2010 Sixth International Conference on Networking and Services

An Empirical Study of the NETCONF Protocol

James Yu, and Imad Al Ajarmeh

DePaul University, Chicago, IL, USA

jyu@cdm.depaul.edu iajarmeh@cdm.depaul.edu

***Abstract*—This paper presents an overview and empirical study of NETCONF, which is a new network management protocol approved by IETF in December 2006. The traditional approaches of CLI, SNMP, and CORBA are discussed, along with their deficiencies in network management. In this paper we present an empirical study based on a standard NETCONF implementation. We highlight the major capabilities of NETCONF, which is a document-oriented approach based on XML, and how these capabilities could be used to address the challenges of configuration management in a complex network environment. To demonstrate the NETCONF capability, we installed an open source implementation of NETCONF, EnSuite (Yencap), on our lab Linux environment. Our preliminary results show that NETCONF provides more functionality (more advanced features), and is more efficient (single transaction for complex configuration data), more secure (embedded in the transport protocol), and easier to develop new services than CLI and SNMP.**

***Index Terms*—Computer Network Management, NETCONF, SNMP, XML-Based Network Management**

I. INTRODUCTION

THE continual growth of telecommunications and data networks in terms of size and service functions result in increased complexity of the network management process. The legacy approach of Command-Line Interface (CLI) is a vendor-dependent approach where each vendor has its own commands to perform the network management functions. There is no concept of network managers and clients in this approach. When there are more nodes on the network, the need is obvious for a central network manager to provision, configure, monitor, and trouble-shoot various network devices at different locations. However, the lack of standard CLIs prevents interoperability of equipment from different vendors where the manager of Vendor-A could administer only devices of Vendor-A, and cannot communicate with devices of Vendor-B.

Over the years, there have been many standards for network management from different organizations, such as OSI-System Management, ITU-T, IETF and OMG. Based on the acceptance of the industry, it is clear that Simple Network Management Protocol (SNMP) is probably the most successful one, as almost every vendor supports SNMP in their network equipment. In addition, there is still an active working group for SNMP since its introduction in 1988 [1]. During the past 20 years, many new functions and security measures were added to the SNMP (in SNMPv2 and SNMPv3). However, the SNMP is primarily used for network fault management and performance management, while its application in configuration management is very limited, especially in system configuration (involving multiple nodes) and service provisioning [2]. The weaknesses of SNMP lead to investigating alternative approaches to network management. This paper explores the newly approved IETF protocol, Network Configuration Protocol (NETCONF) [3] and studies its capabilities based on the EnSuite/yencap implementation [4]. A major issue of NETCONF is a lack of support from the industry, and few publications on the Netconf implementation. This paper bridges this gap and provides performance benchmarks for SNMP and a standard NETCONF implementation.

II. NETWORK MANAGEMENT REQUIREMENTS

The Open System Interconnect (OSI) network management framework specifies five functional areas for managing telecommunications networks, known as FCAPS:

1. Fault management,
2. Configuration management,
3. Accounting management,
4. Performance management, and
5. Security management.

Although there are differences between telecommunications networks and the Internet, these functional areas are still the same. The Internet Architecture Board (IAB) held a milestone workshop on network management in 2002. The fundamental core of the workshop was to establish a common ground between network operators and protocol developers. The workshop published RFC3535 [5] to guide the IETF efforts on future network management work. The recommendations and conclusions of the IAB workshop based on network operators' requirements can be summarized as follows:

1. The network management system must be easy to use for the operators who could perform the configuration of the whole network rather than individual devices.
2. The management protocol should support a standard mechanism to save and restore complete device configuration rather than individual entities.
3. Managed devices should support multiple configurations. The protocol should support the distribution of multiple configurations to devices, and then activate any

978-0-7695-3969-0/10 $26.00 © 2010 IEEE
DOI 10.1109/ICNS.2010.41

configuration. In addition, rollback between configurations should be supported.

4. The management protocol should support configuration transactions across multiple devices simultaneously in order to avoid configuration inconsistency. This function significantly simplifies network configuration tasks.
5. Device configuration should be distributed in human-readable format so that text processing tools and version control systems can be used to manage and process configuration data.
6. The management protocol should provide authentication, secured transport as well as robust access control that are integrated with the existing key and credential infrastructure.

The IAB workshop resulted in a recommendation that the IETF/IRTF should work on the development and standardization of XML-based device configuration and management technologies.

III. Network Management Approaches

A. Command-Oriented Approach

In the Command Line Interface (CLI) approach, the network administrator logins to the device, and enters commands as illustrated in Fig 1.

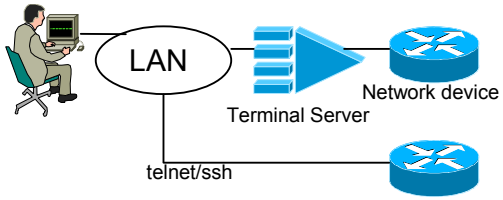

Fig 1. Management Network for CLI

If a device supports IP, the administrator can **telnet** or **ssh** to the device. If the device does not support IP or the IP interface is not configured, the administrator uses a terminal server to access the console interface of the device. To automate the configuration procedure, it is common to compile a sequence of commands in a script file and then send the script file to the device. The following example is a telnet script to show the IP routing table of a Cisco router:

```
#/bin/sh
host="192.168.1.101"
( echo <password>
sleep 1
echo "show ip route"
sleep 1
echo exit ) | telnet $host
```

A major issue with the CLI approach is the lack of standards because each vendor has its own command sets and proprietary procedure for device configuration. Also, there is no centralized scheme for network management, and each vendor has its own tool sets to address the needs. Although there are many issues with the CLI-approach to network management, it is still the most common approach to network configuration management.

B. Variable-Oriented Approach

SNMP is an IETF standard protocol. It is an application-layer network management protocol used to read and write simple variables to/from network devices. These variables have no data structures associated with them. SNMP operations involve using Get, Get-Next, and Get-Bulk requests to *read* variables on devices. It also employs the Set request to *write* (or update) variables on devices. In addition, SNMP employs Trap operations for device monitoring where the managed device sends management data upon certain events (Notifications).

The SNMP management system employs manager-agent architecture as illustrated in Fig 2. Note that the web server and client are not part of the SNMP network, and it shows a typical environment for SNMP-based network management.

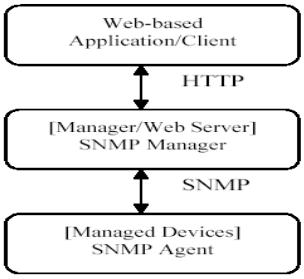

Fig 2. Web-Based SNMP System

The management data structures on the devices are defined and standardized using data structures known as Management Information Base (MIB) modules. MIB's are written in Structure of Management Information (SMI), which is a data-oriented language adopted from Abstract Syntax Notation 1 (ASN.1). It does not support advanced programming concepts such as structured data types, methods, or objects. Therefore, it is difficult to develop practical applications [2]. SNMP is widely supported on almost all managed network devices, and the standards for MIBs are comprehensive, covering almost all data network devices. The increasing complexity of modern networks exposed several serious issues related to using SNMP-Based Network Management systems. A summary of those issues are given as follows [5]:

1. SNMP does not support the retrieval of complete device configuration as a whole. As a result, it does not have the capability to compare the current running configuration with another configuration (for the same or different devices) for consistency or integrity checks.
2. SNMP suffers poor performance for bulk data transfers even for simple task as the retrieval of a routing table.
3. SNMP lacks query and aggregation mechanisms which reduce the efficiency and scalability of the protocol. SNMP Get or Get-Bulk operations can retrieve

management information from only one device at a time. To collect the same information from multiple devices, the manager needs to issue the Get/Get-Bulk request multiple times.

4. The development process of MIB modules is slow and behind the development of devices. When MIBS are released, they usually lack comprehensive documentation and description for their usage. In addition MIB modules often lack writable objects for device configuration.
5. SNMP programming interfaces is too low-level and too time-consuming; therefore, SNMP programming/ scripting is inconvenient for practical use. Tools developed based on SNMP are expensive.
6. SNMP does not employ the standard security mechanisms; instead the security is self-contained within the protocol itself which makes SNMP credentials and key management complex and difficult to integrate with other existing credential and key management systems.
7. SNMP traps do not provide comprehensive description of an event, and Get operations are usually required to collect additional information to describe the event.

Although SNMP is widely used for Fault Management and Performance Management, its capability to support Configuration Management is limited. Few, if any, network administrators would use SNMP (Set request) for configuration management. Many network management tools are still using CLI scripts, instead of SNMP Set, for configuration management.

C. Object-Oriented Approach

The limitations of SNMP (a variable-oriented approach) lead to the research of object-oriented approach to Network Management. Common Object Request Broker Architecture (CORBA) from the Object Management Group (OMG) received a lot of interests in the network community in the late 90's. CORBA is a standard that supports the collaboration of software components written in different languages on different devices. The standard includes an Interface Definition Language (IDL) which provides a formal specification of the network interfaces. IDL could be implemented in Java, C/C++, or other languages. This mechanism allows a management server to communicate with any network device, even a hand-held device that supports the CORBA agent. However, CORBA-based network is simply a portal and it requires another standard body to define management objects. We searched the RFC documents in the IETF web site, and there is ONLY one information-only document (RFC 2714). The interest in the object-oriented approach to network management is diminishing.

D. Document-Oriented Approach

The document-Oriented Approach is based on eXtensible Mark-up Language (XML) which is standardized by the World Wide Web Consortium (W3C). XML is widely used for exchanging documents of web services, and it supports several standard API's for accessing and manipulating XML documents, such as XML Schema, Document Object Model (DOM), Extensible Stylesheet Language (XSL), XSL Transformation (XSLT), and XML Path language (XPath). The XML-Based approach was introduced to solve the weaknesses of SNMP, and to satisfy the demand of managing the current complex networks. The IETF Network Configuration Working Group (NETCONF WG) is responsible for the standardization of XML-Based network management. The working group proposed a new protocol, called NETCONF, to manage diverse network devices manufactured by different vendors.

In the XML-based network management, the device configuration can be specified in an XML document, which is then exchanged between the manager and the managed device (i.e., agent). Unlike a variable-oriented approach, the XML-based approach could provision or update a complex configuration change on a device by a single transaction.

IV. NETCONF

NETCONF protocol is a major step towards an automated XML-Based network management system. It is a new management protocol that defines operations for managing network devices where configuration data can be uploaded, retrieved, and manipulated as a whole or partially. NETCONF protocol is based on a XML-encoded Remote Procedure Call (XML-RPC) to communicate between the manager and the agent. Although NETCONF is proposed for network configuration, it may also be used for network fault management [6].

A. NETCONF Architecture

NETCONF architecture is designed to distinguish between writable configuration data used to control the operation of a device and state data containing device statistics and status description. Configuration data can be retrieved by <get-config> and modified by <edit-config>, <copy-config>, and <delete-config>, whereas <get> is used to retrieve available state and configuration data [3]. In addition, NETCONF distinguishes between three configurations on a managed device [7]:

1. **Running**: configuration currently active on the device
2. **Candidate**: a standby configuration, which can be manipulated without affecting the current device's running configuration.
3. **Startup**: the initial configuration of a device

NETCONF uses a layered architecture for transmitting messages in order to provide a clear separation between management data (content) and the underlying protocol for

transporting the data. In this architecture, the protocol is divided into four layers as shown in Table I [8].

Table I. NETCONF Protocol Layers

Layer	Content and Examples
Content	Device configuration data
Operation	Operations invoked as RPC methods encoded in XML <get-config> and <edit-config>
RPC	A transport independent framing mechanism also in an XML encoding scheme. <rpc> and <rpc-reply>
Transport	Transmission protocol between agent and manager. SSH, SOAP, and BEEP

There are many advantages to using XML such as the flexibility of defining data structures, the availability of free tool kits and API's, human readability, and the ease of transport over existing secured channels.

B. NETCONF Transport Layer

IETF provides three different transport mechanisms for NETCONF to send the XML-based configuration data:

1. Secure Shell (SSH) – RFC 4742 [9] defines how to establish an SSH session to transport the NETCONF data in a secured channel. The default TCP port for the NETCONF SSH session is <830>. The support of SSH is considered mandatory for NETCONF implementation.
2. Simple Object Access Protocol (SOAP) – RFC 4743 [10] defines how to use SOAP to transport NETCONF messages. Although SOAP is a transport independent protocol, it is usually implemented on HTTP(S). Therefore, security consideration is covered in HTTPS. A major feature of SOAP is Web Service Definition Language (WSDL) file, which is the advertisement of the services. A manager could query the device WSDL file to understand the available services and use the services to build management applications.
3. Blocks Extensible Exchange Protocol (BEEP) – RFC 4744 [11] defines an application mapping of NETCONF over BEEP, which is a peer-to-peer protocol. A major feature of BEEP is to support a large number of serially connected devices, even in the face of firewall and Network Address Translators (NAT). Security is supported by the use of Simple Authentication and Security Layer (SASL).

One commonality of these protocols is that security is supported and implemented in the transport layer, and this is an important feature of NETCONF.

C. *NETCONF Implementation*

Given that SNMP is a popular and widely supported network management protocol, it is important that NETCON should interwork with the SNMP-based network environment. Two major approaches have been investigated:

The first approach is to develop a NETCONF manager to interface with SNMP agents via a mediator as shown in Fig 3. Because NETCONF is a new protocol, many legacy devices do not support NETCONF-capable agents. This approach introduces a mediator/gateway to translate between XML and SNMP data.

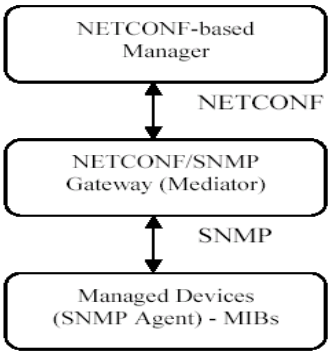

Fig 3. NETCONF and SNMP Interworking

The 2nd approach requires the installation (software upgrade) of a NETCONF agent on the managed device as illustrated in Fig. 4, and it is the ideal solution because it does not involve NETCONF/SNMP gateway for translation.

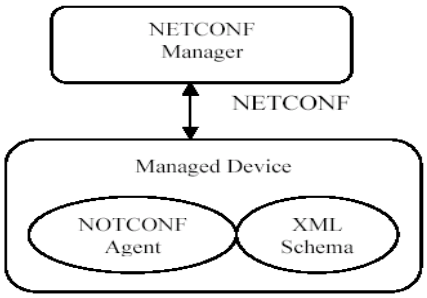

Fig 4. NETCONF Manager and Agents

V. NETCONF EXPERIMENTS

To explore the capabilities of NETCONF, we installed the open source NETCONF implementation Yencap [4] on our lab Linux environment. In order to capture the XML documents exchanged between NETCONF agent and manager, we implemented the instrumentation code to trace the XML messages sent and received on the agent. In this paper we compare the performance and functionality of NETCONF with the legacy SNMP where we used the same environment to install an SNMP agent and manager. The test environment is illustrated in Fig 5.

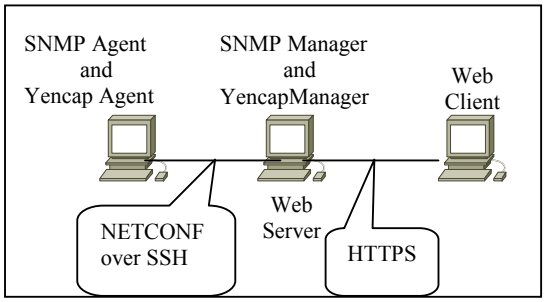

Fig 5. NETCONF Lab Test Environment

We used NETCONF and SNMP to manage VoIP SIP server (Asterisk), and we chose this particular server since we could find a NETCONF module and a SNMP MIB for this server. Our evaluation metrics will be based on the functionality, number of transactions, the overall size of exchanged messages, and the number and average size of packets.

The first experiment is to retrieve all available management information using SNMP and NETCONF. The ASTERISK-MIB used in this experiment contains 62 readable objects. Walking through the entire MIB using SNMP get-bulk results in 227 data objects from our Asterisk server. Although our ASTERISK NETCONF module has more data objects than the MIB, we customized the module to retrieve the same MIB objects for comparison. Table II shows the results of this test.

Table II. Retrieving large number of objects

	Transactions	Size (kB)	Num pkts	Avg pkt size (Byte)
SNMP	23	8.5	45	189
NETCONF	1	9.2	14	676

The second experiment is to retrieve a single data object using SNMP and NETCONF. Table III shows the results.

Table III. Retrieving a single data object

	Transactions	size (kB)	num pkts	Avg pkt size (Byte)
SNMP	1	0.152	2	76
NETCONF	1	1.460	3	486

The third experiment is to write configuration data. Using NETCONF, we are able to write any configuration objects into Asterisk configuration files. On the other hand, the available SNMP ASTERISK-MIB does not have any writable objects. This issue makes SNMP fail to support any configuration for the Asterisk server. Fig. 6 shows an example of NETCONF XML request for adding a SIP peer (i.e., a new subscriber with an assigned telephone number). Fig 7 shows the XML reply message.

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:xc="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="15">
  <edit-config>
    <error-option>stop-on-error</error-option>
    <default-operation>merge</default-operation>
    <test-option>test-then-set</test-option>
    <target>
      <startup/>
    </target>
    <config>
      <netconf xmlns="urn:loria:madynes:ensuite:yencap:1.0">
        <asterisk xmlns="urn:loria:madynes:ensuite:yencap:module:ASTERISK:1.0">
          <file name="sip.conf">
            <section xc:operation="create" name="2500">
              <attribute name="type">friend</attribute>
              <attribute name="username">imad</attribute>
              <attribute name="host">dynamic</attribute>
              <attribute name="nat">yes</attribute>
              <attribute name="secret">TDC364-imad</attribute>
              <attribute name="reinvite">yes</attribute>
              <attribute name="canreinvite">no    </attribute>
              <attribute name="disallow">all</attribute>
              <attribute name="allow">gsm  </attribute>
              <attribute name="allow">ulaw</attribute>
              <attribute name="allow">alaw</attribute>
              <attribute name="allow">ilbc</attribute>
            </section>
          </file>
        </asterisk>
      </netconf>
    </config>
  </edit-config>
</rpc>
```

Fig. 6. NETCONF request (add SIP peer)

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="15">
  <ok/>
</rpc-reply>
```

Fig 7. NETCONF reply (add SIP peer)

The forth experiment is to explore some of the new features of NETCONF that provide the network configuration process with increased security, automation, robustness and consistency. In this experiment we explored the configuration locking features of NETCONF, where a management session can completely or partially lock the configuration on one or multiple devices to ensure consistency.

To demonstrate the locking feature we modified our test environment by adding another management station as illustrated in Fig 8.

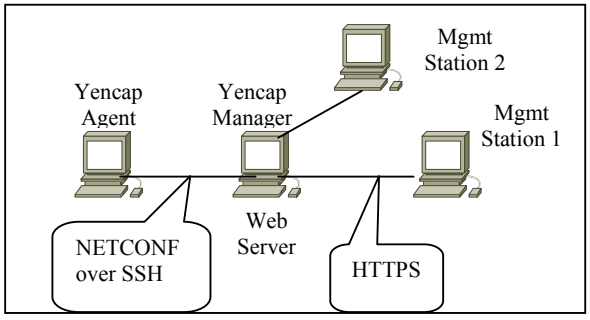

Fig 8. NCONF Lab with 2 mgmt stations

We established two separate NETCONF sessions using station 1 and station 2 (Session-1, and Session-2), and we used the same NETCONF credentials for both sessions. Session-1 locked the startup configuration of the Asterisk server, and the locking mechanism is illustrated in Fig 9. Session-2 tried to lock the same configuration again, and the operation failed as shown in Fig 10. Session-2 tried again to modify the locked configuration by trying to add a new SIP user, and the operation failed as shown in Fig 11.

```
[*************** REQUEST: START *****************]
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <lock>
    <target>
      <startup/>
    </target>
  </lock>
</rpc>
[*************** REQUEST: END *****************]


[*************** REPLY: START ****************]
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
  <ok/>
</rpc-reply>

[*************** REPLY: END *****************]
```

Fig 9. NETCONF Lock Configuration request-response

```
[*************** REPLY: START ****************]
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2">
 <rpc-error>
  <error-type>protocol</error-type>
  <error-tag>IN_USE</error-tag>
  <error-severity>error</error-severity>
  <error-message>Lock failed, lock is already held</error-message>
  <error-info>
   <session-id>1</session-id>
  </error-info>
 </rpc-error>
</rpc-reply>

[*************** REPLY: END *****************]
```

Fig 10. NETCONF Lock operation failure

```
[*************** REPLY: START ****************]
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3">
 <rpc-error>
  <error-type>protocol</error-type>
  <error-tag>OPERATION_FAILED</error-tag>
  <error-severity>error</error-severity>
```

Fig 11. Error modifying a locked configuration

As illustrated in Table II, NETCONF provides significantly better performance in terms of number of transactions and packet utilization compared to SNMP when retrieving many objects at once. The reason is that NETCONF retrieves the entire configuration using a single transaction. It sends one request (get-config) for the required object tree defined by the XML schema, and the response comes in a single XML document. This document is transferred by the underlying transport protocol with high packet utilization.

As shown in Table III, retrieving a single data object requires more bandwidth for NETCONF than SNMP and that is due to the verbose nature of XML documents in addition to the overhead associated with establishing and terminating NETCONF application and TCP transport sessions. In this extreme case, SNMP outperforms NETCONF; however, this case is not very common in network management operations. Data compression can be used to mitigate NETCONF overhead issue at the cost of some increase in CPU usage. For the configuration experiment, SNMP completely fails due to the lack of writable MIB objects. This issue is very common in many MIBs and not restricted to the ASTERISK-MIB used in this experiment.

Yencap supports many of the standard NETCONF features such as Validate Configuration, Lock Configuration, rollback, and multiple configurations on the device (startup, running, and candidate). Our lab experiments were able to validate many of these features, but we present only the Lock Configuration feature and demonstrate it in this paper. As illustrated in Fig 10 and Fig 11 a configuration locked by a certain NETCONF session cannot be locked or modified by any other sessions. This feature is very important to ensure configuration consistency, especially among several devices. SNMP lacks any configuration locking mechanisms. In addition, SNMP lacks the concept of management sessions, leaving the devises open to any SNMP requests to modify the configuration as long as they carry the correct credentials.

VI. CONCLUSION

This study provides an overview of the current approaches to network management, and identifies the major deficiencies of command-oriented and variable-oriented approaches. The new approach based on XML is considered essential in supporting the increasing complex and diverse network environment. NETCONF is the standard to address this challenge. To explore the capability of NETCONF, we conducted an experiment to study an open source implementation based on EnSuite/Yencap. The experiment demonstrates that NETCONF is more efficient, more effective, and more secure than SNMP to support various network configuration functions. NETCONF is associated with more transmission when accessing small number of objects due to XML, application and transport session overheads. This issue can be mitigated by data compression. We also identified the need for a data model to standardize the configuration information, and this issue is been addressed in YANG [12]

REFERENCES

[1] A Simple Network Management Protocol, RFC 1067, August 1988.

[2] Jürgen Schönwälder, Aiko Pras, and Jean-Philippe Martin-Flatin, "On the Future of Internet Management Technologies", IEEE Communications Magazine, October 2003.

[3] NETCONF Configuration Protocol, RFC 4741, December 2006.

[4] EnSuite Software Site, http://ensuite.sourceforge.net/yencap_user_doc.html.

[5] J. Schoenwaelder, "Overview of the 2002 IAB Network Management Workshop", IETF, RFC3535, May 2003.

[6] G. Munz, A. Anthony, F. Dressler, and G. Carle, "Using Netconf for Configuring Monitoring Probes," 10th IEEE/IFIP Network Operation and Management Symposium, April 2006

[7] Torsten Klie, Florian Muller and Stefan Fischer: "Network Monitoring with Asynchronous Notifications in Web Service Environments", in Proc. Communication in Verteilten Systemes (KIVS), Bern, Switzerland, Mar 2007

[8] Sun-Mi Yoo, Hong-Taek Ju, and James Won-Ki Hong, "Web Services Based Configuration Management for IP Network Devices", J. Dalmau and G. Hasegawa (Eds.): MMNS 2005, LNCS 3754, pp. 254 - 265, 2005.

[9] Using the NETCONF Configuration Protocol over Secure Shell (SSH), RFC 4742, December 2006

[10] Using NETCONF over the Simple Object Access Protocol (SOAP), RFC 4743, December 2006

[11] Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP), RFC 4744, December 2006

[12] YANG – A Data Modeling Language for NETCONF, draft-ietf-netmod-yang-03, January 2009.