Grid-Aware Network Resources Allocation using a Policy-Based Approach

Ricardo Neisse, Lisandro Z. Granville,
Maria Janilce B. Almeida, Liane Margarida R. Tarouco

Institute of Informatics - Federal University of Rio Grande do Sul
Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brazil

{neisse, granville, janilce, liane}@inf.ufrgs.br

Abstract. *Computing grids require the underlying network infrastructure to be properly configured for provisioning of appropriate communications among their nodes. The management of networks and grids are currently executed by different tools, operated by different administrative personnel. Eventually, the grid communication requirements need corresponding support from the network management tools, but such requirements are fulfilled only when grid administrators manually asks the network administrators for the corresponding configurations. In this paper, we propose a policy translation mechanism that creates network policies given grid requirements expressed in grid policies. We also present a system prototype that allows (a) grid administrators to define grid policies, and (b) network administrators to define translating rules. These translating rules are used by the translation mechanism proposed by this work for the generation of the necessary underlying network configuration policies.*

Resumo. *Grids computacionais requerem que a infra-estrutura de rede subjacente seja adequadamente configurada para o fornecimento apropriado dos serviços de comunicação entre os nodos do grid. O gerenciamento da rede e do grid atualmente são executados através de diferentes ferramentas, operadas por diferente pessoal administrativo. Eventualmente, os requisitos de comunicação do grid precisam de um suporte de comunicação correspondente das ferramentas de gerenciamento de redes, mas esses requisitos somente são atendidos quando os administradores do grid manualmente requisitam aos administradores da rede as configurações correspondentes. Este artigo propõe um mecanismo de tradução de políticas que cria políticas de rede a partir dos requisitos do grid expressos nas políticas de grid. É apresentado também um protótipo que permite (a) administradores de grid definirem políticas de grid e (b) administradores da rede definirem regras de tradução. Essas regras de tradução são usadas pelo mecanismo de tradução proposto neste trabalho para geração das políticas necessárias para configuração da rede subjacente.*

1. Introduction

Grids are distributed infrastructures that allow sharing of computing resources distributed along several different administrative domains between users connected through a computer network. Resources can be processing, memory, storage, network bandwidth, or any kind of specialized resource (e.g. telescopy, electronic microscopy, medical

diagnostic equipment, etc.). Typical grid applications are: high performance computing, data sharing, remote instrument control, interactive collaboration, and simulation. Usually, applications that require powerful, specialized, or expensive computing resources get benefits from the use of grid infrastructures. Most of these applications are latency and jitter sensitive, and often require high network bandwidth and multicast communication support. Thus, in order to manage a grid infrastructure, the management of the underlying network that provides the communication support is also a requirement.

Besides the network requirements, other factor that may turn the grid management complex is the resource distribution. Since the grid resources are distributed along several administrative domains, the grid operations can only be supported through grid management solutions that coordinately interact with each network administrative domain. In this management scenario, two administrative figures come out: the grid administrator and the network administrator. The grid administrator is responsible for the management of the grid resources (e.g. clusters and storage servers), proceeding with tasks such as user management and access control. The role of the network administrator is to proceed with the network maintenance to allow the users to access the grid resources through the communication network.

The management of the network infrastructure is important because grid users access the shared resources through the network and, if the network is congested or unavailable, such access is likely to be compromised. The configuration of the underlying network allows, for example, reservation of network bandwidth and prioritization of critical flows, which is generally proceeded with the use of a QoS provisioning architecture such as DiffServ or IntServ. The current grid toolkits [Globus 2003] [Steen et al. 1999] [AccessGrid 2003] do not interact with either the network QoS provisioning architecture nor the network management systems. That leads to a situation where the grid and network administrators are forced to manually interact with each other in order to proceed with the required configuration of the communication support. Thus, although the toolkits provide support to the grid resources management, the available support for an integrated management of grids and networks is still little explored.

In this paper we propose a policy translation solution where network management policies are created from the grid management policies. The main objective is to allow an integrated management of the communication infrastructure required for the grid operation. The proposed solution translates grid policies to network policies through a translation architecture. In this architecture the network administrators, in each administrative domain that composes the grid, define translation rules in order to control how to create the network policies based on the grid requirements. We implemented a Web-based prototype, to support the proposed architecture, where the grid administrator is allowed to specify the grid policies, and the network administrators (in each domain) can specify the translation rules.

The remainder of this paper is organized as follows. Section 2 presents related work, where the management support provided by toolkits and an actual typical scenario of grid management is detailed. Section 3 presents the proposed hierarchical policy-based management architecture, and Section 4 shows the prototype developed based on such architecture. Finally, the paper is finished in Section 5 with some conclusions and future work.

2. Related Work

The management of grid resources is not a trivial work, since the grid resources can be located along several different administrative domains. For example, the cluster of a grid could be located in an company, and the storage servers could be located in a university. However, both resources (processing and storage) belonging to the same grid are located in different administrative domains. In this situation, each resource is maintained by a different administrative entity, with different operation policies. Thus, a distributed management coordination of the grid resources is required.

Typical grid management tasks that need to be coordinated in the grid distributed environment are, for example, user authentication and resource scheduling. Considering that most grid infrastructures need a common management support, software libraries, called toolkits, were developed. These toolkits provide basic services and try to reduce the initial work needed to install and manage a grid. Toolkit examples are Globus [Globus 2003], Globe [Steen et al. 1999] and AccessGrid [AccessGrid 2003].

A commonly required network configuration in a conference grid, implemented for instance with the AccessGrid toolkit [AccessGrid 2003], is to reserve network resources for multicast audio and video flows. This configuration must be executed in all administrative domains that are part of the grid, to guarantee a successful audio and video transmission. The current version of the AccessGrid toolkit considers that all needed configuration and network reservations for the grid operation were made, which is not always true.

A toolkit that explicitly considers an integrated network infrastructure management is Globus [Globus 2003], through its Globus Architecture for Reservation and Allocation (GARA) [Foster et al. 2002]. This architecture provides interfaces for processor and network resources reservations. GARA was implemented in a prototype where configurations are made directly in routers to configure queue priorities of the DiffServ architecture. This implementation considers that the toolkit has permission to directly access and configure the network devices.

Globus, in its management support, also explicitly defines the concept of proxy (an important concept for the grid policy definitions to be presented in the next section). A proxy represents a grid resource that runs determined tasks on behalf of the users and have the same access rights that are given to the user. Globus implements proxies using credentials digitally signed by users and passed to the remote resources. A possible proxy configuration could be a user accessing a storage server through a process running in a supercomputer. In this case, the supercomputer acts as a user proxy, since it requests actions in name of the user.

Besides the management support provided by the toolkits, policy-based grid solutions were also proposed by Sundaram et al. [Sundaram et al. 2000] [Sundaram and Chapman 2002]. An example of such grid policies is showed in Listing 1. This policy uses parameters to specify processor execution and memory usage for a user accessing a server during a determined period of time. It is important to notice that this approach for grid policy definition does not allow the specification of network QoS parameters to be applied in the user-server communication and also does not support explicitly the concept of user proxies.

```
machine : /O=Grid/O=Globus/OU=sp.uh.edu/CN=n017.sp.uh.edu
subject : /O=Grid/O=Globus/OU=sp.uh.edu/CN=Babu Sundaram
login : babu
startTime : 2001-5-1-00-00-00
endTime : 2001-5-31-23-59-59
priority : medium
CPU : 6
maxMemory : 256
creditsAvail : 24
```

Listing 1. Grid Policy

Sahu et al. [Verma et al. 2002] define a management service where global grid policies are combined with policies of each local domain. The local policies have high priority, which means that if a global policy defines a 20GB disk allocation in a server, but the local administrator defines a policy that allows only 10GB, the local policy is chosen and only 10GB is allocated. Grid policies in each administrative domain can be influenced by local network policies that can, for some reason (e.g. critical local application), indicate that a local resource or service should not be granted to a grid member. Here, potential conflicts of interest between the grid and network administrator can exist and impact in the definition of grid and network policies. Therefore, for a proper grid operation, the local network administrator and the global grid administrator are supposed to have some kind of common agreement regarding the grid and network resources on the local domain.

Another proposal that uses policies for network configuration aiming grid support is presented by Yang et al. [Yang et al. 2002]. The solution specifies an architecture divided in a policy-based management layer (that follows the IETF definitions of PEPs and PDPs [Westerinen et al. 2001]), and a layer that uses the concept of programable networks (active networks) represented by a middleware. With this middleware, the network devices configuration are done automatically. However, the Yang et al. work does not specify how grid and network policies for the proposed multi-layer architecture are defined.

Sander et al. [Sander et al. 2001] propose a policy-based architecture to configure the network QoS of different administrative domains members of a grid. The policies are defined in a low level language and are similar to the network policies defined by the IETF [Yavatkar et al. 2000]. The Sander et al. approach defines an inter-domain signaling protocol that sequentially configures the grid domains that are member of an end-to-end communication path (e.g. a user accessing a server). The signaling protocol allows the communication between bandwidth brokers located in each grid domain. Such brokers exchange information with each other in order to proceed with the effort to deploy a policy. Although the proposed architecture is based on policies, it does not present any facility to allow the integration with the grid toolkits presented before: it is only an inter-domain, policy-based QoS management architecture.

Considering these solutions we identified a typical scenario of grid and network management. In this scenario the grid administrator coordinate the grid operation using the support provided by the toolkits, and manually interact with the network

administrators in each domain to guarantee that the needed network configurations for the grid operation is executed. Analyzing this scenario, it is possible to notice that every time a grid requirement that imply in a new configuration in the network infrastructure is changed, a manual coordination between the grid and network administrators is needed. The support provided by the toolkits to solve this situation is very limited and, in most cases, it does not even exist. Actually, most toolkits consider that the network is already properly configured for the grid operation, which is not always true.

3. Translation of Grid Policies to Network Policies

The solution presented in this paper defines a translation mechanism where network policies are created by translation rules using as input data information retrieved from the grid policies. Figure 1 shows a general view of the translation process. First, at the top, grid management policies are defined by a grid administrator. The translation mechanism, based on the translation rules defined by the network administrators, creates the network policies. In our solution the network administrator is not supposed to define static network policies anymore, he or she is now supposed to define the translation rules of the translation mechanism. The network policies generated by the translation mechanism are then translated to network configuration actions executed by Policy Decision Points [Westerinen et al. 2001] of a regular policy-based management system.

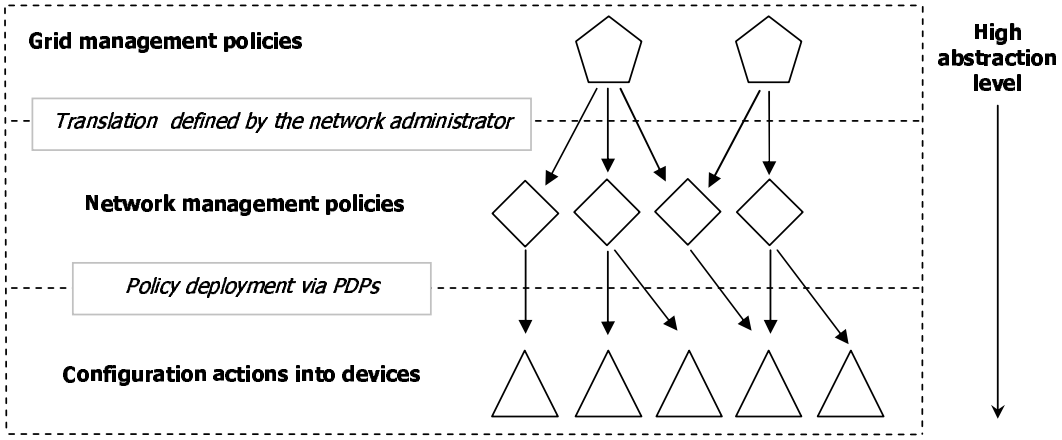

Figure 1. Hierarchy for policy translation

To define grid policies we used an hypothetical language, which is based in previous work on grid policies [Sundaram et al. 2000] [Sundaram and Chapman 2002]. In this language we defined a set of elements to allow the expression of grid policies regarding users, resources, proxies and network QoS requirements. Such elements are not present together in any the current grid policy languages. Some grid policy solutions provide policies to control proxy instantiation and lifetime [Sundaram et al. 2000], but does consider network reservations. The support for grid policies definition could be accomplished by actual established policy languages such as Ponder [Damianou et al. 2001] and PDL [Lobo et al. 1999]. For simplicity we decide in do not use any this languages.

In the definition of the grid policy elements we first identify that grid policies must be defined not only based on grid users and resources, but also based on proxies and

network QoS requirements. We suppose here that a grid policy language supports both proxies and network QoS following the condition-action model from the IETF, where a policy rule is composed by condition and an action statements. A condition is a list of variables and associated values that must evolve to true in order to turn the rule valid and an action is a list of variable attributions triggered when the rule just turned to be valid. Thus, in our approach, a grid policy is composed by a conditional statement (`if`) containing conditional elements related to grid users (`user`), proxies (`proxy`), resources (`resource`), and time constrains (`startTime` and `endTime`).

```
if (user == "mity" and
    resource == "Cluster" and
    startTime >= "11/25/2003 00:00:00" and
    endTime <= "11/25/2003 23:59:59")
{
  allowAccess = true;
  login = gridUser;
  maxProcessing = 50%;
  networkQoS = remoteProccessControl;
}

if (user == "mity" and
       proxy == "Cluster" and
       resource == "Data Server" and
       startTime >= "11/25/2003 00:00:00" and
       endTime <= "11/25/2003 23:59:59")
{
  allowAccess = true;
  maxAllowedStorage = 40GB;
  networkQoS = highThroughputDataIntensive;
}
```

Listing 2. Grid policies examples

In the grid policy rules presented in the Listing 2 the user `mity` is able to access during a specific period of time a grid cluster (`Cluster`). The user is able also to access through the cluster a storage server (`Data Server`). In this example the user does not have direct access to the data server, but he or she is able to store information generated by the cluster in the server. In each one of the rules different QoS requirements are defined in the policy actions: remoteProcessControl and highThroughputDataIntensive.

Two different network paths are used when deploying the `remoteProccess Control` and `highThroughputDataIntensive` network classes of services. For remote process control, the intermediate network devices from the user host and the `LabTec` cluster are supposed to be configured in order to allow a proper remote operation. In the second case, the network devices between the `LabTec` cluster and the `UFRGS Data Server` should be configured to support a high throughput of data transfer. It is important to notice that no specific network device configuration will be executed in the path between the user host and the storage server, since no policy directly binding the user to the storage is defined.

The grid policy language supports rule nesting and domains [Sloman and Moffett 1989]. Rule nesting allow one inner rule to be defined in the

context of another outer rule and domains allows the definition of classes of resources, users and proxies in the policy conditions. The internal rules will be considered only when the conditions of the external rule become valid, which optimizes the policy evaluation process. Using domains the administrator is allowed to define , for instance, that the user `neisse` can access only one grid cluster and two storage servers, but does not designate what specific cluster and storage servers will be used.

In order to define network management policies we used the same language present for grid policies. Listing 3 presents an example of such network policy. This policy states that the traffic generated by host 143.54.47.242 sent to host 143.54.47.17, using any source port (*), addressed to the HTTP port (port 80 over TCP), and with any value as DSCP (*) will have 10Mbps of bandwidth, will be marked with value 1 in the DS field, and will gain priority 4. We define in our work a translation model where the network administrator is able to define a translation rule to generate a network policy given a grid policy and the QoS requirements definition.

```
if (srcAddress == "143.54.47.242" and
    srcPort == "*" and
    dstAddress == "143.54.47.17" and
    dstPort == "80" and
    DSCP == "*" and proto == "TCP" and
    startTime >= "11/25/2003 00:00:00" and
    endTime <= "11/25/2003 23:59:59")
{
    bandwidth = 10Mbps;
    DSCP = 1;
    priority = 4;
}
```

Listing 3. Network policy example

Until now, the grid policies presented state the required network QoS through the `networkQoS` clause and an associated class of service identification (e.g. `remoteProccessControl` and `highThroughputDataIntensive`). Behind these identifications, a set of QoS-related parameters is found. We suppose that the following parameters are available in defining new classes of services: minimum bandwidth, required bandwidth, minimum loss, maximum loss, priority, and a sharing flag that indicates if the bandwidth used by the class of services will be shared among the users (other network-related parameters can be supported depending on the underlying QoS provisioning architecture). The classes of services are supposed to be defined by the grid administrator and stored in a repository to be further used when new grid policies are defined.

Our translation architecture is presented in Figure 2. Each step in a grid policy translation is identified with the numbers from 1 until 9 and are described bellow:

1. the grid administrator defines grid policies and required associated network classes of services through a grid policy editor and stores them in a global grid policy repository;
2. the network administrator of each administrative domain defines a set of translation rules using a translation rule editor and stores them in a local rule repository;

3. once the grid administrator wants to deploy a policy, the translation engine retrieves such policy from the global grid policy repository;
4. the translation engine also retrieves the set of translation rules from the local rule repository;
5. the translation engine translates the grid policies based on the translation rules and consults the toolkit to discover network addresses and protocols information;
6. once the translation engine builds up new network policies related to the local domain, these policies are stored back in a local network policy repository;
7. then, the translation engine signals a set of PDPs in the local domain in order to deploy the just created network policies in a set of PEPs;
8. the signalled PDPs retrieve the network policies from the local repository;
9. the PDPs translate the network policies to configuration actions in order to deploy such policies in the local domain PEPs.

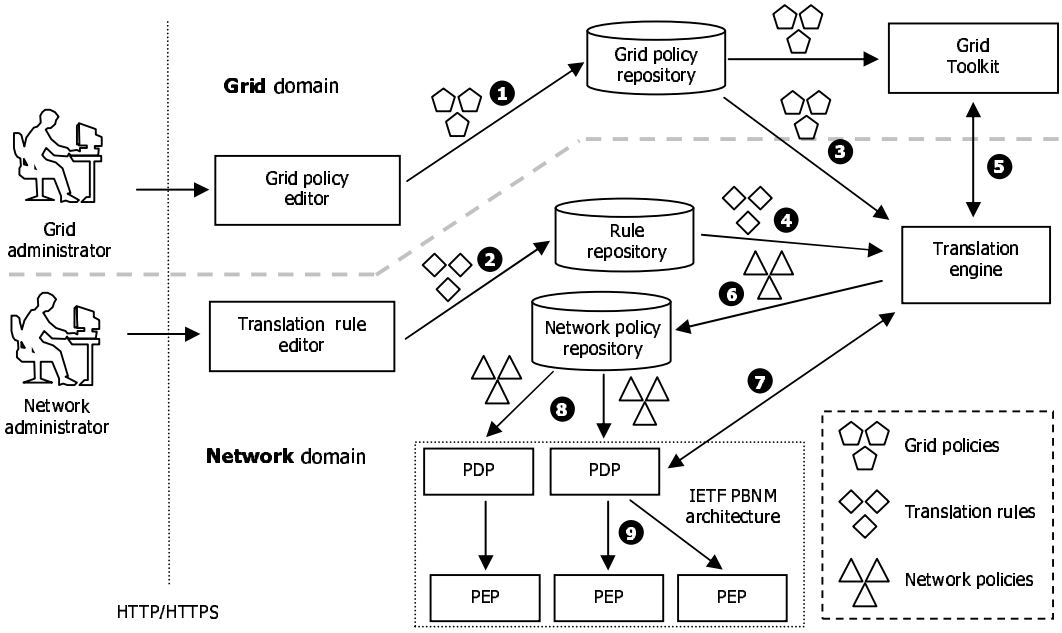

Figure 2. Policy Translation Architecture

We suppose that only one grid administrator is responsible for creating grid policies using the previously presented grid policy language. Although the figure presents just one network administrator several network administrators may interact with the architecture to define the translation rules. An object-oriented/condition-action language (further presented) is used to create the translation rules, which are very similar to policies, except that in this case they control the translation process. Thus, the translation rules may be taken as meta-policies that govern the translation processes of grid policies to network policies.

Translation rules are defined dealing with a set of policy objects that addresses both original grid policies and network policies to be created. Four global objects are implicitly instantiated before a translation rule evaluation: `schedule`, `srcResource`, `dstResource`, and `requiredQoS`. These objects identify a grid communication pair and hold, respectively, the period in which the communication has to be considered, the source

grid resource, the destination grid resource, and the QoS required from the underlying network. The four implicitly instantiated objects have their content values retrieved from the grid policy being translated, and can be used in the conditions or in the actions of a translation rule.

The translation engine evaluates a translation rule parsing its code and accessing the values provided by the four implicitly instantiated objects. At the end of the translation process, the translation engine will provide a set of new network policies. Sometimes, however, the engine is forced to block the translation process if all information required to produce new network policies is not available. That happens because the original grid policy and the translation rule do not always provide all the information required to resolve the communication pairs. The remainder information (not found in the grid policy and in the translation rule) needs to be retrieved from the grid toolkit. Listing 4 presents the evaluation algorithm used in the grid policy deployment by the translation mechanism. In this process the mapping engine resolves the communication pairs defined in the grid policy and executes all the mapping rules defined in the domain for each communication pair.

```
function onApplyGridPolicy (gridPolicy policy) {
  Array communicationPairsList = resolveCommunicationPairs(policy);
  Array translationRulesList = retrieveDomainTranslationRules();
  for (int i=0; i<sizeof(communicationPairs); i++) {
    for (int n=0; n<sizeof(translationRules); n++) {
      translationRule[n].exec(communicationPairs[i]);
    }
  }
```

Listing 4. Algorithm for translation rule execution

In a translation rule, when dealing with network policies, a fifth class called `NetworkPolicy` is used. To create new network policies, a translation rule must first instantiate a `NetworkPolicy` object, and proceed manipulating its content in order to define the network policy conditions and actions. The `addCondition` and `addActions` methods of `NetworkPolicy` help building up the new policy. Listing 5 presents an example of a translation rule that creates two new network policies from a single grid policy.

This translation rule defines the network policies `p1` and `p2` to mark packets and allocate bandwidth in the underlying network, operating with the IETF DiffServ architecture. However, `p1` and `p2` are only created if the original grid policy states that the source resource is located in the local network (143.54.47.0/24) and the destination resource belongs to another network, different than the local one. The network policy `p1` verifies the local and remote addresses, the remote port (80), and the transport protocol (TCP) of the network packets in order to mark the DS field with the DSCP 2. The policy `p2`, on its turn, only verifies the DSCP to guarantee the required bandwidth determined in the original grid policy.

In a conventional policy-based network management system, the network administrator is the one responsible to determine in which devices of the managed network the policies will be deployed. The selection of these devices triggers the policy deployment, although the policies are activated only at scheduled times, due to the time

constraints in the policy rule conditions. In the case of our policy-based grid management, the network devices in which the created network policies will be deployed can not always be determined except when the grid policies become valid. Thus, a mechanism to support the selection of target network devices is supposed to be provided in order to automate this process. We provide such mechanism introducing in the translation rule language the support for dynamic domains [Ceccon et al. 2003]. Such domains are defined through selection expressions introduced in the translation rules. In the example from Listing 5, the previous policy `p1` is deployed in the ingress interface of the first router, while policy `p2` is deployed in the egress interface of all routers in the path (including the first router).

```
if (srcResource.address/24 == 143.54.47.0/24 and
    dstResource.address/24 != 143.54.47.0/24 and
    dstResource.port == 80 and
    dstResource.protocol == TCP)
{
  p1 = new NetworkPolicy();
  p1.addCondition(startTime,">=",schedule.startTime);
  p1.addCondition(endTime,"<=",schedule.endTime);
  p1.addCondition(srcAddress,"==",srcResource.address);
  p1.addCondition(dstAddress,"==",dstResource.address);
  p1.addCondition(dstPort,"==",dstResource.port);
  p1.addCondition(dstProtocol,"==","tcp");
  p1.addAction(DSCP,2);
  inPEPs = select pep
      .within[srcResource.address, 143.54.47.1]
      .direction["in"]
    from
      device.type["DiffServDevice"];
  inPEPs[0].deployPolicy(p1);
  p2 = new NetworkPolicy();
  p2.addCondition(startTime,">=",schedule.startTime);
  p2.addCondition(endTime,"<=",schedule.endTime);
  p2.addCondition(DSCP,2);
  p2.addAction(bandwith,requiredQoS.requiredBandwidth);
  outPEPs = select pep
      .within[srcResource.address, 143.54.47.1]
      .direction["out"]
    from
      device.type["DiffServDevice"];
  outPEPs.deployPolicy(p2);
```

Listing 5. Translation rule example with policy deployment

4. System Prototype

The translation architecture was implemented in a Web-based prototype where the grid administrator can define grid policies, grid class of services and the location of the several translation engines dispersed through the network administrative domains that compose the grid. We provide in the prototype also an interface to allow the network administrator to create translation rules an configure the translation engine, for instance, specify which grid toolkit is used. The prototype is a module of the QoS-Aware Management Environment, a Web-based network management system developed in the PHP language at the Federal University of Rio Grande do Sul.

The Web browser snapshot in figure 3 shows the translation rule editor. In order to define translation rules the network administrator must only select a list of pre-defined language constructors in the user interface. This interface act as a wizard to allow the definition of the translation rule without previous knowledge of the mapping rule language syntax.

The information model used in the implementation is presented in the UML model of figure 4. We extended the core classes of PCIM schema and implemented policies, policy conditions and policy actions for grid and network management. We modelled and implemented also a set of classes to deal with the translation process, regarding grid class of services, grid communication pairs and the grid toolkit interface. Using this model it is possible to provide further extensions to new grid policies to deal, for instance, with network security and also to support other toolkits implementations.

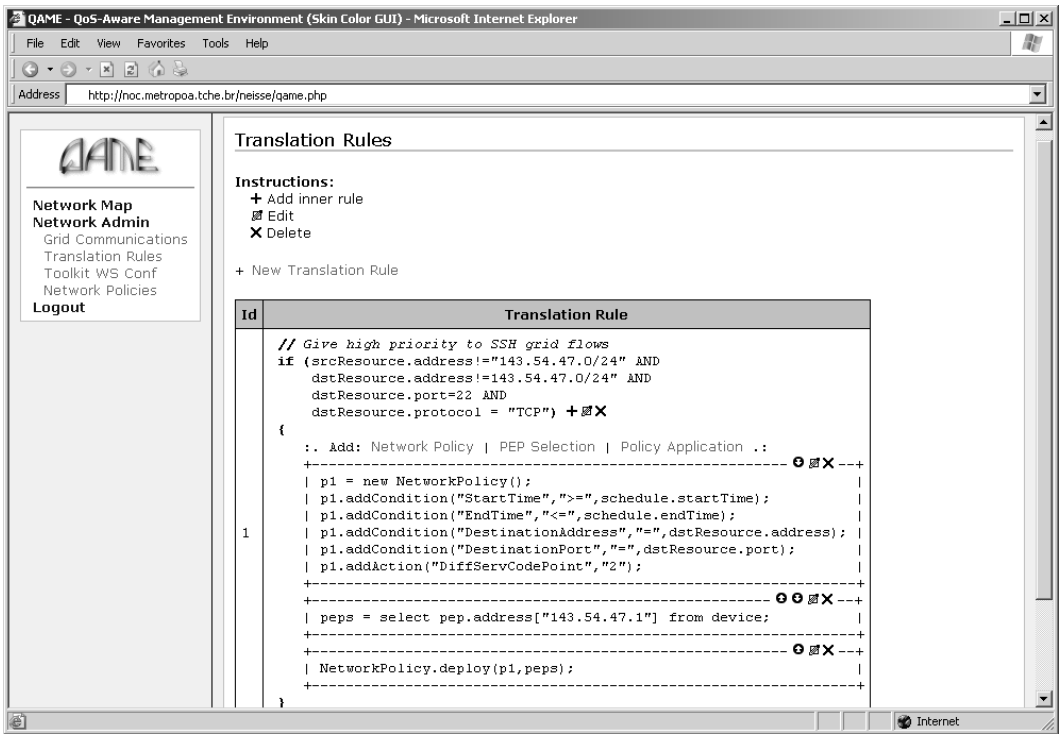

Figure 3. Grid Translation Rule Editor

The distributed operation of the prototype is presented in figure 5. The figures shows the grid and network administrator interaction and, policy and translation rules repositories and the mapping engine location. The prototype stores the grid and network policies in a LDAP repository following our information model schema derived from the IETF Policy Core Information Model (PCIMe) [Moore 2003]. Each network administrator defines the corresponding mapping rules in he or she domain considering the network architecture and topology, the grid resources, and the local network policies.

After the definition of the grid policies and translation rules, the translation engines, distributed over the network administrative domains, are able to create the network policies. Each network administrative domain has a local network policy repository and must have the translation engine running to a proper configuration of

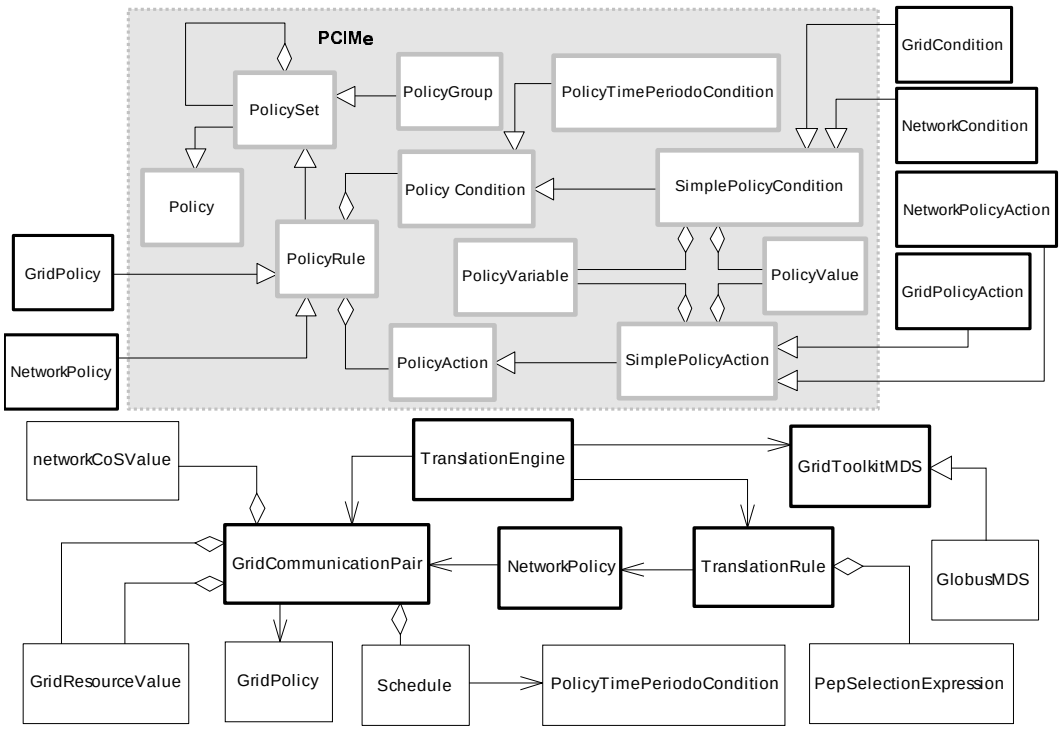

Figure 4. Information model for policy translation (UML)

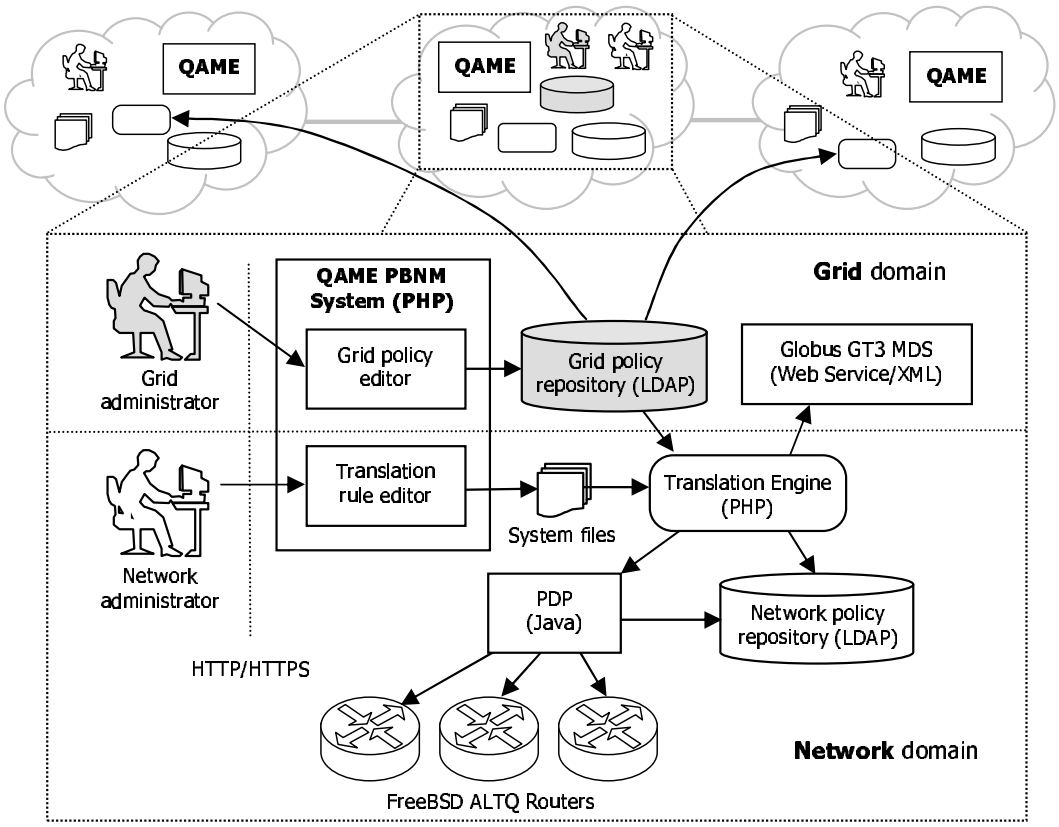

Figure 5. Prototype implementation

the network to support the grid communication. The extra information required by the translation engine to create the communication pair objects, for instance, resources address and protocols, are queried in the Monitoring and Discovery Service (MDS) of a Globus toolkit version (GT3), implemented in our prototype as a Web Service.

5. Conclusions and Future Work

We presented in this paper an architecture that translates grid policies to network policies through a translation engine. In order to operate, such engine uses translation rules defined by network administrators locate along the administrative domains of a grid. Since each domain has its own network administrator that defines local translation rules, which are different than the rules defined by other administrator, a single grid policy is potentially translated to different network policies in each administrative domain. Although the translation rules are flexible, this forces the network administrators to learn a new language to define more adequate translations. We believe that visual wizards would ease the definition of translation rules. We also argued that grid policies are supposed to be defined by grid administrators, instead of network administrators. To do so, we presented a set of grid policy definition elements to be used by grid administrator in defining grid policies. The grid policies defined with such elements, compared to the grid policy definition languages found today, express richer rules, because such grid policies support the concept of proxies and explicitly express network QoS requirements used in the translation process.

The translation engine, besides receiving grid policies and translation rules, needs to interact with both grid toolkit and policy-based network management system in order to build and deploy the expected network policies. We presented a system prototype that uses the QAME management system and the Globus toolkit. Concerning Globus, just a subset of the grid policies can be effectively used, since the toolkit does not support the grid policy language presented. The current communication between QAME, Globus and the translation engine is achieved using Web Services.

Currently, we are investigating the use of more sophisticated user graphical interfaces in order to reduce the complexity of defining grid policies and translation rules. Also, bandwidth consumption investigation is required, although performance and scalability observations of the translation engine seems to be more critical, primarily concerning the number of grid rules, levels of rule nesting, and the number of translation rules defined by the network administrator.

References

Globus, “The Globus project”, 2003, http://www.globus.org.

M. Steen, P. Homburg, and A. S. Tanenbaum, “Globe: A wide-area distributed system”, *IEEE Concurrency*, pp. 70–78, Jan. 1999.

AccessGrid, “The Access grid (ag) user documentation”, 2003, http://www-fp.mcs.anl.gov/fl/ accessgrid.

I. Foster, M. Fidler, A. Roy, V. Sander, and L. Winkler, “End-to-end quality of service for high-end applications”, *IEEE Computer Communications Special Issue on Network Support for Grid Computing*, 2002.

B. Sundaram, C. Nebergall, and S. Tuecke, “Policy specification and restricted delegation in globus proxies”, in *SuperComputing 2000*.

B. Sundaram and B. M. Chapman, “Xml-based policy framework for usage policy management in grids”, in *Grid’02 3rd International workshop on Grid Computing*, 2002.

D. C. Verma, S. Sahu, S. B. Calo, M. Beigi, and I. Chang, “A policy service for grid computing”, in *Grid Computing - GRID 2002, Third International Workshop*, ser. Lecture Notes in Computer Science, pp. 243–255, Nov. 2002.

K. Yang, A. Galis, and C. Todd, “Policy-based active grid management architecture”, *10th IEEE International Conference on Networks*, 2002.

A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, A. Huynh, M. Carlson, J. Perry, and S. Waldbusser, “Terminology for policy-based management”, Request for Comments: 3198, IETF, Nov. 2001.

V. Sander, W. Adamson, I. Foster, and R. Alain, “End-to-end provision of policy information for network qos”, *10th IEEE International Symposium on High Performance Distributed Computing*, 2001.

R. Yavatkar, D. Pendarakis, and R. Guerin, “A framework for policy-based admission control”, Request for Comments: 2753, IETF, Jan. 2000.

N. Damianou, N. Dulay, E. Lupu, and M. Sloman, “The ponder policy specification language”, in *Policies for Distributed Systems and Networks*, ser. Lecture Notes in Computer Science, vol. 1995. LNCS, pp. 18–38, Jan. 2001.

J. Lobo, R. Bhatia, and S. Naqvi, “A policy description language”, in *AAAI/IAAI*, pp. 291–298, 1999.

M. Sloman and J. Moffett, “Domain management for distributed systems”, in *Integrated Network Management I*, pp. 505–516, 1989.

M. B. Ceccon, L. Z. Granville, M. J. B. Almeida, and L. M. R. Tarouco, “Definition and visualization of dynamic domains in network management environments”. Lecture Notes in Computer Science, vol. 2662. LNCS, pp. 828–838, 2003.

B. Moore, “Policy core information model (pcim) extensions”, Request for Comments: 3460, Updates RFC 3060, IETF, Jan. 2003.