A XV-a Conferință internațională – multidisciplinară
„Profesorul Dorin Pavel – fondatorul hidroenergeticii românești”
SEBEȘ, 2015

PROTOCOLUL NETCONF, O NOUĂ SOLUȚIE PENTRU MANAGEMENTUL REȚELELOR

Zaid Ali SHHEDI, Florica MOLDOVEANU

PROTOCOL NETCONF, A NEW SOLUTION NETWORK MANAGEMENT

NETCONF (the Network Configuration Protocol) is an emerging technology for the network management, which was conceived to respond to the challenges of the next generation networks. NETCONF is still with a limited deployment, but it is a Network management protocol with a promising capabilities. It can keep pace with Network management requirements and especially configuration management of network devices. In this article we highlight the NETCONF protocol, as a real replacement of the traditional network management protocols such as SNMP (Simple network management protocol). This paper presents an analysis of the NETCONF protocol session with practical examples, the NETCONF layered structure. In particular we analysed NETCONF protocol operations in details.

Keywords: NETCONF session, NETCONF operations, data stores, configuration management
Cuvinte cheie: sesiune NETCONF, operațiunile NETCONF, magazin de date, managementul configurației

1. Introduction

NETCONF was developed and standardized by NETCONF working group and IETF (Internet Engineering Task Force), and Published as a standard in December 2006, (RFC 4741), several improvements were published in the next years and a revised version

of the base NETCONF protocol was published in June 2011, as RFC 6241 [1], [2].

NETCONF allows controlling the network devices configuration. It enables applying a group of operations on the configuration data stored on distributed network devices, installing or deleting the current device configuration data. The NETCONF protocol messages and device configuration data are represented in XML format, which is a readable language for both human and machines [3], [4]

NETCONF was designed to conform to RFC 3535 [5], which describe the network operators' problems and requirements on network management. It is based on the workshop held by the Internet Architecture Board (IAB) on Network Management in June 2002 in USA, Reston. Among the operators' requirements described in this workshop are:

- The network management technology should be easy to use.
- The configuration data that describes operational state and statistics should be clearly and separately defined.
- The network management technology should enable operators to configure the network as whole rather than individual devices. And mechanism for pulling and pushing configurations are needed by operators as primitive operations.
- Text processing tools are needed to process configurations.
- Devices should be able to hold multiple configurations which can be activated separately
- The need for Role-Based Access Control (RBAC) of users at the level of each device

The most important weak points of SNMP, as defined in RFC 3535 are:

- The performance of the monitoring operations depends on the number of MIB objects involved in an operation (set or retrieved). They are slow for large amounts of data (for example a routing table).
- There are too few deployed writable MIB modules that would be useful for devices configuration; for example, router equipment is usually not fully configurable via SNMP.
- It is difficult to implement MIB operations which need a sequence of SNMP interactions. And SNMP does not provide means for easy retrieval and restore of configurations.
- Standardized MIB modules often are not well documented: it lacks a description of how the various MIB objects can be used to achieve certain management functions.

2. NETCONF layered structure

The NETCONF protocol can be conceptually partitioned into four layers:

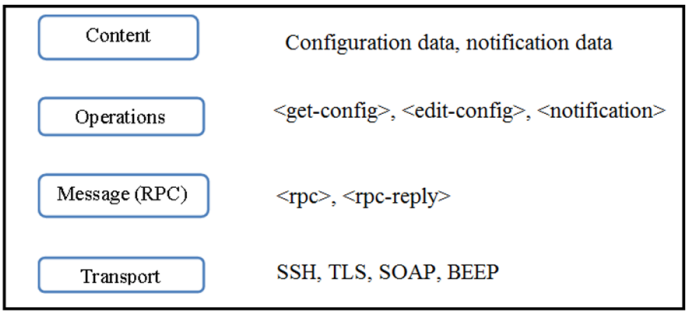

Fig. 1 NETCONF Protocol stack [7]

3. The NETCONF session

NETCONF is based on the client-server paradigm, a session-based and a connection oriented protocol. A NETCONF session means a connection between two NETCONF entities: one is an application that plays the role of a manager (client) and other is a server running on a network device.

The client must initiate the connection (session) which should be terminated when the server is not able to send back a response. The client sends a <RPC> request (XML well-encoded RPC) to the server in a secure NETCONF session and the server in turn sends back XML encoded response (<RPC REPLY>), and it doesn't matter the used transport protocol. The Server response may be positive (<ok> element), or negative (<ERROR> element) with information about the error.

The session or the connection must provide authentication, data integrity, and it must be a permanent connection; and must ensure a safe sequenced data delivery that the client requests should be delivered and processed in the same arriving order by the managed server, and the server response will be in the same order too [5]. The server must authenticate the identity of the entity that asks to establish a session before manipulating any incoming requests. The authenticated entity is called the "NETCONF username". The authentication between both sides is provided at the level of the underlying transport protocol by encrypted connections (xml encoded

RPCs), and the authentication mechanisms must be available on the NETCONF device.

3.1. NETCONF session practical side

The client starts a connection using ssh2 (secure shell version2), as demonstrated in the following command line:

nms1> ssh -s -p830 server.example.com NETCONF [4].

As the connection is established, the server should send its hello message immediately and the client should do the same (handshaking); the contents of the <hello> message determine the NETCONF Capabilities supported by each party, so the manager knows which operations can be submitted to the device. The NETCONF protocol elements are defined in the following namespace: “urn: ietf: params: xml: ns: NETCONF: base: 1.0”

The following XML example illustrates the <hello> message that a client is required to send.

```
<? xml version="1.0" encoding="UTF-8"?>
  <hello Xmlns="urn: ietf: params:xml:ns:NETCONF:base:1.0">
    <Capabilities>
        <Capability>urn:
ietf:params:NETCONF:base:1.0</Capability>
    </Capabilities>
  </hello>]]>]]>  [4].
```

After receiving this message, the server waits for <rpc> requests for processing. For each received <rpc request>; the server should send an <rpc-reply>. All element attributes which are sent in the <RPC> request will be returned back in the <RPC-reply>operation. The "message-id" is an obligatory attribute of the <rpc> element and it is used without decoding or modifying.

```
<? xml version="1.0" encoding="UTF-8"?>
<rpc Xmlns="urn: ietf:params:xml:ns: netconf:base:1.1" message-
id="101">
  <edit-config>
 ……
  </edit-config>
</rpc>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.1"
     message-id="5">
  <ok/>
</rpc-reply>
```

Several errors may occur during the request processing at the server side. In this case, the server must send back the <rpc-error> element included in the <rpc-reply> message. If no errors occurred, an <ok> element is included in the <rpc-reply> message. If the received message was not encoded in UTF-8 or not well-composed XML, the reply must contain the “malformed-message” error.

4. NETCONF Configuration Data Stores

There are three types of data stores, Running is mandatory; Start-up and Candidate are optional.

4.1. The running database<running/>:

It is available constantly; it saves the active configuration data, and conceptual state information for each network device. If the candidate capability was not supported by the server, this means that the candidate data store is not available, so the running data store will be treated directly by the NETCONF operations. There is only one running configuration data store available on some device. The other configuration data stores are allied to capabilities, and are existed only on the devices that declare those capabilities.

4.2. The candidate database <candidate/>:

This database runs as a temporary store (buffer), it enables manipulating and modifying the configuration data, without impacting the network device running configuration, then the results will be committed to the running data store, it is not supported by all devices.

4.3. The start-up database <startup/>:

It represents the initial status of the device configuration data; it saves the configuration that the device loads. It is available if the server supports the: start up capability. It only exists on devices that differentiate between start up and running configuration data stores.

5. NETCONF protocol operations

These operations and their parameters are expressed in XML code, and executed by means of RPC methods, NETCONF operations must be supported by the server. But not all operations have the same sensitivity or importance. The <copy-config> operation is more sensitive than the <edit-config> operation. And when some operation doesn’t have an adequate authorization, it gains an access denied and will not be executed.

<get config> operation is applied on configuration data, but the <get> operation may be applied on both of configuration and state data. But there is no specific operation for the state data, and this may result faults in the network management [8].

<get-config> operation: this operation restores a portion, or the entire queried configuration data store, it has attributes [7]

- Source: it indicates to the name of the configuration data store that the operation applies on.
- Filter: The filter retrieves portions of the device configuration data store based on identified criteria.
- Positive Response: it means that the queried device replies by sending results and a <data> element within the <rpc-reply>
  Negative Response: it means that the queried device replies with an <rpc-error> element within the <rpc-reply> [4], [7].

<edit-config>operation: this operation carries a portion, or the entire source configuration to the destination configuration data store (adding), if the destination data store is present, and if not, it will be created. The destination configuration data store changes is based on the executed operations, and the source data.

<Edit-config> operation attributes:

- The Operation attribute has a mission of identifying the goal within the configuration data store, to execute the required command (performing the operation), the attribute values are as follows: Merge, replace, create, delete and remove [4].

<copy-config>operation: it means to substitute the whole configuration data store with the data of another configuration data store. If the destination data store is present, it is overwritten. If not, a new data store will be made-up, if it is permitted [7].

<copy-config> parameters:

- target: the name of the destination configuration data store
- Source: the name of the source configuration data store, which contains the entire configuration to be copied.
- Positive Response: by sending back an <ok> element included in the <rpc-reply>.
- Negative Response: by sending an <rpc-error> element in the <rpc-reply>.

<delete-config>: It means deleting the specific configuration data store, except the <running >configuration data store which cannot be removed.

<Delete-config>Parameters:

- Target: it refers to the name of the destination configuration data store that to be removed; also it has positive and negative response parameters.

<lock>: It means that the client is able to lock the whole configuration data store, for a short time to perform its intended operations, without the intervention of other NETCONF or non NETCONF clients or other users. The lock operation cannot be performed if any piece of the configuration data store was locked by others (partial lock). In other words, the server reserves the locked resources for the existing session, and doesn't allow any changes by other requesting clients, also when we have a locked node, the partial lock to be applied on any node or sub tree under this locked node will fail, that the sub tree under the locked node is locked too [8]. The partial lock may succeed if the requesting entity has an access rights with a higher priority than the entity that hold a lock. Only the <kill-session> operation can be executed to break a lock specified to another NETCONF session. The duration of the lock starts when the lock is done, and ends when the lock is let off, or the session is ended by the client or by the server. Also it has the positive response, and negative response parameters. The <lock> operation has a general meaning, that if a particular entity carried out a <lock> operation on some resource for reading only, it means that this resource in is not available for other entities who want to write or delete or add or get data, but when NETCONF protocol has a dedicated lock for each operation, that will lead to the provision of resources [8].

<Unlock>: It is the opposite operation of the <lock> operation, used to break the specified lock to some session. To get a successful <unlock> operation, we have to have an active lock presently, also the <unlock> operation should be related to a session different from the session that own the lock. It also has three parameters target, positive and negative responses, as in the previous operations.

<get>: It is able to retrieve running configuration and state information, so using <get> operation may cause an additional overhead for the network, and here the importance of using the filter emerges. <get> operation has the following parameters:

- Filter: to choose what data we exactly need.
- Positive response: It means that the requested sub tree is included in the data part of the <rpc-reply>.
- Negative response: with an <rpc-error> within the <rpc-reply>.

<Close-session>: liberation of all locks and reserved resources owning by the NETCONF session, and closing any related connection. It has the two parameters positive and negative responses.
<Kill-session>: It means cancelling the current operations, releasing the reserved resources, letting the locks off, and terminating the connections owning by the session. If <kill session> request has been received during the execution of a commit operation, the NETCONF protocol retrieves the configuration to the previous state before the commit is executed. If We have two sessions A&B are treating the configuration data at the same time, A is not allowed to deal with some portions of the configuration which they are dedicated to B. (A) needs to treat these portions of configuration; in this case, (B) will be killed forcibly using a programmatic method by knowing this session identifier, [4].
<Commit>: It means transferring and delivering the contents of the candidate configuration database to the running configuration database.
<create-subscription>: creating a NETCONF notification subscription.
<discard-changes>: Cleaning and removing all changes from the candidate configuration data store, and making it match the running configuration data store.
<Validate>: Validating the entire contents of a configuration data store.

6. NETCONF Security

The security side is represented by a set of access rights, or permissions authorized to the user which are pieces of the NETCONF server configuration data. Transport protocol is the base to perform authentication, integrity and confidentiality during a NETCONF session. Authorizations are specified based on transport protocols such as SSH or BEEP. The configuration information are considered as sensitive information, due to they contain secret and important information, such as usernames, passwords. If the integrity procedures are not applied, then configuration information will be endangered by classic attacks, and the devices themselves are exposed to attacks, so they could destroy. The authentication procedure for the client means that the client has been proven as an authenticated identity, and that it has permissions known to the server. NETCONF operations may be able to change the access control rules during a rehabilitation NETCONF

session. The client can change the current access control rules by setting its own appropriate rules [7].

7. NETCONF capabilities

Capabilities are part of the NETCONF protocol space, or are protocol extensions; they annex the main protocol characteristics, supported by the server, and enable clients to benefit from Advantages that the devices display. Exchanging of capabilities between a server and a client starts directly when the session starts and at the same time. That each side announces its capabilities to the other by sending its announced capabilities within messages. Each of the two peers treat with the capabilities that it needs and understand and neglects the rest. One capability may depend on other capabilities, thus, to realize supporting a capability by the server; the server should support those capabilities [4] [7].

8. NETCONF configuration

Configuration data is a writable data, needed to switch a system from the elementary state to its present state. During an authorized session, and If there are several sessions supported by the server, changes on configuration attributes are obvious in all sessions. But when we have configuration attributes dedicated only to one session, it means that these attributes changes will only impact their session. configuration data includes the device sittings, ports numbers, the node status up or down, O.S version, last boot, location, supported configuration types (running or start up), interface (e.g., fast Ethernet 1/0/1), and system information. The state data is different from the configuration data, such as read-only status information, and statistics. The database with the user authentication data may be considered as a configuration data.

9. NETCONF Sub tree Filtering

In implementing the NETCONF protocol, the application can base on a method to choice some sub trees (portions of the entire configuration) and include them in the output for <get>and <get-config> operations, which are called, sub tree filtering.

In case of absence the filter, the whole configuration and state information are retrieved [7].

10. Conclusion

This paper presented the NETCONF protocol as a new solution for network management, especially configuration management. The most important weak points of SNMP protocol have been reviewed. The paper also includes an exploration of the NETCONF protocol layers, and the NETCONF protocol session. The NETCONF operations workflow has been displayed analytically. The security considerations, the NETCONF capabilities and the NETCONF configuration have been clarified precisely. We believe that NETCONF protocol is a good idea for a lot of applications in the field of Internet of Things.

REFERENCES

[1] * * * http://en.wikipedia.org/wiki/NETCONF, accessed 12 June 2014.
[2] Hui Xu, Chunzhi Wang, Wei Liu and Hongwei, NETCONF-based Integrated Management for Internet of Things, using RESTful Web Service International Journal of Future Generation Communication and Networking, Vol. 5, No. 3, September, 2012.
[3] Yanan Chang, Debao Xiao, Hui Xu, Limiao Chen Huazhong, Wuhan. *Second International Conference on Future Generation Communication and Networking, Design and Implementation of NETCONF-Based Network Management System*, China, 2008.
[4] Rui Wang, Bin Zhang, Guohui Li, Yan Li, Xuesong Gao, *The Implementation and Analysis of the Monitoring Module based on NETCONF* International Symposium on Information Science and Engieering, 2008.
[5] * * * http://libnetconf.googlecode.com, accessed 3 June 2014.
[6] * * * Netconf central, Network Configuration Protocol, http://www. netconfcentral.org/netconf_docs, accessed 19 June 2014.
[7] Enns, R., Bjorklund, M., Schoenwaelder, J., Bierman, A., *Internet Engineering Task Force* (IETF), Network configuration protocol (NETCONF), June 2011.
[8] Ji Huang, Bin Zhang, Guohui Li, Xuesong Gao, Yan Li, *First International Workshop on Education Technology and Computer Science*, Challenges to the New Network Management Protocol: ”NETCONF”, 2009.

Ph.D student Zaid Ali SHHEDI
University POLITEHNICA of Bucharest
e-mail: zaid.shhedi@yahoo.com

Prof.Dr.Ing. Florica MOLDOVEANU, Vice-dean
University POLITEHNICA of Bucharest
e-mail: florica.moldoveanu@cs.pub.ro