SEMANTIC WEB-BASED MANAGEMENT OF ROUTING CONFIGURATIONS

Félix J. García Clemente

Departamento de Ingeniería y Tecnología de Computadores

University of Murcia, Spain

fgarcia@um.es

Jose M. Alcaraz Calero

Jorge Bernal Bernabé

Juan Manuel Marín Pérez

Gregorio Martínez Pérez[1]

Antonio F. Gómez Skarmeta

Departamento de Ingeniería de la Información y las Comunicaciones

University of Murcia, Spain

{jmalcaraz, jorgebernal, juanmanuel, gregorio, skarmeta}@um.es

Abstract—**Today, network operators typically reason about network behaviour by observing the effects of a particular configuration in operation. This configuration process typically involves logging configuration changes and rolling back to a previous version when a problem arises. Advanced network operators (more each day) use policy-based routing languages to define the routing configuration and tools based on systematic verification techniques to ensure that operational behaviour is consistent with the intended behaviour. These tools help operators to reason about properties of routing protocols. However, these languages and tools work in low-level, i.e. they focus on properties, parameters, and elements of routing protocols. However, network operators receive high-level policies that must be refined to low level parameters before they can be applied. These high-level policies should consider other properties (e.g. extensibility or reasoning capabilities), parameters (e.g. time period, localization or QoS parameters), and elements (e.g. AAA individuals or resources), when the network configuration is defined. We believe that there is a need of broader approaches in languages and tools for defining routing configurations that are more powerful and integrated to other network elements. This article provides the main ideas behind the specification of routing policies using formal languages which enable the description of semantics.**

[1] Corresponding author; telephone: +34 868 887646; Fax: +34 868 884151

These semantics make easier the policy refinement process and allows describing an automated process for doing conflict detection on these policies.

Keywords- Routing policy, Network Management, BGP, Policy Languages, Conflict Detection, Semantic Web

I. INTRODUCTION

The Internet is composed of a large number of Autonomous Systems (AS) that are independently administered networks by one or more network operators (or network administrators), coupled by using an inter-domain routing infrastructure using the Border Gateway Protocol (BGP) [1]. The exchange of routing information between ASes is controlled by diverse policies, decided locally be each AS, and not directly observable from available BGP data.

The network operator can typically specify the routing policy enforcing BGP data into configuration files directly applied to the managed routers. An advanced network operator can use a policy language such Routing Policy Specification Language (RPSL) [2] to describe their routing policies and store them at various Internet Routing Registry (IRR) databases including RIPE, RADB and APNIC [3]. The network operator can use tools such as IRRToolSet [4] for automated router configuration, routing policy analysis, and on-going maintenance.

However, RPSL works with low-level policies, i.e. RPSL focus on properties, parameters, and elements of routing protocols. Network operators should consider other high-level requirements, when they specify the routing policy. These requirements of policies include maintaining business relationships between different ASes, ensuring resilience requirements even under overload, guaranteeing efficient internal operations and supporting growing customer demands to control their traffic flows. In most cases, network administrator receives a high-level policy that must be refined before it can be applied to the managed routers. There are many cases in which the network operator cannot refine the high-level policy received. One of the reasons is the lack of languages to support and extend high-level concepts. For example, a routing policy that could appear with a simple textual form such as “if the managed AS is multihomed, the internal network of the client A is not connected, and today is Sunday, then removing the peering link with the AS of the Provider A”. This policy can be defined by a customer AS who wants to reduce the bill that she pays to a provider AS for any traffic sent between the two networks. However, this policy implies a complex refinement task for the administrator. Moreover, the administrator is limited to the parameters supported by configuration files that could make impossible to deploy that policy. This policy refinement problem is especially relevant when the high-level policies are related to the description of conflictive situations. For example, the policy “we always should use the chipset connection” can raise a conflict when there are two connections and for any external problem it is being used the non-chipset connection. Usually, there are no parameters in the configuration files to manage this kind of circumstances (and then resolve this conflict, for example).

The routing policy language may include support for other properties (e.g. extensibility or reasoning capabilities), parameters (e.g. time period, localization or QoS parameters), and elements (e.g. AAA individuals or resources) that permit to define high-level routing policy. Moreover, the language may be extensible in order to enable the definition of new concepts and their semantics. The language may enable the definition of methods to detect and resolve conflicts. The adoption of languages that enable the representation of semantic for policy representation, i.e., the use of ontologies and Semantic Web [5] languages to specify them can provide a solution to these problems. For this reason, these languages have been used in this paper. Other existing routing policy languages cannot describe semantics and methods to detect and resolve conflict in the system, which it is an added value of the proposal being presented in this paper.

The main contribution of this article is the definition of a framework based on the Semantic Web languages to specify high-level routing policies. The routing information is defined in Ontology Web Language (OWL) using the BGP concepts available in the Common Information Model (CIM). As a result, Semantic Web Rule Language (SWRL) is used to define high-level routing policies whereas SWRL is also used to detected and resolve conflicts. The automatic reasoning capabilities available in these technologies and the wide set of friendly tools to define policies ease the configuration job of network operators. Although each application domain has different concepts and semantics, previous works have validated our information model for other application domains such as security services [6] and distributed firewall [7].

The rest of this paper is structured as follows. Section II provides the background on BGP concepts and further motivation and requirements of the designed policy language. Some related works are presented in the Section III. Section IV provides a semantics-aware policy representation where different kind of routing policies can be expressed. Section V describes the automated reasoning and the conflict detection in such specifications. Some indications about the deployment of semantic-aware routing policies related to this research work are described in section VI. Section VII discusses the performance and scalability of the proposal. Finally, section VIII outlines the main conclusions and some ideas behind the ongoing work on conflict resolution.

II. BACKGROUND AND MOTIVATION

A. BGP Autonomous System Types and Routing Policies

In many cases, a BGP speaker is connected to more than one different speaker. This provides both greater efficiency in the form of more direct paths to different networks, and also redundancy to allow the internetwork to cope with either device or connection failures. It is possible for a BGP speaker to have neighbour relationships with other BGP speakers both within its own AS and outside it. A neighbour within the AS is called an internal peer, while a neighbour outside the AS is an external peer. BGP between internal peers is sometimes called Internal BGP (iBGP) while use of the protocol between external peers is External BGP (eBGP). You can see an example of BGP topology and the designation of internal and external peers in Fig. 1.

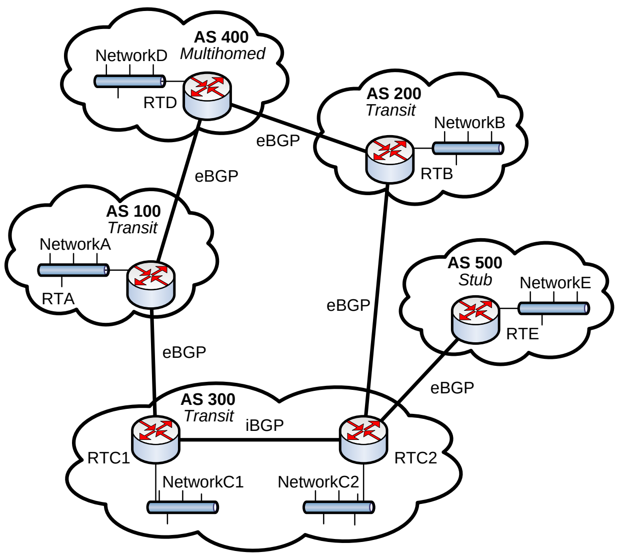

Figure 1. BGP Example Topology and Designations

We can make a similar distinction between different types of ASes, based on how they are interconnected in the overall BGP topology. There are three main types of AS: *Stub AS*, which is an AS that is connected to only one other AS (AS 500 in our example of Fig. 1); *Multihomed Non-Transit AS*, which is an AS that is connected to two or more other ASes and transports only traffic originating or terminating on an host (or server) within the network (AS 400 in our example of Fig. 1); and *Multihomed Transit AS*, which is an AS that provides connections through itself to other networks (AS 100, AS 200, and AS 300 in our example of Fig. 1).

To provide control over the carrying of transit traffic, BGP allows an AS to set up and use routing policies. These are sets of rules that govern how an AS will handle transit traffic. Some of the many options of an AS to handle transit traffic include: *No Transit Policy*: an AS can have a policy that it will not handle transit traffic at all. *Restricted AS Transit Policy*: an AS may allow handling of traffic from certain ASes but not others. In this case, it tells the ASes it will handle that they may send it traffic, but does not say this to the others. And *Criteria-Based Transit Policy*: an AS may use a number of different criteria to decide whether to allow transit traffic. For example, it might allow transit traffic only during certain times, or only when it has enough spare capacity.

In a similar manner, policies can also be set to control how an AS will manage its own traffic received by other autonomous systems. These policies may be based on security considerations (if one connecting AS is deemed more secure than another), performance (one AS is faster than another), reliability or other factors.

For example, Fig. 1 shows an AS 100 defined as a transit AS with criteria based on transit policy where the network operator establishes routing policies that specify which conditions the AS is willing to handle transit traffic. On the other hand, the AS 400 is a multihomed AS with no transit policy. However, the network operator establishes routing policies to determine route selection by specifying the conditions under which either AS100 or AS200 should be used. This example is used as running example on the rest of this paper.

B. Requirements for High-Level Routing Policy Language

One of the tasks in network administration is to transform the business policies into implementable routing policies using a formal representation. To do so, the administrator should use a policy language that assures that the representation of these policies will be unambiguous and verifiable. Other important requirements of any policy language are [8]:

- *Clear and well defined semantics.* A semantics that include all policy concepts necessary to facilitate the task of the network operator.
- *Clear and well defined syntax*. A syntax that can be easily used to share policy information between different kinds of computers, different applications, and different organizations.
- *Flexibility and extensibility.* A policy language has to be flexible enough to allow new policy information to be expressed, and extensible enough to allow new semantics to be added in future versions of this language.

In particular, for a routing policy language, the refinement process between the business policy and the routing configuration requires the system description, i.e. network topology, network state, functionality of each resource, etc. Other important requirements of any routing policy language are:

- *Languages for system description and policy description.* The separation of system description and policy description permits us to analyze both specifications individually using different techniques.
- *Reasoning capabilities about system descriptions.* These capabilities permit to query and access to policy information.
- *Reasoning capabilities about policy description.* It permits rule-based reasoning that can be used to define techniques for conflict analysis, policy scheduling, consistence checking, and etcetera.

A Semantic Web approach for system and policy representation can provide an adequate solution to these requirements since it provides high expressive languages to define concepts, semantics and policies. These languages are formal languages with a well-known semantic, so called Description Logics [9], which provide reasoning capabilities about both system description and policy definitions. Moreover, the added value of enabling the definition of semantics is very useful to define both routing policies and policies to detect and resolve conflictive circumstances in the system.

III. RELATED WORK

There has been significant work in understanding and automating BGP network management. However, most of these approaches have been at the individual device level in terms of creating the right configuration files, and do not consider the high level network objectives. In these sense, several network operators have developed tools to automate the configuration of parts of their routing policies. Among others, [10] describes tools to build the configurations of the eBGP sessions with customers in a large ISP network based on provisioning databases. No information is provided in [10] about the support of complex routing policies. Bohm et al [11] propose an XML-based configuration language that enables the expression of the network-wide routing policies in an ISP network. This XML configuration is then converted in RPSL format and the RPSL tools are used to generate the corresponding router configurations.

In an effort to better understand the dynamics of configuration management, Chen et al [12] use a combination of TACAS logs, static configuration files and the router configuration of a Tier-1 ISP to build a Deterministic Finite Automaton (DFA) representation of network configuration. Feamster et al [13] use static analysis to detect configuration faults in BGP. Particularly, the authors derive configuration constraints from high level policy specifications and check BGP configurations against the derived constraints. Using this approach, they detect path visibility and route validity configuration faults. However, they do not deal with dynamic configuration changes. The Routing Policy Specification Language (RPSL) [2] is an object oriented language for specifying routing policies from which router configurations can be automatically generated.

RPSL was initially proposed as a language to both document and verify that there are no conflicts among routing policies from different ASes. Some ASes rely on RPSL to document their peering relationships and some use the RPSL information to automatically generate part of their filters. However, it does not seem that RPSL is still used to verify the coherence of the routing policies. This is mainly because some operators do not provide their policies in RPSL format. Furthermore, RPSL does not allow operators to easily express complex policies. Moreover, RPSL does not support inference and is limited in expressiveness.

The use of ontologies and Semantic Web in automating network management is new and so there are few proposals in this direction. Kodeswaran et al [14] proposes an alternate mechanism for policy based networking that relies on using additional semantic information associated with routes expressed as OWL ontology. Policies are expressed using SWRL to provide fine-grained control where the routers can reason over their routes and determine how they need to be exchanged, but the policy conflicts are not discussed.

Research in policy conflict analysis is being active for several years. However, most of the work in this area addresses general management policies rather than policies in use in routing. For example, Lupu et al [15] classify possible policy conflicts in role-based management frameworks, and develop techniques to discover them. Some other works, although interesting as background are not directly applicable to the routing scenario; this is the case of Yagüe et al [16] which provides a semantic-aware access control model for web services, or the work of Al-Shaer et al [17] using a tree-based model with advanced techniques for conflicts and anomaly analysis for firewalls.

IV. Semantic Web-based Routing Policy Representation

Our proposal defines the System Description Language (SDL), which consists of an ontology based on Common Information Model (CIM) [18] to describe the system to be managed, and the Security Policy Language (SPL), which uses the concepts defined by SDL to create policy rules to express the desired behaviour for the administered network. Note that SDL and SPL represent a new way to express routing policies combining Semantic Web languages and CIM. The SDL language is defined as the usage of the Ontology Web Language (OWL) [19] in order to describe the system to be managed using the OWL representation of the CIM ontology. This fact enables the definition of semantics related to the system description and the extension of such description. As a result, a high level definition of the system to be managed is achieved in a standard format and all the advantages related to the formal methods associated to OWL such as the reasoning processes can be applied. The SPL language is defined as the usage of the Semantic Web Rule Language (SWRL) [20] to define policies using the concepts defined in the previous SDL language.

OWL enables the definition of semantics like inheritance among concepts, transitiveness among properties, existential constraints, cardinality constraints, and etcetera. SWRL is used to represent rules on the Semantic Web and it extends OWL in order to provide a way to express conditional knowledge. The language itself is not decidable, but a syntactic restriction called DL-Safe context [21] can be applied in order to restore the decidability. The combination of SWRL and OWL can be used to express not only the routing policy, but also more deductive processes on the system which may be relevant for routing decisions. For example, this language enables the definition of policies to detect and resolve conflict in the system. In case the reader is more interested in the syntax and semantics of these languages, OWL [19] and SWRL [20] provide a detailed description.

The following subsections are focused on the routing policy representation using SDL and SPL, and show how different kinds of routing policies can be expressed.

A. Ontological representation for system description

Our representation is based on a well-recognized standard information model such as CIM, thus enabling the management of policy information in a uniform manner and guaranteeing the necessary extensibility to support any kind of routing policy. Moreover, the SDL representation uses the ontological language OWL. Our motivation for using OWL, besides being a W3C standard, is mainly its capabilities for expressing formal semantics, defining class hierarchies and their relationships, associated properties, cardinality restrictions while still retaining decidability and computational completeness. Using OWL for ontology specification makes the framework generic, flexible and more scalable than using proprietary labelling schemes that raise interoperability issues.

Fig. 2 shows the UML representation for the CIM classes related to BGP routing concepts, which we have used in this paper as working example.

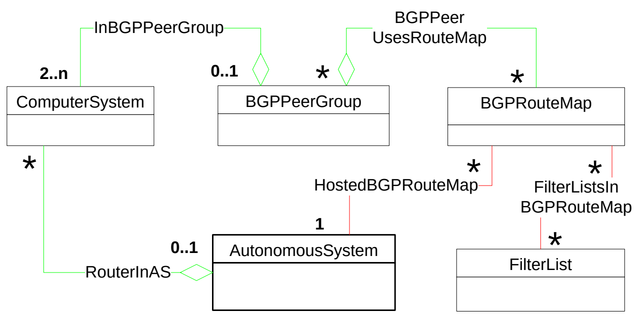

Figure 2. CIM BGP Routing Model

AutonomousSystem is the base class in networking model. The aggregation *RouterInAS* establishes relationships between an *AutonomousSystem* and routers that it contains. *BGPPeerGroup* is a set of BGP speakers (two or more) that share the same set of restrictions that must be followed in order to work correctly with BGP peers. A *BGPRouteMap* is used to control and modify routing information as well as to define when a route is redistributed between autonomous systems. *RouteMaps* may use *FilterLists* to identify the route, and a *BGPRouteMap* is specific to a given *AutonomousSystem* that contains it.

Our approach uses CIM ontology proposed by [22] to represent the routing concepts that are depicted in Fig. 2. The reason for this choice is twofold. On one hand, it holds many semantics description in the translation to OWL in contrast to the approaches which only preserves the expressiveness associated to the Resource Description Framework (RDF) Language [22]. The second reason is related to the way in which the relationships between CIM elements have been mapped. In this sense, some approaches propose the mapping of CIM relationships to OWL classes [23], whereas our choice defends to map CIM relationships to OWL properties. This approach confers simplicity to CIM models and, in turn, it allows to model semantic features applied to these properties. The text available in Table 1 shows an extract of the description in OWL of the administrative domain of the Fig. 1 using RDF/XML syntax.

```
<AutonomousSystem rdf:about="#AS100">
  <ASNumber>100</ASNumber>
  <IsSingleHomed>false</IsSingleHomed>
  <IsTransit>true</IsTransit>
</AutonomousSystem>
```

```
<AutonomousSystem rdf:about="#AS400">
  <ASNumber>400</ASNumber>
  <IsSingleHomed>false</IsSingleHomed>
  <IsTransit>false</IsTransit>
</AutonomousSystem>
```

```
<ComputerSystem rdf:about="#RTA">
  <Name>Router A</Name>
  <Dedicated>router </ Dedicated>
  <RouterInAS rdf:resource="#AS100"/>
  <InBGPPeerGroup rdf:resource="#AS100_AS400"/>
</ComputerSystem>
```

```
<ComputerSystem rdf:about="#RTD">
  <Name>Router D</Name>
  <Dedicated>router </ Dedicated>
  <RouterInAS rdf:resource="#AS400"/>
  <InBGPPeerGroup rdf:resource="#AS100_AS400"/>
</ComputerSystem>
```

```
<BGPPeerGroup rdf:about="#AS100_AS400"/>
  <BGPRouteMap rdf:resource="#RouteMap100"/>
</BGPPeerGroup >
```

```
<BGPRouteMap rdf:about="#RouteMap100">
  <Direction>Both</Direction>
  <Action>Permit</Action>
  <HostedBGPRouteMap rdf:resource="#AS100"/>
</BGPRouteMap>
```

Table 1. Representation of two AS and their BGP link

The text available in Table 1 shows the representation of AS100, AS400, and their BGP peering link. Each autonomous system is represented by the *AutonomousSystem* concept. This concept has some high-level attributes like *ASNumber*, *IsMultihomed*, *and IsTransist*. Both AS100 and AS400 are in the same *BGPPeerGroup* and each of them has a *ComputerSystem* (router), RTA and RTD, respectively. The *BGPRouteMap* associated to AS100 permits the transit traffic in both directions. The rest of the system description is not present because this syntax is very verbose. In any case, this extract is enough to show that the syntax and semantics are clear.

B. *Routing Policies*

The SPL language is represented by SWRL rules. These rules are defined in high-level abstract syntax. SWRL enable to define policies using the SDL ontology mentioned before. SWRL rules are in the form of an implication between antecedents (body) and consequents (head). The intended meaning can be read as: whenever the conditions specified in the antecedents hold, then the conditions specified in the consequents must also hold.

For example, the network administrator may decide this basic routing policy using OWL extract above described: “IF AS400 contains a router, THEN this router has a BGP peer with AS100”. The rule available in Table 2 shows the formal representation of this policy that can be directly mapped to SWRL.

ComputerSystem (?CS) ^ Dedicated (?CS,’router’) ^
RouterInAS (?CS,#AS400)
→
InBGPPeerGroup (?CS,#AS100_AS400)

Table 2. Example of SWRL Routing Policy

The rule body establishes a variable *CS* to specify the computer system. The property *Dedicated* identifies the computer system as a router and the property *RouterInAS* sets that the AS400 contains the router. Then the rule head establishes the association between the router and the BGP peer group that exists between AS400 and AS100.

The network administrator can also decide more complex routing policies that include other parameters not related to routing, i.e. auxiliary measures such as time period, localization or QoS parameters. For example, the network administrator may decide the following routing policy: “IF AS has a peering link with a router located in *ProviderA* and today is Sunday; THEN AS has a peering link with AS100”. The following text shows the logic representation of this policy that can be mapped to SWRL.

AutonomousSystem (?AS) ^
ComputerSystem (?CS) ^ Dedicated (?CS,’router’) ^
RouterInAS (?CS,?AS) ^ LocatedIn (?CS, ‘ProviderA’) ^
DayOfWeek (#Today, ‘Sunday’)

→
InBGPPeerGroup (?CS,#AS100_ALLASES)

Table 3. Example of SWRL Routing Policy

The rule body establishes the localization with the property *LocatedIn*, and the time period with the property *DayOfWeek*. Both properties belong to CIM ontology and they permit the definition of routing policies that consider no BGP parameters. Moreover, the network operator could include elements not related to routing, i.e. identities such as AAA individuals or resources. For example, the network administrator may decide the following routing policy: "IF AS has a number of authenticated individuals greater than 1000, THEN AS has a peering link with AS100".

CIM ontology does not include all concepts that a network operator could need. However, CIM ontology is easy to combine with other ontologies, e.g. SOUPA ontology [24] for concepts related to ubiquitous and pervasive applications, or also SWRC ontology [25] for modelling entities of research communities. It permits a system description as wide as the network operator need it to define high-level routing policies. However, it is worth mentioning that each router managed in the system has to be able to interpret the information provided by the policies.

V. Reasoning Capabilities and Conflict Detection

A. Ontology reasoning and rule-based reasoning

The combination of the OWL-encoded CIM ontology and SWRL to specify behaviour rules for routing policies offers a clear advantage: it allows two different types of automated reasoning. The first one is ontology reasoning (i.e., reasoning over the structure and instances of the ontology) and the second one is rule-based reasoning (i.e., reasoning over the policy rules in system management tasks).

Therefore, we identify an OWL reasoner and a rule based reasoner (see Fig. 3), where we use the word reasoner to refer to a specific program that performs the inference tasks, i.e., the process of deriving additional information that is not explicitly specified) and validation tasks, i.e. the process of ensuring that the system description and the policies are coherent and there is not inconsistencies therein.

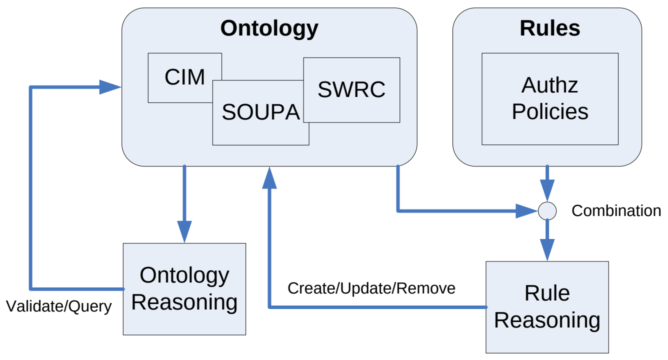

Figure 3. Ontology and Rule Reasoning

The main operations for an OWL reasoner are:

- *Validation.* The OWL ontology language allows constraints to be expressed; the validation operation is used to detect when such constraints are violated by some data set, i.e., the validation consists in a global check across the schema and instance data looking for inconsistencies.
- *Querying the ontology*, including instance recognition (i.e., testing if an instance is belonging to a given class) and inheritance recognition (i.e., testing if a class is a subclass of another class and if a property is a sub-property of another).

For our example, a query using SPARQL [26] to obtain all routers associated to an *Autonomous System* is the following:

SELECT ?cs
WHERE { ?cs RouterInAS #AS100 }';

Table 4. Example of SPARQL language

The rule reasoner performs inference about SWRL rules deriving new information (i.e., new OWL instances). For our previous SWRL rule (shown in Table 2), the rule reasoner infers new data when a new router is associated to the autonomous system AS400. In particular, the rule infers that this new router belongs to the BGPPeepGroup “AS100 AS400”. The text shown in Table 5 match with the OWL representation of the new router, and underlines data are the inferred results for the execution of the previous SWRL rule.

<ComputerSystem rdf:about="#RTX">
<Name>Router X</Name>
<Dedicated>router </Dedicated>

<RouterInAS rdf:resource="#AS400"/>
<InBGPPeerGroup rdf:resource="#AS100_AS400"/>
</ComputerSystem>

Table 5. Example of inference results provided by the reasoner

Moreover, our proposal includes the property *NoInBGPPeerGroup* for the representation of a denied peering link to an Autonomous System. *NoInBGPPeerGroup* has been defined as disjoint property with *InBGPPeerGroup* that is expressed using the *DisjointObjectProperties* construct. In OWL semantics, two disjoint properties are those which can not belong to the same instance at same time. For example, the following statement shows the equivalent property to previous example, but denied:

<NoInBGPPeerGroup rdf:resource="#AS100_AS400"/>

The network administrator could define this statement when a new router is included into AS400. When the ontology does not include peering information for a router, i.e. no *NoInBGPPeerGroup* and *InBGPPeerGroup* is defined to the router, the peering decision is indeterminate, according to the OWL semantics. It is important to remark that this capability to define semantics such as the *DisjointObjectProperties* is an important added value of our proposal because enables the incorporation of semantics into any concept of the domain by the network administrators. The network operators can include this information according to the high-level policies associated to the domain being administrated.

B. Conflict detection

An important functionality of the rule-based reasoner is that it enables the detection of conflicts. For example, conflict detection can discover security failures, undesired behaviours, configuration mistakes and contradictions, among others. Our proposal identifies a conflict between different policies from the point of view of semantics, when the data inferred by the policies generates an inconsistency in the system description. For example, disjoint properties can be a way to generate an inconsistency. Thus, thanks to the automatic generation performed by the reasoner, the system is able to identify when two disjoint properties appear simultaneously (inconsistence).

In our example, an inconsistence can appear when an instance *NoInBGPPeerGroup* exists between AS100-AS400 and a router (meaning that this peering link is prohibited to the router), and then the rule reasoner infers a new instance *InBGPPeerGroup* between them (meaning that there is a peering link). The text in Table 6 shows the inconsistence in OWL using RDF/XML syntax.

<ComputerSystem rdf:about="#RTX">

<Name>Router X</Name>
< Dedicated>router </Dedicated>
<RouterInAS rdf:resource="#AS400"/>
<NoInBGPPeerGroup rdf:resource="#AS100_AS400"/>
<InBGPPeerGroup rdf:resource="#AS100_AS400"/>
</ComputerSystem>

Table 6. Example of an inconsistent situation

The inconsistence is detected by the reasoner because an instance cannot belong to two disjoint properties simultaneously. Moreover, disjoint classes can also be a way to generate an inconsistency (that is expressed using the *DisjointClasses* construct). Thus, two classes which represent a conflictive pair are defined as disjoint classes in the application domain. In our example, an inconsistence could appear whether *ComputerSystem* and *Service* instances are representing the same individual because these classes are declared as disjoint. Additionally, OWL language provides other constructs for equivalence between instances, equivalence between classes, and complement of classes and properties that can be used to model the application domain. If the domain description is not consistent as defined by the OWL semantics, then the reasoner automatically will detect an inconsistence.

Notice that OWL provides a set of semantics that enables to the network administrators to describe conflict in the system to be managed. Moreover, SWRL offers an important added value for which it enables the description of conflictive situations. Notice that an SWRL can define a conflictive situation in the antecedent part of the rule and to raise a conflict in the consequent part. This allows the network administrator to detect any conflict in the system that can be described in OWL/SWRL. For example, the conflict of interest and conflict of duties can be easily detected. While the former occurs when the autonomous system is able to perform actions which may conflict with its own interest, the later occurs when an autonomous system is obligated to perform incompatible actions. In these cases, a simple rule in which the conflictive actions are specified in the antecedent part can insert a disjoint definition among these actions to detect the conflicts.

Finally, some conflict resolution is necessary to provide an automatic solution for conflict analysis. Policy conflicts can be detected using our proposal. However, there is no mean of deciding at run-time what the original intention of the network administrator was; for our example, whether the peering link should be established or not. To solve the conflict resolution, our proposal uses prioritization, where more recent descriptions take priority over older ones (obsolescence). This resolution strategy assumes that domain descriptions defined or inferred more recently are by consequence more ‘up-to-date’, and therefore relevant, while older descriptions become obsolete. Once detected, a decision can be made as how resolving the conflict. For our example, the conflict resolution decides to remove the description *NoInBGPPeerGroup* (meaning that the peering link can be established) because the inferred data (*InBGPPeerGroup*) is more recent. Notice that this resolution technique is independent of the kind of conflict detected. Because it depends on the time on which the information is available in the system rather than the kind of conflict. For this reason, the technique can be applied to all the different kinds of conflict detected in the system.

VI. DEPLOYMENT OF SEMANTIC-AWARE ROUTING POLICIES

SDL and SPL are languages used internally for the management framework. The administrator can use a graphic tool to define the system domain and policies, although they are stored and encoded by SDL and SPL, respectively.

Moreover, the framework needs a PDP (Policy Decision Point) [27] [28] that evaluate the policies and system description to establish the BGP configuration in managed routers. Each managed router includes a PEP (Policy Enforcement Point) that applies BGP data into configuration files (see Fig. 4) and provides network information to the BGP PDP that completes the system description defined by administrator and returns monitoring information.

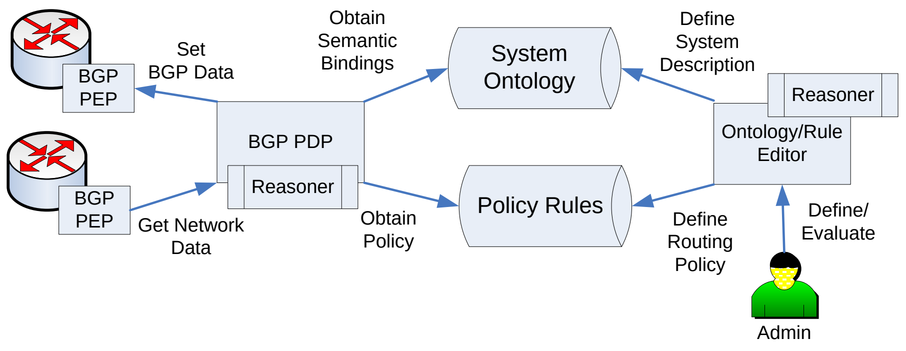

Figure 4. Managing routing policies

The administrator defines the policy creating policy rules follow the if-then paradigm. Our proposal uses the ORE-GUI [29], which is a stand-alone application created in our research group and aimed to graphically edit, test, debug and validate ontology rules in any domain. ORE-GUI is a wizard guiding the administrator when editing rules and interpreting the reasoning results. It also allows debugging the high-level policy definition, which an added-value over other existing approaches. In fact, it helps the network administrator to test a given set of policies before they are applied in the real system. In case some of these policy rules are not behaving well, the network administrator has the possibility to change the definition, before the final policy rules are enforced to the BGP PDP (see Fig. 4). These policies enable the definition of high-level routing policies as well as policies to detect conflict in the system.

A key element in the framework is the reasoner used in the BGP PDP. Our proposal uses Jena [30], as it provides a set of methods for loading and managing ontologies, together with a group of reasoning engines with different capabilities such as Pellet [31]. Jena is employed in the core part of the framework to obtain a working model from the domain ontology and also as a rule reasoning engine. Hence, the working model can be accessed by ORE according to the rule tasks to be performed. For example, ORE builds a domain concept hierarchy to be displayed when editing rules in the GUI, using for this the ontology working model given by Jena.

The PDP/PEP information exchange must be real time to avoid that up-to-date PEP information is not considered on the reasoning process and to guarantee that BGP configurations are real-time enforced. Our proposal uses SNMP protocol, where the BGP PDP retrieves information through GET operations and sends BGP configuration updates through SET operations. The BGP PEPs will send data without being asked using TRAP operations. Moreover, our proposal uses BGP4 MIBs [32] that, for example, are sent when establishing peering or loss of a peer.

The BGP PEP can be integrated on the routing software or be external software that enforces the BGP configuration using an additional protocol or mechanism, e.g. CLI commands via Telnet. Our proposal integrates the PEP functionality into GNU Zebra [33], where the BGP4 MIB support is limited and so we have extended to complete framework functionality.

VII. PERFORMANCE

In order to measure the performance of the proposal, several executions have been performed, starting from a simple scenario and bringing it more and more complex. This complexity is achieved by increasing the number of ontology instances present that, in turn, increase the number of statements of the knowledge base of the BGP PDP reasoner. The number of instances for a given execution is referred to as population. In order to decide the mathematical function to be used to set up the populations' size, our system was stressed until it was overload in terms of memory and time response requirements. Once we knew the top level threshold of the system for the computer in which this experiment was carried out, we designed an exponential function in which the final step provided this number. It is common to make use of an exponential function to determine the size of these populations. Starting from the basis that the biggest population should be the one which shows impracticable performance results in terms of response time or amount of required memory, the following mathematical function has been empirically obtained:

$$f(x) = 2000 \cdot e^{(x/2)}$$

where x takes integer values from the interval [0; 10). These sizes will be used to evaluate the performance and scalability of the proposal. Notice that the last population has more than 180.000 instances which, in turn, will result in a big amount of statements whose exact number depends on the concrete scenario. For instance, a population of 180000 instances leads to more than 2.6 millions of statements since each instance have properties and features associated that will be represented as statements in the knowledge base.

Population	Instances
1	3297
2	5436
3	8963
4	14778
5	24364
6	40171
7	66230
8	109196
9	180034

Table 7. Number of instances by population

Once the population size has been defined, it is necessary to define the percentage of instances which will belong to each class of the application domain. This definition has been done bearing in mind the peculiarities associated to realistic scenarios. For example, typically there should be available much more number of BGP routes than autonomous systems. Moreover, there should be more routers than autonomous systems. In this sense, figure 5 depicts a graphic showing the percentage of instances belonging to each class. The distribution has been done following the common sense and the scenarios provided in other research works [1] [12] [13].

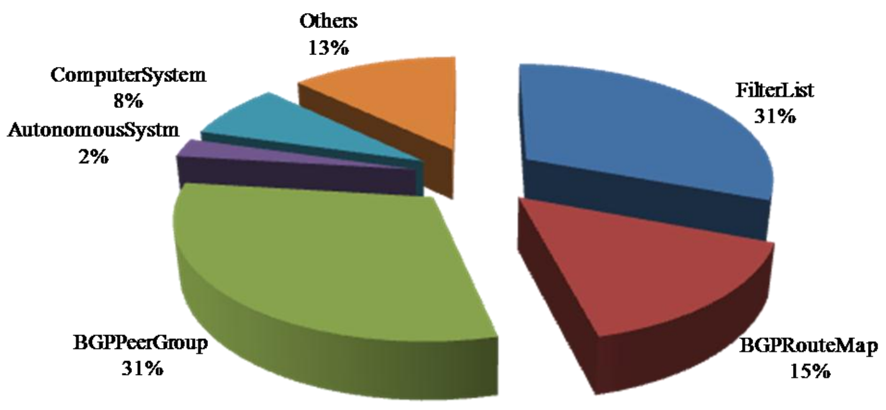

Figure 5. Instance distribution in populations

These percentages intend to model a scenario and they will be used to generate the populations. Populations of instances are created in a driven way for representing realistic features on the generated scenarios. These populations are used to measure different timings associated to load populations, check consistence on knowledge base and get times on detecting conflicts, among others. Using the population size as metric to evaluate the reasoner performance provides a high level measure by directly considering the number of entities which are present in the managed system model. On the other hand, using the number of statements provides a lower level metric which measures the amount of data inside the reasoner. Moreover, reasoners performance is usually determined by the complexity of the ontology and the number of statements rather than considering the number of instances represented by such statements. Therefore, both the statements number and the population size will be taken into account for this analysis.

Figure 6 shows a graphic depicting the relationship between population size and the number of statements for our scenario. As can be observed, the number of statements is bigger than the number of instances. The bigger a population is the greater amounts of statements appear, although this growth is not necessarily proportional.

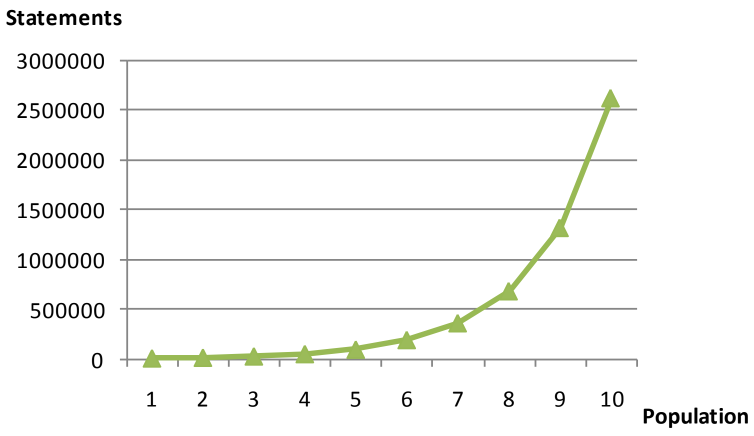

Figure 6. Statements and population relationship

Different simulations have been done in order to evaluate the response time for our proposal. Also the detection method has been tested by measuring the time that is needed to identity a conflict in the knowledge base. The machine used to run the simulations has been a Core 2 Duo T7500 at 2.2 GHz with FSB 800 MHz and 4 GB RAM. Regarding the operating system, a common clear Ubuntu Linux in its version 8.0.4 has been used. Finally the used Java Virtual Machine has been the Sun JDK 1.6 tuned up to 2 GB of maximum heap size.

The testbed is basically composed of three different tests executed against ten incremental populations. The first test deals with the load time, i.e., the time needed to load fully the knowledge base with the information provided by the BGP PEP entities. The second one measures the time that the Pellet reasoner spends in consistence checking when the knowledge base does not hold any inconsistency fact. It should be noticed that a knowledge base is consistent when it does not contain any contradictory statements.

Finally, the third test is about the time that the rule reasoner requires detecting the conflict when there is at least one inconsistency fact present in the ontology. This time deals with the consistency checking operation which will allow detecting the conflict be means of the reasoner.

As a proof of concept, these testbeds have been executed in a prototypical implementation of the BGP PDP element. This software is simulating the BGP information provides by the BGP PEP elements in order to evaluate the performance of the PDP elements which in turn, it is the key element in the architecture. This information conforms each of the previous described populations and it is inserted in the PDP together with the policies. These policies (SPL information) are also automatically generated in the software in order to simulate realistic description of policies defined by administrator. Thus, the PDP perform the reasoning with all these information using the combination of Jena and Pellet reasoners as previously described. Figure 7 depicts the performance results obtained by the execution of the third aforementioned tests with the selected populations in this PDP element. This graphic illustrates two different metrics. One of them defines the time required to perform the consistency operation in a non conflictive scenario, whereas the other one defines the time spent to perform this operation in a conflictive situation. In conflictive scenarios detection rules will be fulfilled, causing the insertion of new facts in the knowledge base which, in turn, will cause inconsistency. It should be noticed that load times are included in the metric values in order to show the overhead introduced by loads. An added value of our approach is the expressiveness provided in the languages used to describe both the system descriptions and routing policies. This added value enables to define high-level policies whereas it also enables the definition of policies to detect conflicts in the system. Due to the well-known trade-off available between expressiveness provided in a language and computational time for carrying out reasoning processes over this language, Figure 7 has been included in order to demonstrate that our proposal provided acceptable convergence times dealing with big BGP systems whereas provide a high-level expressiveness in the language. These results are not comparable with other proposals. Notice that each proposal provides its own expressiveness in the language ranging from a simple configuration file to more complex scenarios with some inference processes therein. Each one has associated a computational time to carry out these inference processes and the high expressiveness provided in this proposal is not comparable to other existent solutions.

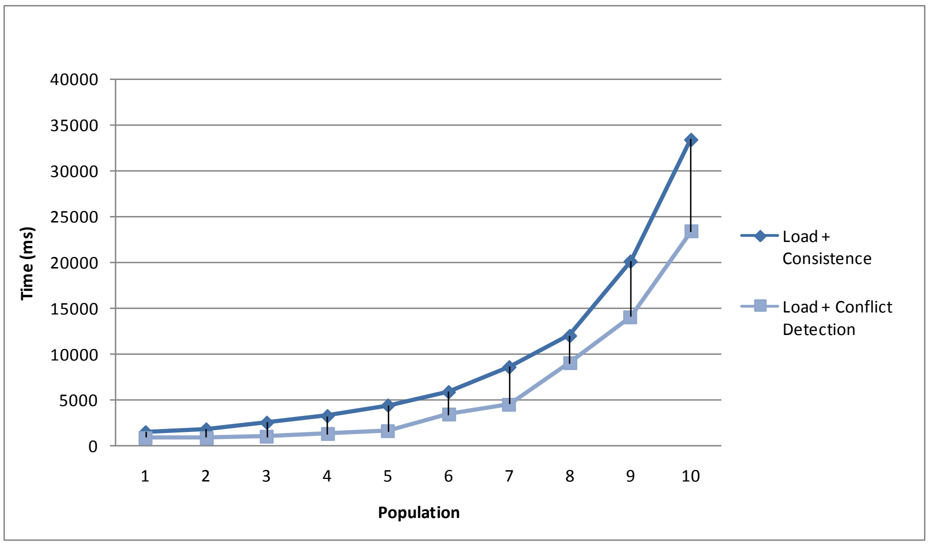

Figure 7. Scenario performance

Some conclusions arise from these results. Firstly, all the ontologies have its own peculiarities and therefore the consistence checking operation shows different statistics, regardless the number of instances. In this sense, a BGP routing scenario is quite complex due to the amount of classes needed to model it. This leads the reasoner to require more time checking the consistence. However, we can establish a ceiling in a population over 15.000 instances (approximately 300 autonomous systems) requires more 5 seconds for checking the consistence. Moreover, these results show that, in general, the time spent by the reasoner to check the consistency is quite higher when dealing with a consistent knowledge base than with an inconsistent one.

Another tested has been simulated in order to analyze the impact of the ranging of the percentages shown in Figure 5 using a fixed population. This tested enables the analysis about how the different concepts available in Figure 5 provides different overload in the system. It has been used the population 5 shown in Table 1 to carried out this testbed. The ranging of the percentages has been ±15% to all the different concepts available in Figure 5. As a result, all the convergence times of the simulated scenarios provides similar convergence times. The results provide only a difference of 3% in the average convergence time of the system; this fact determines that the percentages of the different kinds of concepts available in the system do not affect much to the response time of the routing policies.

VIII. Conclusions and Future Work

Policy-based management of routing configuration is a research field that still needs to make significant efforts to provide a final solution to the problem of automatic provision of configurations. Our proposal is focused on solving certain important limitations in current routing policy languages.

In this sense, this article provides a detailed view of a Semantic Web based representation of routing management policies and a technique defined to detect policy conflicts. This representation has been described from the perspective of some of the advantages related with its use (e.g., reasoning capabilities) and a prototypical implementation where both representation and conflict detection technique have been successfully tested.

As statement of direction [34], current work is being focused on the definition of taxonomy of conflicts existing in the routing area. We are also currently working in advanced resolution techniques that can provide an automated answer when a conflict appears. Prioritization of policies is a technique suitable for resolving the policy conflicts, but there are conflicts more subtle, hence prioritization may not be enough to resolve them.

ACKNOWLEDGEMENTS

This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grants AP2006-4150, CSD2006-00046, TIN2008-06441-C02-02 and TIN2009-14475-C04. Thanks to the Funding Program for Research Groups of Excellence with code 04552/GERM/06 granted by the Fundacion Seneca.

REFERENCES

[1] Y. Rekhter, T. Li, S. H. Ed., “A Border Gateway Protocol 4 (BGP-4)”, IETF, RFC 4271, January 2006.

[2] C. Alaettinoglu et al., “Routing Policy Specification Language (RPSL)”. RFC 2622, June 1999.

[3] Merit Network, List of Routing Registries, http://www.irr.net/docs/list.html, 2009.

[4] Internet Systems Consortium, IRRToolSet, http://irrtoolset.isc.org/, 2009.

[5] T. Berners-Lee, J. Hendler, O. Lassila, “The SEMANTIC WEB”. Scientific American, 2001.

[6] F.J. García, G. Martínez, A. Muñoz, J.A. Botía, A.F. Gómez-Skarmeta, “Towards semantic web-based management of security services”. Annals of Telecommunications, vol. 63(3-4), pp. 183--194. Springer, 2008.

[7] M. Majewska, B. Kryza, J. Kitowski, “Translation of Common Information Model to Web Ontology Language”. International Conference on Computational Science, Part I, pp. 414-417, 2007.

[8] F.J. García, G. Martínez, J.A. Botía, A.F. Gómez-Skarmeta, “Description of Policies Enriched by Semantics for Security Management”. Web Semantics and Ontology, pp. 364--390, Idea Group Inc., 2006.

[9] F. Baader; D. Calvanese D. McGuinness, D. Nardi, P. Patel-Schneider, The Description Logic Handbook: Theory, Implementation and Applications Cambridge University Press, 2003

[10] J. Gottlieb, A. Greenberg, J. Rexford, J. Wang. “Automated Provisioning of BGP Customers”. IEEE Network, 17, 2003.

[11] H. Bohm, A. Feldmann, O. Maennel, C. Reiser, R. Volk. “Network-wide inter-domain routing policies: Design and realization”. NANOG 34, May 2005.

[12] X. Chen, Z. M. Mao, J. van der Merwe, “Towards automated network management: network operations using dynamic views”. Internet network management (SIGCOMM workshops), pp. 242–247, 2007.

[13] N. Feamster, H. Balakrishnan, “Detecting bgp configuration faults with static analysis”. Symposium on Networked Systems Design & Implementation, pp. 43–56, 2005.

[14] P. Kodeswaran,S.B. Kodeswaran, A. Joshi, F. Perich, “Utilizing semantic policies for managing BGP route dissemination”. Automated Network Management (INFOCOM workshops), pp. 1-4, April, 2008.

[15] E. Lupu, M. Sloman, “Conflict Analysis for Management Policies”, Proceedings of IFIP/IEEE International Symposium on Integrated Network Management, 1997.

[16] M.I. Yagüe, A. Maña, J. López, “A Metadata-based Access Control Model for Web Services", Internet Research Journal, Emerald, Vol. 15(1), pp. 99-116, 2005.

[17] E. Al-Shaer, H. Hamed, R. Boutaba, M. Hasan, “Conflict classification and analysis of distributed firewall policies”, IEEE Journal on Selected Areas in Communications 23 (10), pp. 2069-2084, 2005.

[18] Common Information Model (CIM), Distributed Management Task Force (DMTF), http://www.dmtf.org/standards/cim, 2009.

[19] OWL 1.1. Web Ontology Language, http://www.webont.org/owl/1.1/, 2009.

[20] SWRL: A Semantic Web Rule Language Combining OWL and RuleML, http://www.ruleml.org/swrl/, 2004.

[21] B. Motik, U. Sattler, R. Studer, “Query answering for OWL-DL with rules”, Journal of Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 3 (1), pp. 41–60, 2005.

[22] S. Quirolgico, P. Assis, AndreaWesterinen, M. Baskey, E. Stokes, “Toward a formal common information model ontology”, LNCS Web Information Systems 3307, pp. 11-21, 2004.

[23] G. Martinez, F.J. Garcia, A.F. Gomez, “Managing Semantic-Aware Policies in a Distributed Firewall Scenario”, Emerald Internet Research, vol. 17(4), pp. 362-377, 2007.

[24] H. Chen, F. Perich, T. Finin, A. Joshi, “SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications”. International Conference on Mobile and Ubiquitous Systems: Networking and Services, pp. 258--267, 2004.

[25] Y. Sure, S. Bloehdorn, P. Haase, J. Hartmann, D. Oberle, “The SWRC Ontology - Semantic Web for Research Communities”. 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005), pp. 218 - 231. Springer, 2005.

[26] E. Prud'hommeaux, A. Seaborne. SPARQL Query Language for RDF W3C, 2008

[27] J. Strassner, “Policy-Based Network Management: Solutions for the Next Generation”. Morgan Kaufmann, 2003.

[28] G. Martinez, F.J. Garcia, A.F. Gomez, “Policy-Based Management of Web and Information Systems Security: An Emerging Technology”, Book Chapter, Web and Information Security, pp. 173-195, Idea Group Inc, 2006

[29] Ontology Rule Editor (ORE), http://sourceforge.net/projects/ore, 2008.

[30] Jena – A Semantic Web Framework for Java, http://jena.sourceforge.net/, 2009.

[31] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, “Pellet: A practical OWL-DL reasoner, Journal of Web Semantics”, vol. 5 (2), 2007.

[32] J. Haas, S. Hares, “Definitions of Managed Objects for BGP-4”, IETF, RFC 4273, January 2006.

[33] GNU Zebra, http://www.zebra.org/, 2009.

[34] A. Pras, J. Schönwälder, M. Burgess, O. Festor, G. Martínez, R. Stadler, B. Stiller, “Key Research Challenges in Network Management”. IEEE Communications Magazine, vol. 45 (10), pp. 104-110, October, 2007.

BIO

Félix J. García Clemente is an assistant professor of Computer Networks at the Department of Computer Engineering of the University of Murcia. His research interests include security and management of distributed communication networks. He received an MSc and PhD degrees in Computer Science from the University of Murcia.

Jose M. Alcaraz Calero received the Computer Engineering and Master degrees with honours at the University of Murcia. He has been working at University of Murcia since 2004 in several European and international projects whereas he is doing his PhD. Currently he is research staff at Hewlett Packard Laboratories.

Jorge Bernal Bernabé received the Computer Engineering and the MSc in Computer Science from the University of Murcia. Currently, he is a research staff in the Department of Information and Communications Engineering of the University of Murcia. He has been working in several European projects while pursuing his PhD. His scientific activity is mainly devoted to the security and management of distributed systems.

Juan Manuel Marín Pérez is a research staff in the Department of Information and Communications Engineering of the University of Murcia. He has been involved in several European projects. His research interests include security and management of distributed systems. He received Engineering and MSc degrees in Computer Science from the University of Murcia and he is currently doing his PhD.

Gregorio Martínez Pérez is an associate professor in the Department of Information and Communications Engineering of the University of Murcia. His research interests include security and management of distributed communication networks. He received an MSc and PhD in Computer Science from the University of Murcia.

Antonio F. Gómez Skarmeta received the MS degree in Computer Science from the University of Granada and BS (Hons.) and the PhD degrees in Computer Science from the University of Murcia Spain. He is a Full Professor in the same Department and University. He has worked on different research projects at regional, national and especially at the European level in areas related to advanced services like multicast, multihoming, security and adaptive multimedia applications in IP and NGN networks.