POLICY-BASED MANAGEMENT OF DIFFSERV USING XML TECHNOLOGIES

Ntanzi M. Carrilho, Neco Ventura
Department of Electrical Engineering, University of Cape Town, Rondebosch, Cape Town, South Africa

Keywords: Policy-based Network Management, DiffServ, XML-based Network Management, NETCONF.

Abstract: A design, implementation, and evaluation of a Policy-based Management framework for DiffServ networks using XML technologies is presented. XML is used for policy representation throughout the entire policy life cycle, as well as in the communication protocol, reducing development costs, while promoting clarity, simplicity, and interoperability. Furthermore, performance evaluation results show that the proposed scheme generates less network traffic than its COPS-PR counterpart when protocol messages are compressed. Moreover, XML messages generate higher processing delays. The added processing overhead caused by compression to XML messages is very small, thus the gains in reduced network traffic generated outweigh the losses in terms of processing delays.

1 INTRODUCTION

Today Internet traffic is highly diverse. Applications have different requirements in terms of bandwidth, delay, loss and jitter. The Internet protocol suite was originally designed to provide best-effort services; it was designed to deliver packets to their destination without any Quality of Service (QoS) guarantees, resulting in unpredictable behaviour of applications.

In order to counter this behaviour a few technologies were designed to complement the IP infrastructure to it capable of handling traffic with different QoS needs. One of these technologies is the Differentiated Services (DiffServ) (Blake et al., 1998) network architecture. DiffServ was designed to be a scalable service differentiation architecture for the Internet. It achieves scalability by classifying and possibly conditioning traffic streams on the edge of the network, according to their requirements. On the core of the network, resources are allocated to these streams based on their classification or Per-Hop Behaviour (PHB).

The management requirements for DiffServ networks differ from those of simple IP networks. Although the DiffServ technology is relatively simple, the initial setup and configuration of hundreds of devices can be complex and daunting. Policy-Based Network Management (PBNM) provide mechanisms to simplify and automate the management and administration of these networks, therefore reducing management costs.

The Internet Engineering Task Force (IETF) is working on the standardisation of PBNM technologies such as the Common Open Policy Service protocol for Policy Provisioning (COPS-PR) (Chan et al., 2001) and its associated data definition language, the Structure of Policy Provisioning Information (SPPI) (McCloghrie et al., 2001). COPS-PR messages are usually encoded according to the Basic Encoding Rules (BER), a binary encoding scheme. As a consequence the protocol is not easy to use, understand, or debug. Up to now, COPS-PR and SPPI have failed to gain considerable market acceptance (Schoenwaelder et al., 2003), mainly because network operators feel that COPS-PR protocol does not address their requirements (Schoenwaelder, 2003).

On the other hand the limitations of the Simple Network Management Protocol (SNMP) as a suitable mechanism for device configuration management are driving research on the applicability of XML (the eXtensible Markup Language) to device configuration management. The IETF has charted the Network Configuration (NETCONF) Working Group

M. Carrilho N. and Ventura N. (2007).
POLICY-BASED MANAGEMENT OF DIFFSERV USING XML TECHNOLOGIES.
In *Proceedings of the Third International Conference on Web Information Systems and Technologies - Internet Technology*, pages 72-79
DOI: 10.5220/0001281500720079
Copyright © SciTePress

(WG) with the aim of creating a network configuration protocol using XML technologies. NETCONF (Enns, 2006) enables devices to communicate with each other using Remote Procedure Calls (RPC) encoded in XML.

Motivated by the failure of COPS-PR, we present the design, implementation, and evaluation of a policy-based management architecture for DiffServ networks using NETCONF as the device communication protocol, along with XML Schema and XML for policy modelling and representation, respectively. There are several advantages of using XML for both data representation and the exchange protocol. XML provides a format that is both machine and human-readable, which is useful in the debugging and deployment of the tool. XML Schema provides powerful and extensible modelling mechanisms capable of modelling information of arbitrary complexity. Due to the widespread XML document processing tools, the effort needed in the development of the PBNM system is reduced. Perhaps the most important incentive for the use of XML for data representation and protocol messages in a PBNM system is that it improves interoperability between heterogeneous management systems or even between elements within the same system, and provides a common format for the various policy representation levels.

The remainder of this paper is organised as follows: Section 2 presents a brief overview of NETCONF. Section 3 presents an overview of relevant work related to the one presented here. Section 4 introduces the proposed policy framework. Section 5 proceeds by describing the proposed PBNM architecture and its main components. Section 6 provides measurement results and finally, conclusions are drawn in Section 7.

2 THE NETCONF PROTOCOL

NETCONF (Enns, 2006) is an XML based protocol created specifically for network device configuration. It uses a XML RPC mechanism to expose a formal Application Programming Interface (API) through which managers and agents communicate. The protocol defines a base set of operations to retrieve, edit and copy partial or entire device configurations. This base set of operations can be extended by adding capabilities.

NETCONF is a connection-oriented protocol which requires a persistent connection between the manager and the agent although it is not bound to any particular transport protocol. Its connections must provide authentication, data integrity, and privacy. NETCONF depends on the transport protocol for this capability. SSH, SOAP over HTTP and BEEP may be used. Although we are not using any of these protocols in our current implementation, in future we are planning to use SSH to secure sessions.

3 RELATED WORK

This section will review research work addressing issues relating to policy representation and transportation using XML based technologies.

(Beller et al., 2004) propose a framework for defining reusable business-level policies for DiffServ using XML for policy representation. Policies are designed following a modular approach to increase reusability. Xpointers are used to locate reusable policy fragments placed in a reusable policy container. Although this is a good approach, we think that using the XInclude standard would yield a better result, since it would automate the parsing and inclusion of re-usable fragments. This is the approach followed in the research project.

(Clemente et al., 2005) propose a similar PBNM architecture focusing on IP Security (IPSec). Similar to the scheme proposed in our paper, XML is used for representation of policies during their entire life cycle. High-level policies are modelled based on PCIM and lower-level policies are modelled based on the IPSec PIB. A Model-Level mapping scheme is used for lower level policy mapping. PIB attributes are mapped to XML elements instead of attributes, resulting in XML documents that are even more verbose.

The proposals mentioned above only partially solve the problems stated in Section 1. This is due to the fact that they still use COPS-PR to provision policies. (Franco et al., 2006) and recognise in their work the potential of using XML-based exchange protocols for PBNM. They present two communication models for policy provisioning, based on NETCONF and SOAP, respectively. The study shows, as compared to COPS-PR, that both schemes waist more bandwidth. An interesting finding is that if compression is used for XML messages both schemes consume less bandwidth than COPS-PR. The study concludes that if NETCONF or SOAP are to be used for policy provision careful attention must be paid to the modelling of the communication between devices. The architecture presented in this paper uses NETCONF for device communication and the communication model used is very similar to the one presented in (Franco et al., 2006).

In summary, whereas (Beller et al., 2004) and (Clemente et al., 2005) use COPS-PR for policy pro-

visioning, we employ NETCONF. We think that this is a step forward because by using an XML-based provisioning protocol coupled with XML policy representations we simplify the internal structure of the PBNM system and reduce development costs. Moreover, (Franco et al., 2006) use XML-based policy provisioning schemes but their work focuses on low level QoS policies. In this paper we aim to present a more comprehensive policy framework by adding a layer of abstraction that focuses on the network instead of the various managed devices and simplifies the administration process.

4 PROPOSED POLICY FRAMEWORK

The IETF and the Distributed Management Task Force (DMTF) have jointly developed object oriented models for policy representation, namely the Policy Core Information Model (PCIM) (Moore et al., 2001) and its extensions (PCIMe) (Moore, 2003). PCIM and PCIMe define the generic structure of a policy and can be extended to enable developers and administrators to define policies of different types. The Policy QoS Information Model (QPIM) (Snir et al., 2003) is a further extension of PCIM and PCIMe designed specifically for the representation of QoS policies. The policy framework presented here is based on these standards.

Figure 1 presents an overview of the policy framework used in our work. Business-level policies are used to express the general business requirements of the network. They represent the highest level of policy abstraction and are generally presented in a user friendly format. Business-level policies can be modelled using PCIM extended to model policies in a specific domain, e.g. QoS.

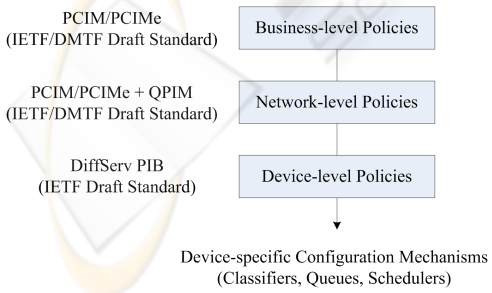

Figure 1: Policy hierarchy.

This work focuses on the two middle levels of Figure 1, more specifically, network and device-level policies. Network-level policies represent generic network operational and configuration information that is independent of any specific device but is dependent of general QoS mechanisms such as the type of queue management strategy employed. We model network-level policies based on the PCIM/PCIMe and QPIM information models. We chose to model network-level policies based on these standards because they are very flexible, widely accepted, and do not depend on any storage technology.

Device-level policies are specific for a given device. The PBNM system translates network-level policies to device-level policies based on the capabilities supplied by each device. While network-level policies provide a common level of abstraction required to manage multiple devices with potentially different capabilities, device-level policies provide adequate policies for each device based on the device's capabilities. In this work we model device-level policies based on the DiffServ Policy Information Base (PIB). We opted to use the DiffServ PIB, in favour of QDDIM (Information model for Describing Network Device QoS Datapath Mechanisms) (Moore et al., 2004) because of its simplicity and acceptance. Furthermore COPS-PR is designed to transfer policies modelled using the DiffServ PIB, and in order to provide a fair comparison between the alternative policy provision scheme presented in this paper and traditional COPS-PR schemes both schemes have to be based on the same model.

Device-level policies provide a common configuration format for multiple network devices. But the specific configuration commands applied to devices may differ. Therefore the final translation stage in the policy life cycle is the translation of device-level policies to device-specific configuration commands.

4.1 Network-Level Policies

As mentioned in the previous section we model network-level policies based on PCIM/PCIMe and QPIM. These models are independent of any storage technology allowing various data models to be designed and implemented according to a single uniform model. Some of their strengths are that they allow for policy rule nesting and re-usability. Rule nesting enables complex policy rules to be constructed form multiple simpler rules enhancing the manageability and scalability of the policy framework.

In order to gain advantage from these features these information models have to be mapped to a suitable data model. By using XML for policy representation and storage we ease that mapping. XML is renowned for its ability to represent data of almost arbitrary complexity, easily enabling data nesting and re-usability.

There are two possible mapping schemes: schema mapping and meta-schema mapping (Youn and Hong, 2003). A Schema Mapping is one in which the XML Schema is used to describe CIM classes, and CIM Instances are mapped to valid XML Documents for that schema. XML element names and attributes are taken directly from the corresponding CIM element names and attributes respectively. A Meta-schema Mapping is one in which the XML schema is used to describe the CIM meta-schema, and both CIM classes and instances are valid XML documents for that schema. CIM element names are mapped to XML attribute or element values, rather than XML element names.

In this work we opted for the schema mapping since it provides a more intuitive representation of PCIM/PCIMe and QPIM in XML. Moreover because XML Schema is used to describe CIM classes, this mapping scheme gives us more validation power. The mapping strategy adopted in this work is as follows.

The mapping of CIM structural classes is straight forward. There is a one-to-one mapping of CIM structural classes to XML elements. CIM class attributes are mapped to the corresponding XML element attributes. Only concrete CIM classes are mapped. Abstract classes are omitted and their attributes are inherited by the closest concrete descendant class.

The mapping of CIM associations is more tricky because of the need to enable data re-usability. When the association does not involve reusable information the association class is not represented. Its attributes are mapped to the structural representing the *Dependent/PartComponent* of the association.

When the association involves reusable information the association class is explicitly mapped to an XML element. The end of the association representing the *Dependent/PartComponent* class is mapped to an XInclude element. XInclude is a World Wide Web Consortium (W3C) standard that enables the inclusion of a fragment of an XML document within another. XInclude uses XPointer, another W3C standard, to determine what part of a document to include. The advantage of using XInclude is that it allows for policies to be created in a modular fashion. Furthermore, some native XML databases already come equipped with an XInclude processor that replaces xinclude elements with the XML fragments they point to, making it very easy to deploy solutions that benefit from XInclude. In order to increase robustness, XML policies are validated before and after inclusion.

The extract below corresponds to a network-level policy that models an Expedited Forwarding (EF) class suitable for IP telephony. In the example xinclude elements are used to locate and include re-usable actions and conditions. The example also demonstrates how associations (e.g. `PolicyConditionInPolicyRule`) that involve re-usable policy elements are used.

```
<PolicyRule Name="EFedgePolicyRule">
  <PolicyConditionInPolicyRule>
    <xi:include href="/db/QoSPolicies/
    ReusablePolicies/
    CompoundPolicyCondition.xml
    #xpointer(//CompoundPolicyCondition[
    @Name='EFedgeConditions'])" />
  </PolicyConditionInPolicyRule>
  <CompoundPolicyAction Name="EFedgeAct">
    <PolicyActionInPolicyAction>
      <xi:include href="/db/QoSPolicies/
      ReusablePolicies/
      QoSPolicyPoliceAction.xml
      #xpointer(//QoSPolicyPoliceAction[
      @Name='EFedgePolicer'])"/>
    </PolicyActionInPolicyAction>
    <PolicyActionInPolicyAction>
      <xi:include href="/db/QoSPolicies/
      ReusablePolicies/
      QoSPolicyBandwidthAction.xml
      #xpointer(//QoSPolicyBandwidthAct[
      @Name='EFedgeBand'])"/>
    </PolicyActionInPolicyAction>
  </CompoundPolicyAction>
</PolicyRule>
```

4.2 Device-Level Policies

Device-level policies are modelled based on the DiffServ Policy Information Base (PIB). The DiffServ PIB (Chan et al., 2003) follows the principles of the DiffServ architecture described in (Blake et al., 1998) and its informal management model presented in (Bernet et al., 2002). It describes a structure for specifying management information that can be transmitted to a network device using COPS/COPS-PR. The DiffServ PIB models TCB elements as Provisioning Classes (PRCs) and Instances of these classes (PRIs), identified by unique Provisioning Instance Identifiers (PRIDs).

As with network-level policies, there are two possible schemes for mapping SPPI-based PIB to a XML format: model-level mapping and meta-model-level mapping (Youn and Hong, 2003). A model-level mapping is one in which each PIB object is mapped to a XML Schema fragment. XML element names or attributes are taken directly from PIB object names. A meta model-level is one in which the XML Schema gives a generic description of the PIB. In this case, instead of mapping PIB objects to XML element names or attributes, PIB objects are mapped to XML attributes values defined by the XML Schema.

For this work we used the model-level mapping

because it provides a more intuitive representation of the PIB in XML, and it gives us more validation power.

For the mapping strategy each PRC is mapped to a XML element and instances of that PRC (i.e. PRIs or PIB objects) are mapped to corresponding XML attributes. We chose to map PRIs to XML attributes instead of child elements as in (Clemente et al., 2005) because this way the resulting XML documents become less verbose. As an exception, PRIs or objects that refer to other Provisioning Instance Identifiers (PRIDs), such as the `ClfrElementSpecific` object, are mapped to child XML elements. This is due to the fact that PRIs containing PRIDs are at times optional, and can be left out. These elements have an attribute `Location`, which has as value, an Xpath expression locating the desired PRC element. PRCs of the same type are grouped within the same PRC container, so that XML documents become more convenient for reading and processing.

Figure 2 depicts a very simplified version of a TCB. In the figure the blocks are PRIs. The arrows depicted are linking PRIs using their PRIDs as reference.

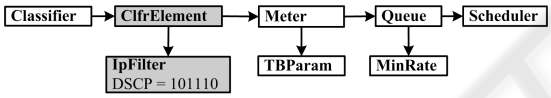

Figure 2: DiffServ PIB TCB.

The XML fragment bellow corresponds to the mapping of the highlighted portion of Figure 2. Note that although all elements that represent the DiffServ PIB belong to a specific namespace (i.e. `ds-pib`), we chose to leave the namespace prefix out of the following example for simplicity.

```
<IpFilterTable>
  <IpFilterTable BaseFilterPrid="3"
  IpFilterDscp="101110"/>
</IpFilterTable>
...
<ClfrElementTable>
  <ClfrElementTable ClfrElementPrid="1">
    <ClfrElementSpecific Location=
    "//IpFilterTable[@BaseFilterPrid=3]"/>
    <ClfrElementNext Location=
    "//MeterTable[@MeterPrid=1]"/>
  </ClfrElementTable>
</ClfrElementTable>
<MeterTable>
  <MeterTable MeterPrid="1">
  ...
  </MeterTable>
</MeterTable>
```

There are many advantages of using a XML over binary SPPI for policy representation. XML-based PIB is clear, easy to use and debug. The Xpath-based naming scheme presented here is more clear and easy to understand than the Object Identifier (OID) naming scheme employed by SPPI-based PIBs. As depicted in the XML fragment above, the generated XML document is easy readable and self-describing. Furthermore, because XML parsers are widely available, developers spend less time worrying about the details of the parser itself and more time working on the application. This certainly reduces the development cycles of PBNM applications.

5 PROPOSED ARCHITECTURE

Figure 3 presents an overview of the our proposed PBNM architecture. It is based on the PBNM architecture defined by the IETF (Yavatkar et al., 2000). The components are the same, whereas the technologies used are different.

Firstly, a native XML database serves as the policy repository. This is a natural choice taking into consideration that we only use XML for policy representation. Some native XML databases already come equipment with an XInclude processor and an XML parser and 'validator'. That way we do not have to worry about the parsing, validating and processing of XML documents, focusing only on policy authoring. Furthermore, we can organise policies in the database in a modular fashion, as described in Section 4.1, using XInclude to include the various reusable XML policy fragments.

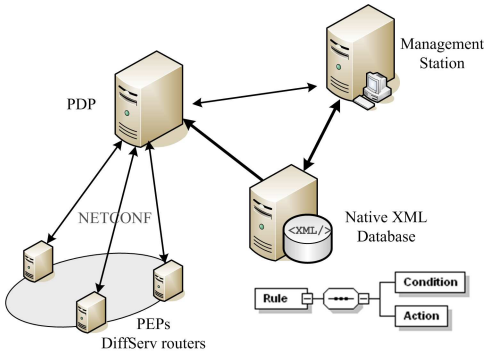

Figure 3: Proposed PBNM architecture.

Secondly, policy provisioning is done through NETCONF instead of COPS-PR. Because NETCONF is XML-based it is simple, portable, easy to understand, program and debug. Additionally we can use the same tools to process protocol messages and configuration data, greatly reducing development costs.

As per Figure 3, the management station serves

as the interface between the administrator and the PBNM system. It is used to create, insert and edit policies in the XML database through the standard API exposed by the database server. It also monitors the operation of various PDPs.

The native XML database stores policies in XML format. It automatically validates them against well defined XML Schemas. The database server includes a XInclude processor that automatically processes XInclude elements, locating and including fragments of policies stored in reusable policy collections. The other system components interact with the database server through a standard API.

The PDP is the central part of the system. It is responsible for fetching network-level policies stored in the database using the roles supplied by PEPs to discriminate between policies. It validates the fetched policies. Translates network-level policies into device-level based on the device capabilities supplied by PEPs; sets up a scheduler according to the time conditions attached to each policy and sends device-level policies to PEPs using the NETCONF API exposed by them.

The PEPs enforce policies on the DiffServ routers. They are responsible for receiving and translating common device-level policies into a set of commands specific for the type of router being managed, and feeding the resulting commands to the router.

5.1 The Communication Model

The communication model employed in this scheme is different from that employed by schemes that use COPS-PR. COPS-PR sees the PDP as the server and the PEP as the client. The PEP opens the connection with the server. In case of failure the client is responsible for re-establishing the lost connection.

On the other hand NETCONF employs a client/server model where the PDP is the client whereas the PEP is the server. The PDP is configured with a list of PEPs that it manages. It is responsible for initiating communication with all PEPs, requesting roles and capabilities of each device, and sending policy decisions. The PDP has to keep track of all open sessions to PEPs. In case of connection failure the PDP is responsible for re-establishing the lost connection. This characteristic makes this scheme less attractive than the COPS-PR scheme, with regard to failure recovery.

5.2 Implementation Details

This Section gives a brief overview of some of the details regarding the implementation of the proposed architecture.

The native XML database used was the eXist (Meier, 2006) open source database. It has XQuery support to run server side applications. It also comes equipped with an XInclude engine to process xinclude elements. We use the standard XML:DB API provided to interact with the repository.

The PDP was programmed as a multi-threaded Java client. We used the Java API for XML Processing (JAXP) to ease the parsing, validation, and processing of policies. JAXP provides an abstraction layer between the PDP application and the XML documents through the Document Object Model (DOM) and the Simple API for XML (SAX). The PDP has a simple NETCONF stack used to communicate with PEPs.

The Yenca (Bourdelon and State, 2004) NETCONF agent prototype was used as basis for the PEP server. Yenca is a simple NETCONF agent written in C by the LORIA-INRIA Lorraine Madynes Research team, in France. The agent has a modular design allowing users to extend it with modules that are loaded dynamically. We extended the Yenca agent with a DiffServ module that supports the DiffServ PIB. The module validates policies and translates them to router-specific configuration commands.

6 PERFORMANCE STUDY

The advantages of using XML in any system are its simplicity, clarity, and portability. On the other hand that design choice can be costly in terms of network usage and processing overhead because of the verbose nature of XML documents. This Section presents a performance comparison between the XML-based PBNM system proposed in this paper and a PBNM system using COPS-PR as the management protocol with a SPPI-based DiffServ PIB.

6.1 Provisioned Policies

DiffServ policies were provisioned to core DiffServ routers in four iterations, going from simple to complex. Table 1 gives a summary of the four iterations.

In iteration 1 the PDP sends policies corresponding to the Expedited Forwarding (EF) class suitable for IP telephony. In iteration 2 it adds an Assured Forwarding 4x class (AF4x) designed to support multimedia conferencing applications. In the third iteration we add an AF3x class for multimedia streaming applications. Finally, in iteration 4, a AF1x class suitable for applications that generate high throughput data is

Table 1: Provisioned policies.

Iteration	Provisioned Classes	Number of Objects
1	EF	67
2	EF, AF4x	213
3	EF, AF4x, AF3x	293
4	EF, AF4x, AF3x, AF1x	373

added. These policy classes were created following the guidelines recommended by (Babiarz et al., 2006).

6.2 Experimental Environment

Our experimental setup is composed of two machines acting as a PDP and PEP respectively. The PDP is an AMD Sempron 2600+, with 705MB RAM, running a Fedora Core 4 operating system. The PEP is a Pentium II, 233 MHz, with 450 MB RAM, running a Fedora Core 4 operating system. The PEP serves as proxy to an edge DiffServ router implemented using the Linux traffic control libraries. The the two machines are connected to each-other via a 100 Mbps switch.

The overhead imposed by our implementation was compared to that of the COPS-PR/DiffServ implementation developed by the Networks and Protocols Group of the Tampere University of Technology. We chose this implementation because like Yenca, it is written in C, and it is simple and small, providing a good basis for comparison.

The parameters taken into consideration when comparing the performance of both schemes were network usage when the PDP sends decisions to the PEP, and the processing delays generated when the PEP parses and saves policy decisions.

6.3 Network Usage

We compared the network traffic generated by each scheme for each policy provisioning iteration. Given the verbose nature of XML documents we decided to add zlib compression to XML messages and verify its effect on the traffic generated. Figure 4 shows the resulting measurements.

The results show that NETCONF generates more traffic than COPS-PR. As the number of objects transmitted increases, the amount of traffic NETCONF generates as compared to that of COPS-PR also increases.

When zlib compression is added to NETCONF messages, and the number of objects transmitted is small, both schemes generate roughly the same network traffic. However, as the number of objects increases NETCONF outperforms COPS-PR. This is due to the fact that XML documents are structurally repetitive, resulting in good compression levels for large documents.

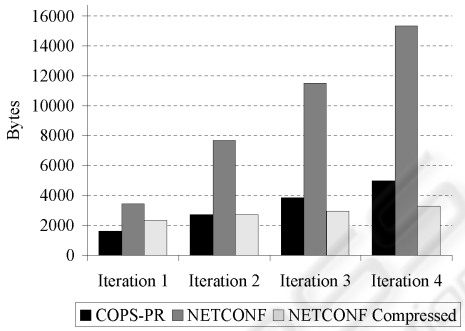

Figure 4: Network usage.

In cases where bandwidth is scarce, compression schemes such as XMill can be used. These schemes are tailored specifically for XML documents, achieving better results than general compression schemes.

6.4 Processing Delay

The second parameter considered was the time taken for the PEP to parse, decode and save policy decisions. Figure 5 shows the resulting measurements.

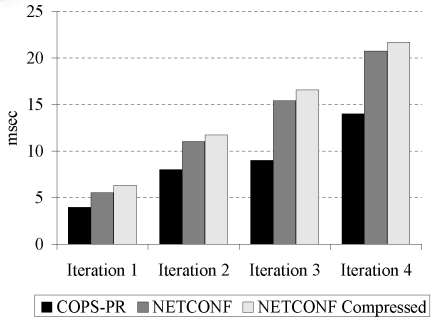

Figure 5: Processing delays.

Not surprisingly the measurements show the scheme proposed in this paper generates more processing overhead. The XML messages are more processor intensive than BER messages because they go through complex parsing and validation before data is extracted. However, as network devices become more powerful the processing overhead caused by XML documents becomes a less relevant issue.

The added processing overhead generated by adding compression to documents is very small, therefore with the use of compression the gains in re-

duced network traffic generated outweigh the losses in terms of processing overhead.

7 CONCLUSIONS

In this paper we presented the design, implementation, and evaluation of a PBNM system based on XML technologies. XML is used for policy representation throughout the entire policy life cycle, as well as in the communication protocol, reducing development costs, while promoting clarity, simplicity, and interoperability. As exemplified, XML-based (network and device-level) policies are self-describing, and easy to use.

The performance comparison results show that the scheme proposed in this paper generates less network traffic than its COPS-PR counterpart when protocol messages are compressed. The difference becomes more significant as the number of objects transferred increases.

On the other hand the measurements show that the XML-based scheme proposed here generates more processing delays than COPS-PR due to the fact that XML messages have to go through complex parsing and validation before data is extracted. However, as network devices become more powerful the processing delays caused by XML messages becomes a less relevant factor.

The added processing overhead generated by adding compression to documents is very small, therefore by using compression the gains in reduced network traffic generated outweigh the losses in terms of processing delays. In the near future we plan to verify the memory consumption levels at the PEP. This can be a concerning factor because DOM parsers have to reconstruct entire XML documents in memory in order to process it.

REFERENCES

Babiarz, J., Chan, K., and Baker, F. (2006). Configuration Guidelines for DiffServ Service Classes. RFC 4594 (Informational).

Beller, A., Jamhour, E., and Pellenz, M. E. (2004). Defining Reusable Business-Level QoS Policies for DiffServ. In *DSOM*, pages 40–51.

Bernet, Y., Blake, S., Grossman, D., and Smith, A. (2002). An Informal Management Model for Diffserv Routers. RFC 3290 (Informational).

Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., and Weiss, W. (1998). An Architecture for Differentiated Service. RFC 2475 (Informational). Updated by RFC 3260.

Bourdelon, J. and State, R. (2004). YENCA Network Management Framework. http://sourceforge.net/projects/yenca/.

Chan, K., Sahita, R., Hahn, S., and McCloghrie, K. (2003). Differentiated Services Quality of Service Policy Information Base. RFC 3317 (Informational).

Chan, K., Seligson, J., Durham, D., Gai, S., McCloghrie, K., Herzog, S., Reichmeyer, F., Yavatkar, R., and Smith, A. (2001). COPS Usage for Policy Provisioning (COPS-PR). RFC 3084 (Proposed Standard).

Clemente, F. J. G., Pérez, G. M., and Gómez-Skarmeta, A. F. (2005). An XML-Seamless Policy Based Management Framework. In *MMM-ACNS*, pages 418–423.

Enns, R. (2006). NETCONF Configuration Protocol. Technical report, Internet Engineering Task Force, NetConf Working Group.

Franco, T. F., Lima, W. Q., Silvestrin, G., Pereira, R. C., Almeida, M. J. B., Tarouco, L. M. R., Granville, L. Z., Beller, A., Jamhour, E., and Fonseca, M. (2006). Substituting COPS-PR: An Evaluation of NETCONF and SOAP for Policy Provisioning. In *POLICY '06: Proceedings of the Seventh IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY'06)*, pages 195–204, Washington, DC, USA. IEEE Computer Society.

McCloghrie, K., Fine, M., Seligson, J., Chan, K., Hahn, S., Sahita, R., Smith, A., and Reichmeyer, F. (2001). Structure of Policy Provisioning Information (SPPI). RFC 3159 (Proposed Standard).

Meier, W. (2006). eXist Version 1.0 - Open Source Native XML Database. http://exist.sourceforge.net/.

Moore, B. (2003). Policy Core Information Model (PCIM) Extensions. RFC 3460 (Proposed Standard).

Moore, B., Durham, D., Strassner, J., Westerinen, A., and Weiss, W. (2004). Information Model for Describing Network Device QoS Datapath Mechanisms. RFC 3670 (Proposed Standard).

Moore, B., Ellesson, E., Strassner, J., and Westerinen, A. (2001). Policy Core Information Model – Version 1 Specification. RFC 3060 (Proposed Standard). Updated by RFC 3460.

Schoenwaelder, J. (2003). Overview of the 2002 IAB Network Management Workshop. RFC 3535 (Informational).

Schoenwaelder, J., Pras, A., and Martin-Flatin, J.-P. (2003). On the Future of Internet Management Technologies. *IEEE Communications Magazine*, 41(10):90–97.

Snir, Y., Ramberg, Y., Strassner, J., Cohen, R., and Moore, B. (2003). Policy Quality of Service (QoS) Information Model. RFC 3644 (Proposed Standard).

Yavatkar, R., Pendarakis, D., and Guerin, R. (2000). A Framework for Policy-based Admission Control. RFC 2753 (Informational).

Youn, K. S. and Hong, C. S. (2003). A Network Management Architecture Using XML-Based Policy Information Base. In *ICOIN*, pages 876–885.