SOLVING TRANSACTIONAL CONTROL IN CURRENT MANAGEMENT FRAMEWORKS

Vitor Roque
Polytechnic Institute of Guarda, ESTG, Guarda – Portugal
email: vitor.roque@ipg.pt

José Luís Oliveira
University of Aveiro, DET, Aveiro – Portugal
email: jlo@det.ua.pt

Rui P. Lopes
Polytechnic Institute of Bragança, ESTiG, Bragança – Portugal
email: rlopes@ipb.pt

Keywords: Policy based management, policies, network management, transactions.

Abstract: Policy Based Network Management has been presented as a paradigm for efficient and customisable management systems. The IETF has provided a framework to describe the concept but some aspects still open like transactional control. In fact transactional control mechanisms are receiving today great attention in the scope of network management. In here, we identify the lacks of current management paradigms concerning transactional control and we propose a policy-based network management system that allows specify operations over aggregations of agents and that provides high-level atomic transactions.

1 INTRODUCTION

Today's information systems are typically based on a large numbers of heterogeneous computing devices connected through communication networks, and joining together various resources, services, and user applications. These resources and applications are now indispensable to organizations, but as the whole system becomes increasingly larger and more complex, also a higher number of elements can be the source for the disruption of critical business operations. In fact, network management has gained in the last years great importance due the increased dependence of the enterprises on their computer systems, networks and networked applications. This dependence has made *availability* and *performance* of the network infra-structure and network services more critical than ever. In addition, the growth in size and complexity of modern networks increases the need of standard configuration mechanisms for an efficient network management. It is expected that these mechanisms are strongly related to fault-tolerance systems as well with performance management systems. The concept of policy-based management has emerged during the last years as an adequate paradigm to deal with this type of requirements and this concept has been widely supported by standards organizations such as the IETF and DMTF. In fact the Policy Working Group is chartered to define a scalable and secure framework for policy definition and administration.

The development of policy-based management applications, due to the diversity and type of equipments, can be very complex in structure, with complex relationships between their constituent parts. Because of these, the success of network operations (configuration operations and others) is a critical issue in network management thus deserving great attention. In fact transactional control mechanisms are receiving today great attention in the scope of network management. In here, we identify the lacks of current management paradigms concerning transactional control and we propose a policy-based network management system that allows specify operations over aggregations of agents and that provides high-level atomic transactions.

2 NETWORK MANAGEMENT

PBNM technologies have been developed to reduce the configuration complexity of the network and its nodes. It is desirable that a network management system technology will be provided with the ability to automatically manage the network configuration based upon high-level rules, more or less in the same way business-oriented requests are issued (Sloman 1994). In fact, policies definition aim at replacing the dependency on vendor and device specific configuration commands, thus making network management a homogenous task independent of the installed equipment. For example, a management system should be capable, in a specific management situation, of offering facilities to reconfigure the whole system without the network administrator having to worry about the configuration details of network equipment.

The *policy* concept is quite wide (Moore, Ellesson et al. 2001) – policies can be applied in QoS management, access control, security or other areas. Policies are defined by users, such as network administrators or operators, stored and handled in policy servers, and deployed on network nodes. The execution of a policy depends on the evaluation of a check-action rule that is activated when the implicit condition or conditions are verified. Although the main ideas are simple, the development and wide adoption of policy-based management applications have been complex. Issues such as policy format, enforcement mechanisms, conflict avoidance, low level protocol mapping, user-interface representation and edition, just to name a few, are being a mater of research and standardization. Today several standardization organizations are working on this subject, namely IETF and DMTF. As the result of their work new proposals have been developed such as COPS (Durham, Boyle et al. 2000; Chan, Seligson et al. 2001), SNMP for Configuration (MacFaden, Partain et al. 2003), PCIM (Moore, Ellesson et al. 2001).

These works also stimulate the definition of the policy framework architecture, composed of four functional entities namely the Policy Management Tool (Policy Console), the Policy Repository, the Policy Decision Point or Policy Server (PDP) and the Policy Enforcement Points (PEP) (Kosiur 2001).

This model describes the key components but it does not prescribe any kind of implementation details such as distribution, platform or language.

Policies and Transactions

The configuration activity and policies execution causes state changes in network elements and it is critical that this operation is executed atomically to maintain the network elements in a consistent state.

PBNM systems must have mechanisms that in case that the configuration of any equipment fails, all the other network equipment which is involved in that configuration must return to the last good configuration installed.

Considering this situation, PBNM systems must implement the ACID properties of Distributed System theory, i.e., the applications must have the capability of monitoring the execution of configuration operations. If an operation would compromise one of the ACID properties the system must have the ability to return to the previous configuration state (Gray and Reuter 1994; Coulouris, Dollimore et al. 2001). We can associate the term transaction with the term policy, i.e., we may consider a policy as a transaction, where we must execute all of the operations (rules – conditions/actions) or none, in case something fails.

The definition of a transaction mechanism is also necessary due the wide-nature of the networks, i.e., in large networks the time necessary to configure the whole network, all the network elements, could be long, running for minutes, hours or even days. Although the network transportation protocols have mechanisms, like “timeout” and “keep alive” messages on COPS, to control the fault-tolerance, this mechanisms/functions seems to be insufficient because in certain situations the configuration effort is too valuable to be undone.

3 A MECHANISM FOR TRANSACTIONAL CONTROL

The definition of a mechanism for transactional control on PBNM systems is essential because the trust on transactional integrity at the protocol level seems insufficient (MacFaden, Partain et al. 2003). The mechanism here proposed uses the concepts of server (PDP) and agent (PEP) of the IETF conceptual policy model. The communication model adopted is unidirectional, from PDP to PEP, because the network management will be done in a centralized way at the PDP, which will also have the responsibility of querying the PEPs concerning the policies installation. In this mechanism the configuration process is based on five basic commands: (1) *examine* *(PDP->PEP)* – Tells the agent to carry out the operations related to the policy

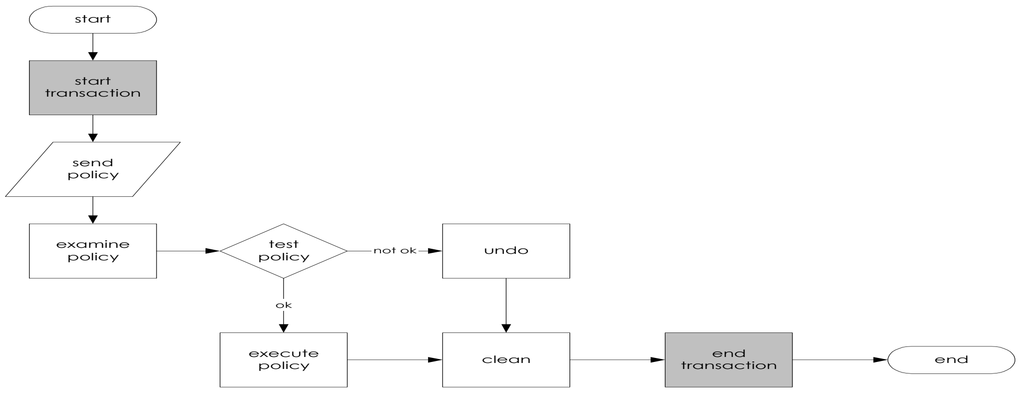

Figure 1 - Flow of the proposed fault-tolerance mechanism.

without applying them; (2) _test_ *(PDP->PEP)* – this command is issued by the server to ask the agent if the policy installation can be applied with success; (3) _execute_ *(PDP->PEP)* – it is indicated to the agent that it must install the policy; (4) _undo_ *(PDP->PEP)* – if error, this command is issued by the server to indicate agent to forget about the policy installation and (5) _clean_ *(PDP -> PEP)* – it will be issued by the server to indicate that the agent should release the resources used by the temporary storage of the policy.

The working mode of this mechanism is the following: suppose that is planned to install a policy on the network elements of an administrative domain. In this mechanism it is assumed that the PDP is the central element in the policies distribution and the PEP´s (agents) must be always in a listening state, waiting to be contacted by the server. The configuration information (policies) is sent to the various agents through the _examine_ command, and it is of the responsibility of the agent to verify if the policies can be installed, i.e., if there is no incompatibility concerning the possibility of policies installation. Next, the server issues the _test_ command, to verify if no problems were detected and that the policies are prepared to be installed in the agent. This process is repeated in all agents of the PEPs in the administrative domain. On the next phase, the server (PDP) requests to all agents (PEPs) to _execute_ the command – the effective installation of the policy. If the query made to the _test_ command was affirmative by all the agents, the server issues the _clean_ command for the agents to release the resources allocated during the operations. In the case of non affirmative answer by one or more agents to the _test_ command, the server issues the _undo_ command for the reposition of the precedent state configuration. Finally, the server issues the _clean_ command to all the agents, telling them to release the resources allocated during the operations. On Figure 1 it is represented the flow information diagram of the mechanism proposed.

The configuration information is sent in a high-level file format, following the Extensible Markup Language (XML) (W3C 2003). This file content is then translated in accordance with the language and protocol used by the PEP. All these tasks are of the responsibility of the server (PDP) and are independent of the transactional control mechanism proposed. If the transaction was successful, it is possible to assure, that the network has passed from one consistent state to another consistent state.

4 TRANSACTIONS REPRESENTATION

Transactions, or the definition of atomic sets of policies, should be easy to define, in a graphical user interface. The purpose of this mechanism is related with the previous work, namely the build of a prototype for policy representation in a graphical way (Roque, Oliveira et al. 2003).

With the development of this mechanism it will be possible to endow the visual policy editor with transactional control, i.e., to make available the interfaces of the transactional mechanism to be used by the application (visual policy editor), keeping in this way the separation between application – mechanism.

The policy editor allows the user to define policies using a single specification language: XML. The resulting information is then transferred to the network elements using a common syntax. The XML file must follow the rules defined in a standard template (DTD/Schema) where it is referred all the relevant information that the policy must have. After validation, the policy must be sent to the network

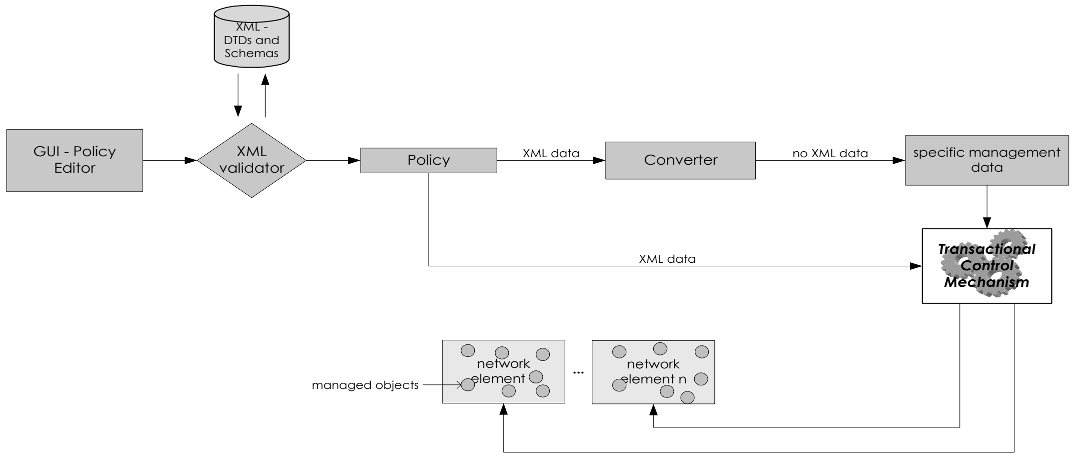

Figure 2 – XML model for policies.

elements (Figure 2). If the network elements (agents) are XML compatible, the policy is immediately delivered through the transactional control mechanism, in the case that network elements do not support XML, the policy will have first to pass through a converter. The converter will transform the XML policy into a specific language that the network element could understand. After the transformation the policy is delivered to the network element by the transactional mechanism.

Having the network elements (agents) received the policies, the transactional mechanism grantees that the policies are manipulated in accordance with the ACID properties.

5 CONCLUSIONS

Policy-Based Configuration Management is a methodology wherein configuration information is derived from rules and network-wide objectives, and is distributed to many potentially network elements with the goal of achieving a consistent network behaviour.

The configuration activity causes state changes in the network elements and it is critical that the configuration system treats all the configuration operations atomically.

The development of a transactional mechanism for the management of policies installation in the various network elements, is because on the work that comes being developed, we discovered some gaps in the existing management models about this matter, i.e., this kind of work is left to the central management systems, that will have to handle, case by case, its implementation.

Within this paper we made some considerations about the importance of the transactional mechanisms at the level of policies specification and presented some solutions that could be used in today systems. The definition of this mechanism is based on the supposition that simple *commit/rollback* semantics of an ACID transaction is enough. With this supposition, even in the presence of failures, the state transition of the network elements is guarantee by the atomicity of the transactions.

6 REFERENCES

Chan, K., J. Seligson, et al. (2001). COPS Usage for Policy Provisioning (COPS-PR) - RFC3084, IETF.

Coulouris, G., J. Dollimore, et al. (2001). Distributed Systems - Concepts and Design, Addison-Wesley.

Durham, D., J. Boyle, et al. (2000). The COPS (Common Open Policy Service) Protocol - RFC2748, IETF.

Gray, J. and A. Reuter (1994). Transaction Processing: Concepts and Techniques, Morgan Kaufmann.

Kosiur, D. (2001). Understanding Policy-Based Networking, Wiley.

MacFaden, M., D. Partain, et al. (2003). Configuring Networks and Devices With SNMP - RFC3512, IETF.

Moore, B., E. Ellesson, et al. (2001). Policy Core Information Model Specification v1 - RFC3060, IETF.

Roque, V., J. Oliveira, et al. (2003). Visual Composition of Management Policies. 4th Conference on Telecommunications (ConfTele2003), Aveiro - Portugal.

Sloman, M. (1994). "Policy driven management for distributed systems." Journal of Management Information Systems **2**(4): 333-360.

W3C (2003). W3C XML Documents.