WIRELESS WORLD
RESEARCH FORUM

A Policy-Based System for Network-Wide Configuration Management

R. Romeikat, B. Bauer, University of Augsburg, Institute of Computer Science
H. Sanneck, C. Schmelz, Nokia Siemens Networks, Network Technology

***Abstract*—We propose the vision and concept of a system for management of configuration data in a mobile network. Policies encapsulating operators' high-level knowledge will enable the network with self-configuration capabilities on a network-wide focus, thereby reducing the amount of work and time to be spent by human operators for operation, administration, and maintenance of the network. In such a system, several aspects need to be considered such as an appropriate policy language and engine to define and enforce correct network behavior, a simulation environment and effective conflict detection mechanisms. A toolbox will support operators in development, deployment, and runtime management of high-level policies and corresponding low-level rules. Our approach of automated configuration management is intended to contribute to the vision of a self-managing mobile network.**

***Index Terms*—mobile network, policy, rule, self-configuration.**

Introduction

IN mobile communications, a crucial task to every mobile network operator (MNO) is the operation, administration, and maintenance (OAM) of the underlying mobile network and the comprised network elements (NE). To this day, the tasks and workflows for OAM have mainly been performed by human operators, whereas only a few tasks and workflows have been automated so far. In the future, MNOs and their human operators will be facing some challenges. Evolution leads to more dynamic and distributed network architectures with large numbers of specialized NEs. Increasing network dynamics will require more frequent and faster (re)configurations of NEs. In addition, a broader feature offer and increasing network size and complexity will generate more configuration data and require faster processing of configuration tasks compared to today. Another factor contributing to this development is the co-existence of several network generations and technologies in parallel, e.g. GSM, UMTS, and 3G LTE, resulting in more effort for OAM and ascending operational expenditures (OPEX). In order to address these challenges, complex and specialized solutions are required helping to automate configuration tasks and workflows to the greatest extent possible.

A building block of such a solution for automated configuration management (CM) is provided with [1], which deals with network-wide configuration management in a highly distributed mobile network. Assuring CM data consistency on a network-wide focus is a non-trivial task due to the distribution of the NEs and interdependencies between their configurations. Moreover, particular attention has been directed to data consistency for the fast and incremental roll-out and alignment of CM data as illustrated in fig. 1, in contrast to rolling out and aligning complete CM data stores, e.g. performed when preparing a new network configuration plan. As a result, a middleware layer in form of a data management subsystem is proposed allowing flexible synchronization and rollback of multiple configuration changes between NEs and the associated Element Management System (EMS) simultaneously. However, human interaction is still necessary to take decisions and initiate appropriate actions in case some of the configuration changes fail.

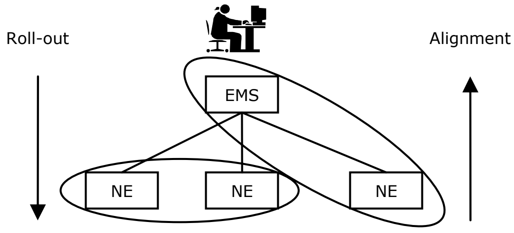

Fig. 1. Consistency model for roll-out and alignment of CM data

In order to perform the next step towards management automation, policies are an appropriate means. To a certain degree, policies can add autonomous decision capabilities to network management by specifying actions that must be taken when a set of associated conditions are met. For configuration management, such decisions could be e.g. to reschedule some of the failed configuration changes in dependence on the failure cause and the network state. Thereby, the degree of configuration automation can be increased and operational expenditures be lowered by reducing the necessary degree of human interaction. The process of how these policies can be developed is addressed in the following chapters of our contribution.

In the next chapters, background information on the proposed CM data consistency mechanism and an introduction to policies are provided. Both will influence advanced policy mechanisms to be developed. Furthermore, we introduce the vision of a policy-based system supporting all phases of the configuration management process, also including details about features, components, and requirements. Finally, we present our work plan towards such a system.

Technical Background

In this chapter, we give background information on the present network configuration mechanisms, including the role policies play in that approach. We then focus on policies and describe which role they play in the field of network management. The technical details given in this chapter represent influencing factors for the development of a policy-based management system for network configuration which is presented afterwards.

Transaction-based Configuration Mechanism

For purposes of configuration data consistency in a mobile network, [1] proposes a concept based on "distributed, adaptive transactions". We first introduce the concept of transactions, then explain their role in the configuration work cycle, and finally describe how adaptiveness is supported by the use of policies.

For developing a consistency concept based on transactions, ideas from distributed databases were adopted. This is why some of the terminology employed is inspired by database management systems. However, this is only for conceptual reasons and does not imply any requirements regarding a concrete database system or type to be used. The approach avoids any product dependency as any transactional functionality is realized by hand and a database is only used as a primitive data store.

Transactions are defined as individual, indivisible operations based on the ACID (Atomicity, Consistency, Isolation, and Durability) paradigm [2]. Network configuration changes are communicated as transactions between the NEs and their EMS. Such a transaction is called NE transaction (NET); it refers to one NE and consists of one or more atomic interdependent operations (e.g. insert, update, and delete operations). Execution of a transaction transforms the configuration data of an NE from one consistent state to another one. A master-replica paradigm is adopted where the replicas (=NEs) may autonomously commit configuration data as "tentative" and the master (=EMS) follows the state of the replicas, but may force a rollback later. In this context, a commit carries out the configuration change whereas a rollback withdraws the new configuration and returns to the previous consistent state. Dependent on the success or failure of a transaction, changes are finally committed or rolled back as a whole. Transactional semantics between an individual NE and the associated EMS assure that both of them always converge to a consistent state.

Multiple transactions having interdependencies are arranged as a transaction group (TG). In contrast to a single transaction, a TG represents configuration dependencies between different NEs. As different NEs are affected, we also speak of a distributed transaction mechanism. Once the TG is committed, the whole group of NEs is transferred from one consistent state into another one. In the presence of transaction failures, it is possible to either completely rollback the TG or to partially commit the TG and re-assess the configuration dependencies. TGs with no interdependencies may be executed in parallel. The execution of these transaction groups is controlled by a centralized transaction manager (TM) located at the respective EMS. Multiple TGs can again be grouped as network transaction (NT). NTs are independent of each other and are executed consecutively.

The configuration work cycle is characterized by the rollout of configuration data from an EMS to numerous NEs and the alignment of the actual configurations of the NEs towards the EMS. Based on the transaction mechanism described, it consists of several phases. By employing TGs covering several NEs, dependencies between NEs are expressed in the "plan" phase. The distributed transaction mechanism is made adaptive in the sense that the original expression of dependencies may be revised during the "execution phase" of the TG. Dependent on the overall state of the TG after rollout to the respective NEs, it can be rolled back during the "monitor" phase, or individual NETs can be committed and aligned to the EMS at an operator-defined time to finish the configuration process. In order to support this process, simple low-level policies are already considered and implemented in the TM, like "commit the whole group if 90% of the individual transactions have been committed". This approach represents a two-phase commit and considerably improves an all-or-nothing approach. It has a weaker consistency requirement, allowing a temporarily inconsistent state until the failed transactions have been committed at a later date. By correlating the states of the individual transactions in the "analyze" phase, more intelligent decisions supported by policies are performed by the operator in the further configuration process. All phases of the configuration work cycle are influenced by knowledge held in the EMS and an appropriate policy repository. With regard to the characteristics of the transaction-based configuration approach as a potentially "autonomic" system, fig. 2 shows a mapping of the major building blocks described to the Monitor-Analyze-Plan-Execute (MAPE) model known from autonomic computing [3].

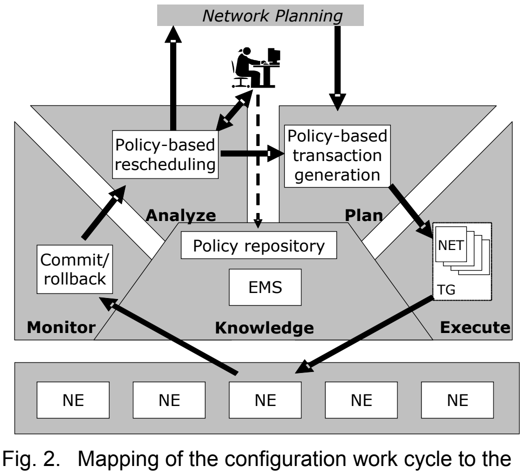

Fig. 2. Mapping of the configuration work cycle to the MAPE model

The ability to arrange multiple interdependent configuration changes in a transaction and multiple transactions in a transaction group is a major achievement. In this respect, the approach provides a basic concept for configuration automation as it provides a means for controlling the network configuration on an NE group rather than individual NE level and thus helps avoiding inconsistent configuration states. Like that, human operators are freed of the treatment of simple, recurring problems in the roll-out of network reconfigurations. From the operator's point of view, a group-wide rollback may appear only as a single operation. However, human interaction is still necessary to take decisions and initiate appropriate actions after the rollback of a transaction or transaction group. In case all transactions within a group are successful except one, e.g. due to a connection loss, rescheduling the failed transaction to a later date may be an appropriate means, allowing a temporarily inconsistent state until the affected connection has recovered. Support for more complex decisions is desirable in the "analyze" phase by correlating the states of the individual transactions. Besides that, it is feasible to include other relevant network state information into an automated decision process. These goals can be reached by encapsulating human operator knowledge in a machine-executable way, driving automated transaction generation and management. In this context, policies together with an appropriate management system are an effective means for further automating network configuration.

Policies

During the last couple of years, policies have gained attention both in research and industry. A main goal of policy-based management is to perform management tasks at a high level of abstraction. Policies are considered as an appropriate means for controlling the behavior of complex systems. They are used in different domains for automating system administration tasks, such as configuration, security, or Quality of Service (QoS) monitoring and assurance. The idea behind policy-based system management is allowing administrators to modify system behavior without changing source code or requiring information about the dependencies of the components being governed [4]. The system can continuously be adjusted to externally imposed constraints by changing the determining policies.

According to [5], policy management is the usage of rules to manage the states of managed entities and to accomplish decisions. In this context, a policy is defined as an aggregation of policy rules. Each policy rule consists of a set of events, conditions, and a corresponding set of actions. An event model is used to trigger the evaluation of a policy condition clause. The conditions define whether the policy rule is applicable, whereas the actions are allowed to execute as soon as the corresponding set of conditions are met. Fig. 3 shows these three building blocks and their relationships as classes modeled in Universal Modeling Language (UML) [6]. It can be seen that each event, condition, and action is assigned to at least one rule. The other way round, a rule is comprised of at least one event, condition, and action.

According to its set of actions, a policy rule is triggered statically or dynamically. Static policy rules apply their set of actions dependent on pre-defined parameters. A simple example would be “no network access for employees from 9pm to 6am”. In contrast, dynamic policy rules are only enforced when needed as they are based on the current network state which changes over time. An example in this case would be “when network is congested, emails are limited to 1 MB.” It is worth to be mentioned that there are also implicit policy rules when negating explicit rules. In case of the two example rules mentioned above, implicit rules are allowing network access to employees from 6am to 9pm and allowing emails to be larger than 1 MB when there is enough network traffic left respectively.

Researchers have proposed different approaches for specifying policies, starting from formal policy languages that can be interpreted and processed directly by a computer, over rule-based policy notations using if-then-else constructs, up to representation of policies within a table consisting of multiple columns which describe the details of each policy. Another criterion for differentiating policy languages is the level of abstraction. High-level languages let managers concentrate on high-level business processes rather than implementation details. In order to conduce to several application domains, such languages are very generic and operate at a high level of abstraction. In contrast, low-level languages are target-oriented, vendor- and technology-specific and usually designed in a logical or programming-like style. They express constraints unambiguously and also regard a concrete enforcement mechanism [7]. The gap between high level and low level of abstraction is also referred to as “semantic gap”. Policy languages can also be differentiated regarding the usage of semantic concepts such as metadata or ontologies. Modern languages often use semantic approaches, bringing along some advantages concerning runtime extensibility and adaptability of the system [8]. Furthermore, semantically-rich policy representations can reduce the probability of programming errors, simplify policy analysis, reduce policy conflicts, and facilitate interoperability [9].

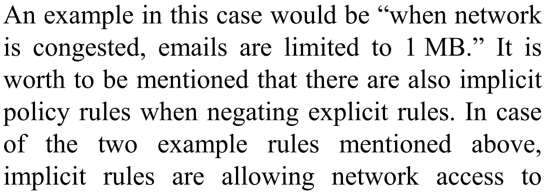

Fig. 3. A simplistic policy model

In the network management field, policies are expressed as sets of rules representing choices in the network behavior [5]. Regarding mobile communications, the role of the administrator is assigned to the network operator and the system to be managed is represented by the mobile network. There are some fields in network management where policies could play an effective role. For purposes of performance management, policies could be a means for aggregating and processing performance data in an automated and load-dependent way. In the field of alarm correlation and root cause analysis, policies could be used for performing effective fault management. Besides that, distributed data management and configuration management for automated re-configuration in case of dedicated triggers are interesting fields of application.

Vision of a Policy-based Configuration Management System

The general goals of policy management are promising and can be illustrated as a simple means to control the network behavior. This has led to some commonly heard value propositions, where the most important for our purposes are defining the behavior of the network, simplifying network management, and requiring less personnel for network configuration [5]. We intend to take advantage of policies by capturing operating logic associated with certain conditions that occur in the network. By providing a management system driven by policies and rules, mobile network management is simplified through abstraction. This refers to configuration management (CM) primarily in our focus, but may be extended to other fields such as

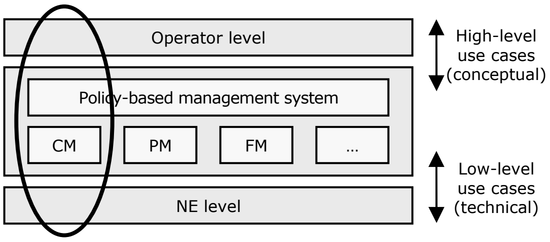
Operator level
Policy-based management system
CM
PM
FM
...
NE level
High-level use cases (conceptual)
Low-level use cases (technical)

Fig. 4. Functional view of a policy-based management system

performance management (PM) or fault management (FM) as illustrated in fig. 4. Consequently, operators can concentrate on high-level and conceptual tasks, rather than bothering with low-level use cases such as recovering from failures by hand. Like that, the dynamically changing environment can be controlled more easily and skilled personnel are relieved of simple technical tasks.

In the following, we explain our vision of an advanced system that meets operator's requirements for network-wide configuration management and realizes the idea of simplicity from the operator's point of view by the use of policy-related techniques. A concrete system implementing our vision is intended to accomplish and extend the transaction-based configuration mechanisms proposed in [1]. We first describe the features and requirements such a system should accomplish and then work out important components and explain the way how they interact and are put together in an appropriate system architecture.

Features and Functionality

The basic functionality of a policy-based system for network-wide configuration management is to specify abstract guidelines for the network configuration and to control the network configuration and behavior autonomously according to the guidelines specified. This is achieved by encapsulating operators' high-level knowledge within policies and policy rules. Like that, low-level configuration tasks are transferred from the operators into the system, which is enabled with self-configuring capabilities and can adapt itself to its environment. In other words, network configuration is lifted up to a higher level from the operator's point of view by providing an abstract interface. An appropriate toolbox shall support the operator in all phases of the policy-driven configuration management process. We divide the management process and the tasks enclosed into three phases: configuration development, configuration deployment, and configuration enforcement. According to these phases, we now explain crucial features such a system should provide.

It is the main goal of the configuration development phase to formalize the determining parameters and to specify the behavior of the network. This implies strong involvement of the network operator with the system, since it is necessary to formalize operator processes and knowledge. Concrete features in this phase of the management process are creation, modification and deletion of policies. Managing different versions of a policy or policy rule are a desirable feature as well. This feature is inspired from software engineering, where versioning systems such as Concurrent Versions System (CVS) [10] or Subversion [11] allow recording the history of source files and collaboration of several developers. Regarding mobile networks, this is a crucial feature as a huge number of operators are usually involved in network configuration processes and operators are likely to be located in different geographical areas.

In the deployment phase, policies are put into effect. For convenience, the system shall be able to distinguish between active and inactive policies and policy rules. Active ones operate on the network, whereas inactive ones represent a reservoir of templates, drafts, or similar. Before a policy can be activated on the network, a systematic consistency check has to be offered by the system verifying there is no contradiction with policies being already active. Imagine an active policy rule telling "commit the whole transaction group as soon as 90% of the individual transactions have been committed". Activation of a second one telling "do not commit transactions from 5am to 10pm" would cause a conflict in case transaction groups are committed in daytime. For such cases, rules of precedence may be a way out, specifying the exact sequence of execution in ambiguous cases. In contrast to activation, deactivation of a policy rule doest not require a consistency check as no conflict emerges if the overall number of constraints is reduced and there has been no inconsistency before. Furthermore, the system shall consider implicitly existing rules that arise when negating explicit ones. Besides detecting and dealing with conflicting policies, a smooth migration process has to be offered when updating an active policy with a new version, avoiding abrupt changes in the network behavior.

The configuration enforcement phase deals with the operating mobile network. It is the task of the system to autonomously monitor all components which have influence on the network state. As soon as the conditions of a policy rule are met, the respective actions shall be executed automatically. In addition, the system shall

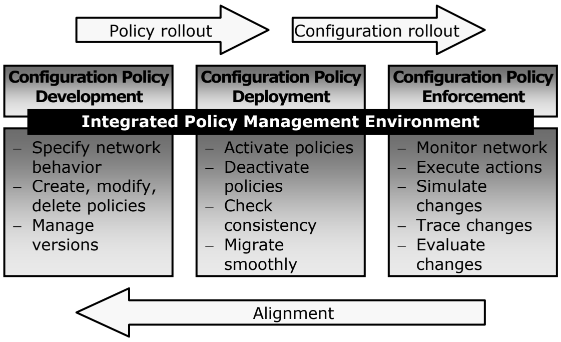

Fig. 5. The configuration policy management cycle

simplify the learning and optimizing process. Since there may be a large number of active policy rules, making it a challenging task for the operator to keep the system consistent, it is recommended to provide a simulation environment where modifications made can be tested prior to their activation. Furthermore, an evaluation environment that monitors the network state or at least dedicated indicators after the activation of new policies is desired to help detecting possibly strange behavior at an early stage.

As there are cyclic dependencies between the three phases described above, we also talk about a configuration policy management cycle. Before a configuration policy can be deployed, it first has to be developed. Deployed policies then need to be enforced at runtime. Experience gained in runtime execution again influences the configuration development process. An appropriate toolbox is ought to support the operator when performing the corresponding tasks. We call this toolbox Integrated Policy Management Environment and give more details in the next chapter. A summary of the management cycle phases, dependencies, corresponding features, and the toolbox is given in fig 5.

Components and System Architecture

Having pointed out the general idea and basic functionality of a policy-based system for network-wide configuration management in the previous chapter, we now point out structural details of such a system. We determine important components which realize the functionality described in the previous chapter and represent a reasonable partitioning of the overall system. We also address requirements coming along with the components. Furthermore, we explain the way how the components enclosed collaborate and are put together in an appropriate system architecture.

First of all, the adoption of a policy based-approach for realizing a configuration management system requires an appropriate policy representation. For specification of policies and policy rules, an appropriate policy language is required fulfilling some properties.

- Well-defined: The syntax and structure of the language must be clear and non-ambiguous and it must be able to express all procedures of the application domain. Here it is a crucial task to be compliant with the transaction mechanisms proposed in [1]. The meaning of a policy written in this language must be independent of its particular implementation.
- Flexible and extensible: It must allow new policy information to be expressed and allow new types of policies to be added in future versions of this language.
- Semantically-rich: The language shall be capable for semantic information models in order to describe the contents of the policies with semantic concepts. As far as different services or applications are described semantically as well, they are enabled to communicate with each other in an interoperable way according to the behavior stated in the respective policies.

In order to support the operator in development, deployment, and runtime management of policies and policy rules, an adequate toolbox is desired. For this purpose, we adapt the idea of Integrated Development Environments in software engineering and call such a toolbox an Integrated Policy Management Environment (IPME). Elements of such a toolbox are:

- Viewer and editor: This component shall support the development process of policies by offering appropriate visualization and editing mechanisms.
- Transformator: High-level policy definitions shall be transformed into concrete low-level policy rules in executable form by this component. Thus, the semantic gap between high-level policies and low-level rules shall be overcome.
- Analyzer: An analyzing component shall enable simulating and evaluating policy rules with respect to the extent they influence the network behavior.
- Graphical User Interface: Of course, a Graphical User Interface (GUI) easy to handle shall be offered, so the operator is able to carry out the tasks arising as comfortable as possible.

Once policies and rules have been defined, a storage mechanism is required to store policies and rules permanently. For this purpose, an appropriate policy repository holding active and inactive policies and rules is to be realized. For enforcement of policy rules, a policy engine is needed executing active policy rules automatically when necessary. This runtime task is to be performed by the following components:

- Monitor: A monitor component observes relevant network state information in order to determine if a set of conditions is met that requires corresponding actions to be undertaken. Network state changes may be reported to the monitor by events.
- Policy Enforcement Point (PEP): This component is responsible for executing actions defined within policies and rules as soon as the monitor has determined a matching set of conditions.
- Conflict handler: In case a conflict between active policy rules has not been detected in the deployment phase, a conflict handling component is needed within the policy engine as well performing appropriate detection and resolving mechanisms.

The policy engine and language together are also called policy framework. In order to monitor and execute configuration changes on the NEs, connections to the NEs are required, of course. An endpoint of such a connection is called Configuration Enforcement Point (CEP). Last but not least, the design of the execution engine must allow efficient functioning even under a high system load due to a huge number of active rules, NEs, and configuration changes.

Finally, the system should be able to interoperate with other components or systems that may exist in the administrative domain. The system architecture should be flexible enough to allow addition of new types of devices with minimal changes or updates necessary. For this purpose, well-defined interfaces are necessary which are independent of the particular component implementations in use. Interfaces between the components need to be clear and non-ambiguous. Fig. 6 shows the basic components of a policy management system and how they collaborate via designated interfaces.

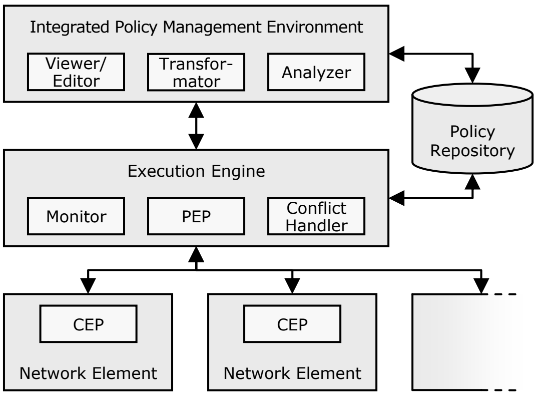

Fig. 6. The policy management architecture

Our Work Plan

In this chapter, we present detailed tasks which will bring us closer to realizing the vision of network-wide configuration management and to the policy-based system presented. The existing transaction-based configuration mechanisms represent a starting point for us. First of all, we will realize the low-level policy mechanisms already existent using a proper policy language and engine, which has not been the case so far. In order to do so, we will formalize the requirements for a concrete policy language arising from the present mechanisms. Some requirements already determined are e.g. the distinction between atomic policies and parameter controlled policies or timer controlled policies, as well as boolean combinations of those. Next, we will evaluate the cost-value ratio of using an existing policy framework that best fulfills the requirements compared to developing an individual framework. It has to be noted that the latter task may imply a considerable effort, but may also generate the most satisfying results. It might be a reasonable compromise to develop an individual Domain Specific Language (DSL) and realize a mapping to an existing policy framework. For this purpose, techniques from compiler construction and model-driven software development might be appropriate means. However, the policy framework and engine should not be limited to configuration mechanisms, but be extendible enough to address other fields in network management such as PM and FM at a later date, as mentioned before. We will then realize the present configuration mechanisms using the policy framework chosen. Extensions are to be made by introducing high-level concepts such as derived or iterative policies. This also represents a step from low-level rules to high-level policies. However, it is not yet clear where exactly the boundary between both levels will be. The answer to this question also has a high impact on the tranformator component that translates abstract policy definitions into concrete rule definitions, which then will be executed by the policy engine. Besides that, other components of the IPME such as the editor and analyzer component are to be developed after a proper policy framework has been established.

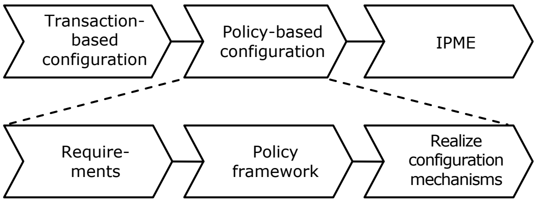
Transaction-based configuration
Policy-based configuration
IPME
Require-ments
Policy framework
Realize configuration mechanisms

Fig. 7. Work plan

Some open questions remain to be answered. Firstly, it has to be examined whether active policy rules should be held centralized within the policy engine or decentralized by transferring them into the network. Furthermore, it has to be evaluated to what extent ideas from distributed systems management will impact our work, based on the premise that management is itself a distributed activity. The answer to this question depends on the mobile network architecture and to what extent decentralized collaboration of the NEs is possible. Besides that, it is currently unclear to what extent standardization efforts play a role for our ideas. There have been some standardization efforts toward common policy information models and frameworks. The Internet Engineering Task Force (IETF) has been focusing on the specification of protocols and object-oriented models for representing policies and later released recommendations for the use of terms used in and related to policy-based management [12], [13],. However, it did not fit well with the existing focus of the IETF and work within the policy framework working group ended. The Policy RuleML Technical Group proposed the use of RuleML as a way to interoperate between different policy systems [14]. They investigate scenarios of usage of RuleML as an interchange vehicle for policy languages, and to develop standards for such interchange suitable for a useful set of such scenario areas. Additionally to the ones mentioned, other standardization efforts need to be investigated and evaluated.

Conclusion

In this position paper, we gave an introduction into network-wide configuration management for mobile networks. To reduce the amount of manual work necessary, the vision of autonomic computing seems appealing. We presented an approach for achieving configuration data consistency based on transaction mechanisms and an appropriate middleware concept. We also presented the vision of an advanced configuration management system and explained how we intend to adopt policy-based approaches to realize such a system. We pointed out important features and components of an appropriate management system and divided the network management process into three phases being policy development, policy deployment, and runtime policy management. The main idea is to encapsulate operators' high-level knowledge by the use of policies which enable the network with self-configuring capabilities. Furthermore, we proposed a policy management architecture showing the structural correlations between the components involved. By the use of an appropriate management system, less human effort will be needed for managing the overall network configuration. With our ideas, we intend to contribute to establishing ideas from policy-based management to mobile networks.

References

[1] H. Sanneck, C. Schmelz, C. Gerdes, C. Kleegrewe, J. Sokol, A. Southall, “Transaction-based Configuration Management for Mobile Networks,” 1st Annual Workshop on Distributed Autonomous Network Management Systems, Dublin, June 2006.

[2] International Organization for Standardization, ISO/IEC 10026-1:1992, Section 4, 1992.

[3] J. O. Kephart, D. M. Chess, “The vision of autonomic computing,” Computer, vol. 36 no. 1, January 2003, pp. 41-50.

[4] N. Damianou, N. Dulay, E. Lupu, M. Sloman, “The Ponder Policy Specification Language,” Proceedings of the International Workshop on Policies for Distributed Systems and Networks, Bristol, January 2001, pp. 18-31.

[5] J. C. Strassner, “Policy-based network management: solutions for the next generation,” Morgan Kaufmann, San Francisco, 2004, pp. 10-24.

[6] Object Management Group, “UML Superstructure Specification, v2.0,” August 2005.

[7] V. Casola, “A policy-based methodology for the analysis, modelling and designing of security infrastructures,” Ph.D. Thesis, University of Napels Federico II, Napels, November 2004, pp. 27-28.

[8] F. J. García, G. Martínez, J. A. Botía, A. F. Gómez Skarmeta, “Representing security policies in web information systems,” Proceedings of the 14th International WWW Conference, Chiba, May 2005, pp. 61-66.

[9] G. Tonti, J. M. Bradshaw, R. Jeffers, R. Montanari, N. Suri, A. Uszok, “Semantic web languages for policy representation and reasoning: a comparison of KAoS, Rei, and Ponder,” Proceedings of the 2nd International Semantic Web Conference, Sanibel Island, October 2003, pp 419-437.

[10] D. R. Price, “cvs – Concurrent Versions System,“ December 2006, viewed 2 April 2007, http://www.nongnu.org/cvs.

[11] CollabNet, “Subversion,” 2006, viewed 2 April 2007, http://subversion.tigris.org.

[12] IETF Network Working Group, “Policy Core Information Model – Version 1 Specification,” RFC 3060, February 2001.

[13] IETF Network Working Group, “Terminology for policy-based management,” RFC 3198, November 2001.

[14] H. Boley, M. Dean, B. Grosof, M. Kifer, S. Tabet, G. Wagner, “RuleML Position Statement,” W3C Workshop on Rule Languages for Interoperability, Washington, April 2005.