Performance Improvement Methods for NETCONF-Based Configuration Management*

Sun-Mi Yoo[1], Hong Taek Ju[2], and James W. Hong[3]

[1] Samsung Electronics, Korea
sunmi.yoo@samsung.com
[2] Dept. of Computer Engineering, Keimyung University, Korea
juht@kmu.ac.kr
[3] Dept. of Computer Science and Engineering, POSTECH, Korea
jwkhong@postech.ac.kr

Abstract. IETF's NETCONF WG has taken efforts in standardizing configuration management protocol, which allows high interoperability of configuration management. In addition to interoperability, high performance is also required in configuration management, but many researches have often discarded the performance aspect of it. In order to fill that void, this paper proposes methods to improve performance with layers of NETCONF. It demonstrates and compares the performance evaluations to verify the efficiency of the proposed methods. This evaluation can be used as a guideline to effectively implement NETCONF. Moreover, it also demonstrates the process of performance evaluation of configuration management.

1 Introduction

The network configuration management sets up operation values of devices that constitute the network and collects and analyzes the values. For example, it sets up routing parameters of routers or security values of firewalls and monitors the values. A centralized server manages various remote devices for configuration management, which is essential on current networks. IETF has proposed the Network Configuration (NETCONF) [1] standard for configuration management of remote devices. The NETCONF standard [2] assumes that the current network is composed of various devices from diverse vendors. These standards can formulate the remote configuration management more effectively.

Along with interoperability, the efficiency of configuration management is an important factor to consider. Since the configuration management is carried out against many devices, the efficiency is required thus more. For instance, when monitoring the values of many devices, an efficient process is mandatory to achieve

* This research was supported in part by the MIC (Ministry of Information and Communication), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2005-C1090-0501-0018) and by the Electrical and Computer Engineering Division at POSTECH under the BK21 program of the Ministry of Education, Korea.

Y.-T. Kim and M. Takano (Eds.): APNOMS 2006, LNCS 4238, pp. 242 – 252, 2006.

correct values. Moreover, fast configuration management is essential to avoid causing distress in networks when modifying operation values of devices. The NETCONF standard has been explored for functional requirements in various ways, whereas there is little discussion on the efficiency issue. So far, the performance enhancement methods for configuration management using NETCONF have yet to be proposed in any literature. Furthermore, NETCONF performance evaluation result and its analysis, along with the methods for efficient use of NETCONF have not been discussed, while existing technical reports on NETCONF implementation do not report on its performance.

This paper proposes several methods that improve NETCONF protocol performance. NETCONF can be conceptually partitioned into four layers by its functions. This paper considers three core layers that have a great effect on NETCONF performance; the application protocol layer, the RPC layer and the operations layer. These three layers are used to propose methods to improve performance in each layer and the performance is evaluated to verify the effect of the methods. The proposed methods include the ones that follow the original NETCONF standards, as well as the ones that do not. In addition, since the method of approaching by layers is applied, these methods can be adopted to various configuration environments.

To achieve statistical significance, we have measured the response time, network usages, and computer resource usages of each layer. Our methods manage smallest network usages to reduce the response time and the load on network, which produces rapid configuration management. Furthermore, computing resource usages are reduced to enable the manager system to manage many devices, and the NETCONF agent system to apply to the embedded devices.

The remainder of this paper is organized as follows. Section 2 introduces NETCONF implementations and research of network management performance. Section 3 presents the testing environment and architecture of XCMS used for the test. Section 4 presents the performance evaluation results of transport protocol layer. Section 5 presents the performance evaluation results of RPC protocol layer. Section 6 presents the performance evaluation results of operations layer. Finally, we summarize our work and discuss future work in section 7.

2 Related Work

In this section, we present implementations of XML-based network configuration management using NETCONF by various earlier works. We also present related work on network management performance.

2.1 Implementations of NETCONF

The NETCONF standard is an early stage that does not have many implementations yet. This section briefly introduces a system that is implemented by our previous work and an open source project called EnSuite [7].

In our previous work, we have developed an XML-based configuration management system, XCMS [8], which implemented the first internet draft of IETF NETCONF for IP network devices. XCMS used the XPath [8], which has been standardized since the fourth draft and SOAP over HTTP [3] as a transport mechanism. XCMS supports the latest internet draft of IETF NETCONF protocol [2] as well as three transport protocols; SOAP over HTTP [3], BEEP [5] and SSH [4]. In XCMS, a centralized manager controls the configuration information of network devices equipped with NETCONF agents. XCMS can manage configuration information of diverse devices depending on the proprietary operating systems of the vendors.

EnSuite [7] is a network management platform prototype based on NETCONF and is an open source project from LORIA-INRIA in France. LORIA-INRIA had developed a NETCONF agent called YENCA, in their previous work. YENCA was implemented by C language but had limited functions. LORIA-INRIA then developed a new configuration management system, EnSuite, which has improved upon its functions and architecture. EnSuite consists of a NETCONF web-based manager, a NETCONF agent and a set of extension modules that are implemented in Python.

2.2 Research on Network Management Performance

This section discusses research work that focus on the performance of network management using XML technologies.

Aiko Pras et al have presented Web Services for management on the performance differences between SNMP and Web Services-based management [13]. To compare their performances, they investigated bandwidth usage, CPU time, memory requirements, and round trip delay. To conduct the tests, they implemented several Web Services-based prototypes and compared their performance to various SNMP agents. These tests showed that there is a significant difference in the bandwidth requirements of SNMP and Web services. They concluded that SNMP is more efficient for cases where only a single object is retrieved although Web Services-based management is more efficient for larger number of objects. Thus, Web Services-based management is more suitable for large scale networks.

Another interesting study on the performance of network management has been conducted by Apostolos E. Nikolaidis et al [12]. This study mentions that the Universal Plug and Play (UPnP) protocol undertakes the Lan configuration and management. Due to relatively high bandwidth and the limited number of devices in this paper, the traffic issues are of secondary importance. They examined the unnecessary management traffic volume that may occur due to the verbose nature of XML technology, used by the protocol. Their solution exploits some capabilities provided by the protocol and uses real-time compression in order to effectively reduce the management traffic, while keeping the response times at the same or even lower levels. The solution mainly comes from the application of the Lempel-Ziv compression algorithm, with minimal additions in the proposed DSL Forum standard. They evaluate the performance and usability of the solution, implementing a typical configuration example with the CPE WAN management protocol [12].

3 Measurement Environment and Implementations

We applied our XML-based configuration manager and agent, XCMS, as NETCONF implementations. Our XCMS manager and agent are connected by a 100Mbps Ethernet and run on Linux servers with Pentium IV 2.4GHz CPU and 512MB RAM. The XCMS agent manages network information of Linux system configuration; 'interface'. For the statistical significance, we have measured the response time and memory usage of XCMS manager and agent for 1000 times and averaged the results.

The NETCONF standards specify a communication process of the XCMS manager and the agent as follows: First, the manager attempts to correspond with the agent. When the session is opened, each peer (both the manager and the agent) must send a <hello> element containing a list of the peer's capabilities. If the <hello> element does not have any wrong elements, the session is a NETCONF session and a manager begins sending NETCONF request messages. Each transport protocol has a different process for opening a NETCONF session. Hence, we only measured the communication between NETCONF request messages and response messages.

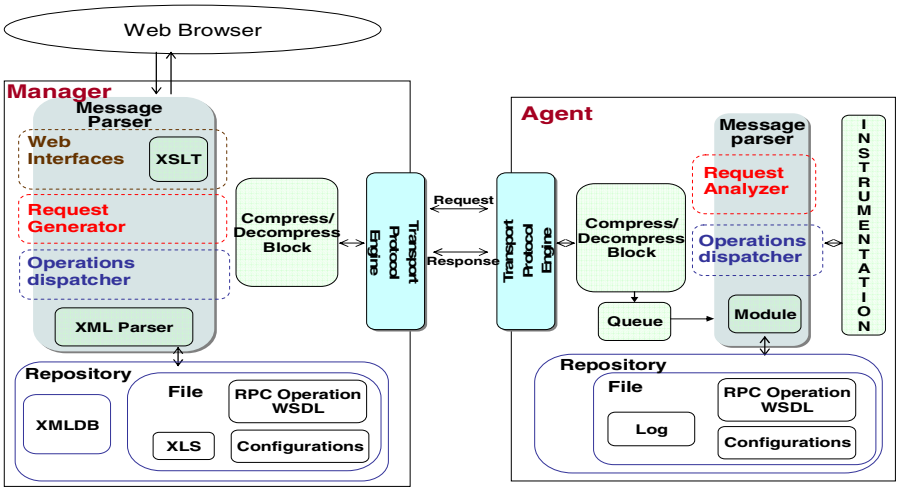

Fig. 1. Architecture of XCMS

As mentioned earlier, the experiments are performed by using our XML-based configuration management system; XCMS. Figure 1 illustrates the architecture of XCMS consisting of a manager and an agent. The XCMS manager and the agent are implemented in C. They use ZLIB for compression and decompression and open source implementations for transport protocol. They use gSOAP implemented with C/C++ languages to exchange SOAP over HTTP RPC messages and Roadrunner implemented with C/C++ languages for BEEP and a port forwarding method for SSH. We used the libxml, which is the lightweight one among existing XML parsers as an XML parser in the manager and the agent. The XCMS agent is implemented with the latest NETCONF drafts and we have added our proposed solutions to XCMS.

4 Transport Protocol Layer

The transport protocol layer of NETCONF provides a communication path between the manager and the agent. The NETCONF protocol allows multiple sessions and transport protocols. The NETCONF protocol [2] is currently considering three separate transport protocol bindings for transport; Secure Shell (SSH) [4], Block Extensible Exchange Protocol (BEEP) [5] and SOAP over HTTP [3]. Both the functions and the performance of the protocols are examined to select a transport protocol for configuration management. This section demonstrates how the performance of transport protocol affects the overall performance, and proposes a mechanism to improve the application protocol layer. Finally, the performance is evaluated to verify the effect of the proposed mechanism. We have performed several 'get-config' operations, which read configuration information of various sizes to check the transport protocol performance. The manager constructs a request message, which performs 'get-config' operation with a subtree filtering and sends the message to the agent. We operated the network interface configuration management for this experiment. The number of network interface is increased in intervals of 2 due to the changed information on sizes.

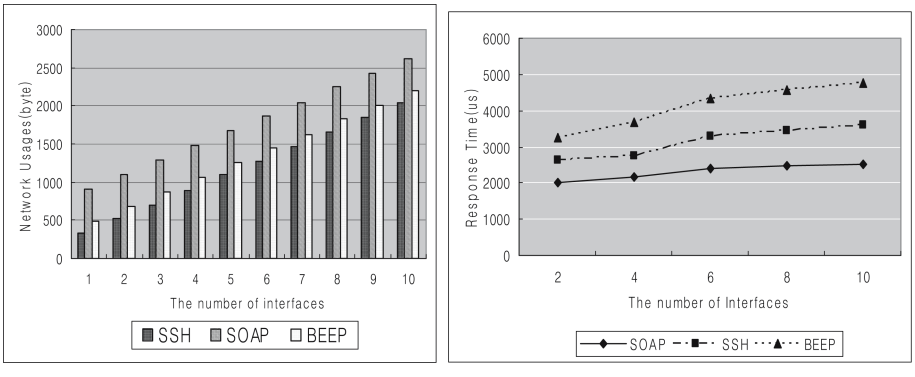

Fig. 2. Network usages and Response time by Transport Protocols

The network usages and the response time have been measured to examine NETCONF transport protocol performance. First, each transport protocol's network usage is examined. SOAP over HTTP has the heaviest network usages, whereas SSH has the least, as illustrated in Figure 2 left graph. Both SOAP over HTTP and BEEP are based on XML technology, and they both append extra messages for header contents. Despite the fact that a message conveys only one parameter and one value, the size of the corresponding XML message is bigger.

Next, by using each transport protocol, the response time of configuration management is measured. In our analysis, the response time is defined as the time interval between sending and receiving a NETCONF message. Although it was predicted that the result of the response time would be similar to the network usage, it has been found to differ, as Figure 2 right graph demonstrates.

The implementation methods are considered as the reason for such result. The XCMS manager and agent use the port forwarding mechanism for SSH. It is a simple implementation mechanism but due to the port forwarding process, some overhead occurs. However, the actual time difference is very little in the consideration of the unit of time and the time is round trip time.

Although the response time of the three transport protocols are similar, the network usages are quite different. When the network usages in configuration management have an immense effect on the network traffic, a mechanism for reducing the sizes of NETCONF messages is needed. For solution, we used the compression method since the repetitive nature of text patterns in typical XML message are produced by NETCONF. Moreover, many papers have also proposed the compression mechanism for reducing the sizes of XML. We have compressed payloads using the ZLIB library.

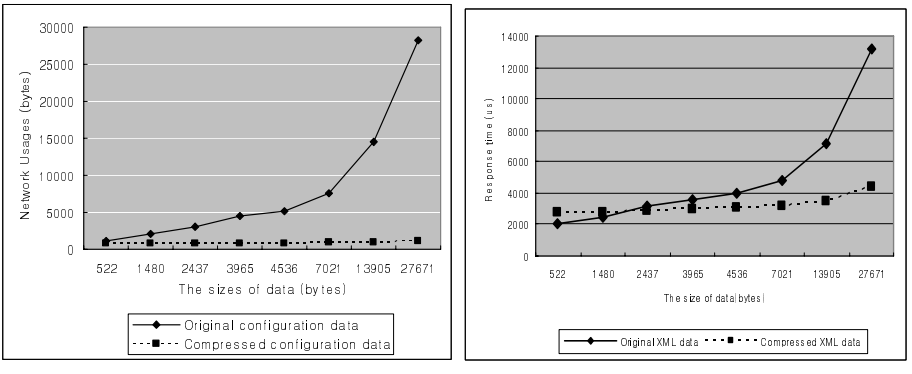

Fig. 3. The Network Usage and Response time of NETCONF Messages Compression

Experiments were conducted with the increasing sizes of messages to test the effect of the compression mechanism. The response time and the network usage were measured. Figure 3 left graph illustrates that the size difference between the original data and the compressed data increases as the size of the data increases. Figure 3 right graph shows that the response time depends on the network usage. Although the process for compressing data produces overhead time, the large size data can ignore this fact, as the number of packet fragmentations reduces.

5 RPC Layer

The NETCONF uses an RPC-based communication model. The RPC layer provides a simple, transport-independent framing mechanism for encoding RPCs. The <rpc> element is used to enclose a NETCONF request sent from the manager to the agent. Next, the <rpc-reply> element encloses a NETCONF message sent in response to the <rpc> operation on the RPC layer. The <rpc> element can only have a NETCONF method and should have one-to-one communication with the response message. The NETCONF provides two mechanisms for operating commands, with no pipelining and pipelining. With no pipelining, a NETCONF manager waits for a response

message of the previous request before sending next request message. This mechanism has some inter-request time and low efficiency. Therefore, NETCONF provides pipelining in order to lower the elapsed time and to improve the efficiency. The pipelining mechanism serially sends request messages before the previous requests have been completed, instead of waiting on a response with pipelining. Furthermore, the NETCONF agent must send the responses only in the order the requests were received. We investigate the effect of pipelining and propose a mechanism for improving the efficiency on RPC layer.

We increased the number of requests containing a NETCONF command to measure the effect of the pipelining mechanism. The command processes the 'get-config' operation to obtain interface information and has a response message of around 465bytes. The response time is measured by SSH and the other protocols have the same result. Figure 4 demonstrates that the performance of pipelining is better than the one of no pipelining. However, the pipelining mechanism has some risks. The configuration management process could be destroyed if a request message is corrupted during processing.

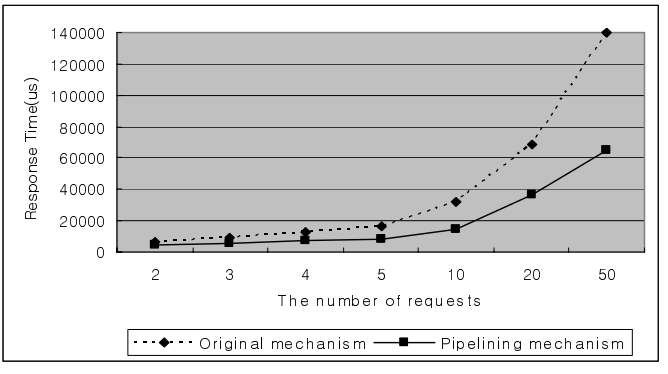

Fig. 4. No Pipelining/Pipelining response times (μs)

The pipelining mechanism does not improve the network usage performance. The number of packets at a point of time on the network increases compared to the case without the pipelining mechanism. The NETCONF protocol draft states that the <rpc> element only has the NETCONF operation for its sub element. However, several NETCONF operations related with each other are needed for a completed configuration management. For instance, in order to complete setting the configuration, operations for obtaining the current data as well as for setting the new data are required.

We propose the Multi-Command mechanism for improving the efficiency of network usage. Multi-Command is a NETCONF request message with several commands under an <rpc> element. For example, the following three operations can be put into a <rpc> element with Multi Command: an <edit-config> operation to <candidate> data, <commit> operation for applying the <candidate> data to the <running> data and <get-config> operation for checking the result of previous operations. An agent sequentially provides the Multi-Command mechanism processes

on the request operations and creates a response. This mechanism also has some risks that are similar to the pipelining mechanism. If any errors occur while processing a request message, all operations of the request message are canceled by the <roll-back> function.

The network usages and the response time have been measured to verify the effect of the Multi-Command mechanism. We have performed a similar process to the experiment of the pipelining mechanism and compared to three communication mechanisms on a RPC layer. Three protocols show similar results of using the Multi-Command mechanism, which does not have a great effect on SSH, since it adds a small header to a payload. However, SOAP over HTTP and BEEP show larger difference of network usages. Moreover, the communication process of the multi command mechanism and the pipelining mechanism is similar. This mechanism also has no inter request time but has a little additional processing time.

We have compared our results with the pipelining of HTTP. The NETCONF pipelining has similar effect to the HTTP pipelining on response time. The requests are buffered before transmission so that multiple HTTP requests can be sent with the same TCP segment. As a result, HTTP pipelining does not generate unnecessary headers and reduces the number of packets required to transmit the payload. On the other hand, the NETCONF pipelining uses the RPC communication and transfers a request message containing an operation at once. The NETCONF pipelining cannot reduce the number of packets or network usages. The proposed Multi Command mechanism uses the RPC communication and transfers a request message containing several operations at once. Therefore, this mechanism can reduce the number of packets and the total network usages similar to the HTTP pipelining.

6 Operation Layer

The NETCONF operations for managing configuration information of devices are processed on an operation layer. The NETCONF protocol provides filtering methods in selecting particular XML nodes. In particular, both of <get> and <get-config> operations use two different filtering mechanisms. One is the subtree filtering, stated as default by NETCONF specific and another is the XPath capability of NETCONF protocol. These two mechanisms have their own advantages and disadvantages. The subtree filtering is easy to comprehend but difficult to implement correctly, since the NETCONF draft has ambiguous pats and is short of examples of accuracy on its usage.. On the other hand, the XPath allows common XPath expressions. Thus, it is simple to implement, but rather difficult to comprehend.

Subtree filtering and the XPath demonstrate the difference of performance. NETCONF WG has pointed out that the XPath has a heavy memory usage and response time for configuration management, and suggested subtree filtering as a substitute for the XPath. The XPath requires loading the whole XML message to DOM document before applying the request. This process efficiently creates XML messages, but the XPath uses many memory usages regardless of complexity and sizes. NETCONF WG has proposed that the subtree filtering is lighter than the XPath and has the same results with the XPath. We have experimented in order to verify these claims to confirm the difference of performance between the subtree

filtering and the XPath. We have referenced other researches [15] of the difference between the subtree filtering and the XPath and compared with ours. We used the <get-config> operation and two sets of equivalent requests. The 'Request Message_1' shown in Figure 5 needs to merge different processes. The result of this request includes all interfaces whose name equals 'eth0' along with the interface names which 'mtu' equals '1000'. The subtree filtering and the XPath expression are 'Request Message_1' in Figure 5. The 'Request Message_2' shown in Figure 5 needs to parse simply and does not need to merge. The result of this request includes the names of all interfaces.

Figure 5 demonstrates our experiment results of processing time and memory usauge. The two experiments have produced rather different results. In the first experiment, the subtree filter takes a longer time than the XPath, with the merging. However, in the second experiment, the XPath takes a longer time than the subtree filtering, without the merging. Moreover, these two experiments have different XPath results. As mentioned above, the XPath [9] builds a DOM document and then applies the XPath expression on the DOM tree. It can travel efficiently on the DOM tree for selecting property nodes and shows more power in complicated structure. In contrast, the subtree filtering is reliant on our implementation, which uses a recursive top down approach for selecting nodes. The subtree filtering process is repeated when different nodes need to be merged, which takes more time. The XPath uses the same memory usage on both of two experiments, but it uses more memory than the subtree filtering.

	Subtree Filtering	XPath
Request Message_1 (Using merge)	<filter type="subtree"> <interfaces> <iface> <name>eth0</name> </iface> <iface> <mtu>1000</mtu> <name/> </iface> </interfaces> </filter>	<filter type='xpath'> //name[.='eth0']|//mtu[.='1000'] </filter>
Processing Time	2.085 ms	1.901 ms
Memory Usage	1816 KB	1828 KB
Request Message_2 (Using simple)	<filter type="subtree"> <interfaces> <iface> <name/> </iface> </interfaces> </filter>	<filter type='xpath'> //name </filter>
Processing Time	1.716 ms	1.954 ms
Memory Usage	1812 KB	1828 KB

Fig. 5. Subtree Filtering/XPath

7 Concluding Remarks

In this paper, we have proposed mechanisms for effective reduction of traffic volume, response time and computer resource usage in configuration management using NETCONF. We have also provided an analysis of performance results by NETCONF layers in our implementation; XCMS. In particular, we have investigated network usages, memory requirements and response times.

The response time of transport protocol layer is hardly affected by the transport protocols. The network usage of transport protocol layer is the sum of the request/response message that is used to manage configuration information and the headers of each transport protocol. Also, according to our experiments, clearly, the response time depends on the network usages. We presented the compression mechanism for reducing both the network usages and the response time. We also compared our experiments to other researches of XML compression.

RPC layer provides the pipelining mechanism for more efficient configuration management. The pipelining mechanism can reduce the total response time, but it cannot affect the network usage. We proposed the solution of Multi-Command for reducing total network usage, which is similar to HTTP pipelining. Our solution reduces the response time as well as the network usages from our measurements. Naturally, this mechanism follows a RPC communication method.

We have measured and compared the processing time and the memory usages of the XPath and subtree filtering in our implementation. The XPATH is suitable for processing messages since it requires fewer sources in merging the results as well as in embedding the systems.

References

1. IETF, “Network Configuration,” http://www.ietf.org/html.charters/netconf-charter.html.
2. R. Enns, “NETCONF Configuration Protocol”, draft-ietf-netconf-prot-11, http://www.ietf.org/internet-drafts/draft-ietf-netconf-prot-11.txt, February 2006.
3. T. Goddard, “Using the Network Configuration Protocol (NETCONF) Over the Simple Object Access Protocol (SOAP),” draft-ietf-netconf- soap-06, http://www.ietf.org/internet-drafts/draft-ietf-netconf-soap-06.tx t, September 16, 2005.
4. M. Wasserman, T. Goddard, “Using the NETCONF Configuration Protocol over Secure Shell (SSH)”, http://www.ietf.org/ internet-drafts/draft-ietf-netconf-ssh-05.txt, Oct. 2005.
5. E. Lear, K. Crozier, “Using the NETCONF Protocol over Blocks Extensible Exchange Protocol,” http://www.ietf.org/internet-drafts/draft-ietf-netconf-beep-07.txt, Sept. 2005.
6. R. Fielding, J. Gettys, J. Mogul, H. Frystyk Nielsen, L. Masinter, P. Leach and T. Berners-Lee, “Hypertext Transfer Protocol - HTTP/1.1”, RFC 2616, IETF HTTP WG, June 1999.
7. INRIA-LORIA, EnSuite, http://libresource.inria.fr/projects/ensuite.
8. Hyoun-Mi Choi, Mi-Jung Choi, James W. Hong, “Design and Implementation of XML-based Configuration Management System for Distributed Systems,” Proc. of the IEEE/IFIP NOMS 2004, Seoul, Korea, April 2004, pp. 831-844.
9. W3C, “XML Path Language (XPath) Version 2.0,” W3C Working Draft, November 2005.
10. W3C, "Web Services Description Language (WSDL) Version 1.2" July 2002.

11. Sun-Mi Yoo, Hong-Taek Ju, James Won-Ki Hong, “Web Services Based Configuration Management for IP Network Devices,” Proc. of the IEEE/IFIP MMNS 2005, LNCS 3754, Barcelona, Spain, Oct., 2005, pp. 254-265.
12. Apostolos E. Nikolaidis et al, “Management Traffic in Emerging Remote Configuration Mechanisms for Residential Gateways and Home Devices,” IEEE Communications. Magazine, Volume 43, Issue 5, May 2005, pp. 154-162.
13. A. Pras, T. Drevers, R. v.d. Meent and D. Quartel, “Comparing the Performance of SNMP and Web Services-Based Management,” IEEE eTNSM, Vol. 1, No. 2, Dec. 2004, pp. 1-11.
14. Mi-Jung Choi et al, “XML-based Configuration Management for IP Network Devices,” IEEE Communications Magazine, Vol. 41, No. 7, July 2004. pp. 84-91.
15. Vincent Cridlig, et al, "A NetConf Network Management Suite:ENSUITE", Proc. of the IEEE IPOM 2005, LNCS 3751, Barcelona, Spain, Oct., 2005, pp. 152-161.
16. Henrik Frystyk et al, "Network Performance Effects of HTTP/1.1, CSS1, and PNG," W3C, June 1997.