A SEMANTIC APPROACH TO POLICY-BASED MANAGEMENT
Linking Policies to Ontologies

S. Avallone, S. D'Antonio and S. P. Romano
Dip. Informatica e Sistemistica, University of Napoli Federico II, Via Claudio 21, Napoli, Italy

Keywords: Policy-based network management, ontologies, heterogeneous networks.

Abstract: The possibility of dynamically managing Quality of Service (QoS) in heterogeneous networks represents a key element for telecom operators which aim at making their communication infrastructures able to support new emerging multimedia services. The large success of triple play services together with the ongoing migration of voice carriers towards the NGN (Next Generation Network) architecture imposes a new way of controlling and utilizing network resources in order to fulfill user's requirements. The use of network level policies to configure and manage QoS-based communication networks allows network operators to automatically adapt the managed systems to changing requirements of the new operational and business scenarios. In this paper we present an innovative approach to policy-based network management which ensures flexibility and effectiveness to all processes composing the service life cycle, from negotiation to delivery. In order to allow users and applications to modify their QoS requirements, thus triggering the SLA re-negotiation, at the same time optimizing the configuration of network elements we implemented a novel framework capable to automatically translate user's requirements into network policies. Such framework relies on the adoption of ontologies as means to describe heterogeneous realms such as those associated with the fruition and delivery of innovative services. Ontologies are used to represent user's needs and preferences, which feed the policy creation and enforcement process. Different real world scenarios are depicted to validate the effectiveness of the proposed approach.

1 INTRODUCTION

In this paper we present an architecture for the dynamic creation and adaptation of network configuration policies. The architecture has been conceived in the framework of the European project NetQoS (Net,) and brings in a number of innovative solutions for the automated management of Next Generation Networks. The specific contribution of this work is twofold: on one hand we describe a hierarchical approach to the definition of high level QoS (*Quality of Service*) goals, by means of a semantic process based on the exploitation of ontologies; on the other hand, we discuss how such high level (i.e. user-oriented) goals can be progressively refined in order to arrive at the definition of related low-level (i.e. network-oriented) configuration policies.

The policy-based part of the overall architecture is based on the utilization of the XACML standard language, which, though originally conceived for role-based access control, can be fruitfully exploited in the wider context of network management.

The paper is structured as follows. Section 2 briefly presents the scope and objectives of the NetQoS project, which is a sixth framework programme funded initiative. Section 3 starts delving into the details of our contribution, by providing some highlights about the use of ontologies for the specification of user-level preferences/goals. On the grounds of this discussion, section 4 explains how to exploit XACML policies in order to appropriately configure QoS-enabled network elements. A crucial component of the overall architecture, the so-called Context Manager is thoroughly analyzed in section 5, which focuses on the most advanced autonomic aspects of the devised framework. The core of our contribution will be highlighted in section 6, which deals with the awkward task of automatically translating high-level ontologies into lower-level policies. Finally, section 7 presents an interesting case study related to the delivery of an advanced Video on Demand service. Section 8 concludes the paper and provides some information about future work directions.

2 THE NETQOS PROJECT

The NETQOS project is concerned with the design and development of a policy-based network management architecture. The main focus is on an autonomous management approach with the objective of providing enhanced quality of service (QoS) and resource utilization in varying operational environments. The proposed approach maps requirements and preferences from the actors of the system (users, service providers, applications, operators) into network and transport level policies, which are then enforced. In the course of time, changes in the operational environment emerge due to many reasons like the mobility of users, varying service demands, altered QoS requirements etc. Policy management automation allows for the system to autonomously adapt to these changes. Equally, a suboptimal QoS delivery and resource utilization triggers an adaptation of the employed policies. The NetQoS approach modifies existing policies, activates/deactivates existing policies and assembles new policies at run-time while respecting the requirements imposed by the different actors.

In order to fulfill the above mentioned objectives, it is necessary that the NetQoS components are capable to autonomously manage the network by both handling user requests and reacting to all possible events. To this end, all the components need to cooperate to carry out the task of the NetQoS system. Hence, the need arises for a component that coordinates all the others. In the NetQoS architecture (Fig. 1), such a coordinating component is denoted as the Context Manager (CM). The Context Manager is in charge to identify the "context", i.e., the operational status of the NetQoS system and all the relevant events. In case an event that requests actions from some components occurs, the Context Manager must become aware of it and notify the appropriate components. Examples of events that the Context Manager has to be aware of are the launch of an application, the violation of a policy, the addition of new policies, etc.

The Context Manager is primarily an event sink and source: it gets notified about important events and forwards this information to the relevant components. The detection of the events themselves is realized within the NetQoS monitoring and measurement tool platform (MoMe Tool). The MoMe Tool is the component that is responsible for providing information about the current state of the network. Clearly, this view of the network status is indispensable in the realization of an autonomous management approach.

The Policy Description (PD) represents a set of ontologies used to specify the actors' preferences, requirements, profiles, etc. The actors of the NetQoS system (users, applications, service providers, operators) use the Actor Preference Manager (APM) to specify their needs through ontologies.

This paper focuses on the automatic translation of those ontologies into policies. The Automated Policy Adaptor (APA) enforces the operational policies and updates them as needed when the system evolves.

3 ONTOLOGIES

Ontologies were originally conceived and introduced in the context of the Semantic Web. They provide formal specification of concepts and their interrelationships, and play an essential role in complex web service environments, semantics-based search engines and digital libraries. One important purpose of these formal specifications is sharing of knowledge between independent entities. Although there are yet many definitions for ontology, the most general and complete one looks at an ontology as "an explicit and formal specification of a shared conceptualization", which:

- is explicit because it defines the concepts, properties, relationships, functions, axioms and constraints that compose it;
- is formal because it is machine readable and interpreted;
- is a conceptualization because it is an abstract model and a simplified view of the existing things it represents;
- is shared because there has been previously a consensus about the information and it is accepted by a group of experts.

In brief, ontology aims at defining a set of concepts and properties, together with axioms providing the rules that govern them. In the last years, ontology languages have been in a rapid development. The OWL (Bechhofer et al., 2004), proposed by the W3C for the definition of ontology in the Semantic Web, is based on RDF and RDF Schema (RDFS) and provides greater machine interpretability of Web content

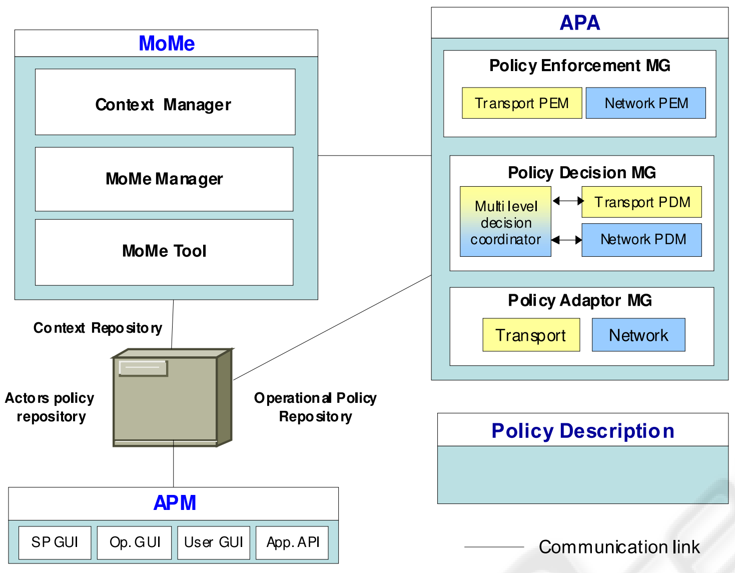

Figure 1: Architecture of the NetQoS system.

by offering additional vocabulary along with formal semantics.

3.1 Using of Ontology for Service Creation and Delivery

In the framework of service creation and provisioning, different entities interact by exchanging and sharing information concerning different realms and domains. The heterogeneity of both the communication networks and the user's devices currently used to deliver innovative services represents a further element which prompts the adoption of an ontology-based approach to service definition and negotiation. User's requirements and preferences can be optimally described by means of ontologies, which, in turn, can be effectively and automatically translated in network configuration and management policies. Indeed, policy based networks provide high levels of flexibility by allowing definition of packet handling rules, resource allocation strategies, network management techniques, access control restrictions, levels of service, etc. All these policies are then enforced by configuring the network devices with the appropriate primitives so that the desired actions are performed on the data streams.

One of the main challenges frequently faced is ensuring that network configuration settings and policies are applied consistently throughout the network. However, this task is often error-prone and difficult to manage especially when there is heterogeneity of network devices and protocols. In order to effectively define and enforce policies such as “allow traffic from host A to take priority over traffic from host B” and “permit user A to receive video at 128 Kbps” onto heterogeneous networks, the need arises to develop a framework capable to increase the *expressibility* of policies thanks to the use of ontologies as means to describe concepts, their taxonomy, interrelations and rules.

In the following of this paper we present a framework which allows users to specify their service requirements and preferences by exploiting predefined ontologies and automatically translates such ontologies into network level policies to be used for QoS management. To develop ontologies we used Protégé (Pro,), a graphical tool for ontology editing and knowledge acquisition, which allows thinking about domain models at a conceptual level without having to know the syntax of the language ultimately used on the Web. Protégé provides full-fledged support for knowledge modeling and acquisition: classes, instances of these classes, slots representing attributes of classes and instances, as well as facets expressing additional information about slots. Protégé enables developers to instantiate ontology that defines the abstract data on which specific operations are performed with corresponding domain concepts. It uses the instantiated ontology to generate a domain-specific knowledge-acquisition system. Application experts can then exploit such system to enter the specific content knowledge on which the method operates to solve particular application tasks.

4 POLICIES IN XACML

Extensible Access Control Markup Language (XACML) (XAC,) is an XML-based language designed specifically for creating policies and automating their use. XACML has two basic components. The first is an access-control policy language that lets developers specify the rules about who can do what and when. The other is a request/response language that presents requests for access and describes the answers to those queries. XACML provides for fine-grained access control of resources based on several criteria, including the following:

- attributes of the user requesting access (e.g., "Only division managers and above can view this document.");
- the protocol over which the request is made (e.g., "This data can be viewed only if it is accessed over HTTPS.");
- the authentication mechanism (e.g., "The requester must have been authenticated using a digital device.").

The key top-level element of the policy definition language is the PolicySet which aggregates other PolicySet elements or Policy elements. The Policy element is composed principally of Target, Rule and Obligation elements and is evaluated at the Policy Decision Point to yield resource access decision. The Target element is used to associate a requested resource to an applicable Policy. It contains conditions that the requesting Subject, Resource, or Action must meet for a Policy Set, Policy, or Rule to be applicable to the resource. Rules provide the conditions which test the relevant attributes within a Policy. Any number of Rule elements may be used, each generating a true or false outcome. Combining these outcomes yields a single decision for the Policy, which may be 'Permit", "Deny", "Indeterminate", or a "NotApplicable" decision. Attributes provide the typed values that represent both a resource requester and the Policy's condition predicates. Finally, obligations are a set of operations that must be performed by the Policy Enforcement Point in conjunction with an authorization decision. An obligation may be associated with a positive or negative authorization decision. In our case, obligations play a major role, since they trigger (and appropriately configure) the dynamic instantiation/updating of NetQoS policies, as it will become apparent in the next section.

5 THE ROLE OF THE NETQOS CONTEXT MANAGER

The Context Manager component is a central element of the NetQoS system, which interacts both with the other components of the Monitoring and Measurement (MoMe) framework - specifically the MoMe Tool and the MoMe Manager - and with the two main repositories, namely the Ontology Repository and the Policy Repository. This interaction is needed in order to make the NetQoS system capable to dynamically react upon the detection of specific events related to changes in the overall execution context. Under this perspective, we can state that the CM helps the system become autonomic, since it provides it with the following properties:

1. Automatic: this essentially means being able to self-control its internal functions and operations. An autonomic system must in fact be self-contained and able to operate without any external intervention.
2. Adaptive: an autonomic system must be able to change its operation. This allows the system to cope with temporal and spatial changes in its operational context.
3. Aware: an autonomic system must be able to monitor its operational context as well as its internal state in order to be able to assess if its current operation serves its purpose. Awareness controls adaptation of its operational behavior in response to situation or state changes.

Based on the considerations above, we designed the CM on the basis of a "Publish/Subscribe" paradigm: such element "produces" information to be used by the companion MoMe components. As an example, when a user launches an application, the CM has to notify the MoMe Manager of the occurrence of such event. At the same time, the CM also "consumes" information produced both by the repositories and by so-called "NetQoS-aware applications" (or, in the presence of NetQoS-unaware applications, by an ad hoc designed Traffic Identification Engine which is also part of the NetQoS system). We herein focus on the former scenario, which helps clarify the overall functionality of the CM, at the same time justifying the design choice related to the adoption of an asynchronous notification paradigm. Hence, with respect to the interaction with repositories, let us consider the case when new information is inserted into either the Ontology or the Policy repository: in such case, the CM has to get a notification if other components requested it to be kept updated with respect to

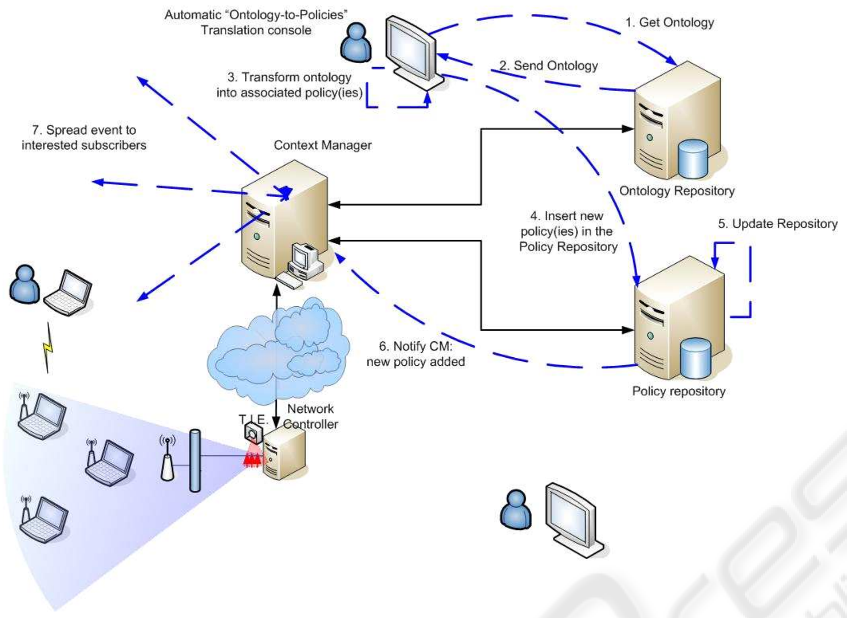
Automatic "Ontology-to-Policies" Translation console
1. Get Ontology
2. Send Ontology
3. Transform ontology into associated policy(ies)
7. Spread event to interested subscribers
Context Manager
Ontology Repository
4. Insert new policy(ies) in the Policy Repository
5. Update Repository
6. Notify CM: new policy added
Policy repository
T.I.E.
Network Controller

Figure 2: Automatic production of policies starting from an ontology.

the addition/deletion of policies/ontologies. The mentioned "subscriber components" will in turn be notified by the CM of the occurrence of the cited event. We have identified three different scenarios associated with CM's interaction with repositories:

1. Updating of the Ontology Repository;
2. Updating of the Policy Repository;
3. Automatic translation of ontologies into policies, which requires dynamic updating of the Policy Repository.

The three scenarios above have all been implemented within the project. For the sake of conciseness, we will herein just focus on the third one, which is strictly related to the subject of this paper.

6 AUTOMATIC TRANSLATION OF ONTOLOGIES INTO POLICIES

This case is by far the most complicated one, since it envisages the orchestrated operation of both repositories. Indeed, NetQoS foresees a possibility to dynamically create policies starting from higher-level ontology-based descriptions. This work is a fundamental part of the dynamic adaptation functionality and clearly requires that advanced "inference" techniques be applied. This stated, we have to account for the scenario depicted in Fig. 2.

The figure shows a human operator sitting in front of an automatic "ontology-to-policy" translation console, used to first retrieve (steps 1 and 2 in the picture) and then define constraints on the existing ontologies. Such constraints will then be transparently translated (step 3 in the picture) into one or more policies, which will in turn be stored in the policy repository (step 4 in the picture). From this point on, we have the following steps: (i)repository updating - step 5; (ii) CM notification - step 6; (iii) event spreading - step 7.

The way the CM gets the triggering notification depends on whether or not the repository is capable to directly raise events towards the external world (as indicated in Fig. 2). We show in Fig. 3 the alternative situation, dealing with the case when the trigger to the CM comes from the console rather than the policy repository.

The CM has been realized as a stateful web service with notification capabilities. In other words, it is compliant with an extension of web services, defined and standardized by the OASIS organization under the name of Web Services Resource Framework (WSRF), which gives a web service the ability to keep state information. The service and the resource are nonetheless kept completely separate: the state is not "in" the web service, but rather in a separate entity called "resource" and each resource is assigned a unique 'key', used by the web service whenever a stateful interaction is needed. Support for notification capabilities is made possible by implementing the "Observer/Observable" design pattern, as specified in the WS-Notifications family of specifications

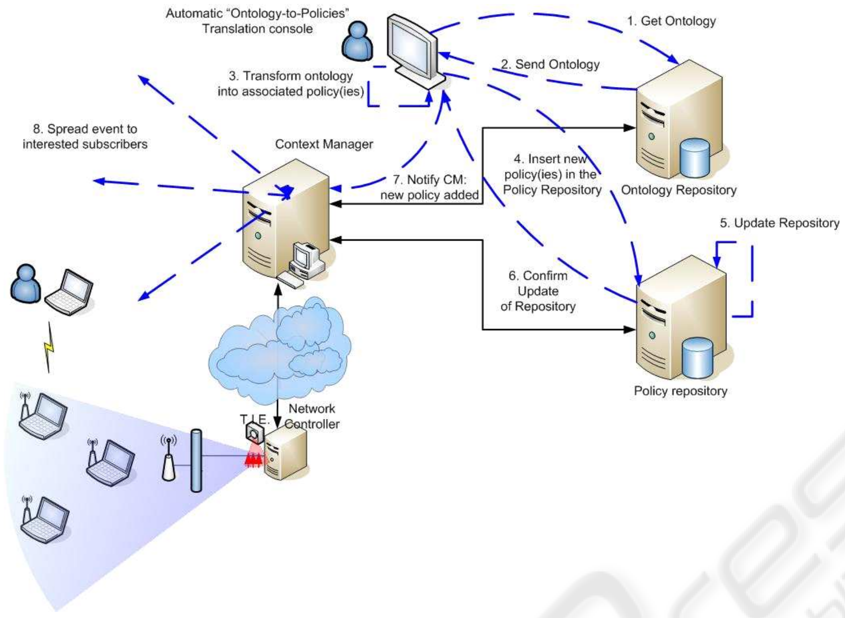
Automatic "Ontology-to-Policies" Translation console
1. Get Ontology
2. Send Ontology
3. Transform ontology into associated policy(ies)
8. Spread event to interested subscribers
Context Manager
4. Insert new policy(ies) in the Policy Repository
Ontology Repository
7. Notify CM: new policy added
5. Update Repository
6. Confirm Update of Repository
Policy repository
Network Controller
T.I.E.

Figure 3: Automatic translation with notification from the console.

also defined and standardized by the OASIS organization. WS-Notification is based on the concept of a "topic", i.e. an "item of interest for subscription". Notification producers expose a "subscribe" operation that notification consumers can use to request a subscription. Notification consumers, in turn, expose a "notify" operation that notification producers can use to deliver the notification.

7 A CASE STUDY: VIDEO ON DEMAND SERVICE

In this section we describe how an ontology-based approach can be adopted to specify user's requirements in the context of a Video on Demand service. Let us consider a generic user who wishes to receive video contents at different quality levels depending on the equipment he is using. For instance, he prefers to receive a movie at the highest resolution in case the used equipment is a multimedia workstation equipped with a wide-screen LCD monitor, while he requires a low-level quality version of the same content if the user terminal is a PDA or a UMTS phone. A proper ontology can be created to allow the user to specify his requirements and preferences. With reference to the Video on Demand service, a possible ontology can comprise the following classes:

- User, which includes information and attributes related to the VoD user,
- Terminal, which describes the user's device used to deliver the service,
- Service, which is intended to specify the service parameters and elements.

During the service negotiation phase, the user is required to specify his preferences by providing values to be associated with the attributes of the predefined ontology.

Such negotiation will automatically bring to the production of the XACML policies associated with the ontology itself. In the VoD example above, the produced XACML policies would contain, for example, the following elements with the associated values:

- Subject: such field is filled in with information about the identity of the user for which the policies are being generated;
- Resource: this field represents the specific category of movies (e.g. thriller, action, fantasy, etc) for which the user wants to specify his own preferences;
- Condition: it might represent the terminal type (as well as terminal capabilities) used to access the resource.
- Obligation: as already anticipated, such field is of paramount importance, since it contains all policy directives to be associated with the specified resource in order to appropriately configure the network infrastructure used to deliver the service, in a specific execution context and with the specified features. In the VoD example, the Obligation

field might for instance indicate that a satisfactory service fruition requires that a high (where high might be associated, e.g., with the *gold* class of service in a network capable to provide for service differentiation bases on the so-called *olympic model*) quality level is guaranteed.

8 CONCLUSIONS AND FUTURE WORK

In this paper we presented an approach to the automatic creation of network configuration policies, starting from higher-level ontologies associated with a semantic view of a service. We described how the NetQoS architecture, which has been conceived at the outset with this goal in mind, is capable to dynamically create and adapt policies based both on context and on higher-level information related to user's preferences/profiles.

The ideas we presented have so far been only partially implemented, since the NetQoS project is now approaching its first year of work. The work is currently in full swing and we are confident that within the next few months further steps will be done towards the completion of the overall architecture we presented.

ACKNOWLEDGEMENTS

This work has been carried out with the financial support of the European project NetQoS, which is partially funded by the EU as part of the IST Programme, within the Sixth Framework Programme.

REFERENCES

NetQoS, Policy Based Management of Heterogeneous Networks for Guaranteed QoS. http://www.netqos.edu.

OASIS eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/xacml/charter.php.

The Protégé Ontology Editor and Knowledge Acquisition System. http://protege.stanford.edu/.

Bechhofer, S. et al. (2004). OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/.