Network Configuration Protocol (NETCONF) Proxy
draft-wangzheng-netconf-proxy-01

Abstract

This document presents Network Configuration Protocol (NETCONF) Proxy through which NETCONF requests can be forwarded to a target host. It would be useful when a client does not have direct network access to a target host.

Table of Contents

1. Introduction

This document proposes a NETCONF Proxy mechanism. The mechanism extends the NETCONF protocol [RFC6241] which provides a standard way to set up NETCONF session. At its core, the mechanism defined here introduces a set of new operations which allow a client to forward NETCONF requests to a target host through an intermediary NETCONF proxy server, especially in case where client would otherwise not have direct network access to a target host. The document also includes YANG data model which extend the model and RPCs defined within [RFC6241].

1.1. Motivation

NETCONF provide a RPC-based mechanism to facilitate communication between a client and a server. The client can be a script or application typically running as part of a network manager. The server is typically a network device [RFC6241]. However, the network manager may not have direct network access to the target network

devices. For example, some target network devices may locate in a network with private addresses behind a NAT device or a firewall. Thus, network manager cannot direct communicate with these target devices.

NETCONF Call Home [RFC8071] provides a mechanism that allows NETCONF Servers to initiate a connection with a NETCONF client, reversing the normal direction of NETCONF session setup. This allows a NETCONF Server, e.g. a networking device that needs to be managed, to reach out to a NETCONF Client, e.g. an Operations Support System of an SDN controller, in order to be managed. By reversing the direction in which NETCONF sessions are normally set up, problems such as establishing connectivity with devices behind a firewall can be alleviated. However, NETCONF Call Home requires that the server knows its client by way of configuration or discovery. It does not address the scenarios as presented below:

1. In some NFV scenarios, VNF instances are running in a private network. To reduce the management resources (like IP resources, bandwidth, etc) of large-scale management activities, these VNF instances may not be assigned IP addresses. Then the element management system (EMS), which located in public network, cannot be aware of the addresses of VNF instances. Therefore, the element management system (EMS) is difficult to manage these VNF instances via NETCONF protocol. More details please see section 1.2.1.

2. And in some cloud network scenarios, the gateway network element (GNE) and non-gateway network elements (N-GNEs) communicate with each other using some private protocol. And these non-gateway network elements (N-GNEs) may not IP devices. Therefore, the cloud centre EMS (element management system) cannot be aware of the addresses of N-GENs. Thus, the element management system (EMS) is difficult to manage these N-GNE devices via NETCONF protocol. More details please see section 1.2.2.

To solve that problem, this document proposes a NETCONF Proxy mechanism. The proxy can acts as an intermediary between manager and target device, therefore the client can set up a NETCONF session to a target through a NETCONF Proxy.

The mechanism allows the client to subsequently direct NETCONF requests to the server, to receive responses, and to subscribe to notifications from the server. While the NETCONF Proxy can be used to traverse NAT boundaries, it should be noted that it does not apply NAT mappings for contents carried as part of the NETCONF payload; specifically, it does not substitute IP address information that is carried as part of data nodes.

1.2. Netconf Proxy Use Case

1.2.1. Using Netconf Proxy to manage VNF Elements

Figure 1 illustrates EMS manage the VNF instances.

```
                +--------------+
                |      EMS     |
                +-------+------+
                        |
                        |  NetConf Over SSH/TLS
                        |
      +.................|.........................+
      .          +------+--------+                .
      .          |     VNFM      |                .
      .          +-------+-------+                .
      .             /    |     \                  .
      .            /     |      \ Neconf Over ITPC.
      .           /      |       \                .
      .          /       |        \               .
      .         /        |         \              .
      .     +--------+ +--------+ +--------+      .
      .     | VNFI_1|  | VNFI_2|  | VNFI_3|       .
      .     +--------+ +--------+ +--------+      .
      .            VNF Network                    .
      +...........................................+
```

Using Netconf Proxy to manage VNF Elements

EMS is connected to VNFM through public network. To reduce the cost of management resources (like IP resources, bandwidth, etc) for large-scale management activities, the VNFIs(VNF instances) are running a TIPC(Transparent Inter-process Communication) protocol, and these VNIs are not assigned IP address. The management data of VNFIs will be transported to VNFM via TIPC. Within the VNF Network, the TIPC protocol will provide the data to the respective application i.e. NETCONF.

To manage the VNFIs, EMS will access the VNFIs via the Netconf Proxy which located in the VNFM. EMS is access to Netconf Proxy through Netconf over SSH. Within the VNF network, the NETCONF data will be transported from VNFM to VNFIs over Transparent Inter-process Communication (TIPC) protocol. And the VNFIs will report their IDs and other information to netconf proxy. The netconf proxy will store these information in the "target-list". According to these

information, the EMS can manage the VNFIs via Netconf Proxy, more details see section 2.

1.2.2. Using NetConf Proxy to manage the Non-Gateway Elements of OSN (Optical Switch Network)

Figure 2 illustrates EMS manage the Non-Gateway Elements of Optical Switch Network (OSN) via Netconf Proxy.

```
                                            ----------------
                                        ////          +--------+\
                                     ///              |       | \\\
                                   //   +--------+    |IP Device   \\
                                 ||     |       |     +--------+     |
                                 |      | IP Device                   |
                                |       +--------+                     |
                                |                                      |
                                 |               +--------+           |
                                 ||              |       |           |
                                   \\            |IP Device         //
                                     \\\ /       +--------+      ///
                                        X\\\                 ////
```

```
                                      //    ----------------
                                     /
                               +-------+
                               |       |
                               | EMS   |
                               +-  /---+
                                  /
                        +---------+
                        |Netconf Proxy ------------
                        |         +--              -\\\\
                      //| Gateway NE -----      +-------+-\\\
                    // -+-----+---+       ----  |Non-Gateway  \
                  //          |               --+ NE    |      \\
                 |            |                 +----+---+       |
               ||             |                      |            |
               |              |                      |             |
               |              |                      |             |
               ||         +---+---+             +----+----+       |
                 |        |Non-Gateway          | Non-Gateway    |
                  \\      |NE      |-------------- NE      |     //
                    \\    +-------+             +---------+  //
                      \\\\            OSN               /////
                           \\\\--                 --////
                                 -----------------
```

Using Netconf Proxy to manage VNF Elements

The network between EMS and GNE is IP Accessible whereas the network between GNE and N-GNE is not IP Accessible. Therefore, the EMS cannot be aware of the address of N-GENs. Note that the Non-Gateway Elements are not IP devices, thus the N-GNE cannot support NAT. The management data of N-GNE will be transported to GNE on OSN's private transmission layer. Within the OSN Network Elements, the OSN private transmission protocol (i.e. via QX interface [G.773]) will provide the data to the respective application i.e. NETCONF.

To manage the non-gateway network elements, NMS will access the non-gateway NE via the Netconf Proxy which located in the gateway network element (GNE). EMS is access to Netconf Proxy through Netconf over SSH. Within the OSN, the NETCONF data will be transported from GNE to Non-GNEs over OSN private transport protocol. And the Non-GNEs will report their IDs and other information to netconf proxy. The

netconf proxy will store these information in the "target-list". According to these information, the EMS can manage the Non-Gateway Elements of OSN via Netconf Proxy, more details see section 2.

1.3. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Solution Overview

The diagram below illustrates how the client can set up a NETCONF session to a target through the NETCONF proxy.


```
         client                          proxy                    target ID:A1
            |                              |                           |
            |                              |                           |
            |          <hello>             |                           |
            |<---------------------------->|                           |
            |                              |                           |
            |       <get>                  |                           |
            |        <target-list>         |                           |
```

```
            |---------------------------->|                          |
            |                             |                          |
            |<----------------------------|                          |
            |     <reply target-list:     |                          |
            |            targetID=A1,     |                          |
            |            protocol,        |                          |
            |            Authen...>       |                          |
            |                             |                          |
            |                             |                          |
            |    <rpc messageID="101"     |                          |
            |---------------------------->|                          |
            |         target="A1"         |                          |
            |         xmlns="urn:xxxx">   |                          |
            |         <rpc>               |                          |
            |                             | case 1: target_id can be |
            |                             |------------------------->|
            |                             |         found            |
            |                             |  Forward <RPC> message   |
            |                             |      <rpc-reply>         |
            |                             |<-------------------------|
            |       <rpc-reply>           |                          |
            |<----------------------------|                          |
            |                             |                          |
            |                             | case 2: target_id cannot |
            |                             |         be found         |
            |   <rpc-reply>               |                          |
            |<----------------------------|                          |
            |     <rpc-error>             |                          |
            |       <error-type>          |                          |
            |        invalid-target       |                          |
            |       </error-type>         |                          |
            |        ...                  |                          |
            |     </rpc-error>            |                          |
            |   </rpc-reply>              |                          |
```

This diagram makes the following points:

1. The client initiates the connection using the SSH/TLS transport protocol. When the NETCONF session is established, the client

and proxy MUST send a <hello> element containing a list of that

peer's capabilities. The proxy SHOULD send at least the "netconf" and "proxy" capabilities. And other rules of capabilities exchange described in section 8 of [RFC6241].

2. The client sends a <get> RPC to proxy to retrieve the "target-list" of the proxy.

3. The proxy responds with a <get-reply> RPC which containing "target-list" attributes. The "target-list" attributes includes the target's information such as target-id, protocol, authentication, etc.

4. The client receives a the <get-reply> RPC from the proxy, and retrieves the target information according to the received "target-list". Subsequently, the client can direct NETCONF requests to the target according to the received "target-list", to receive responses, and to subscribe to notifications from the target. For example, the client wants to retrieve the configuration information of a target. The client should construct a <get-config> message according to the received "target-list". This <get-config> message SHOULD contain at least a "target-id" attribute. And then client sends this <get-config> message to proxy and waits for a reply.

5. The proxy receives the RPC message and checks the value of "target-id" attribute:

   If the target is not found, then an "invalid-target" error will be returned.

   If the target can be found, then proxy forwards the RPC message, which received from client, to corresponding target.

6. The target receives the RPC message. And then sends an <rpc-reply> message in response to the received RPC message.

3. The NETCONF Client

The term "client" is defined in [RFC6241], Section 1.1 "client". In the context of network management, the NETCONF client might be a network management system for example a EMS (element management system).

The client operation describes as follows:

1. The client initiates a connection to proxy using the SSH/TLS transport protocol [RFC6242]. How to establish an SSH/TLS transport connection is described in [RFC6242]

2. When the NETCONF session is established, the client sends a <hello> message to proxy, then waits for a reply. This <hello> message contains a list of client's capabilities.

3. After capabilities exchange, the client sends a <get> RPC to proxy to retrieve the "target-list" of the proxy, then waits for a reply.

4. The client receives the <get-reply> RPC from the proxy, looks up the value of "target-list", and then constructs a RPC message according to the received "target-list".

For example, the client wants to retrieve the configuration information of a target A1. The client should construct a <get-config> message. This <get-config> message SHOULD contain at least a "target-id" attribute:

```
<rpc message-id="101"
   target-id="A1"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/schema/1.2/config">
        <users/>
      </top>
    </filter>
  </get-config>
</rpc>
```

And then client sends this <hello> message to proxy and waits for a reply.

5. If the reply containing a <data> element which satisfied with "Positive Response" condition of corresponding RPC (Section 7 of [RFC6241]), it means that the client has successfully managed the target device.

6. If the reply contains the "invalid-target" error, the process turn to step (4) or aborts.

7. Otherwise, the client interprets the error and aborts.

4. The Proxy

The Proxy should ensure that requests given by client for a particular target device should reach the target device and the operations should be executed on that target device and the response should be given back to the client.

The proxy operation describes as follows:

1. When the NETCONF session is established, the proxy sends a <hello> element containing a list of proxy's capabilities. The proxy SHOULD send at least the "netconf" and "proxy" capabilities. And other rules of capabilities exchange described in section 8 of [RFC6242].

2. The proxy receives the <get> RPC and then responds with a <get-reply> RPC which containing "target-list" attributes. The "target-list" attributes SHOULD includes the target's information such as target-id, protocol, etc. The following example shows a "target-list":

```
<target-list>
 <target-id>A1</target-id>
 <protocol>protocol-foo</protocol>
</target-list>
```

3. The proxy receives the RPC message and checks the value of "target-id" attribute:

   If the target is not found in target-list, then an "invalid-target" error will be returned.

   If the target can be found, then proxy forwards the RPC message, which received from client, to corresponding target.

4. In this Netconf-Proxy model, the proxy reads data from both the client and the target, and writes any data received to the other end, without interpreting the data. If any side of the connection is closed, the proxy closes the other side.

5. The Target

The term "target" is equivalent to the term "server" which is defined in [RFC6242], Section 1.1 "server". In the context of network management, the target is typically a network device.

The target operations describes as follows:

If the connection between the proxy and the target established. And target receives the RPC message from the proxy, and then responds a <rpc-reply> message.

If the target can satisfy the RPC request, the target sends an <rpc-reply> element containing a <data> element which satisfied with "Positive Response" condition of corresponding RPC (Section 7 of [RFC6241]).

If an error occurs during the processing of an <rpc> request, the target sends an <rpc-reply> element which includes a corresponding <rpc-error> element (Section 7 of [RFC6241]).

6. New attribute: target-id

A proxy can be used by a client to connect to several servers and to maintain multiple NETCONF sessions. A client may use the proxy even to maintain multiple NETCONF sessions with the same NETCONF server. When issuing a NETCONF request, a client must therefore differentiate between NETCONF sessions. To solve this problem, a new attribute "target-id" is defined. This attribute allow the proxy to forward RPC to corresponding target.

For example:

The following <rpc> element invokes the NETCONF <get> method and includes the "target-id" attribute:

```
<rpc message-id="101"
     target-id="A1"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get/>
</rpc>

<rpc-reply message-id="101"
     target-id="A1"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <!-- contents here... -->
  </data>
</rpc-reply>
```


7. YANG DATA MODEL

7.1. Overview

The YANG data model for NETCONF Proxy is depicted in the following figure. Following Yang tree convention in the depiction, brackets enclose list keys, "rw" means configuration, "ro" operational state data, "?" designates optional nodes, "*" designates nodes that can have multiple instances. A "+" at the end of a line indicates that the line is to be concatenated with the subsequent line.

```
module: ietf-netconf-proxy
    +--rw proxy {proxy}?
       +--rw proxy-name?    string
       +--rw target-list* [target-id]
          +--rw target-id         string
          +--rw protocol?         string
          +--rw authentication?   string
```

7.2. YANG Module

```
<CODE BEGINS> file "ietf-netconf-proxy@2017-03-09.yang"
module ietf-netconf-proxy {
```

```
namespace "urn:ietf:params:xml:ns:yang:ietf-netconf-proxy";

prefix np;

organization
  "IETF NETCONF (Network Configuration) Working Group";

contact
  "WG Web:   http://tools.ietf.org/wg/netconf
   WG List:  netconf@ietf.org

   WG Chair: Mehmet Ersue
             mehmet.ersue@nsn.com

   Editor:   zitao wang
                 wangzitao@huawei.com";

description
  "NETCONF Protocol Data Types and Protocol Operations.

   Copyright (c) 2011 IETF Trust and the persons identified as
   the document authors.  All rights reserved.
```



```
   Redistribution and use in source and binary forms, with or
   without modification, is permitted pursuant to, and subject
   to the license terms contained in, the Simplified BSD License
   set forth in Section 4.c of the IETF Trust's Legal Provisions
   Relating to IETF Documents
   (http://trustee.ietf.org/license-info).

   This YANG module describe how to define a netconf proxy";

revision 2017-03-09 {
  description
    "Initial revision";
  reference
    "draft-wang-netconf-proxy";
}

feature proxy {
  description
```

```
            "Netconf proxy";
        }

        container proxy {
         if-feature proxy;
         leaf proxy-name{
          type string;
          description
              "Proxy name";
         }
         list target-list {
          key "target-id";
              leaf target-id{
               type string;
               description
               "Target ID";
              }
              leaf protocol {
               type string;
               description
               "Support protocols";
              }
              leaf authentication {
               type string;
               description
               "Authentication";
              }
              description
              "List for target information";
         }
```



```
         description
         "Container for NETCONF Proxy";
        }

      }
      <CODE ENDS>
```

8. Security Considerations

The security considerations described in [RFC6242] and [RFC7589], and by extension [RFC4253], [RFC5246] apply here as well.

9. IANA Considerations

TBD

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC4253] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, DOI 10.17487/RFC4253, January 2006, <http://www.rfc-editor.org/info/rfc4253>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <http://www.rfc-editor.org/info/rfc5246>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <http://www.rfc-editor.org/info/rfc6242>.

[RFC7589] Badra, M., Luchuk, A., and J. Schoenwaelder, "Using the NETCONF Protocol over Transport Layer Security (TLS) with Mutual X.509 Authentication", RFC 7589, DOI 10.17487/RFC7589, June 2015, <http://www.rfc-editor.org/info/rfc7589>.

[RFC793] Postel, J., "TRANSMISSION CONTROL PROTOCOL", STD 7, September 1981, <https://www.ietf.org/rfc/rfc793.txt>.

10.2. Informative References

[G.773] "Protocol suites for Q-interfaces for management of transmission systems", ITU-T Recommendation G.773, 1993.

Authors' Addresses

Zitao Wang
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing
China

EMail: wangzitao@huawei.com

Guangying Zheng
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing
China

EMail: zhengguangying@huawei.com