Processing Multiple Replies for One Request in NETCONF

(draft-liu-netconf-multiple-replies-00)

Bing Liu (Ed.) , Guangying Zheng, Mahesh Jethanandani, Kent Watsen

IETF92 @Dallas, Mar 2015

Background

- This draft intends to merge following works
  - draft-liu-netconf-fragmentation
  - draft-mahesh-netconf-persistent-00
  - Some relevant mailing list discussion
- Currently, focusing on scenarios collection and requirements analysis
- After that, wish to come out a unified solution for these similar requirements

Scenarios & Requirements

- Bulk <rpc-reply>
  - response data might be very big
- Persistent <rpc-reply>
  - Persistent replies for ping, tracert .etc
- Long time <rpc-reply>
  - Multiple responses for monitoring progress
- “data-push”
  - client subscribes for datastore update; server pushes the updates

- Requirements for NETCONF messaging
  - Handle for correlating multiple <rpc-reply> messages to a given <rpc> message
  - terminate response at any time
  - be able to cancel the request in pipeline scenarios

Next Steps

- Complete the scenario collection and requirement analysis (welcoming contributions)
- Provoke solutions discussion
- Add the requirements into NETCONF charter?

Comment?
Thank you!

Backup Slides

Scenarios

- Bulk <rpc-reply>
  - Discussed in *draft-liu-netconf-fragmentation*
  - Problem: response data might be very big (e.g. routes, statistics, synchronizing)
  - Proposed solution: fragmented replies, controlled via a newly defined <get-block> operation
- Persistent <rpc-reply>
  - Discussed in *draft-mahesh-netconf-persistent*
  - Problem: multiple responses might be returned for an operation (e.g. ping, tracert)
  - Proposed solution: linked replies, adding an “link-id” element in response messages
- Long time <rpc-reply>
  - Discussed in mailing list
  - Problem: some operations might take a long time to perform (e.g. network link performance validation)
  - Proposed solution: initial responses returns handle which the client uses to monitor progress till the final result (no detailed solution yet)

Open Question

- According to the requirements, should we also include the “data-push” scenario as well?
  - Discussed in *draft-netmod-clemm-datastore-push*
  - defines a subscription and push mechanism to allow client applications to request updates from a datastore, which are then pushed by the server to the client per a subscription policy, without requiring additional client requests.