Protocol Efficiencies of NETCONF versus SNMP for Configuration Management Functions

Brian Hedstrom, Akshay Watwe, Siddharth Sakthidharan
b.hedstrom@cablelabs.com, akshay.watwe@colorado.edu,
Sakthidharan.Siddharth@colorado.edu

A capstone paper submitted as partial fulfillment of the requirements for the degree of Masters in Interdisciplinary Telecommunications at the University of Colorado, Boulder, May 2, 2011. Project directed by Mark Dehus.

1 Introduction

This paper focuses on the quantitative analysis of the performance characteristics of two different network management protocols, the Simple Network Management Protocol (SNMP) [1] and the Network Configuration Protocol (NETCONF) [2]. SNMP is a legacy protocol which has been widely deployed and in use for several decades and is based on a request-response messaging protocol using a connectionless transport, generally the User Datagram Protocol (UDP) [3]. NETCONF is an emerging technology that has limited deployment. NETCONF is based on an XML messaging protocol using a connection-oriented transport, generally the Transmission Control Protocol (TCP) [4]. Additionally, NETCONF is often invoked within a secure shell (SSH) session [5].

Today's complex and vast communication networks span the globe and include multiple equipment venders and technologies. The problem that Communication Service Providers (CSPs) face today is that SNMP no longer scales as an efficient and effective means for performing many network management functions (e.g., configuration) in such complex environments. New network management protocols have been designed, such as NETCONF, with goals of overcoming limitations encountered with SNMP. Several questions are posed with the advent of this new technology. The first question posed is: Can it be established that NETCONF could meet the current and future needs as a scalable, efficient, and effective method for performing configuration management functions within Communication Service Providers' networks? The second question posed is: Is the NETCONF protocol more efficient, in terms of protocol bandwidth utilization, number of packets, number of transactions and operation time, than SNMP for performing configuration network management functions? And the third question posed is: By what factor is NETCONF more efficient than SNMP, if it is truly more efficient?

If quantitative measurements can show that NETCONF is more efficient (in terms of bandwidth utilization, packet counts, transaction counts and operation time), than SNMP, and by some factor of *N*, this may be applied to the SNMP scalability problem where NETCONF could be the replacement protocol to achieve the desired scalability in configuration network management functions.

2 Definitions

This section defines the terms used in this paper.

A Communication Service Provider (CSP) is any network operator owning and operating a network covering one or more geographic areas and offering voice, video and/or data services to their residential and/or business customers. Examples include Verizon, Comcast, AT&T, etc.

Network management includes the processes and functions which a network operator performs in order to maintain, provision, administer, monitor and operate their network [6]. Configuration management is the function specific to provisioning components within the network (e.g., configuration of one or more devices to enable a specific service or feature).

Simple Network Management Protocol (SNMP) is a protocol used to perform management functions including monitoring and provisioning [7]. The protocol is based on a manager-agent model where one or more manager applications reside within a CSPs operations center and many agents are deployed in the network, embedded within networking devices. Managers make requests of agents. The SNMP SET operation is used for configuration management functions.

Network Configuration Protocol (NETCONF) is another protocol used to perform management functions, mainly targeted at provisioning but capable of monitoring certain configuration and operational state information [3]. The protocol is based on a client-server architecture where one or more client applications reside within a CSPs operations center and many servers are deployed in the network, embedded within network devices similar to how SNMP agents are embedded. Remote procedure calls (e.g., <rpc>, <rpc-reply>) are invoked to exchange information, and the NETCONF <edit-config> operation is used for configuration management functions.

A network management transaction is the exchange of one request and a corresponding response to that request. A single transaction may be composed of several protocol messages (e.g., Ethernet frames) due to protocol message size limits. For example, since NETCONF is a session-based protocol, an <edit-config> operation may require several Ethernet frames to transmit the full session containing the request-response transaction.

Managed objects are parameters within a network device which may be configured using the SET SNMP operation [1] or the <edit-config> NETCONF operation [3].

Python is an interpreted high level programming language and was used to develop the manager-side applications for performing the device configuration as explained in the Analysis section of this paper.

A log-log plot is a data graph with logarithmic scales on both the horizontal X-axis and vertical Y-axis.

Operation time is the duration, in seconds, for a configuration operation to complete once the manager sends the configuration change request and the agent acknowledges the change has been completed. Operation time does not include any manager processing time used to dynamically build the configuration.

Data models are implementation specific representations of managed objects and are specified using a defined syntax or language. In the scope of this paper, data models were developed using SMIv2 [8] for SNMP and YANG [9] for NETCONF. Data models are implemented at both the manager and agent side.

3 Scope and Assumptions

This paper is intended for architects and engineers who are planning and designing the next generation network management architectures for Communication Service Providers. This

network management architecture evolution might be rolled out as emerging services are deployed (e.g., such as new cloud based services or Ethernet services), since new services have higher demands for various network management functions such as monitoring (e.g., quality assurance). As such, the target audience includes network architects planning and designing these new services. The intention is to demonstrate the performance of NETCONF vs. SNMP within the constraints of device configuration. If NETCONF yields superior results, it is believed this paper could stimulate the CSP architects to create purchase orders or request for product to their vendors that contain requirements for the NETCONF protocol, so that they can begin deploying the technology. This paper might also be useful for standards organizations such as the TeleManagement Forum and the ITU-T. Each organization has working groups which focus on standards setting activities within the scope of network management (e.g., ITU-T Study Group 2 Question 8).

Characterizing the configuration network management efficiencies of the two protocols is limited to a CSP's network which provides services to their end customers. The CSP's internal IT network or Enterprise network is not within this scope. Network exchanges across different Service Provider boundaries are not within scope (e.g., scope is limited to a single CSP network footprint). SNMPv3 will not be considered, since this protocol is rarely used within CSP network operations due to its operational complexities. It is assumed that the SNMP agent and NETCONF server devices all have equivalent memory and processing capabilities.

SNMP uses a data modeling language based on the Structure of Management Information [8]. NETCONF uses a data modeling language called YANG, based on XML [9]. Although many data types exist and map between the two data modeling languages, not all do. For the purposes of this paper, equivalent data types will be used when constructing the SNMP and NETCONF data sets.

4 Importance

This research is important in the evolution of complex and expanding communication networks, specifically with the growing need to efficiently and effectively maintain, administer and provision these vast networks while reducing operational costs at the same time. As the networks, and the network devices that form the building blocks of those networks, grow smarter and more complex in their capabilities, so should the tools that manage them. Figure 1 below highlights this linear relationship of the increasing complexity of both the network and the network device.

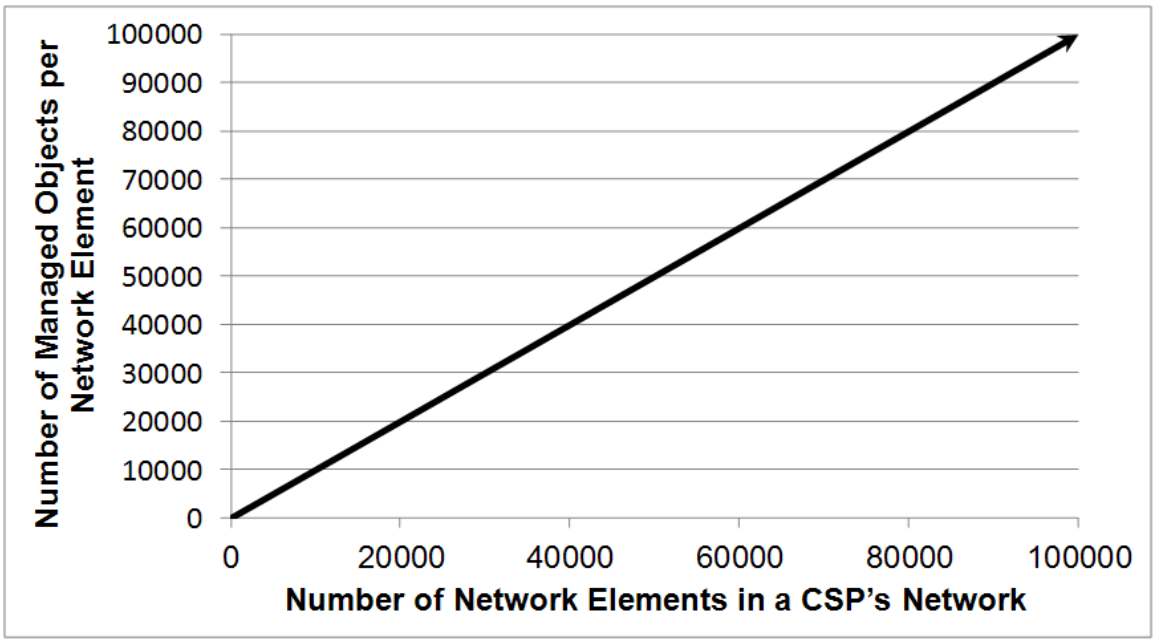

Figure 1 – Complexity of Network vs. Network Element

The services being offered by these networks are becoming more complex, with higher customer demands, lending to a greater demand for richer network monitoring and provisioning applications and protocols. Tail-f [10] provides an example of a new generation management system platform built around the NETCONF protocol and YANG data modeling language with these specific goals in mind. If the research can show that NETCONF is significantly more efficient than SNMP, this fact could be used to advocate for CSPs to transition their network management technology away from SNMP towards a newer, XML-based technology. Furthermore, if standards organizations such as ITU-T and TeleManagement Forum begin to adopt this new technology and its associated data modeling language [9] within the specifications they develop it will reshape the network operations industry.

Previous research has been lacking in characterizing a side-by-side comparison of the efficiencies of SNMP versus NETCONF for Configuration Management functions. One research group [11] examined performance improvements within the NETCONF protocol, but their testing and characterization were based on an early IETF draft version of the NETCONF standard and a prototype implementation. Today, off the shelf vendor implementations of NETCONF clients and servers are available. Another research group performed a comparison of SNMP to Web Services [12] for network management performance characterization. Although this compared the performance of SNMP to an XML-based technology, it did not compare SNMP to the new NETCONF protocol; this paper fills that gap. Finally, an empirical study of NETCONF was performed by a research team [13] where SNMP and NETCONF were comparatively tested under a lab environment using an open source Yencap NETCONF server implementation. While this study performed a comparative analysis of a few hundred managed object queries with both protocols, it performed no comparison of Configuration Management functions using these protocols. This paper focuses strictly on the device configuration aspects and pushes the boundaries of 100,000 managed objects, as discussed in the Analysis section which follows.

5 Analysis

The objectives for performing quantitative measurements on the performance of the two protocols were to perform configuration operations on a number of managed objects *M* in a staircase-type function where *M* = 1, 10, 100, 1000, 10000, and 100000. In order to perform measurements for the configuration operations, a Linux based client-server architecture was constructed as illustrated in Figure 2.

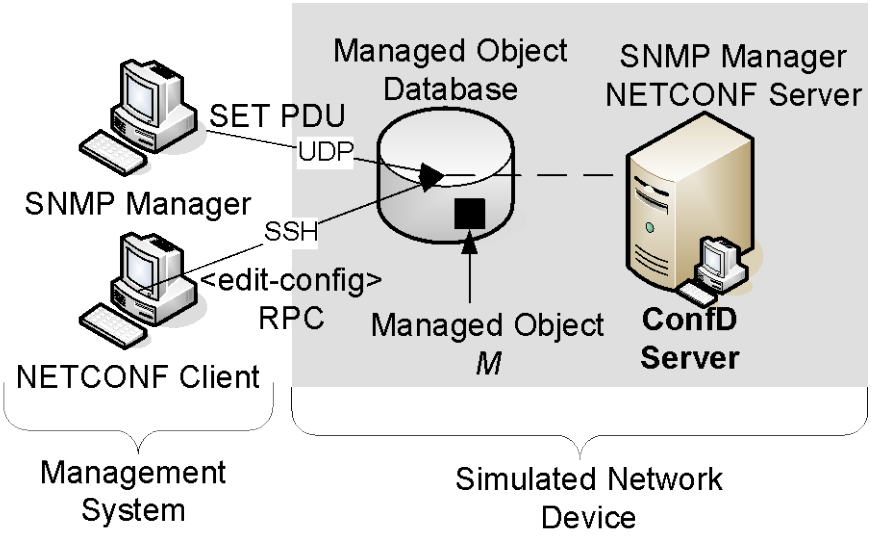

Figure 2 – Lab Environment for Protocol Testing and Data Collection

The SNMP Manager and NETCONF Client applications were developed as Python modules which leveraged the Net-SNMP [14] and ncclient [15] open source Python libraries. As the scalability of this project exceeded 50,000 managed objects, several software bugs were found in the ncclient Python library requiring the authors to work with the open source library owner to identify and resolve the issues [15]. The ConfD NETCONF Server, which contains the managed object database, is a commercial application available from Tail-f Systems [16]. The ConfD Server also included an embedded SNMP Agent [16]. This simulates a network device that contains configuration information which may be updated by both the SNMP and NETCONF protocols.

The managed object database within the ConfD Server was constructed by developing an SNMP MIB and YANG Module which were equivalent data models. This created an environment where a single object existed within the ConfD Server and managed object database, rather than defining separate database objects for each protocol. Figure 3 illustrates the Unified Modeling Language class definition on how the two data modules were developed in order to achieve the goal of reaching a maximum of 100,000 managed objects. A *List-N* object was defined with an Integer id which was a key or index. Ten *leaf[1..10]* attributes were defined, with each being a String type, with a maximum string length constraint of 255 characters (SMIv2 introduces this constraint). The SNMP MIB and YANG Modules definitions are not included in this paper.

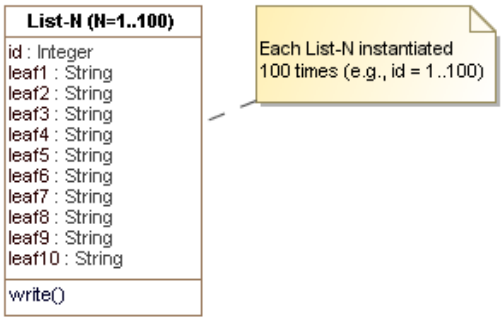

Figure 3 – Class Definition for Data Model Design for MIB and YANG Modules

Three Manager-side applications were developed using the Python programming language as follows:

- NETCONF client application to perform configuration of *M* managed objects and record operation time
- SNMP Manager application to perform configuration of *M* managed objects and record operation time
- Packet sniffer and packet analysis application to perform measurements and calculations of protocol applications

The NETCONF client and SNMP Manager applications included methods to dynamically build the configurations to be sent to the simulated network device. For example, in the NETCONF client application, where *M*=100,000, the application dynamically constructed the YANG data tree instance populating values for all manage object instances. To maintain a comparative test environment, both applications configured the *leaf[1..10]* attributes with identical values; each attribute was configured with a 255 length character string to maximize the payload size of the configuration operations.

The SNMP Manager application, which utilized the SET operation, was optimized to carry more than a single object request per message (referred to as a SET Protocol Data Unit). It was determined through testing that a single SET PDU could carry up to 36 managed object configuration requests, along with their corresponding 255 length character string values. Therefore, this optimization approach was utilized for all *M* staircase-style operations for the SNMP protocol. A non-optimized approach would have included only a single managed object request per SET PDU.

In order to perform the analysis and measurement calculations for the configuration management applications, a packet sniffer and analysis application was developed using the pcap [17] and dpkt [18] Python Library modules respectively. This application was run simultaneously with the NETCONF and SNMP management applications to capture, analyze and compute the performance statistics of the management traffic exchanged between the management system and simulated network device.

5.1 Measurement Results

This section details the measurement results obtained for both the SNMP and NETCONF protocols. For the measurement calculations, each experiment was executed twenty times (number of transactions was performed five times). The mean values for each performance statistic is plotted on a log-log plot, along with error bars on the Y-axis as shown in the graphs

below. A table of the mean values and standard deviations is also provided. The Q_1 and Q_2 values shown on each plot represents the percentage ratio of SNMP-to-NETCONF for the *M*=1 and *M*=100,000 values respectively. It should be noted that a linear regression analysis was also performed to determine if the data sets for each performance parameter would fit to a straight line model, which they did not.

Figure 4 and Table 1 present the measurement results for the number of Ethernet frames (i.e., data packets). The results illustrate that as *M* increases, SNMP continually utilizes fewer Ethernet Frames to carry the configuration payload when compared to NETCONF. However, at smaller values of *M*, NETCONF requires a higher number of Ethernet frames, due to connection-oriented TCP transport. As *M* approaches 10,000, the gap between the two protocols diminishes considerably. In general, NETCONF will require greater processing overhead on the network, especially at the lower number of managed objects. Refer to the next section for a discussion on the messaging and bandwidth overhead and the tradeoffs in the features NETCONF adds along with this overhead.

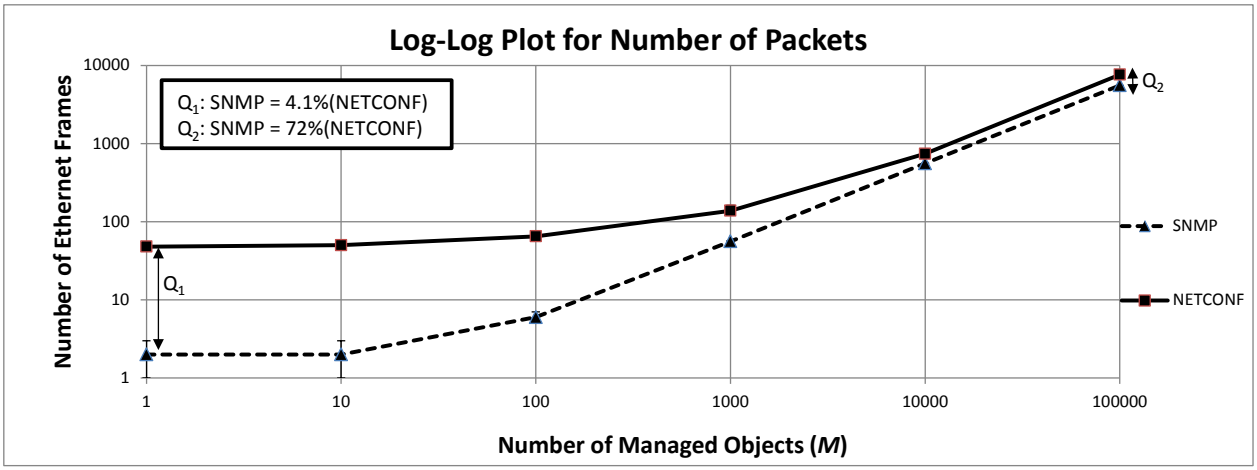

Figure 4 – *M* versus Number of Ethernet Frames

Table 1: Mean Values for Number of Ethernet Frames

Operation on # of Managed Objects (*M*)	Number of Ethernet Frames		Standard Deviation	
	SNMP Protocol	NETCONF Protocol	SNMP Protocol	NETCONF Protocol
	SET	<edit-config>	SET	<edit-config>
1	2	48	0	0
10	2	50	0	0
100	6	65	0	0
1000	56	138.2	0	2.852
10000	557.8	740.3	0.616	23.295
100000	5558	7622.4	0	82.289

Figure 5 and Table 2 present the measurement results for the number of bytes. The results illustrate that the NETCONF protocol consistently carries a higher amount of bytes in the protocol payload as compared to SNMP and thus has higher bandwidth utilization requirements. This is likely due to several reasons. It should be noted that the measurements included the UDP and TCP header lengths (UDP header length is fixed at 8 bytes whereas the minimum TCP header length is 20 bytes and the maximum is 60 bytes, therefore the packet header overhead is higher for NETCONF). NETCONF is running over SSH and therefore has security built into the transport, requiring further protocol overhead and messaging. For this experiment, SNMPv2 was utilized, which includes no security mechanism. In addition, NETCONF is a session-based protocol, where-as SNMP is a session-less protocol running over UDP. This adds additional overhead for the connection handshaking and reliability of TCP. Finally, the NETCONF protocol adds a level of handshaking to allow a client and server the ability to discover capabilities in a <hello> message exchange. All of these added features which SNMP does not have include additional messaging overhead at the protocol layer.

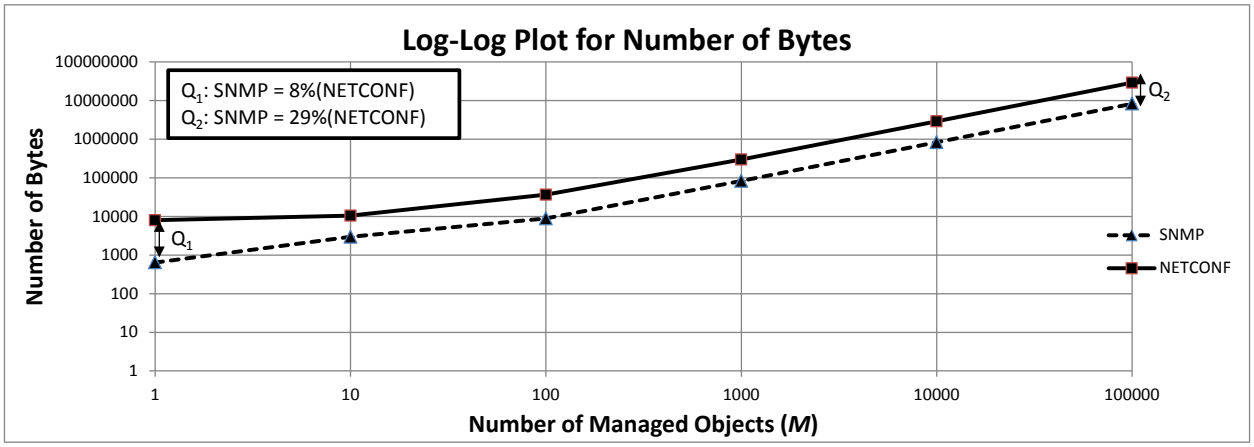

Figure 5 – *M* versus Number of Bytes

Table 2: Mean Values for Number of Bytes

Operation on # of Managed Objects (*M*)	Number of Bytes[2]		Standard Deviation	
	SNMP Protocol	NETCONF Protocol	SNMP Protocol	NETCONF Protocol
	SET	<edit-config>	SET	<edit-config>
1	642	7931	0	0
10	2960	10475	0	0
100	8800	36539	0	0
1000	82880	295891.4	0	96.716
10000	825544	2885174.4	911.069	1580.762
100000	8225840	28820676.5	0	3840.356

[2]Number of bytes includes the TCP and UDP headers

Figure 6 and Table 3 present the measurement results for the operation time. The results illustrate that even with the entire message overhead that NETCONF introduces, the operation time of the protocol performed faster when compared to SNMP as *M* approached 1000 and beyond. Figure 6 shows that when *M* has a small value (*M* < ~500) SNMP operations perform faster; however as *M* increases in size, NETCONF becomes operationally faster than SNMP. This may likely be due to the transaction benefits of NETCONF which will be discussed next.

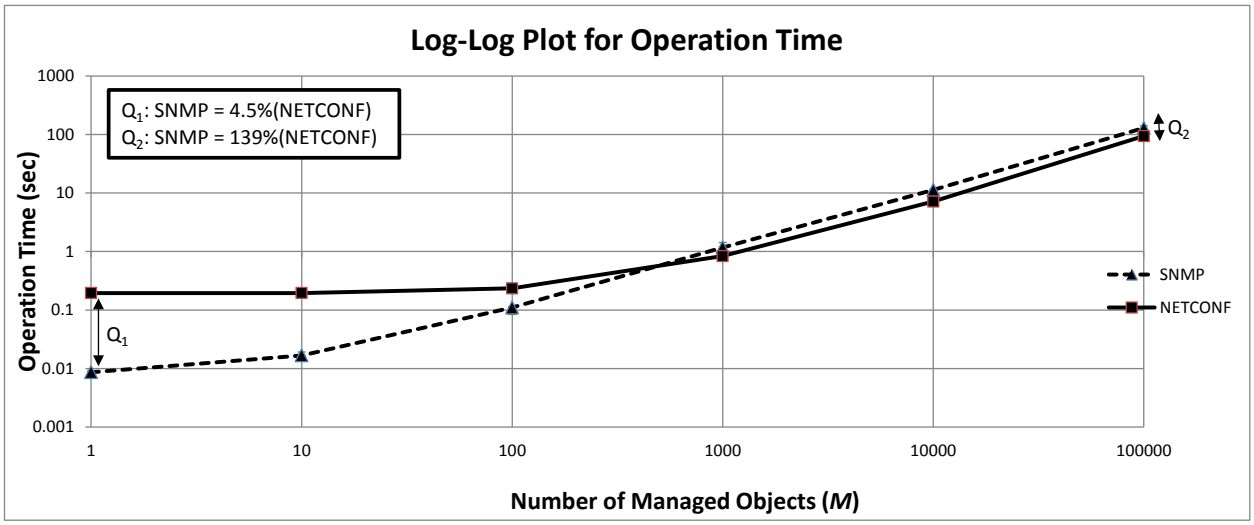

Figure 6 – *M* versus Operation Time

Table 3: Mean Values for Operation Time

Operation on # of Managed Objects (*M*)	Operation Time[3]		Standard Deviation	
	SNMP Protocol	NETCONF Protocol	SNMP Protocol	NETCONF Protocol
	SET	<edit-config>	SET	<edit-config>
1	0.00865	0.19555	0	0.036
10	0.01675	0.19505	0.002	0.033
100	0.11045	0.23520	0.003	0.026
1000	1.17280	0.83565	0.254	0.040
10000	11.27050	7.16690	0.469	0.241
100000	129.05590	93.90900	12.735	11.007
[3]Operation time excludes processing time for dynamically building the configuration				

For the operation time performance attribute, the 99% confidence intervals were calculated as shown in Table 4 below. It's important to highlight that for *M*=100,000 the calculations in Table 4 show there is significance in the data which supports that NETCONF performs faster than SNMP.

Table 4: 99% Confidence Intervals for Operation Time

Number of Managed Objects	Operation Time SNMP SET		Number of Managed Objects	Operation Time NETCONF edit-conf	
1	Mean	0.00865	1	Mean	0.19555
	Confidence Level(99.0%)	0.0003131		Confidence Level(99.0%)	0.0231992
	Lower Limit	0.0083369		Lower Limit	0.1723508
	Upper Limit	0.0089631		Upper Limit	0.2187492
10	Mean	0.01675	10	Mean	0.19505
	Confidence Level(99.0%)	0.0014949		Confidence Level(99.0%)	0.0211751
	Lower Limit	0.0152551		Lower Limit	0.1738749
	Upper Limit	0.0182449		Upper Limit	0.2162251
100	Mean	0.11045	100	Mean	0.2352
	Confidence Level(99.0%)	0.0016536		Confidence Level(99.0%)	0.0167793
	Lower Limit	0.1087964		Lower Limit	0.2184207
	Upper Limit	0.1121036		Upper Limit	0.2519793
1000	Mean	1.1728	1000	Mean	0.83565
	Confidence Level(99.0%)	0.1627253		Confidence Level(99.0%)	0.0257721
	Lower Limit	1.0100747		Lower Limit	0.8098779
	Upper Limit	1.3355253		Upper Limit	0.8614221
10000	Mean	11.2705	10000	Mean	7.1669
	Confidence Level(99.0%)	0.3000636		Confidence Level(99.0%)	0.1543879
	Lower Limit	10.970436		Lower Limit	7.0125121
	Upper Limit	11.570564		Upper Limit	7.3212879
100000	Mean	129.0559	100000	Mean	93.909
	Confidence Level(99.0%)	8.1467486		Confidence Level(99.0%)	7.0415331
	Lower Limit	120.90915		Lower Limit	86.867467
	Upper Limit	137.20265		Upper Limit	100.95053

Figure 7 and Table 5 present the measurement results for the number of transactions. The results highlight the performance parameter where NETCONF is superior to SNMP and where SNMP cannot compete. When $M \leq 10$ both protocols perform identically, but beyond this point, SNMP's performance dramatically decreases as the number of transactions required spike. NETCONF can configure 100,000 managed objects in a network device with a single transaction. In the optimized SNMP case, this takes 2,779 transactions. This equates to nearly 3000 UDP messages that have the possibility of not reaching their destination. This leads to a significantly higher amount of development complexity in developing the management tools and on-device software to handle those individual 3000 operations. In addition, this eliminates the ability to perform a backup and restore of the device's configuration using the SNMP protocol. With a single transaction, this type of operation can be easily performed. In addition, SNMP cannot perform a configuration validation, where a full configuration is checked against the configuration stored on a management system. Just imagine if an operator had 100,000 devices on their network, and each device had 100,000 managed objects and they needed to perform a full configuration of all the devices. These results exemplify a significant disadvantage of the performance of the SNMP protocol for configuration management functions.

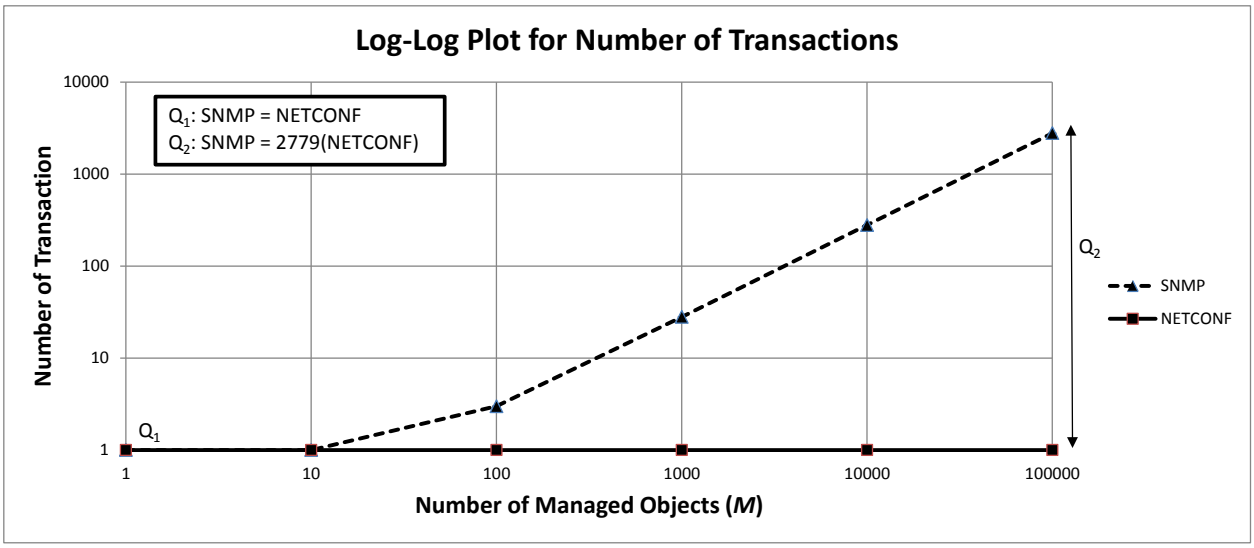

Figure 7 – *M* versus Number of Transactions

Table 5: Mean Values for Number of Transactions

Operation on # of Managed Objects (*M*)	Number of Transactions Required[1]		Standard Deviation	
	SNMP Protocol	NETCONF Protocol	SNMP Protocol	NETCONF Protocol
	SET	<edit-config>	SET	<edit-config>
1	1	1	0	0
10	1	1	0	0
100	3	1	0	0
1000	28	1	0	0
10000	279	1	0	0
100000	2779	1	0	0

[1]Management transactions include request and response message pair for the Operation

6 Conclusions

This paper has brought both the NETCONF and SNMP protocols head-to-head in a performance comparison to help answer the research question, is the NETCONF protocol more efficient, in terms of protocol bandwidth utilization, number of packets, number of transactions and operation time, than SNMP for performing configuration network management functions? By what factor is NETCONF more efficient than SNMP, if it is more efficient? A lab environment was constructed to exercise both protocols under a controlled scenario using a staircase-style function of managed objects configured in a simulated network device. Using quantitative measurement results and statistical data analysis, it was determined that the NETCONF protocol had a higher bandwidth utilization for number of Ethernet frames (i.e., packets) and bytes due to the overhead required for security, connection-oriented (session-based) and capability exchange features that are lacking in the SNMP protocol. However, these are

necessary features for managing complex networks. At a lower number of managed objects, SNMP had a faster operation time, but as the managed objects exceeded ~500, NETCONF's operation time efficiency surpassed SNMP. Finally, and most importantly, NETCONF is most powerful when it stacks up against SNMP for number of transactions. NETCONF is able to configure 100,000 managed objects in a single transaction, using XML configuration data as its payload, while SNMP's best case scenario is 2779 transactions for the same number of managed objects. This performance advantage alone outweighs the higher bandwidth utilization that NETCONF carries with the protocol (but again, these include those necessary features listed above). Due to the number of performance variables analyzed, it cannot be established by what factor NETCONF is more efficient than SNMP.

The conclusions are clear; NETCONF is a viable alternative solution to SNMP for Configuration Management functions for today's complex networks and emerging new generation back office systems. The NETCONF protocol clearly has the ability to displace SNMP as the incumbent Configuration Management protocol and provide Communication Service Provider's with a more powerful back office tool to meet their higher network operation demands.

References:

[1] William Stallings, *SNMP, SNMPv2, SNMPv3, and RMON 1 and 2*, Addison-Wesley, Third Edition, 1999.

[2] R. Enns, "NETCONF Configuration Protocol", IETF RFC 4741, December 2006

[3] J. Postel, "User Datagram Protocol", IETF RFC 768, August 1980.

[4] J. Postel, "Transmission Control Protocol", IETF RFC 793, September 1981.

[5] M. Wasserman, T. Goddard, "Using the NETCONF Configuration Protocol over Secure SHell (SSH)", IETF RFC 4742, December 2006.

[6] ITU-T, "Series M: TMN and Network Maintenance: International Transmission Systems, Telephone Circuits, Telegraphy, Facsimile and Leased Circuits; Telecommunications management network; TMN management functions", ITU-T Recommendation M.3400, February 2000.

[7] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "A Simple Network Management Protocol (SNMP)", IETF RFC 1157, May 1990.

[8] K. McCloghrie, *et al.*, "Structure of Management Information Version 2 (SMIv2)", IETF RFC 2578, April 1999.

[9] M. Björklund, "YANG – A Data Modeling Language for the Network Configuration Protocol (NETCONF)", IETF RFC 6020, October 2010.

[10] Tail-f Systems White Paper, "EMS/NMS – Beyond Alarms and Maps", 2011, Available at http://www.tail-f.com/ems-beyond-alarms.

[11] S. M. Yoo, H. T. Ju, and J. W. Hong, "Performance Improvement Methods for NETCONF-Based Configuration Management", *Proceedings from 9th Asia-Pacific Network Operations and Management Symposium*, APNOMS 2006, Busan, Korea, September 27-29, 2006, pp. 242–252.

[12] 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.

[13] J. Yu, I. Ajarmeh, "An Empirical Study of the NETCONF Protocol", *Sixth International Conference on Networking and Services*, ICNS 2010, Cancun, Mexico, March 7-13, 2010, pp. 253-258.

[14] G. S. Marzot, The Python 'netsnmp' Extension Module for the Net-SNMP Library, v5.4.3, Project Documentation Page, viewed March 26, 2011, https://net-snmp.svn.sourceforge.net/svnroot/net-snmp/trunk/net-snmp/python/README.

[15] S. Bhushan, ncclient – A Python Library for NETCONF Clients, v0.3.1, Project Page, viewed March 2, 2011, http://oss.schmizz.net/ncclient/.

[16] Tail-f Systems, ConfD – A Commercial On Device Management Solution, v3.3.4, Product Marketing Page, viewed March 26, 2011, http://www.tail-f.com/products-and-services/confd, Software obtained under Non-disclosure Agreement.

[17] D.Song, pcap – A Python packet capture library, Project Page, viewed March 26, 2011, http://pypi.python.org/pypi/pcap.

[18] D.Song, dpkt – A fast, simple packet creation /parsing library, with definitions for basic TCP/IP protocols, Project Page, viewed March 26, 2011, http://code.google.com/p/dpkt/.