

Distributed Policy-based Network Management with NETCONF

Ntanzi Carrilho and Neco Ventura
Department of Electrical Engineering, University of Cape Town,
Rondebosch, Cape Town, South Africa
{ntanzi, neco}@crg.ee.uct.ac.za

Abstract—The limitations of the Simple Network Management Protocol are driving research for emerging network management technologies which can cope with today's fast growing networks. Considering that XML is a simple yet powerful technology for information representation and systems integration, and that Policy-based Management is a mechanism which allows for the composition of a management system from building blocks that can be introduced, modified and withdrawn at any time, without interfering with device operation, we present an architecture that integrates the XML-based Network Configuration protocol with Policy-based Management principles. This results in a distributed system capable of autonomous operation guided by management policies.

Index Terms—Netconf, Distributed Management, XML-based Management, Policy-based Management

I. INTRODUCTION

SINCE its introduction more than a decade ago the Simple Network Management Protocol (SNMP) framework standardized by the Internet Engineers Task Force (IETF) has been the dominant technology in the area of management of IP networks. Within this protocol, physical resources are represented by managed objects residing in a virtual information store i.e., the Management Information Base (MIB), which conform to the Structure of Management Information (SMI). The management information model used by SNMP is limited and does not support the hierarchical information model that is often used in network management, and particularly in device configuration management [1]. The SNMP MIBs store management information using simple Object Identifiers (OIDs) which are redundant and verbose. SNMP is a domain-specific protocol which makes the development process of SNMP agents and related applications slow and expensive [2].

The shortcomings of SNMP are driving research on the applicability of XML (the eXtensible Markup Language) [3] in network management. In this context IETF has charted the Network Configuration (Netconf) Working Group (WG) [4] with the aim of providing a network configuration protocol for network devices using XML technologies. Netconf [5] enables devices to communicate with each other using Remote Procedure Calls (RPCs) encoded in XML.

Management data are organized in XML documents which support hierarchical information models. Although the protocol was envisaged mainly for the purpose of configuration management it can also be used in other areas of network management [6].

Another technology that has gained popularity with the internet management community is Policy-Based Network Management (PBNM). The IETF has defined a standard framework for PBNM [7], setting management interfaces for creating policies, repositories for storing policies, policy decision points for evaluating policies and policy enforcement points where policies are enforced [8].

In this paper we present an architecture for distributed policy-based network management based on the IETF Netconf protocol. The architecture uses the Netconf protocol for communication between devices and leverages some principles from the IETF PBNM and Distributed Management (Disman) frameworks [9]. We introduce a Distributed Manager (DM) between the management station (manager) and managed devices (agents). DMs are distributed along the management system, therefore decentralizing the management operations. These entities are capable of operating autonomously according to management policies uploaded by the manager.

The remainder of this paper is organized as follows: Section II presents an overview of the PBNM and Distributed Management architectures developed by the IETF. Section III describes of the Netconf protocol, which forms the basis of the work presented in this paper. Section IV gives further motivation for the work. Section V describes the proposed architecture and its implementation details. Finally, conclusions are given in Section VI.

II. RELATED WORK

A. Policy-based Management within the IETF

Extensive research has been done in the area of Policy-based Management. The groundwork regarding policy-based management for distributed systems resulted from research done at the Imperial College [10], [11]. In that work management policies were divided into two groups: authorization and obligation policies. Policies were viewed as objects that define relationships between subjects (managers) and targets (managed objects). Groups of objects to which the same policies applied were classified as Domains. One of the outcomes of this research was a language for specifying management and security policies for distributed systems (Ponder) [12].

The work presented in this paper is based on the IETF network policy specifications which were widely embraced both by industry and academia.

1) IETF Policy Framework

The IETF initially proposed the mechanism of policy-based management as a framework for providing policy-based admission control decisions in the context of the Integrated Services (InterServ) model [7]. The same principles were applied to Differentiated Services (DiffServ) as well as IPsec environments. However, it was soon realized that those approaches could also be applied in other areas of network management [8]. In general PBNM allows for automated configuration of network devices through the use of policies, which are often expressed as condition-action rules.

The IETF policy framework depicted in Fig. 1 includes four functional blocks: Policy Management Application, Policy Repository, Policy Decision Point (PDP), and Policy Enforcement Point (PEP). The policy management application provides an interface to create, edit, deploy, and store policies in a repository. It also monitors the status of the PBNM environment. The policy repository is used as storage for policies. Stored policies can be retrieved by policy decision points by means of a repository access protocol. IETF recommends the Lightweight Directory Access Protocol (LDAP) for repository access even though other mechanisms such as HTTP, SNMP or SSH are possible.

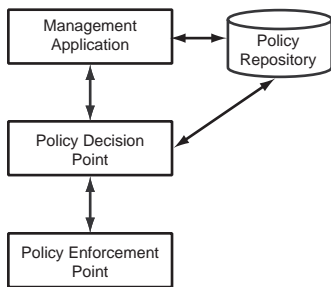


Fig. 1. Architecture of the IETF policy framework.

The policy decision point retrieves policies, interprets them, detects policy conflicts, receives policy requests from policy enforcement points and returns policy decisions to them. In general policies are expressed as condition-action rules. If a condition evaluate to TRUE, an action is triggered, which often entails (re)configuring target elements. Policy decisions are triggered by special events, polling or explicit requests.

Policy enforcement points are responsible for enforcing policy decisions made by a given PDP. The PEP hosts the target elements to which actions are applied. The PEP may also request policy decisions from the PDP. According to the IETF framework one PDP is responsible for multiple PEPs although they may also be combined and co-located. This framework defines the Common Open Policy Service (COPS) as the protocol for policy exchange between the PDP and PEP. Nonetheless other protocols such as HTTP, FTP or SNMP may also be used. The framework does not suggest any specify protocol for communication between the management application and the PDP. Policies are defined by the Policy Core Information Model (PCIM) [13].

PCIM may be used when entering policies, when storing them in repositories, and when evaluating them at PDPs. PCIM is sufficiently generic so that it can be used in various contexts.

PBNM allows for device configuration to be done either by outsourcing or provisioning. In the outsourcing model, PEPs request policy decisions from PDPs according to certain events. In the provisioning model the PDPs initiate communication and send policies that PEPs should enforce.

2) Configuration Management with SNMP

Within the IETF the Configuration Management with SNMP (SNMPconf) WG [14] recognized the positive impact policy-based management principles can have on the ever expanding networks. Within this context it defined a set of management objects for the Simple Network Management Protocol that are used to distribute policies in a common form throughout the network. The outcome of this work was an integrated management architecture capable of coping not only with the traditional network monitoring already handled by SNMP, but also with the always difficult problem of network configuration.

B. IETF Distributed Management

The increasing complexity of networks has motivated the evolution of management paradigms from the traditional centralized to distributed paradigms. The IETF Distributed Management (Disman) WG [9] defines an architecture for distributed management applications based on SNMP. Fig. 2 depicts this architecture. The architecture uses distributed managers (DM) capable of acting in a manager role to perform management functions and in an agent role where they are remotely controlled and observed. Some of the modules defined by Disman are: Script MIB, Expression MIB, Schedule MIB, and Event MIB.

The Event MIB provides the ability to monitor objects locally or remotely, taking an action when a certain trigger condition occurs. This is a very basic form of network management based on simple policies. The Expression MIB is used to define complex expressions based on local objects. This MIB is complemented nicely by the Event MIB. The Schedule MIB can be used in conjunction with the Script MIB to trigger the local execution of management scripts. Refer to [9] for a more detailed explanation of these modules and other modules defined by the Disman WG.

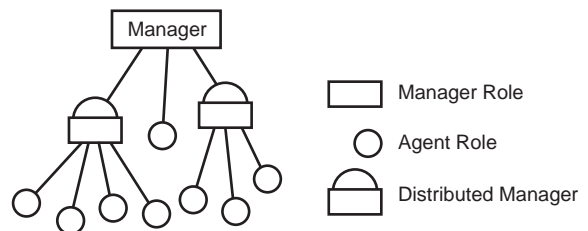


Fig. 2. IETF Distributed Management architecture.

III. NETCONF OVERVIEW

The Network Configuration (Netconf) WG [4] was formed in 2003 with the aim of developing a protocol suitable for configuration management. The WG defined a configuration protocol, its transport mappings and event notifications.

Netconf [5] uses an RPC mechanism to expose a formal Application Programming Interface (API). Managers and agents communicate with each other by sending and receiving RPCs encoded in XML. The characteristics of the protocol are as follows:

- A Netconf session is the logical connection between a network administrator or network configuration application and a network device. A device must support at least one Netconf session.
- Management information is separated into two classes, namely configuration data and state data. Configuration data are writable while state data are read-only and represents network device statistics.
- 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.

As illustrated in Fig. 3, Netconf can be conceptually separated into 4 layers: content layer, operations layer, RPC layer and transport layer. The transport protocol layer provides a communication path between the client and server. It must be connection oriented requiring a long-lived and persistent connection between the manager and the agent. Netconf is defined to be transport independent, allowing mappings to be defined for multiple transport protocols as long as they comply with the transport requirements of the protocol. The WG has selected SSH as the mandatory transport protocol because it is widely deployed and provides security and confidentiality [6]. However, SOAP over HTTP and BEEP may also be used as transport protocols (Fig. 3(b)).

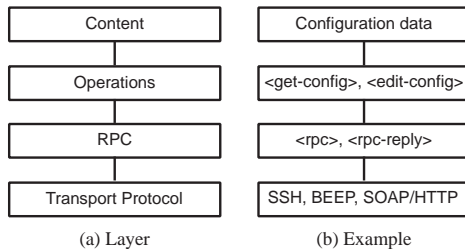


Fig. 3. Netconf Protocol Layers.

The RPC layer provides a simple, transport-independent framing mechanism for encoding RPCs. XML messages are defined by using RPC elements `<rpc>` and `<rpc-reply>`. Netconf peers use these elements to frame requests and responses in the following manner:

- `<rpc>`: is used to enclose requests sent from the manager to the agent.
- `<rpc-reply>`: is used by the agent to enclose a reply to the manager. In case the operation is successful the `<ok>` element is sent to the manager in the `<rpc-reply>`. Otherwise the agent sends a `<rpc-error>` element with the details of the error(s).

The operations layer defines a small set of base operations invoked as RPC methods with XML-encoded parameters. These operations provide the ability to write, retrieve, copy, edit and delete management information in the agent. Netconf operations are executed against

configuration datastores, which are complete sets of configuration data. An active configuration running in an agent is defined in the `<running>` datastore. Other datastores allowed are `<startup>` and `<candidate>`. The following base operations are defined by the protocol (Refer to [5] for details regarding other operations.):

- `<get>`: retrieves device configuration and state management information.
- `<get-config>`: retrieves device configuration information only.
- `<edit-config>`: changes configuration information by applying one of four operations to the target management document. It has an operation attribute that is embedded in an XML element that marks the point of the XML document affected by the operation. Operations allowed are: *merge*, *replace*, *create* and *delete*.
- `<copy-config>`: merge or replace an entire configuration datastore with new contents.
- `<delete-config>`: deletes a configuration datastore.
- `<lock>` and `<unlock>`: allows the manager to lock the configuration system in the client, and releases a configuration lock previously obtained with a `<lock>` operation, respectively.

Netconf provides a mechanism to filter management information accessed by its base operations. Filters can either be XML subtree filters or xpath expression filters [5].

The content layer presents configuration data and is outside of the scope of the protocol. It depends on a particular Netconf implementation although in future it is expected that a separate effort to specify a standard data definition language and standard content will be undertaken.

One of the strengths of Netconf is that its base functionality can be easily extended by adding capabilities that augment the protocol's base operations; they describe both additional operations and the content allowed inside operations. Netconf managers can discover additional capabilities supported by agents and use any operations, parameters, and content defined by those operations. Additional capabilities (standardized or proprietary) can be defined at any time in external documents, allowing the set of capabilities to expand over time.

The Netconf Event Notification protocol [15], which is still a work in progress, defines a mechanism whereby the manager application indicates interest in receiving event notifications from a Netconf agent by creating a subscription to receive event notifications. If the operation is successful the agent begins sending event notifications to the management application as events occur within the system.

IV. MOTIVATIONS FOR THIS WORK

The SNMP framework was designed in an era where network bandwidth and machine processing power was scarce. This protocol was designed to be very simple and efficient. Today's networks are growing at a very fast pace. They are characterized by high speed links and fast processors. New protocols are needed to cope with the management requirements of these networks, and Netconf is one of them.

One of the strengths of Netconf is that it is based on XML. Some arguments for using XML in network management are:

- It is a well established technology used by a large community of professionals, and a large number of software tools available, making the development of management applications more efficient.
- XML-based technologies are easily integrated with other applications.
- It provides powerful modeling features for structured management information.
- XML related technologies such as XML Schema [16], Document Object Model (DOM) [17] and XML Path language (Xpath) [18] provide a rich set of APIs for XML document access and manipulation.
- XML-based representations are both human and machine readable.

One other reason that makes Netconf an attractive technology for network management in general is that it provides an extensible API. Its base operations can be easily extended by adding capabilities and supporting operations.

However Netconf is not the panacea of network management. It was mainly designed with configuration management in mind, following a centralized management approach. In general configuration management operations are not executed regularly [19]. Therefore the centralized approach is often acceptable in small to medium sized networks. On the other hand in networks with Quality of Service (QoS) requirements configuration actions are required more often. Also when management of very large networks is concerned, autonomous and distributed management approaches are more favorable. In light of this argument the present work aims at creating a policy-based distributed and autonomous network management system capable of coping with today's growing networks. We use policies because they allow for the composition of a system from building blocks which can be introduced, modified and withdrawn at any time, without having rigorously tested the resulting system in every modification [20], thus resulting in a highly programmable system. Policies also provide a means of integrating network configuration management, fault and performance monitoring, and Quality of Service management (in domain specific scenarios such as DiffServ).

V. THE PROPOSED ARCHITECTURE

A. General Overview

In order to achieve the stipulated goals, we designed an architecture for distributed management leveraging some of the principles described in Section II.

Fig. 4(a) depicts this architecture, where the principal element of the design is the Distributed Manager (DM) introduced between the management application and managed devices. Managed devices are assumed to be running Netconf agents. One DM is capable of taking over management operations for its network domain according to predefined policies that are uploaded by the management application. This localizes the associated management traffic. This type of setup is mainly suited for applications that require minimal user interaction, would potentially

consume a significant amount of network resources due to frequent polling or large data retrieval, or require close association with the device(s) being managed [21]. DMs are autonomous entities which are capable of operating in “off-line” mode, therefore minimizing the use of slow and expensive WAN links.

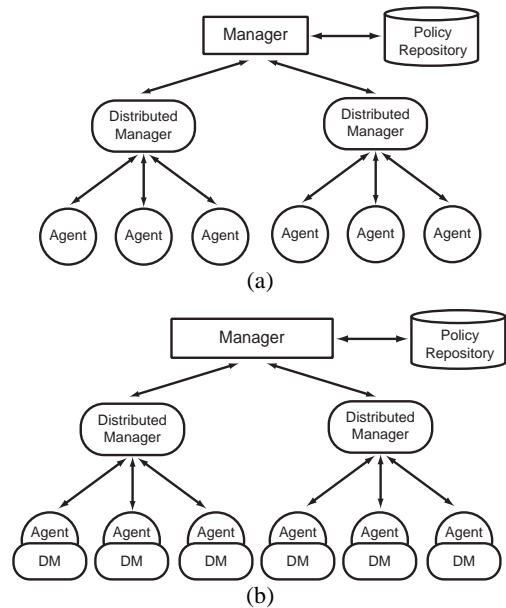


Fig. 4. The proposed architecture. The general management scenario is depicted in (a), where one DM is in charge of managing multiple Netconf agents. In (b), DMs are co-located with Netconf agents (suitable for specific management domains).

While Fig. 4(a) illustrates the general management scenario, Fig 4(b) shows that it is also possible to incorporate some of the DM functionality in the managed devices. The second setup is more favorable in situations where management operations are tailored for specific network management domains (e.g. DiffServ). This situation presents similarities to the policy framework defined by the IETF. In this case a DM would correspond to a PDP, whereas Netconf agents complemented with DM functionality would correspond to PEPs. In 4(a) and 4(b) network elements communicate with each other using Netconf RPCs over SSH.

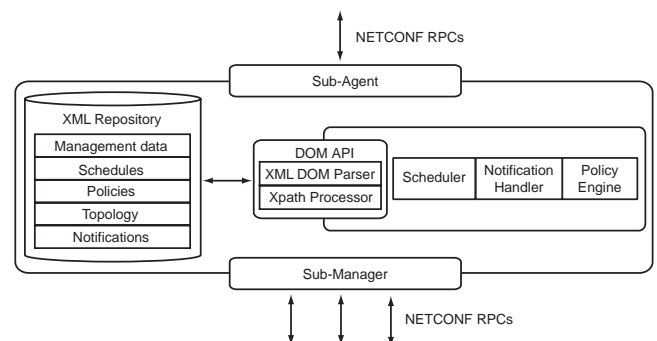


Fig. 5. Architecture of a Distributed Manager.

The Policy Repository is a native XML database, represented as a collection of XML documents. Because the Netconf manager is also XML-based, integration with the database is seamless. The manager uses standard XML APIs to access the database in order to list, insert, update, or

delete management policies using a Graphical User Interface (GUI). The manager can re-use policies stored in the database or create new policies based on existing ones.

Fig. 5 illustrates the general architecture of a DM. The Sub-Agent (SA) deals with the agent side of the DM. It is responsible for handling RPCs sent by the management application and sending notifications back to the manager. The XML Repository is used to store configuration information, management policies, XML Schemas, notifications, and network topology information. The Scheduler is responsible for maintaining a schedule of events that trigger policy enforcement. The Policy Engine is in charge of controlling the application of policies stored in the repository. The policy engine is triggered by events controlled by the scheduler. The Sub-Manager sends RPCs to the various agents and processes their replies. Agent replies can either be stored in the repository or temporarily in a cache in order to be easily accessible. The Notification Handler stores and forwards notifications sent by agents to the manager. It is also responsible for passing notifications defined in the DM to the Sub-Agent, which will forward them to the manager. The Sub-Manager makes use of threads for concurrent execution. All the modules make use of the DOM interface in order to access the XML repository, manipulate XML documents, or validate them against schemas. DOM reconstructs XML trees in memory and provides an interface to delete, add, or manipulate XML document subtrees.

B. Implementation Details

XML Schemas are used to define the structure of management information. The management information in the DM is divided into four main elements: Policy, Schedule, Target, and Notification. Fig. 6 illustrates a portion of a Schema definition corresponding to the Policy element¹. The Role attribute depicted in Fig. 6 is central to the design of the PBNM system. One role can be associated with one or more Rules. Each agent is assigned one or more roles. Therefore the applicability of policies to agents is specified by assigning a role to it. The use of roles facilitates the selection of one or more policies from a larger set of available policies. Roles can also be used to provide access control to the system.

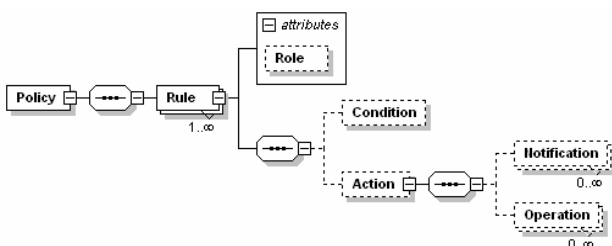


Fig. 6. Policy information model¹

From Fig. 6 a policy is defined by a condition-action pair that forms a rule. When a condition evaluates to TRUE an action is executed against target agents. An action may

¹ Dashed boxes correspond to elements (complexType) which have zero minimum occurrences. Multiple boxes correspond to elements (complexType) that have unbounded maximum occurrences (theoretically).

entail sending Netconf RPCs to agents, sending notifications to the manager station, or both.

Conditions may be evaluated by using predefined Xpath expressions stored in the repository. Xpath is an XML language used to navigate, select, or extract information from XML documents. Xpath may also be used to evaluate arithmetic, logical, and comparison expressions. This functionality can be useful in cases where complex formulas or expressions relating different elements have to be evaluated (e.g., formulas to evaluate device link utilization [19]). Because the base Netconf xpath filter only returns document subtrees [5], the implementation has to evaluate expressions locally. At times unconditional execution of actions is required; the DM also supports this mode of operation. This is very useful in cases where the manager wants to send the same RPC to a number of devices.

The Schedule element contains a list of scheduled events, both active and inactive. A scheduled event may be activated by activating a role associated with it. When an event is activated it triggers the enforcement of policies linked to this event by the same role. Scheduled events can be periodic, calendar, or one-shot [22]. For the sake of simplicity the DM only supports triggering of policy enforcement by events which are time-based (i.e., raised by a timer). However it is also useful to trigger policies by means of other kinds of events, such as monitoring events.

The target element contains a list of all the devices being managed, their network addresses, security related information, and the roles associated with each agent. When a certain policy is activated, its corresponding role will define the agents to which the policy is executed.

```
<rpc message-id="1">
  <policy>
    <activate>
      <roles>
        <role>utilizationThresholds</role>
      </roles>
    </activate>
  </policy>
</rpc>
```

Fig. 7. XML RPC used to activate a role “utilizationThresholds”.

Fig. 7 provides a simple example for a scenario where the manager application activates a role named utilizationThresholds. This role corresponds to policies defined to monitor interface link utilization in target agents (i.e., agents labeled with the same role), and send notifications to the manager in case thresholds are crossed.

The Notification element is used to store notifications sent by agents. In order to manage bandwidth the manager has the option of instructing the DM to only forward notifications having high priority; notifications with a low priority may be stored in the repository for later access.

All the mentioned management information is stored in the repository. This information can be accessed and modified by the manager using base Netconf operations described in Section II.A.

VI. CONCLUSIONS

In this paper we introduced an architecture for distributed network management based on the IETF Netconf protocol. The system makes use of policies for defining management

operations. Policies allow for the composition of a management system from building blocks which can be introduced, modified and withdrawn at any time, without interfering with device operation. The architecture introduces a Distributed Manager which is responsible for managing multiple network devices. DMs may also be combined and co-located with managed devices in domain specific management scenarios (e.g. DiffServ).

By using Netconf the system inherits all the advantages of XML-based applications such as short development cycles, easy integration with other applications, powerful modeling features for management information using XML Schemas, rich set of APIs for application development, and interoperability between heterogeneous systems. The Netconf-based system can be extended by adding capabilities to the Netconf protocol and by uploading new policies to Distributed Managers.

The combination of Netconf and Policy-based Management results in a very simple and efficient way of deploying, accessing, modifying, and controlling management policies for large scale networks. The resulting system is capable of operating autonomously based on management policies defined by the network manager.

The components described in Section V. A. are being developed as a proof-of-concept prototype that will be used to demonstrate and test the proposed architecture.

ACKNOWLEDGMENT

The authors would like to thank Telkom SA, Siemens, the National Research Foundation (NRF) and the Department of Trade and Industry (DTI) for supporting this research project.

REFERENCES

- [1] M.-J. Choi, J. W. Hong, and H.-T. Ju, "XML-Based Network Management for IP Networks," *ETRI Journal*, vol. 25, pp. 445-463, Dec. 2003.
- [2] F. Strauß and T. Klie, "Towards XML Oriented Internet Management," *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management*, Colorado Springs, 2003.
- [3] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, *Extensible Markup Language (XML) 1.0 (Third Edition)*, World Wide Web Consortium (W3C) recommendation, Feb. 2004; <http://www.w3.org/TR/2004/REC-xml-20040204/>
- [4] S. Leinen and A. Bierman, Network Configuration Working Group (NETCONF), <http://www.ietf.org/html.charters/netconf-charter.html>
- [5] E. R. Enns, *NETCONF Configuration Protocol*, Internet draft, Feb. 2006
- [6] C. Mi-Jung, C. Hyoun-Mi, J. W. Hong, and J. Hong-Taek, "XML-Based Configuration Management for IP Network Devices," *Communications Magazine, IEEE*, vol. 42, pp. 84-91.
- [7] R. Yavatkar, D. Pendarakis, and R. Guerin, *A Framework for Policy-based Admission Control*, IETF RFC 2753, Jan. 2000; <ftp://ftp.rfc-editor.org/in-notes/rfc2753.txt>
- [8] P. Martinez, M. Brunner, J. Quittek, F. Straub, J. Schönwälder, S. Mertens, and T. Klie, "Using the Script MIB for Policy-based Configuration Management," *Proceedings of the IEEE Network Operations and Management Symposium (NOMS'02)*, Florence, Italy, 2002.
- [9] R. Presuhn, Distributed Management Working Group (DISMAN), <http://www.ietf.org/html.charters/disman-charter.html>
- [10] M. Sloman, "Policy Driven Management for Distributed Systems," *Journal of Network and Systems Management*, Plenum Press, vol. 2, pp. 333-360, 1994.
- [11] D. Marriott and M. Sloman, "Management Policy Service for Distributed Systems," in *IEEE 3rd International Workshop on Services in Distributed and Networked Environments (SDNE'96)*. Macau, 1996.
- [12] N. Damianou, "A Policy Framework for Management of Distributed Systems," Ph.D. dissertation, University of London, London, UK, 2002.
- [13] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, *Policy Core Information Model - Version 1 Specification*, IETF RFC 3060, Feb. 2001; <ftp://ftp.rfc-editor.org/in-notes/rfc3060.txt>
- [14] J. Saperia and D. Partain, Configuration Management with SNMP (SNMPconf), <http://www.ietf.org/html.charters/OLD/snmpconf-charter.html>
- [15] S. Chisholm, K. Curran, and H. Trevino, *NETCONF Event Notifications*, Internet draft, work in progress, Jan. 2006
- [16] W3C, *XML Schema Part 0, 1, 2*, World Wide Web Consortium (W3C) recommendation, May 2001; <http://www.w3.org/XML/Schema>
- [17] L. Wood, *Document Object Model (DOM) Level 1 Specification*, World Wide Web Consortium (W3C) recommendation, Oct. 1998; <http://www.w3.org/TR/REC-DOM-Level-1/>
- [18] J. Clark and S. DeRose, *XML Path Language (XPath) Version 1.0*, World Wide Web Consortium (W3C) Recommendation, Nov. 1999; <http://www.w3.org/TR/xpath>
- [19] A. Leinwand and K. F. Conroy, *Network Management: A Practical Perspective*, 2nd ed: Addison-Wesley Professional, 1996.
- [20] P. Flegkas, P. Trimintzios, G. Pavlou, I. Andrikopoulos, and C. F. Cavalcanti, "On Policy-based Extensible Hierarchical Network Management in QoS-enabled IP Networks," *Proceedings of the IEEE Workshop on Policies for Distributed Systems and Networks (Policy '01)*, Bristol, UK, 2001.
- [21] D. Gavalas, E. Department, D. Greenwood, M. Ghanbari, and M. O'Mahony, "Hierarchical network management: a scalable and dynamic mobile agent-based approach," *Computer Networks: The International Journal of Computer and Telecommunications Networking*, vol. 38, pp. 693 - 711, April, 2002.
- [22] D. Levi and J. Schoenwaelder, *Definitions of Managed Objects for Scheduling Management Operations*, IETF RFC 3231 Jan. 2002; <ftp://ftp.rfc-editor.org/in-notes/rfc3231.txt>

Ntanz Carrilho received his BSc in Electrical and Computer Engineering at the University of Cape Town in 2003 and is pursuing an MSc in Electrical Engineering at University of Cape Town.