial updates, it will introduce more errors.

Alia: When an operation comes in, are the correlation made. It sounds like NETCONF has this capability.

Andy: The two major NETCONF vendors do not support things. It is an order of magnitude harder.

Alia: My assumption was that you stored only the valid, and send errors for the nodes. I had not understood. I agree that is complexity

Alia: Uplevel – sub-teams: Parts for capability mechanisms. How do we want to break up this work? Do you think that separating out ephemeral from pub/sub

Kent: My thought is to pick a solution so that we believe in. Maybe it is panes of glass. Let's try to make the simple, easy to implement decision. Then let's compare this to the requirements. Then let's see how we close the gaps.

Alia: This makes this sense for the ephemeral state. This sounds good. There are other pieces:

a) What happens if you are relating to dynamic state? How does this impact your validation rules?

Andy: You must think about what requirements on Yang are real?

- We understand for panes of glass – how ephemeral
- An example of how the persistent storage would be useful on how it reaches into datastores?
  [Susan did not respond on mail list]

Alia: The example that I have been using is the LSP tunnels, I want to send a flow of traffic across that tunnel. To refer to the tunnel you will need an indicator for the LSP tunnel. If the LSP goes down, then it goes down. This is a straight-forward example. You do ephemeral config.

Andy: You are talking about a leaf-ref to operational state. I have been trying

[Sue missed a piece here]

Andy: It is not going to tell you – you can now put your data in. What was there is lost, and onw you have a new data.

Alia: I think the pub/sub ties deeply to it in that regard.

Andy: The pub/sub messages do not align with the datastore push.

Alia: The datastore push datastore does give you something for the data store. I really think we need the ability to articulate in the pub/sub subscription what form we need. Brainstorm the sub-teams:

- Validation of ephemeral vs. datapush (pub/sub) that Netconf adopted.
- Data-push

Alia: Wiki page problems and requirements that bump into this sub-team – augment for call?

Andy: I started working on a draft on ephemeral data store. Let's decide how this is organized. This is the mechanics on how it works. You need to know

Kent: A concrete solution at a high-level would be help. Let's lock down an anchor. If we start with ephemeral "true" are anchor points.

Alia (personal perspective): One piece is key for the anchor point – is the client feedback loop (having notifications sent to).

Andy: A whole draft on multi-headed control would be useful.

Jan: (missed0

Alia: You need the notifications without the multi-applications.

Kent: Leaf-ref to LSP state goes away.

Andy: That is existing notifications?

Kent: We do not support notifications for operational state, and we do not have the ability to do a select on an /XPATH.

Andy: Does this fall out of the pub/sub package?

Kent: I think it mostly comes out.

Jan: Pub/sub can be separated from notifications? The notifications for the data-tree are separate from the events coming the data store.

Alia: Do you think is separate from pub/sub is separate from notifications or events? … Let me give he design center:

1) Feedback look for applications,
2) Being able to tie ephemeral to config

While multi-head control (first-in, priority) is useful. However, you could work around this multi-headed control (see controller).

The next two would come down to the pub/sub notifications.

Kent: I think that this focus will speed things around. Jan could you tell me what the difference between pub/sub and notifications.

Jan: We have data change on the datastore and then secondly we have sending the information on pub/sub notifications.

Kent: I do understand how ODL works in this point, and I thought we were talking about the sending information.

Jan: I think we are talking about the datastore notification.

Kent: With the ephemeral having a leaf-ref, the goal will be to get the notification of what happens to the leaf-ref of the LSP. How the subscription happens, is that we state that the leaf-ref model implicitly had that point.

Sue: Description of the RIB-IB,

Kent: Write-up these subscription cases for the leaf-ref. It would be good to have subscription of how this works.

Jan: You have a leaf-ref from operational to operational, or a config-ref to config-ref.

Keyur: Checking the validation state on the route. If you hit an invalid RPKI tag on the BGP route.

Jan: The data-notifications are a critical mechanism that impacts performance. You must be careful with this mechanisms.

Kent: It is good to have you on the call.

Jan: Robert Varga is the one who did the pub/sub implementation.