

# NETCONF agent for link state monitoring

Diogo Loureiro<sup>1</sup>, Pedro Gonçalves<sup>2</sup>, and António Nogueira<sup>1</sup>

<sup>1</sup>University of Aveiro, DETI/Instituto de Telecomunicações

<sup>2</sup>University of Aveiro, ESTGA/ Instituto de Telecomunicações  
Campus Universitário de Santiago, 3810-193 Aveiro, Portugal  
{diogo.loureiro, pasg, nogueira}@ua.pt

**Abstract**— Fault detection mechanisms have a major importance for network managers, allowing them to respond to system failures, resolving or mitigating their effects. Network and system management technologies should include fault detection mechanisms. NETCONF is a new technology that includes monitoring capabilities and defines a fault-detection mechanism based on sending notifications. This paper documents the development of a NETCONF agent for link-state monitoring and analyzes/evaluates the most important features and capabilities of this management protocol. Additionally, it also analyzes the encoding of the monitoring messages that is performed by NETCONF and evaluates the amount of overhead introduced by the gSOAP API.

**Index Terms**— Network Management, Fault-detection, Network monitoring, NETCONF.

## I. INTRODUCTION

AVAILABILITY is a major issue for Internet providers and telecom operators, because when resources are unavailable they cannot be used by customers and can potentially generate huge losses. In order to regulate the availability of service offered to customers, agreements between customers and vendors usually define values of service availability and include minimum penalties for the suppliers whenever they do not achieve them.

In order to ensure a high availability for a resource or service, it is essential that it can be monitored to detect possible failures, thus allowing an immediate action that can minimize its downtime. Management technologies that were developed in the last decades include mechanisms for networks and systems monitoring.

SNMP (Simple Network Management Protocol) [1] is a binary technology developed in the late 80s by the IETF (Internet Engineering Task Force) and became the *de facto* management technology, despite several security and scalability flaws that were detected. SNMP enables equipment monitoring based on two different techniques: a continuous pooling of equipment configuration, which is continuously read by the manager; asynchronous information transmission, where the managed element sends sporadic information to the manager element reporting an event occurrence.

Following a strategy of network and system integrated management, the DMTF (Distributed Management Task Force) developed WBEM (Web Based Enterprise Management) [2], an approach that uses W3C (World Wide

Web Consortium) technologies for the representation and transport of management information and uses a data model that integrates management information. Similarly to the SNMP technology, WBEM enables monitoring by means of information pooling or events reception.

Later, with the emergence of service-oriented architectures, two new initiatives emerged: the Web Services (WS) Management [3] from DMTF and the WSDM-MUWS (Web Services Distributed Management-Management Using Web Services) [4] from OASIS (Organization for the Advancement of Structured Information Standards), which uses the W3C standards for web-services. Both OASIS and DMTF included monitoring support in their web services-based management technologies.

Recently, the NETCONF [5] (Network Configuration) protocol was standardized by the IETF as a new approach in network management. NETCONF uses XML (Extensible Markup Language) for encoding, provides support for several secure transport protocols and uses YANG [6], a language that was specifically created to be used with this protocol, to describe management information.

This paper describes the implementation, and the corresponding performance evaluation, of a network link-state monitor based on NETCONF. The monitor allows event subscription by the element manager, while also allowing the agent that is responsible for monitoring the link status to send events to the manager element. The implementation was based on the gSOAP platform, since it automatically generates the communication interfaces code based on the provided data models. Several functional tests were conducted, as well as tests to evaluate the notifications transmission time and the signaling traffic between manager and agent.

The organization of this article is as follows. Section II gives an overview of the fault management technologies relevant for this work. Section III presents the management scenario that is envisaged for the proposed framework. Section IV presents the proposed implementation of the network monitoring agent, using SOAP (Simple Object Access Protocol) for transport. Section V presents and discusses the results that were obtained from the evaluation tests. Finally, Section VI presents some conclusions about the NETCONF performance and its feasibility for the future management and network monitoring.

## II. TECHNOLOGY OVERVIEW

During the last decades, several network and system

management technologies that provide monitoring support have been standardized. This section presents some relevant details of the most important ones.

#### A. SNMP

SNMP is a management technology standardized in the late 80s by the IETF and has been widely used by the academic and industry communities [7], becoming the *de facto* standard for network management. It uses SMI (Structure of Managed Information) for management data description, through a relational model using MIBs (Management Information Base). SNMP provides operations to read (*GET*, *GETNEXT*, *GETBULK*) and write (*SET*) the MIBs and also for asynchronous (*Trap*) and synchronous (*Inform*) event notifications.

This standard has been successively revised, especially due to security issues. Due to its inability to manage complete configurations, the use of UDP (User Datagram Protocol) for transport, the table organization of the data and the late and complex security features, the SNMP technology was not applied according to its original role, being mainly adopted for monitoring and fault management tasks.

#### B. WBEM

WBEM is a DMTFs unifying standard for the enterprise management area. It uses an object oriented data model named CIM [8] (Common Information Model) for information representation, encodes the management information in CIM-XML [9] and transports management information using the HTTP protocol [10]. WBEM follows a client-server architecture and is composed by four components: a WBEM client that usually runs as a graphical application and interfaces the human administrator; a CIM Object Manager (CIMOM) that acts as the management server and keeps the management information in a repository; providers, who are responsible for interfacing with the management elements and implementing their specific communication details; the managed elements.

WBEM technologies have a broad support: there are several open-source implementations, as well as a huge number of commercial WBEM products.

WBEM allows equipment monitoring through a pooling process where the CIMOM continuously queries the provider about some management element or by using an asynchronous event dispatching process, where the provider notifies the CIMOM that an event has occurred. According to the WBEM terminology [11], event notifications are called *Indications* and represent the occurrence of some event, such as the deletion of an element or the exceeding of a threshold. To subscribe *Indications* the manager has to create an instance of the *IndicationSubscription* class that references instances of an *IndicationFilter* class, set the filter rule to select the objects of the indication and an instance of an *IndicationHandler* class setting the encoding and transport definitions of the indication.

#### C. WS technologies

Following the SOA (Service Oriented Architectures) trend, DMTF and OASIS proposed, respectively, the WS-

Management [3] and the WSDM [12] approaches. They based their efforts in the W3C standards for web-services, like SOAP and WSDL (Web Service Description Language), to define solutions that can remotely access and exchange management information in distributed environments.

The event notification mechanism implemented by WS-Management is very similar to WBEM notifications; the only difference refers to the inclusion of an unsubscribe message that is sent by the indication consumer when it no longer wants to receive indications.

OASIS WSDM was divided in two sub-standards: MOWS (Management Of Web Services) and MUWS. MUWS implements monitoring by sending event information but, unlike WS-Management, it does not plan a subscription process. Albeit less flexible, WS-Management shows better performance than WSDM due to a simpler specification and a small number of operations [13].

#### D. NETCONF

NETCONF is a network management technology standardized by the IETF Management and Operations workgroup. It follows a client-server architecture and is conceptually composed by four layers: a transport layer that must provide information transport and includes the security features of the protocol, providing definitions for the use of SSH [14] (Secure Shell), SOAP [15], BEEP [16] (Blocks Extensible Exchange Protocol) or TLS [17] (Transport Layer Security), with the SSH implementation being mandatory; a Remote Procedure Call (RPC) layer that uses XML encoding to provide a transport independent data exchange; the operation layer, which includes the set of NETCONF operations that can be invoked by the RPC methods; the content layer that includes the device configuration data.

NETCONF was designed to distinguish between status and configuration data, providing operations as *<get-config>* or the more generic *<get>* for data retrieval. For manipulating the management objects, operations like *<edit-config>*, *<copy-config>* and *<delete-config>* are available.

NETCONF devices support multiple configurations: the *startup* configuration should to be applied during the boot process; the *candidate* configuration can be freely manipulated, with no consequences to the device operating status; the *running* configuration represents the active configuration of the device.

NETCONF event notifications [18] are implemented by asynchronous messages, named *notifications*, that are sent from the NETCONF agent to the manager. The manager can subscribe notifications of events streams using a synchronous message named *subscription*: when an event occurs, the agent sends the notification to the stream subscriber. For better granularity, the manager is able to set a filter to the subscribed stream, avoiding receiving unwanted notifications and, thus, improving the overall performance. Figure 1 illustrates the event notification model.

A reduced version of the original NETCONF protocol, named NETCONF Light [19], which includes a subset of the original protocol functionalities, was proposed for devices

with limited computing resources, particularly those with low memory. Regarding the differences from the original protocol, we can highlight that NETCONF light operations lack support for configuration filtering functionalities. Despite maintaining the original protocol operations, the light version removes the possibility of defining a filter that limits the operations scope over the equipment configuration.

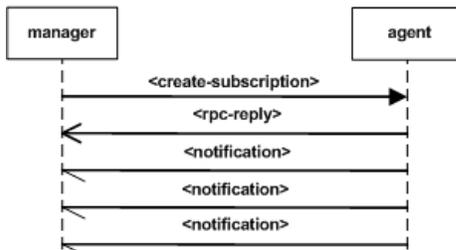
This version requires a small number of sessions and only one data repository, the running configuration. The *get*, *get-config*, *copy-config*, *lock*, *unlock*, *close-session* and *kill-session* operations are mandatory. The *delete-config* operation no longer makes sense, since there is only the running configuration. Filtering is optional, since it is a very resource consuming feature. Consequently, the *edit-config* operation

may not be supported when managing only complete configurations, which should not be considered as a problem since it is probably a reduced size configuration.

Table 1 summarizes the key features of the analyzed management technologies. Since our application scenario belongs to the network management area and the standardization entity (IETF) developed a new technology (NETCONF) that has been receiving a tremendous attention from academia and industry, we chose NETCONF as the basis for the development of our monitoring solution. Moreover, this technology offers a tremendous flexibility, allowing to perform notification subscription and avoiding unwanted notifications, while keeping a centralized agent configuration.

**Table 1 – Comparison between existing fault management technologies**

	SNMP	WBEM	WS-Management	WSDM	NETCONF
Standardization entity	IETF	DMTF	DMTF	OASIS	IETF
Year	1990	1997	2008	2005	2006
Scope	Network management	Enterprise management	Enterprise management	Enterprise management	Network Management
Subscription support	No	Yes	Yes	Yes	Yes
Messages	Trap, inform- response	Subscription, indication	subscription, indication	subscription, notification	subscription, notification
Message transport	UDP	SOAP (HTTP/HTTPS)	SOAP (HTTP/HTTPS)	SOAP (HTTP/HTTPS)	SSH, BEEP, SOAP, TLS
Event selection	UUIDs	WQL/SQL filter	WQL/SQL filter	XPath filter	Subtree or XPath filter
Subscription termination	No	Timeout	Timeout, Unsubscribe message	No	Timeout
Usage	Network equipment	Desktop, networking	Desktop, networking	Desktop, networking	Network equipment



**Fig. 1 – NETCONF event notification model**

### III. MANAGEMENT SCENARIO

Currently, many configuration management solutions have a layered approach with three layers: the Service Layer that defines concepts related to the day-to-day business flows of a service provider, using service models typically defined using SID (Shared Information/Data Model), UML (Unified Modeling Language) or proprietary languages; the Resource Layer that provides a mapping from the Service Layer to actual device manipulations, modeling individual devices (like switches, routers or DSLAMs) using XML, UML or proprietary languages; the Mediation Layer that maps changes to the local data structures in the Resource Layer to actual configuration change commands on the devices.

NETCONF intends to simplify all these layers by defining how to execute configuration changes, by using stringent

YANG models to define device configurations and using technologies such as XMLBeans, Castor, Xgen or JAXB to obtain a set of Java classes that can be used to manipulate the configuration instances.

In fact, as already said, NETCONF is an XML-based protocol specifically designed to configure and manage the most demanding network situations by providing automated configuration management, improved network security and reliability, and robust configuration changes. NETCONF actions are mandatorily communicated across the network in a secure way.

Figure 2 depicts a hypothetic NETCONF deployment scenario, where different network configurable devices can be centrally configured/controlled using several transport protocols. Setting up routing parameters of network routers or the security values of firewalls, while monitoring their behavior, can be efficiently performed using a remote centralized server. Besides, depending on the capabilities of monitoring probes, their configuration can comprise parameters for flow metering and aggregation, packet sampling, and/or the export of monitoring data.

The use of a device-specific command line interface (CLI) or a configuration file can be cumbersome and complicated, especially if used in heterogeneous networks consisting of different device models. The NETCONF standard simplifies

all these tasks, while assuming that real operating networks are composed by various devices from diverse vendors.

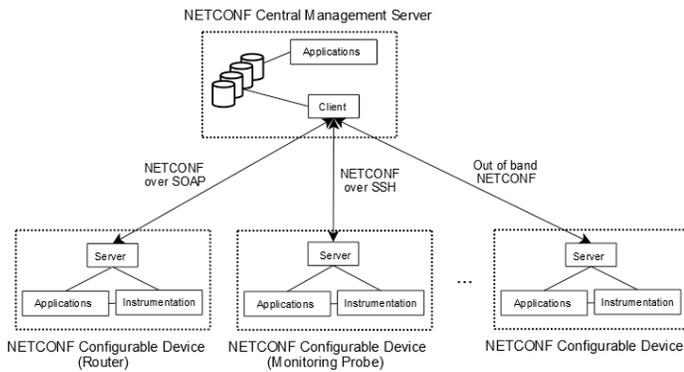


Fig. 2. NETCONF deployment scenario

#### IV. NETWORK MONITORING AGENT

The monitoring solution followed RFC 5277, and NETCONF over SOAP transport was the chosen approach. The NETCONF light version was chosen due to its applicability in resource-constrained devices, and a subset of the operations defined in the standard were developed.

In order to minimize the overhead created by SOAP, we choose gSOAP since it offers better performance than other web service frameworks [20, 21]. For the same reason, we decided to implement our software using the C language, thus reducing the required computational resources.

The gSOAP framework generates NETCONF SOAP communication based on XML Schemas provided by RFC 4147 for the NETCONF base operations and RFC 5277 for the event notifications operations. Additionally, it supports RPC-XML and asynchronous message exchange, which are essential features for this implementation.

Figure 3 illustrates the subscription creation process that was implemented: the manager performs a subscription creation and sends it to the agent which, after validating the request, sends back the answer to the manager. Accepted subscription requests lead to the creation of independent threads: the manager creates a listener thread to receive notifications, while the agent creates a monitoring thread to detect link failures.

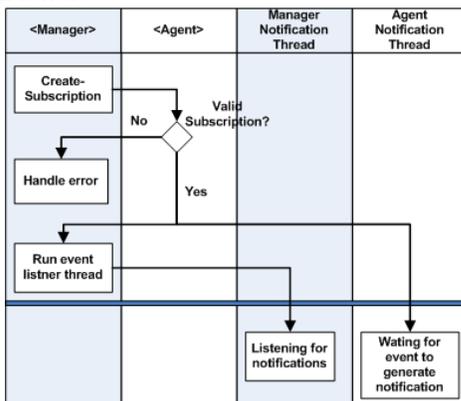


Fig. 3 – <create-subscription> process

Unlike the <create-subscription> message, <notification>

messages are asynchronous. Once a link failure event is detected, the agent sends a notification message to the managers that subscribed the event stream.

#### V. RESULTS ANALYSIS

In order to validate the developed management solution, some functional tests were conducted. Additionally, in order to evaluate the applicability of the proposed solution to a real network, we also evaluated the signaling overhead, the protocol encoding efficiency, the response time and the memory usage level.

Tests were conducted in a machine with an Intel Mobile Core 2 Duo, at 2.2GHz and having 2GB of RAM, with a native installation of Ubuntu 10.10. The traffic that was generated during the experiments was captured and analyzed. Figure 4 illustrates the <notification> message format corresponding to a link-up event.

```
<ns2:notification xsi:type="ns2:NotificationType"> <ns2:eventTime> 2011-10-17T17:16:27Z</ns2:eventTime><ns2:info>link up</ns2:info> </ns2:notification>
```

Fig. 4. NETCONF notification

The traffic analysis allows us to evaluate the influence that the protocol can have in network performance. So, network data regarding the number of packets and number of bytes transferred during the monitoring operations was gathered. Several link-down and link-up events were caused and all messages exchanged during the communication between the manager and the agent were captured.

Figure 5 depicts the traffic analysis results. Both the total number of packets and the amount of signaling data increased with the number of events reported from the agent. The increase was more or less linear, except for small numbers of events where the increase is exponential. Each notification message produces two packets and generates 935 bytes.

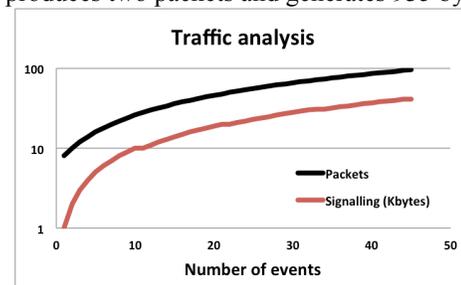
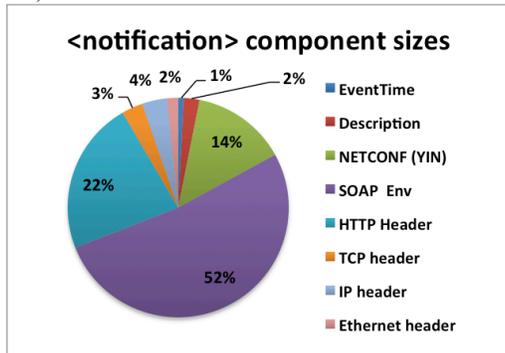


Fig. 1 – Traffic Analysis

A deeper inspection to the captured traffic confirms the verbose characteristic of NETCONF, which was already visible in Figure 4. NETCONF technology encodes notifications in a 869 bytes Ethernet packet, which is acknowledged by a TCP ACK packet.

If we measure the individual sizes of the message components (Figure 6), we can see that useful information is less than 4% of the total information that is sent in the network packet. In addition, we note the preponderance of the component related to the SOAP envelope, which represents more than 50% of the information contained in the Ethernet

packet. Although a high overhead generated by the SOAP framework was expected, its dimension is quite impressive, being even higher than the sum of the HTTP component with the component associated to the NETCONF data description (YIN).



**Fig. 6 - Individual component sizes of <notification>**

In order to evaluate the response times of the monitoring system, we changed the agent code to generate a higher number of event notifications. The time that is necessary to send notifications increased with the number of notifications sent by the agent. The conducted tests have also shown that the time required to send a notification is approximately equal to 200 ms.

## VI. CONCLUSIONS

This paper documents the implementation of a network-monitoring agent. Several management technologies that support fault-detection were analyzed and NETCONF was chosen as the basis for the development.

NETCONF over SOAP transport was chosen and the implementation was based on the gSOAP platform. The gSOAP platform automatically generates the communication interfaces code based on the provided data models, which greatly facilitates the development process.

Several functional tests were conducted, as well as tests to evaluate the notifications transmission time and the signaling traffic between manager and agent. The time to produce notifications seemed to be appropriate, although the message size was enormous.

NETCONF encoding imposes a very considerable amount of overhead on monitoring signaling: although it was somehow a predictable result, the amount of overhead was very impressive. By adding the contribution of all components related to the NETCONF technology, we could easily achieve a percentage of signaling due to NETCONF higher than 80%. The SOAP envelope size and its weight in the overall signaling (52%) represents a very considerable overhead for the management platform. This effect could be mitigated by the use of a tailored [22] SOAP implementation, allowing slightly better results.

There are several comparative studies between management technologies, but typically looking at their performance in configuration management scenarios [7, 23, 24] and not in network monitoring tasks. In these scenarios the overhead associated to NETCONF verbosity is somehow compensated

by the gain that is obtained with NETCONF filtering mechanisms, allowing NETCONF technology to perform better than SNMP for higher volumes of management information. Monitoring scenarios are substantially different, because they have many small size messages, which suggests that NETCONF performance is worse than the SNMP performance.

As future work, we are planning to test other NETCONF transport alternatives in monitoring scenarios and compare its performance with other fault detection technologies.

## REFERENCES

- [1] J. Case, M. Fedor, M. Schoffstall, and J. Davin, Simple Network Management Protocol (SNMP), RFC 1157, 1990.
- [2] J. P. Thompson, "Web-based enterprise management architecture", Communications Magazine, IEEE, vol. 36, pp. 80-86, 1998.
- [3] DMTF, Web Services for Management (WS-Management), 2005.
- [4] I. Sedukhin, Web Services Distributed Management: Management of Web Services (WSDM-MOWS) 1.0., 2005.
- [5] R. Enns, NETCONF Configuration Protocol, RFC 4741, 2006.
- [6] M. Bjorklund, YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF), RFC 6020, 2008.
- [7] L. Andrey, O. Festor, A. Lahmadi, A. Pras, and J. Schönwälder, "Survey of SNMP performance analysis studies", International Journal of Network Management, vol. 9999, p. n/a, 2009.
- [8] DMTF, "Common Information Model (CIM) Specification - Version 2.28", 2011 2011.
- [9] DMTF, Specification for the Representation of CIM in XML, 2002.
- [10] DMTF, Specification for CIM operations over HTTP version 1.2, 2007.
- [11] DMTF, CIM Event Model White Paper - DSP0107, 2003.
- [12] W. Vambenepe, Web Services Distributed Management: Management Using Web Services (MUWS 1.0), 2005.
- [13] G. Moura, G. Silvestrin, R. Sanchez, L. Gasparly, and L. Granville, "On the Performance of Web Services Management Standards - An Evaluation of MUWS and WS-Management for Network Management", in 10th IFIP/IEEE International Symposium on Integrated Network Management (IM 2007) Munich, Germany, 2007, pp. 459-468.
- [14] M. Wasserman and T. Goddard, Using the NETCONF Configuration Protocol over Secure Shell (SSH), RFC 4742, 2006.
- [15] T. Goddard, Using NETCONF over the Simple Object Access Protocol (SOAP), RFC 4743, 2006.
- [16] E. Lear and K. Crozier, Using the NETCONF Protocol over the Blocks Extensible Exchange Protocol (BEEP), RFC 4744, 2006.
- [17] M. Badra, NETCONF over Transport Layer Security (TLS), RFC 5539, 2006.
- [18] S. Chisholm and H. Trevino, NETCONF Event Notifications, RFC 5277, 2008.
- [19] V. Perelman, J. Schoenwaelder, and M. Ersue, Network Configuration Protocol for Constrained Devices (NETCONF Light), draft-schoenw-netconf-light-00.txt, 2011.
- [20] M. R. Head, M. Govindaraju, A. Slominski, L. Pu, N. Abu-Ghazaleh, R. van Engelen, K. Chiu, and M. J. Lewis, "A Benchmark Suite for SOAP-based Communication in Grid Web Services", in Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference, 2005, pp. 19-19.
- [21] M. Govindaraju, A. Slominski, K. Chiu, P. Liu, R. van Engelen, and M. J. Lewis, "Toward characterizing the performance of SOAP toolkits", in Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on, 2004, pp. 365-372.
- [22] K. Chiu, M. Govindaraju, and R. Bramley, "Investigating the limits of SOAP performance for scientific computing", in High Performance Distributed Computing, 2002. HPDC-11 2002. Proceedings. 11th IEEE International Symposium on, 2002, pp. 246-254.
- [23] P. Gonçalves, J. L. Oliveira, and R. L. Aguiar, "An evaluation of network management protocols", in 11th IFIP/IEEE International Symposium on Integrated Network Management (IM 2009), New York, USA, 2009.
- [24] A. Pras, T. Drevers, R. v. d. Meent, and D. A. C. Quartel, "Comparing the performance of SNMP and Web services based management", IEEE electronic Transactions on Network and Service Management, vol. 2, Nov. 2004.