

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL  
INSTITUTO DE INFORMÁTICA  
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RICARDO NEISSE

**Um Modelo Hierárquico Baseado em  
Políticas para o Gerenciamento Integrado  
de Redes de Computadores e Grids  
Computacionais**

Dissertação apresentada como requisito parcial  
para a obtenção do grau de  
Mestre em Ciência da Computação

Profa. Dra. Maria Janilce Bosquioli  
Almeida  
Orientadora

Porto Alegre, novembro de 2004

## CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Neisse, Ricardo

Um Modelo Hierárquico Baseado em Políticas para o Gerenciamento Integrado de Redes de Computadores e Grids Computacionais / Ricardo Neisse. – Porto Alegre: PPGC da UFRGS, 2004.

69 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR-RS, 2004. Orientadora: Maria Janilce Bosquioli Almeida.

1. Computação em grid. 2. Gerenciamento de grids. 3. Gerenciamento de redes. 4. Gerenciamento baseado em políticas. 5. Hierarquias de políticas. I. Almeida, Maria Janilce Bosquioli. II. Título.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Reitor: Prof. José Carlos Ferraz Hennemann

Vice-Reitor: Prof. Pedro Cezar Dutra Fonseca

Pró-Reitora Adjunta de Pós-Graduação: Prof<sup>a</sup>. Valquiria Link Bassani

Diretor do Instituto de Informática: Prof. Philippe Olivier Alexandre Navaux

Coordenador do PPGC: Prof. Carlos Alberto Heuser

Bibliotecária-Chefe do Instituto de Informática: Beatriz Regina Bastos Haro

*“Se A é o sucesso, então é igual a X mais Y mais Z.  
O trabalho é X; Y é o lazer; e Z é manter a boca fechada.”*

— ALBERT EINSTEIN

## AGRADECIMENTOS

Em primeiro lugar gostaria de agradecer à minha família e amigos, que me apoiaram de fato não somente no desenvolvimento deste trabalho, mas em tudo que aconteceu durante essa etapa da minha vida. A minha mãe Zelita e minha irmã Fernanda, que não me deixaram na mão em nenhum momento. Aos meus grandes amigos Otávio Cavalett, José Carlos Bortolotti II, Marcelo Thiesen e Cláudia Cavalett, pessoas fantásticas com as quais também sempre pude contar.

Agradeço à minha orientadora, Profa. Maria Janilce Bosquiroli Almeida, sem a qual esse trabalho não seria possível. Não poderia faltar um agradecimentos muitíssimo especial ao Prof. Lisandro Zambenedetti Granville, que além de um excelente professor foi um exemplo de competência e dedicação que seguirei durante toda a minha vida. Gostaria de agradecer também aos professores Liane Margarida Rockenbach Tarouco, Juergen Rochol e João César Netto, do Grupo de Redes de Computadores, pelo apoio e ensinamentos recebidos durante todo o desenvolvimento deste trabalho, dentro e fora da sala de aula.

Agradecimentos especiais aos meus colegas de mestrado e de festas Leandro Vaguetti, Evandro Della Vecchia Pereira, Tiago Fioreze e Michelle Denise Leonhardt. Gostaria de agradecer também aos demais colegas e professores do Instituto de Informática da UFRGS e ex-colegas e ex-professores da Universidade de Passo Fundo. A todos as pessoas que contribuíram de alguma forma para este trabalho meu sincero MUITO OBRIGADO!

## SUMÁRIO

<b>LISTA DE ABREVIATURAS E SIGLAS</b> . . . . .	6
<b>LISTA DE FIGURAS</b> . . . . .	8
<b>LISTA DE TABELAS</b> . . . . .	9
<b>LISTA DE CÓDIGOS</b> . . . . .	10
<b>RESUMO</b> . . . . .	11
<b>ABSTRACT</b> . . . . .	12
<b>1 INTRODUÇÃO</b> . . . . .	13
<b>2 ESTADO DA ARTE</b> . . . . .	17
2.1 Toolkits para Grids . . . . .	17
2.2 Gerenciamento de Grids Baseado em Políticas . . . . .	21
2.3 Análise Crítica e Proposta . . . . .	24
<b>3 SOLUÇÃO PARA TRADUÇÃO DE POLÍTICAS DE GRID EM POLÍTICAS DE REDE</b> . . . . .	26
3.1 Arquitetura de Gerenciamento Baseado em Políticas do IETF . . . . .	28
3.2 Linguagem para Definição de Políticas de Grid . . . . .	30
3.3 Linguagem para Definição de Políticas de Rede . . . . .	35
3.4 Linguagem para Tradução de Políticas de Grid em Políticas de Rede . . . . .	36
3.5 Modelo para Suporte à Tradução de Políticas de Grid em Políticas de Rede . . . . .	45
<b>4 IMPLEMENTAÇÃO DO PROTÓTIPO</b> . . . . .	48
4.1 Detalhes de implementação . . . . .	48
4.2 Modelo de informações e interfaces . . . . .	49
4.3 Módulo de Gerenciamento de Grid . . . . .	51
4.4 Módulo de Gerenciamento Integrado de Grid e Rede . . . . .	54
<b>5 CONCLUSÕES E TRABALHOS FUTUROS</b> . . . . .	58
<b>REFERÊNCIAS</b> . . . . .	60
<b>ANEXO A ARTIGO PUBLICADO</b> . . . . .	64

## LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
BNF	Backus Naur Form
CORBA	Common Object Request Broker Architecture
COPS	Common Open Policy Service
COPS-PR	COPS Usage for Policy Provisioning
CoS	Class of Service
DS	Differentiated Services
DSCP	Differentiated Services Code Point
DMTF	Distributed Management Task Force
GARA	Globus Architecture for Reservation and Allocation
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol over Secure Socket Layer
IETF	Internet Engineering Task Force
LDAP	Lightweight Directory Access Protocol
MIB	Management Information Base
MPI	Message Passing Interface
MDS	Monitoring and Discovery Service
NFS	Network File System
OSGA	Open Services Grid Architecture
PBNM	Policy-Based Network Management
PEP	Policy Enforcement Point
PDP	Policy Decision Point
PCIM	Policy Core Information Model
PCIMe	Policy Core Information Model Extensions
PQIM	Policy QoS Information Model
QoS	Quality of Service

SNMP	Simple Network Management Protocol
SOAP	Simple Object Access Protocol
UDP	User Datagram Protocol
UML	Unified Modeling Language
WS	Web Service
XML	eXtensible Markup Language

## LISTA DE FIGURAS

Figura 1.1:	Exemplo de grid . . . . .	14
Figura 2.1:	Protocolos para operação e gerenciamento do grid . . . . .	18
Figura 2.2:	Gerenciamento de recursos GARA com reservas DiffServ . . . . .	20
Figura 2.3:	Acesso à um recurso usando um proxy intermediário . . . . .	21
Figura 2.4:	Sinalização dos bandwidth brokers para configuração de QoS . . . . .	22
Figura 2.5:	Ferramenta EZGrid para definição de políticas de grid . . . . .	24
Figura 2.6:	Visão geral do cenário de gerenciamento de grid e rede . . . . .	24
Figura 3.1:	Hierarquia para tradução de políticas . . . . .	27
Figura 3.2:	Níveis de abstração das políticas de grid e rede . . . . .	27
Figura 3.3:	Modelo de gerenciamento baseado em políticas do IETF . . . . .	29
Figura 3.4:	Diagrama de classes do modelo de informações do PCIME . . . . .	30
Figura 3.5:	Caminhos de reserva de QoS na rede . . . . .	39
Figura 3.6:	Modelo de tradução de políticas . . . . .	41
Figura 3.7:	Modelo de informação estendido . . . . .	44
Figura 3.8:	Modelo de tradução de políticas . . . . .	46
Figura 4.1:	Implementação do protótipo . . . . .	49
Figura 4.2:	Modelo UML das condições e ações das políticas de rede . . . . .	50
Figura 4.3:	Implementação do protótipo . . . . .	50
Figura 4.4:	Modelo UML das condições e ações das políticas de grid . . . . .	51
Figura 4.5:	Modelo UML das classes responsáveis pela tradução . . . . .	51
Figura 4.6:	Editor de políticas do grid . . . . .	52
Figura 4.7:	Editor de classes de serviço do grid . . . . .	53
Figura 4.8:	Configuração dos mecanismos de tradução . . . . .	53
Figura 4.9:	Visualização dos pares de comunicação do grid . . . . .	54
Figura 4.10:	Configuração do toolkit no mecanismo de tradução . . . . .	55
Figura 4.11:	Editor de regras de tradução . . . . .	56
Figura 4.12:	Visualização das políticas de rede criadas . . . . .	57



## LISTA DE TABELAS

Tabela 3.1:	Pares de comunicação resolvidos da política . . . . .	37
Tabela 3.2:	Pares de comunicação resolvidos da política . . . . .	39

## LISTA DE CÓDIGOS

Código 2.1	Política de Grid . . . . .	21
Código 2.2	Política de Grid para Proxies . . . . .	23
Código 3.1	BNF simplificada das políticas de grid e rede . . . . .	31
Código 3.2	Exemplo de políticas de grid . . . . .	32
Código 3.3	Regras de políticas aninhadas . . . . .	33
Código 3.4	Políticas de grid com domínios . . . . .	34
Código 3.5	Exemplo de política de rede . . . . .	35
Código 3.6	Política de grid com resolução dos pares de comunicação . . . . .	37
Código 3.7	Política de grid . . . . .	38
Código 3.8	Exemplo de políticas de grid . . . . .	38
Código 3.9	Política de grid com domínios . . . . .	40
Código 3.10	BNF simplificada das regras de tradução . . . . .	41
Código 3.11	Regra de tradução . . . . .	42
Código 3.12	Regra de tradução . . . . .	42
Código 3.13	Algoritmo para execução das regras de tradução . . . . .	43
Código 3.14	BNF da linguagem de criação de domínios dinâmicos . . . . .	44
Código 3.15	Exemplos de expressões de seleção de PEPs . . . . .	44
Código 3.16	Regra de tradução com aplicação da política de rede . . . . .	45

## RESUMO

O gerenciamento dos recursos computacionais em um *grid* é necessário para permitir uma operação adequada dos serviços oferecidos aos usuários. Além disso, *grids* requerem que a infra-estrutura de rede subjacente também esteja configurada adequadamente, para que a comunicação entre os usuários e recursos do *grid* seja possível. Atualmente, os gerenciamentos de rede e de *grid* são executados por diferentes entidades administrativas, usando ferramentas diferentes. As comunicações do *grid* requerem que configurações sejam feitas na rede, por exemplo, para reserva de recursos, mas esses requisitos somente são atendidos quando os administradores de *grid* contatam manualmente os administradores de rede para que as configurações correspondentes sejam executadas. Nesse cenário, um gerenciamento integrado de *grid* e de rede facilitaria o processo.

Esta dissertação de mestrado propõe um modelo hierárquico de gerenciamento baseado em políticas onde duas linguagens são usadas para integração do gerenciamento de *grids* e de redes. Primeiramente, definiu-se uma linguagem hipotética, inspirada no modelo de gerenciamento baseado em políticas do IETF, onde são especificados elementos para representar as políticas de *grid* e de rede. A segunda linguagem é usada para definir regras de tradução, que definem como as políticas de rede são geradas a partir das políticas de *grid* em cada domínio administrativo de rede componente do *grid*.

Além do modelo de gerenciamento integrado, este trabalho apresenta também um protótipo, desenvolvido como módulo do sistema de gerenciamento de redes QAME, que permite aos administradores do *grid* a definição de políticas e classes de serviço de *grid*, e aos administradores da rede definir as regras de tradução. O protótipo é acessível através da Web e foi desenvolvido usando a linguagem PHP, o serviço de diretório LDAP como repositório das políticas, e a tecnologia de Web Services para comunicação entre os elementos do protótipo.

Como resultados constatou-se que o modelo de tradução de políticas proposto permite automatizar o gerenciamento da infra-estrutura de *grid* e rede. Além disso, a solução apresentada fornece novas facilidades de gerenciamento em comparação as soluções de gerenciamento de *grid* baseadas em políticas encontradas na literatura. A implementação do protótipo do modelo junto ao sistema QAME permitiu que o gerenciamento do *grid* e da rede fosse realizado de maneira integrada usando uma hierarquia de tradução de políticas.

**Palavras-chave:** Computação em grid, gerenciamento de grids, gerenciamento de redes, gerenciamento baseado em políticas, hierarquias de políticas.

## **An Hierarchical Policy-based Model for Integrated Management of Computer Networks and Computational Grids**

### **ABSTRACT**

The management of the resources in a computational grid is required in order to allow a proper operation of the services offered to the users. Besides, grids require the underlying network infrastructure to be properly configured in order to have appropriate communications among the grids nodes. The management of networks and the management of grids are currently executed by different tools, operated by different administrative personnel. Eventually, the grid communication requirements will need corresponding support from the network management tools, but such requirements are fulfilled only when grid administrators manually asks network administrators for corresponding configurations. In this scenario, an integrated management of the grid and network resources will turn the process easier.

This master dissertation proposes an hierarchical management model based on policies using two languages to integrate the management of grids and networks. Firstly, an hypothetical language was defined, based in the policy based network management model from IETF, where elements were specified to represent network and grid policies. The second language is used to define translation rules, which specify how the network policies are generated from the grid policies in each network administrative domain participating in the grid.

In addition to the integrated management model, this work presents also a prototype, developed as a module of the network management system QAME, which allows the grid administrators the definition of grid policies and class of services, and the network administrators the definition of the translation rules. The prototype is Web based and was developed using the PHP language, the LDAP directory service as the policy repository, and Web Services as the communication technology between the prototype elements.

As results we verified that the policy translation model allow the automation of some grid and network management tasks. Besides, the solution presented provides new management facilities in comparison to the grid policy based management solutions found in the literature. The prototype implementation as a QAME module allowed the integrated management of the grid and network infrastructure using a policy translation hierarchy.

**Palavras-chave:** grid computing, grid management, network management, policy-based management, policies hierarchies.

# 1 INTRODUÇÃO

*Grids* são infra-estruturas de computação distribuída que permitem o compartilhamento de recursos computacionais entre usuários conectados através de uma rede de computadores (FOSTER; KESSELMAN, 1999). Recursos podem ser: processamento, memória, disco, banda da rede ou dispositivos especializados (e.g. telescópio, microscópio eletrônico, equipamento de diagnóstico médico, entre outros). Existem projetos de pesquisa em diversas áreas do conhecimento usando *grids*, e o cenário não é diferente no mundo dos negócios, onde soluções usando *grids* estão sendo exploradas por grandes empresas como IBM (2004), Sun (2004) e Oracle (2004).

O benefício principal alcançado com o uso de *grids* é o compartilhamento de recursos computacionais localizados em domínios administrativos diferentes. Através de um *grid* é possível que um grande número de recursos computacionais distribuídos sejam utilizados em conjunto, o que permite, por exemplo, que instituições que possuam recursos computacionais ociosos colaborem com outras instituições na realização de uma tarefa que necessite de muito poder computacional ou de algum dispositivo especializado. Essa colaboração requer, entretanto, o estabelecimento de acordos entre as instituições interessadas e definições de políticas de compartilhamento e uso dos recursos computacionais. Essas políticas levam em conta os benefícios econômicos obtidos pelas instituições participantes do *grid* no uso dos recursos.

Aplicações típicas encontradas em *grids* são: computação de alto desempenho, compartilhamento de dados, controle remoto de instrumentos, colaboração interativa e simulações. Geralmente, aplicações que exigem recursos computacionais poderosos ou muito especializados, que são na maioria dos casos caros, obtêm benefícios no uso de infra-estruturas de *grids*. Analisando-se essas aplicações é visível que um conjunto significativo delas apresenta um ou mais requisitos especiais de rede como grande largura de banda disponível, sensibilidade ao atraso (latência), sensibilidade à variação do atraso (*jitter*) e suporte para comunicação *multicast*.

O Nasa Information Power Grid (IPG) (JOHNSTON; GANNON; NITZBERG, 1999), o European Union DataGrid (HOSCHEK et al., 2000) e o Teragrid (NSF, 2004) são exemplos de *grids* atualmente em produção. O Teragrid, por exemplo, é um *grid* localizado nos Estados Unidos que possui aproximadamente 11 teraflops de poder de processamento e mais de 800 terabytes de espaço de armazenamento. Grande parte desses recursos são fornecidos por *clusters* baseados em Linux e servidores de armazenamento distribuídos em cinco instituições geograficamente dispersas, que são: o Centro Nacional para Aplicações de Supercomputação (NCSA), o Centro de Supercomputação de San Diego (SDSC), o Laboratório Nacional de Argonne (ANL), o Instituto de Tecnologia da Califórnia (CALTECH) e o Centro de Supercomputação de Pittsburgh (PSC). Além de processamento e armazenamento, o Teragrid disponibiliza também conexões de rede

de alta velocidade, ambientes de visualização de alta resolução e suporte de softwares específicos para computação em *grid*. A figura 1.1 mostra, a título de exemplo, uma visão geral da estrutura do Teragrid, onde são apresentados os recursos compartilhados em cada instituição e as conexões principais de rede (*backbone*) entre elas.

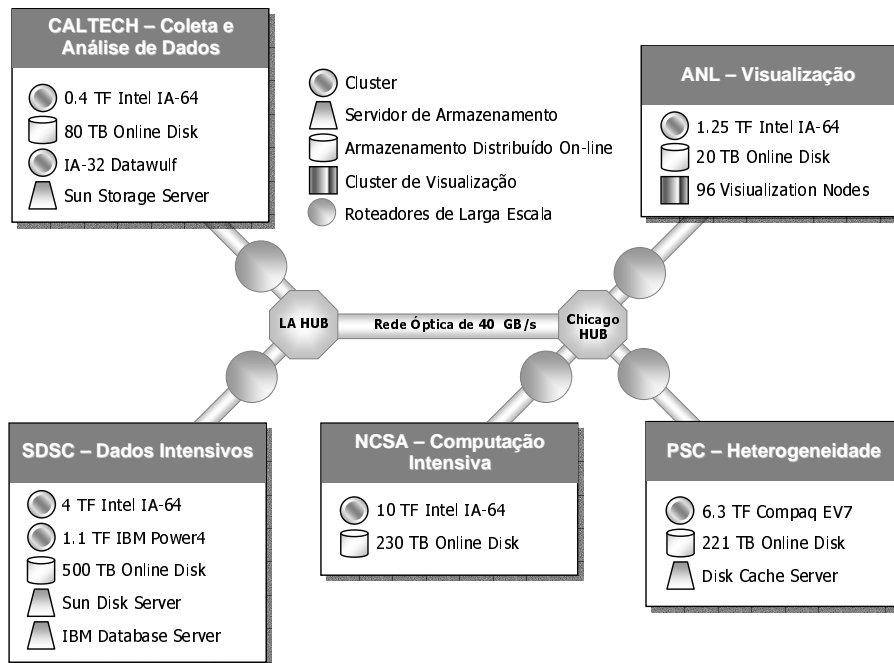


Figura 1.1: Exemplo de grid

Em um *grid*, o conjunto de recursos em cada sítio é denominado um nodo do *grid*, que pode possuir seus próprios administradores e usuários. O administrador do nodo do *grid* é responsável pela instalação, configuração e pelo gerenciamento dos recursos do nodo que participa do *grid*. Do ponto de vista dos usuários, o *grid* é visto com menor complexidade: ele é visto simplesmente como uma coleção de recursos computacionais acessados usando algum software específico (FERREIRA et al., 2003).

*Grids* são tipicamente distribuídos através de vários domínios administrativos de rede diferentes, como pode-se perceber pela configuração do Teragrid. Portanto, para manter o *grid* operando, é necessário que exista um gerenciamento coordenado dos recursos em todos os domínios participantes. Nesse gerenciamento, destacam-se duas figuras administrativas que precisam interagir: o administrador do *grid* e o administrador da rede. O administrador do *grid* é o responsável pelo gerenciamento dos recursos compartilhados no *grid* (e.g. *clusters* e servidores de armazenamento), pelo gerenciamento dos usuários do *grid* e permissões de acesso. A função do administrador da rede, por outro lado, é a de manter a rede operando para que o acesso aos recursos do *grid* seja possível. É importante destacar que administração do *grid* pode ser centralizada ou distribuída, dependendo do software usado e das políticas de operação, mas a administração da rede em cada domínio é realizada de maneira independente em relação à administração do *grid*.

Devido aos requisitos de rede das aplicações e a alta distribuição dos recursos, a implementação e o gerenciamento de uma infra-estrutura de *grid* não é uma tarefa trivial. Por esses motivos, foram desenvolvidas bibliotecas de software, chamadas de *toolkits*, que fornecem as funcionalidades e serviços básicos de gerenciamento para a manutenção de *grids* (GLOBUS, 2004) (NATRAJAN et al., 2002) (STEEN; HOMBURG; TANENBAUM, 1999) (ACCESSGRID, 2004). Os *toolkits* possuem facilidades para que

o administrador do *grid* gerencie os recursos e mantenha os serviços críticos operando de maneira adequada. Grande parte dessas bibliotecas de software são implementadas usando *middlewares* (e.g. CORBA), onde os recursos são abstraídos e gerenciados através de objetos distribuídos (STEEN; HOMBURG; TANENBAUM, 1999). Em algumas implementações mais recentes, a interface de acesso e gerenciamento do *grid* é baseada em serviços, implementados através da arquitetura de Web Services (GLOBUS, 2004).

Independentemente da tecnologia de *middleware* usada na implementação do *toolkit*, é um requisito para o funcionamento do *grid* que a infra-estrutura de rede também seja gerenciada. Como visto anteriormente, isso é importante pois é através da rede que os usuários do *grid* acessam os recursos compartilhados e, caso a rede esteja muito lenta, ou inoperante, o acesso aos recursos compartilhados será comprometido. A configuração da rede para operação do *grid* inclui, por exemplo, a reserva de recursos de rede (e.g. largura de banda), que geralmente é feita pelos administradores da rede em um domínio através de arquiteturas de fornecimento de QoS como DiffServ (BLAKE et al., 1998) ou IntServ (BRADEN; CLARK; SHENKER, 1994). Nesse caso, o ideal seria que os *toolkits* interagissem diretamente com as infra-estruturas de gerenciamento de redes, e realizassem as configurações de rede necessárias, mas o que se encontra na prática são sistemas independentes, onde o administrador do *grid* e o administrador da rede precisam interagir de maneira não automatizada para que o suporte de comunicação requerido para a operação do *grid* seja adequadamente configurado. Apesar dos *toolkits* fornecerem suporte para o gerenciamento dos recursos do *grid*, o suporte para o gerenciamento integrado da infra-estrutura de rede ainda é muito pouco explorado.

Um requisito de configuração da rede necessário em um *grid* de conferência, implementado, por exemplo, usando o *toolkit* AccessGrid (ACCESSGRID, 2004), é habilitar na rede o suporte a transmissões *multicast* de áudio e vídeo entre os nodos do *grid* com a garantia de uma determinada largura de banda, com atraso e variação do atraso pequeno e possivelmente com uma baixa probabilidade de descarte de pacotes. Essa configuração deve ser realizada em todos os domínios administrativos participantes do *grid*, para garantir que as transmissões de áudio e vídeo das conferências sejam realizadas com sucesso. Não é comum o administrador do *grid* ter direitos para realizar configurações de QoS nos dispositivos da rede através dos diversos domínios administrativos que compõem um *grid* e, mesmo que ele possuísse esses direitos, seriam necessárias ferramentas específicas para isso, pois em geral os *toolkits* de *grid* não fornecem facilidades para resolver esse tipo de situação. Observando esse cenário é possível constatar que, todas as vezes que um requisito do *grid* que implique em uma nova configuração na infra-estrutura de rede for alterado, uma coordenação entre os administradores do *grid* e da rede é necessária.

Na tentativa de resolver os problemas de integração entre gerenciamento de *grid* e gerenciamento de rede este trabalho propõe um modelo de gerenciamento baseado em políticas e regras de tradução. No modelo proposto, as políticas de gerenciamento da rede, necessárias em cada um dos domínios administrativos, são derivadas a partir das políticas de gerenciamento do *grid*. A partir desse modelo de gerenciamento foi proposta uma arquitetura para dar suporte a esse modelo. A arquitetura possui um mecanismo de derivação de políticas que utiliza regras de tradução, definidas pelos administradores da rede de cada domínio participante do *grid*.

A implementação da arquitetura proposta neste trabalho foi resultou em um sistema de gerenciamento baseado em políticas disponível através da Web, onde o administrador do *grid* define as políticas do *grid*, e o administrador da rede em cada domínio especifica

as regras de tradução. A partir das regras de tradução e políticas do *grid* o sistema gera automaticamente as políticas de rede, que especificam o suporte de comunicação necessário para operação do *grid*.

Esta dissertação está organizada como segue. O capítulo 2 apresenta os trabalhos relacionados onde detalha-se o suporte de gerenciamento fornecido pelos *toolkits* e também aborda-se as implicações relacionadas com o gerenciamento integrado do *grid* e da rede. O capítulo 3 apresenta o modelo hierárquico de gerenciamento proposto e, em seguida, o capítulo 4 mostra o protótipo implementado baseado na arquitetura. O capítulo 5 valida a arquitetura mostrando um estudo de caso e o protótipo aplicado para gerenciar um domínio de rede participante de um *grid*. Finalmente, esta dissertação é finalizada no capítulo 6, onde são apresentadas as conclusões e os trabalhos futuros.



## 2 ESTADO DA ARTE

O gerenciamento dos recursos de um *grid* não é uma tarefa trivial, uma vez que os recursos podem estar localizados através de vários domínios administrativos diferentes. Por exemplo, o *cluster* de um *grid* pode estar localizado em uma empresa, enquanto os servidores de armazenamento podem estar localizados em uma universidade. Entretanto, ambos os recursos (processamento e armazenamento), ainda que pertencendo ao mesmo *grid*, são administrados em domínios diferentes. Nessa situação, cada recurso do *grid* segue as políticas de operação do domínio onde está localizado.

Portanto, para que o *grid* opere adequadamente, é necessário um gerenciamento distribuído coordenado dos recursos em todos os domínios que compõem o *grid*. Este capítulo apresenta as soluções mais significativas para o gerenciamento de infra-estruturas de *grid*: *toolkits* e ferramentas baseadas em políticas. Este capítulo é encerrado com uma análise crítica das soluções com o objetivo de motivar a solução de gerenciamento integrado proposta nesta dissertação.

### 2.1 Toolkits para Grids

Tarefas típicas de gerenciamento que precisam ser coordenadas em um ambiente distribuído de *grid* são, por exemplo, autenticação de usuários e escalonamento dos recursos. Considerando que a maioria das infra-estruturas de *grid* precisam de um suporte de gerenciamento comum para coordenar essas atividades, bibliotecas de software chamadas *toolkits* foram desenvolvidas. Os *toolkits* fornecem os serviços básicos e tentam minimizar o esforço inicial necessário para implantar e gerenciar um *grid*. Exemplos de *toolkits* disponíveis são: Globus (GLOBUS, 2004), Legion (NATRAJAN et al., 2002), Globe (STEEN; HOMBURG; TANENBAUM, 1999) e AccessGrid (ACCESSGRID, 2004).

Na implementação dos *toolkits* citados, destacam-se as tecnologias de objetos distribuídos, Web Services e gerenciamento baseado na Web. O *toolkit* Globe, por exemplo, implementa os serviços de gerenciamento em objetos distribuídos que se comunicam uns com os outros através da arquitetura CORBA (*Common Object Request Broker Architecture*) (OMG, 2004). O *toolkit* Globus baseia-se em uma arquitetura aberta de serviços integrados denominada Open Grid Services Architecture (OSGA) (FOSTER et al., 2002). Já o *toolkit* AccessGrid, por sua vez, fornece ferramentas para gerenciar os recursos do *grid* através de páginas HTML geradas dinamicamente. Nota-se, considerando os exemplos citados, que não existem tecnologias e interfaces de comunicação padronizadas de gerenciamento para *grids*.

Além disso, pode-se classificar os *toolkits* em *específicos* para um tipo de aplicação ou *genéricos* para qualquer tipo de aplicação. O AccessGrid é, por exemplo, um

*toolkit* específico, pois fornece aplicações e serviços de gerenciamento padronizados para implementação somente de *grids* de conferência. Já o *toolkit* Globus é genérico, pois disponibiliza, além dos serviços básicos de gerenciamento, uma biblioteca com padrões de projeto e generalizações para facilitar o desenvolvimento de qualquer tipo de aplicação para um ambiente de *grid*. *Toolkits* que fornecem suporte para o desenvolvimento de extensões e aplicações genéricas caracterizam-se por possuir interfaces de comunicação entre seus elementos bem definidas e documentadas.

Assim como o suporte para o desenvolvimento de aplicações, os protocolos usados em uma infra-estrutura de *grid* também podem ser divididos em dois grupos: protocolos para o gerenciamento do *grid* e protocolos para operação do *grid*. Por exemplo, no *toolkit* AccessGrid, o gerenciamento de conferências (usuários e recursos) é feito através de páginas HTML geradas dinamicamente (sobre HTTP ou HTTPS), mas as transmissões de áudio e vídeo entre os nodos é feita usando *multicast* UDP. No *toolkit* Globus, a arquitetura de gerenciamento é implementada através de Web Services, mas a transferência de arquivos entre os elementos do *grid* é feita com o protocolo GRIDFTP (uma extensão do protocolo FTP criada para operar em um ambiente distribuído de *grid*). A figura 2.1 apresenta um exemplo de uma possível comunicação em um *grid* implementado com o *toolkit* Globus, onde usuários, administradores e recursos são gerenciados através de Web Services, e as transferências de arquivos são feitas através do protocolo GRIDFTP.

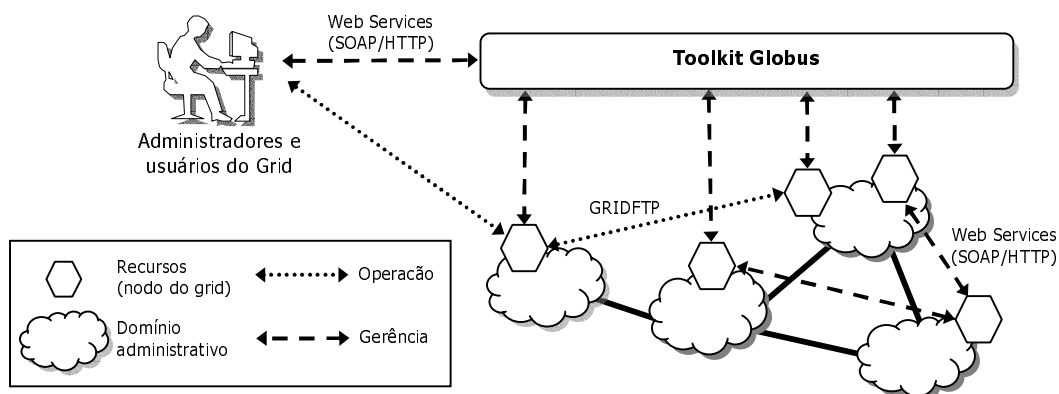


Figura 2.1: Protocolos para operação e gerenciamento do grid

Com base nesse cenário, pode-se dividir o suporte de comunicação necessário em um *grid* em dois grupos distintos: (A) o suporte de comunicação para o gerenciamento dos recursos e (B) o suporte de comunicação para execução de aplicações. Esses dois grupos podem usar os mesmos protocolos ou protocolos diferentes, como foi apresentado no exemplo da figura 2.1.

Embora os protocolos possam ser semelhantes, os requisitos de rede para o gerenciamento (A) são definidos no momento da instalação e configuração da infra-estrutura do *grid*, e somente são alterados quando o conjunto de recursos compartilhados no *grid* é alterado. Por exemplo, um *grid* que é formado por 10 *clusters* tende a usar sempre os mesmos protocolos e banda da rede para tarefas de gerenciamento comuns como monitoração e controle de acesso. Os requisitos de rede são alterados quando um novo *cluster* é incluído no *grid* ou algum software de gerenciamento é modificado. Mudanças na infra-estrutura ou no software implicam então em mudanças nos requisitos de gerenciamento (A).

Por outro lado, o suporte de comunicação usado pelas as aplicações (B) tende a ser mais dinâmico, pois ele depende do comportamento dos usuários e de decisões de escalonamento. Em um determinado momento, durante o uso do *grid* para realizar alguma tarefa, um usuário pode alocar um *cluster* em um domínio administrativo, o que pode implicar em uma reserva de rede para permitir a comunicação. Alguns minutos depois, em função de sobrecarga de equipamentos ou liberação de recursos mais adequados, outro *cluster* pode ser alocado para o mesmo usuário, portanto, uma nova reserva deve ser feita. Alterações na alocação de recursos da rede em virtude da operação do *grid* (B) ocorrem com uma frequência maior que mudanças na infra-estrutura ou software de gerenciamento do *grid*.

As decisões de escalonamento de recursos em um *grid* são, na maioria dos casos, tomadas em tempo de execução, de acordo com as políticas de operação definidas pelo administrador e outros fatores como carga dos enlaces de rede e utilização de um determinado servidor. Por exemplo, eventualmente um usuário do *grid* pode precisar salvar um arquivo em um servidor usando algum serviço fornecido pelo *grid*. A determinação exata do servidor no qual o arquivo será salvo pode variar em função do espaço em disco disponível nos servidores e na velocidade e utilização atual do enlace entre o servidor e o usuário. Caso o enlace da rede entre o usuário e servidor esteja congestionado, outro servidor pode ser escolhido, até que uma situação adequada seja encontrada. Uma forma de garantir que a comunicação entre os elementos do *grid* ocorra com a qualidade desejada pode ser através de uma reserva de banda na rede, por exemplo, definindo-se prioridades nos roteadores para os pacotes desse fluxo.

O ponto importante aqui é que esses roteadores podem estar dispersos através de diferentes domínios administrativos, com diferentes políticas de operação da rede, que podem não estar de acordo com as políticas do *grid*. Por exemplo, uma configuração de rede frequentemente necessária em um *grid* de conferência, implementado com o *toolkit* AccessGrid (ACCESSGRID, 2004), consiste em reservar banda da rede para fluxos *multicast* de áudio e vídeo com uma determinada qualidade. Essa configuração precisa ser executada em todos os domínios administrativos que fazem parte do *grid*, para garantir uma transmissão de áudio e vídeo com sucesso. A versão atual do *toolkit* AccessGrid, apesar de fornecer ferramentas interessantes no que diz respeito ao gerenciamento das conferências, considera que todas as configurações da rede para operação do *grid* já foram feitas, o que nem sempre é verdadeiro. Essa situação também se repete com os *toolkits* Legion (CHAPIN et al., 1999) e Globe, pois eles não fornecem facilidades para um gerenciamento integrado da infra-estrutura de rede.

Um dos *toolkits* que explicitamente considera um gerenciamento integrado da infra-estrutura de rede é o Globus. Seu módulo de escalonamento define uma arquitetura denominada GARA (Globus Architecture for Reservation and Allocation), que fornece interfaces para reservas co-aloçadas de recursos da rede e de processamento. A arquitetura GARA encontra-se implementada em um protótipo (FOSTER et al., 2002), onde configurações de qualidade de serviço são feitas diretamente em roteadores com suporte DiffServ para configurar prioridade em filas. Essa implementação considera que o *toolkit* tem permissão para acessar e configurar diretamente os dispositivos da rede e não considera a existência de um sistema de gerenciamento de redes no domínio.

A figura 2.2 apresenta, do lado esquerdo, as principais APIs usadas internamente na arquitetura GARA. No lado direito são detalhados os componentes principais implementados em um protótipo da arquitetura que realiza reserva de banda em roteadores Cisco com suporte DiffServ. A arquitetura define múltiplos níveis de

APIs para maximizar a flexibilidade e funcionalidades fornecidas aos usuários. O objetivo principal dessa estratégia é facilitar o re-uso de código na implementação das aplicações. Reservas de recursos são tratadas e realizadas por gerentes locais, chamados de *Local Reservation and Allocation Manager* (LRAM), que tem permissões para acessar diretamente os recursos no domínio local e tratam aspectos relacionados com a autenticação e autorização. Funcionalidades relacionadas, por exemplo, com estratégias de recuperação em caso de falhas, são tratadas em um nível mais alto da arquitetura.

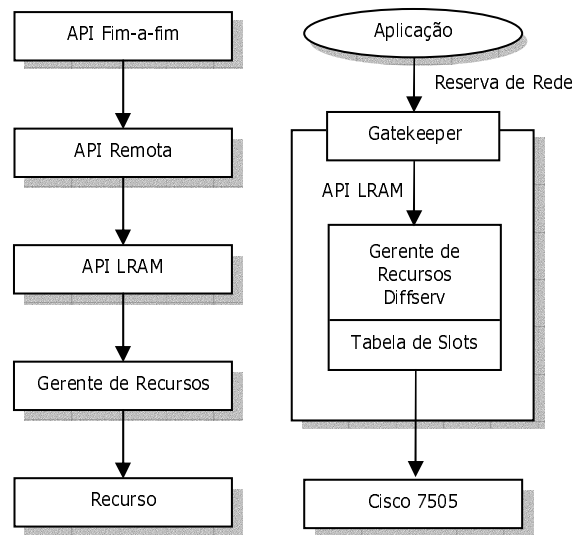


Figura 2.2: Gerenciamento de recursos GARA com reservas DiffServ

Na arquitetura GARA as reservas de recursos são sinalizadas diretamente pelas aplicações e tratadas por *gatekeepers* dispersos através dos domínios administrativos que compõe o *grid*. As reservas de recursos na arquitetura não precisam ser realizadas exclusivamente pelas aplicações: é possível que outros processos realizem as reservas em outras camadas. Apesar de existir a possibilidade da reserva ser feita por terceiros, nenhum exemplo dessa abordagem é apresentado na descrição do protótipo implementado a partir dessa arquitetura (FOSTER et al., 2002). A arquitetura GARA, portanto, faz parte de um conjunto de elementos definidos pelo *toolkit* Globus para implementar e gerenciar um *grid*.

O *toolkit* Globus também define explicitamente em seu suporte de gerenciamento o conceito de *proxy*. Um *proxy* representa um recurso do *grid* que executa determinadas ações no *grid* no interesse de um usuário. Um *proxy* possuirá os mesmos direitos de acesso do usuário que está usando o dispositivo como *proxy*. O *toolkit* Globus implementa *proxies* usando credenciais assinadas digitalmente por usuários, que são então passadas aos recursos remotos. A figura 2.3 apresenta uma possível configuração de *proxy* onde um usuário acessa um servidor de armazenamento através de um processo sendo executado em um supercomputador. Nesse caso, o supercomputador atua como um *proxy* do usuário, uma vez que ele requisita ações em nome do usuário. O conceito de *proxy* é importante para a definição de políticas de *grid* que serão apresentadas no próximo capítulo.

Avaliando-se os demais *toolkits*, além do Globus, nota-se que o conceito de *proxy* também existe, porém de maneira implícita. A maioria dos *toolkits* permite que processos disparados por um usuário em um recurso remoto do *grid* acessem outros recursos e realizem operações em nome do usuário. Esses processos podem, por exemplo, acessar dados em um servidor e disparar novos processos em processadores de um *cluster*. Essa

situação torna a tarefa de gerenciamento do *grid* bastante complexa, no que diz respeito a autenticação e autorização, e ferramentas usadas nesses casos são políticas de acesso e criptografia através de sistemas de chaves públicas e privadas. A complexidade aumenta também quando considera-se a possibilidade de encadeamento de *proxies*, onde processos de usuários estão distribuídos e atuando em diversos elementos do *grid* ao mesmo tempo. A próxima seção apresenta soluções de gerenciamento de *grid* baseadas em políticas no que diz respeito ao gerenciamento dos recursos em geral e também considerando o gerenciamento da criação e autenticação de *proxies*.

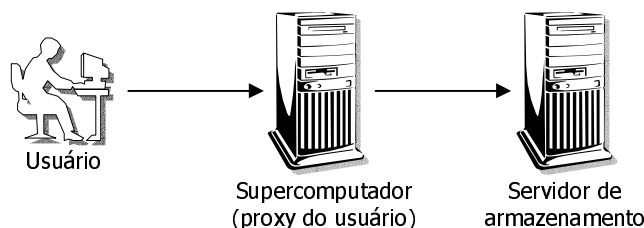


Figura 2.3: Acesso à um recurso usando um proxy intermediário

## 2.2 Gerenciamento de Grids Baseado em Políticas

Em complemento às soluções de gerenciamento encontradas nos *toolkits*, soluções de gerenciamento de *grids* baseado em políticas estão sendo propostas para tornar tal gerenciamento mais fácil e simples (SUNDARAM; NEBERGALL; TUECKE, 2000) (SUNDARAM; CHAPMAN, 2002). Um exemplo de uma política de *grid*, definida por Sundaram et al. (SUNDARAM; CHAPMAN, 2001), é apresentado na listagem de código 2.1. Essa política especifica parâmetros de qualidade na execução de processador (*priority*, *CPU* e *creditsAvail*) e uso de memória (*maxMemory*) para um usuário (*subject*) acessando um servidor (*machine*) durante um determinado período de tempo (*startTime* and *endTime*). É importante notar que essa abordagem para definição de política de *grid* não permite a especificação de parâmetros de QoS da rede a serem empregados na comunicação usuário-servidor. Essa abordagem, portanto, não considera que o acesso à rede pelo usuário receberá qualquer tipo de distinção em relação aos demais fluxos ativos de comunicação.

```

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
  
```

Código 2.1: Política de Grid

Sahu et al. (VERMA et al., 2002) definem um serviço de gerenciamento onde políticas de *grid* globais são combinadas com políticas locais de cada domínio administrativo. As políticas locais possuem uma prioridade maior, o que significa que se uma política global definir uma alocação de 20GB em um servidor, mas o administrador local definir um política que permite somente 10GB, então a política local será escolhida e somente 10GB serão alocados. Políticas de *grid* em cada domínio administrativo podem ser influenciadas por políticas de rede locais que, por alguma razão (e.g. aplicação local crítica), podem indicar que um recurso ou serviço não deve ser concedido

para um usuário do *grid*. Nessa situação, potenciais conflitos de interesse entre o administrador do *grid* e da rede podem existir e impactar na definição das políticas. É necessário então, para uma operação adequada do *grid*, que o administrador local da rede e o administrador global do *grid* estabeleçam algum tipo de acordo comum relativo à alocação dos recursos da rede no domínio local para a operação do *grid*.

Outra proposta que usa políticas para configuração da rede com o objetivo de fornecer suporte de comunicação ao *grid* é apresentada por Yang et al. (YANG; GALIS; TODD, 2002). Essa solução define uma arquitetura dividida em uma camada de gerenciamento de redes, baseada em políticas seguindo as definições de *Policy Enforcement Points* (PEPs) e *Policy Decision Points* (PDPs) do *Internet Research Task Force* (IETF) (WESTERINEN et al., 2001), e uma camada que usa o conceito de redes ativas (*active networks*), implementada em um *middleware*. Através desse *middleware* a configuração dos dispositivos da rede é feita automaticamente através de interações diretas com a arquitetura de gerenciamento de rede. Embora a arquitetura definida por Yang et al. obtenha algum nível de automatização, em nenhum momento políticas de *grid* e rede são definidas nem exemplificadas nesse trabalho.

Sander et al. (2001) também propõem uma arquitetura baseada em políticas para configurar o QoS de diferentes domínios administrativos participantes de um *grid*. As políticas são bastante similares às políticas definidas pelo IETF (YAVATKAR; PENDARAKIS; GUERIN, 2000) e permitem a reserva de banda da rede. A abordagem de Sander et al., apresentada na figura 2.4, define um protocolo de sinalização inter-domínio que configura seqüencialmente os domínios participantes do *grid* presentes no caminho de uma comunicação fim-a-fim (e.g. um usuário de um domínio acessando um servidor de dados em outro domínio). O protocolo de sinalização permite a comunicação entre elementos chamados *bandwidth brokers* (BBs), distribuídos através dos domínios administrativos de rede que compõe o *grid*. Os BBs trocam informações entre si para verificar a possibilidade de aplicação de uma política no domínio. Embora a arquitetura proposta por Sander et al. seja baseada em políticas, não são apresentadas quaisquer facilidades para permitir a integração com os *toolkits* de *grid* apresentados anteriormente. A arquitetura resume-se a uma solução independente de gerenciamento de QoS, baseada em políticas inter-domínios.

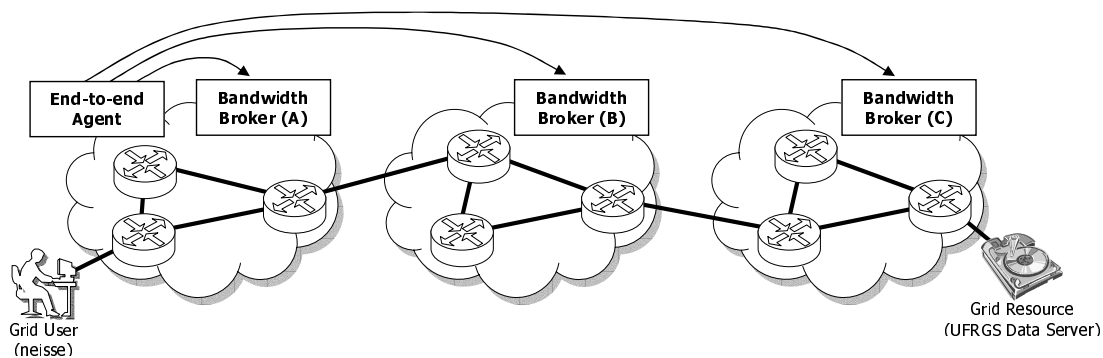


Figura 2.4: Sinalização dos bandwidth brokers para configuração de QoS

Existem soluções de gerenciamento baseado em políticas que tratam especificamente do gerenciamento dos *proxies* em um *grid*. Essas políticas tem como objetivo expressar regras para criação e validação de *proxies*. É possível definir, por exemplo, quais recursos podem atuar no *grid* como *proxies* dos usuários durante um determinado período. A dificuldade de gerenciamento da criação dos *proxies* tende a aumentar em *grids* onde a definição dos recursos alocados aos usuários (e.g. escalonamento) seja dinâmica e existam interdependências na localização de tarefas disparadas simultaneamente. Isso complica o processo de escalonamento e também o processo de alocação/identificação dos recursos, pois o usuário pode possuir mais de um endereço de rede associado nos diversos *proxies*. Isso ocorre pois vários processos no *grid* podem estar atuando

como *proxies* identificados pelas credenciais do usuário.

A listagem de código 2.2 apresenta um exemplo de política de criação e validação de *proxy*. Nessa política o usuário possui restrições de local (`OperateFromList`, `IntermediateSites`, `OperateFrom`) e horários (`UserTimeList`, `TimeRequirements`, `TimeRestrictions`) no qual ele pode criar *proxies* e acessar o *grid*. A definição de políticas desse tipo não é uma tarefa fácil, considerando a quantidade de informações que precisam ser fornecidas e a sintaxe utilizada. Com o objetivo de facilitar o gerenciamento dos *grid*, propostas de gerenciamento baseado em políticas de mais alto nível e ferramentas como EZGrid (CHAPMAN et al., 2004) para facilitar essas definições foram desenvolvidas.

```

Default policy [
  login = Specific.login;
  subject = Specific.subject;
  OperateFromList = {"moleman.mcs.anl.gov", "clarinet.mcs.anl.gov"};
  UserTimeList = {[valid = false, start = "09:00:00"; end = "17:00:00"],
                  [valid = false, start = "09:00:00"; end = "17:00:00"],
                  [valid = true, start = "09:00:00"; end = "17:00:00"],
                  [valid = false, start = "09:00:00"; end = "17:00:00"],
                  [valid = true, start = "09:00:00"; end = "17:00:00"],
                  [valid = false, start = "09:00:00"; end = "17:00:00"],
                  [valid = false, start = "09:00:00"; end = "17:00:00"]
  };
  ctime = CurrentTime();
  DayOfWeek = GetDayOfWeek(ctime);
  TimeRequirements = Specific.UserTimeList[DayOfWeek].valid &&
                    ctime >= Specific.UserTimeList[DayOfWeek].start &&
                    ctime <= Specific.UserTimeList[DayOfWeek].end;
  IntermediateSites = {
    "/O=Grid/O=Globus/CN=pitcairn.mcs.anl.gov",
    "/O=Grid/O=Globus/CN=moleman.mcs.anl.gov",
    "/O=Grid/O=Globus/CN=clarinet.mcs.anl.gov"
  };
  LocationRequirements = Member(
    Specific.OperateFromList, other.dynamic.localhost
  );
  Req = Specific.TimeRequirements && Specific.LocationRequirements &&
        Other.dynamic.subject = Specific.Subject;
  Specific = [
    Subject = "/O=Grid/O=Globus/OU=mcs.anl.gov/CN=John Doe";
    login = "jdoe";
    OperateFrom = {"pitcairn.mcs.anl.gov", "homer.mcs.anl.gov"};
    Req = TimeRestrictions && Subject==other.dynamic.subject;
  ]
  Requirements = Specific.Req;
]

```

## Código 2.2: Política de Grid para Proxies

Na ferramenta EZGrid o usuário (administrador do *grid*) pode definir as políticas de operação do *grid* e também características dos processos que serão executados pelos usuários. Como pode ser observado na captura de tela da ferramenta (figura 2.5) pode-se definir características dos processos (*jobs*) executados, tais como: processamento (CPU's Required), memória (Memory Required), redirecionamento da saída padrão (Standard Output) e da saída de erro (Standard Error). A captura de tela apresentada na figura foi retirada da página Web do desenvolvedor do EZGrid, pois a ferramenta não está disponível para *download*. Novamente, nota-se que não existe preocupação na interface da ferramenta EZGrid com a definição dos requisitos de rede necessários pela aplicação assim como não existe suporte à definição de *proxies*.

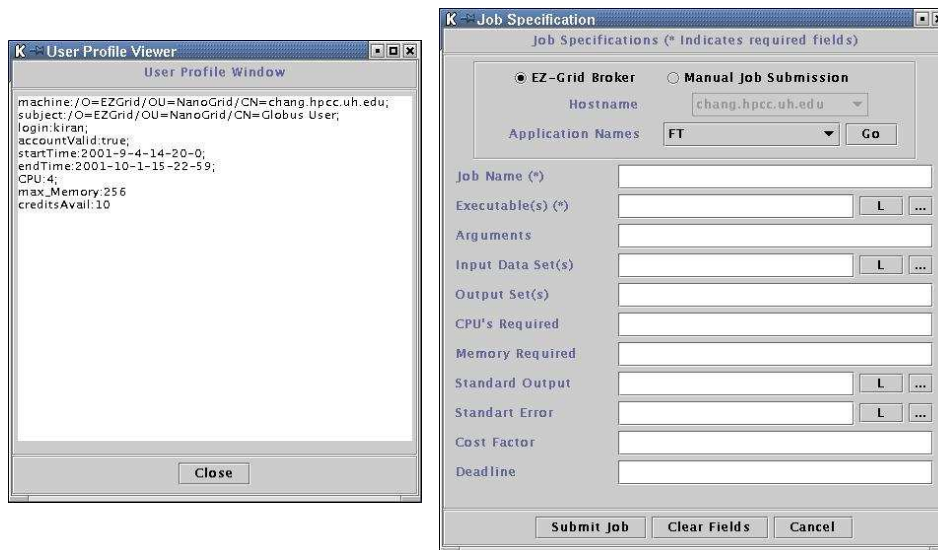


Figura 2.5: Ferramenta EZGrid para definição de políticas de grid

## 2.3 Análise Crítica e Proposta

Embora existam propostas de gerenciamento de *grid* utilizando *toolkits* e também através de políticas, essas soluções não permitem as definições de parâmetros de QoS da rede para alocar recursos de comunicação. A definição de parâmetros de QoS é importante pois a rede, além de ser o meio de acesso aos recursos compartilhados no *grid*, também é um recurso a ser compartilhado entre os usuários. Além disso, as arquiteturas de gerenciamento de *grid* baseado em políticas não estão integradas com nenhum dos *toolkits* mencionados previamente, embora algumas propostas citem esforços de integração futuros (e.g. como o *toolkit* Globus).

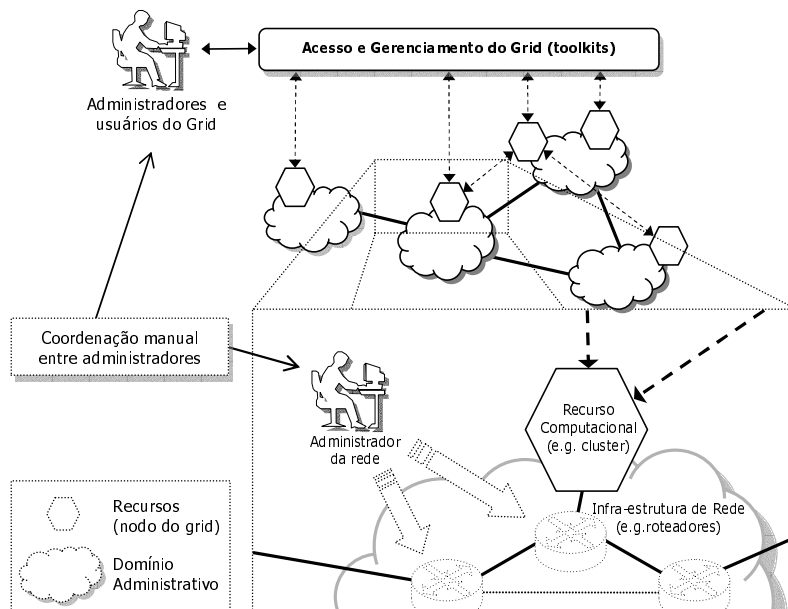


Figura 2.6: Visão geral do cenário de gerenciamento de grid e rede

A figura 2.6 mostra, de maneira abstrata, um cenário atual típico de gerenciamento de *grid* e rede. Nesse cenário, o administrador do *grid* coordena a operação do *grid* usando o suporte fornecido pelos *toolkits*, e interage manualmente com os administradores de rede em cada domínio administrativo para garantir que as configurações de rede necessárias para operação do *grid* sejam



executadas. Observando esse cenário é possível notar que toda vez que um requisito do *grid* que implica em uma nova configuração na infra-estrutura de rede é alterado, uma coordenação não automatizada entre os administradores do *grid* e da rede é necessária.

Como já observado, o suporte fornecido pelos *toolkits* para realizar o gerenciamento integrado do *grid* e das infra-estruturas de rede é bastante limitado e, na maioria dos casos, ele não existe. De fato, a maioria dos *toolkits* considera que a rede já está corretamente configurada para a operação do *grid*, o que não é sempre verdadeiro (NEISSE et al., 2004a).

O modelo de gerenciamento integrado proposto neste trabalho considera em conjunto as implicações inerentes ao gerenciamento de um *grid* computacional e da infra-estrutura de rede, levando em conta a independência dos diversos domínios administrativos através dos quais os nodos de um *grid* podem estar distribuídos. A solução também considera as necessidades do administrador da rede de cada domínio, que precisa definir restrições nas configurações da rede, pois nem todas as operações que podem ser realizadas na rede devem ser permitidas ao *grid*. Além da reserva de recursos, outras tarefas como gerenciamento de grupos *multicast* e definição de regras em um *firewall* também são necessárias no gerenciamento integrado de uma infra-estrutura de *grid*, mas não são abordadas neste trabalho, apesar da possibilidade de extensão do modelo para suportar novas tarefas.

### 3 SOLUÇÃO PARA TRADUÇÃO DE POLÍTICAS DE GRID EM POLÍTICAS DE REDE

Este trabalho propõe um modelo de gerenciamento baseado em políticas que tem como objetivo integrar o gerenciamento de *grid* e da infra-estrutura de rede subjacente. Pretende-se com esse modelo fornecer uma solução de gerenciamento que permita automatizar a geração de políticas de rede a partir de políticas de operação do *grid* definidas. Nesse contexto, as políticas de *grid* expressam as condições de operação do *grid* enquanto as políticas de rede expressam as configurações de rede (e.g. reservas de banda) necessárias para que o *grid* opere adequadamente.

No modelo proposto, as políticas de operação do *grid* são regras definidas pelo administrador do *grid* que governam o uso dos recursos compartilhados (por exemplo, quais usuários tem direitos de usar determinados recursos e quando o acesso será permitido). As políticas de *grid* podem indicar também a qualidade no acesso aos recursos, através de reservas de processador, memória ou banda da rede. Políticas de *grid* são definidas em um nível global, e fazem referência ao conjunto completo de usuários e recursos compartilhados do *grid* que podem estar dispersos através de diversos domínios administrativos de rede independentes.

As políticas de rede, por sua vez, são regras definidas pelos administradores da rede em cada domínio administrativo que expressam, por exemplo, quais fluxos terão uma maior ou menor largura de banda ou qual a prioridade que os pacotes dos fluxos receberão na rede em relação aos demais pacotes. O administrador da rede define as políticas de operação de sua rede de acordo com os interesses locais no seu domínio. Nos sistemas de gerenciamento baseado em políticas tradicionais as políticas de rede são definidas manualmente pelos administradores de rede através de uma ferramenta de gerenciamento.

O modelo proposto neste trabalho tem como objetivo fornecer algum nível de automatização na criação das políticas de gerenciamento de rede em função dos requisitos dos fluxos do *grid* expressos pelas políticas de *grid*. Assim, o modelo fornece uma solução integrada de gerenciamento de *grid* e rede, que tem como objetivo configurar o suporte de comunicação requerido para operação do *grid* na infra-estrutura de rede. No modelo de gerenciamento proposto o administrador da rede não define mais manualmente as políticas de rede, ele define agora regras de tradução que são as responsáveis pela criação automática das políticas de rede.

O processo de criação de políticas de rede, a partir das políticas de *grid*, foi denominado neste trabalho de "processo de tradução" e as regras que alimentam esse mecanismo são chamadas de "regras de tradução". A figura 3.1 apresenta uma visão geral do processo de tradução onde, primeiramente, no topo da figura, as políticas de gerenciamento de *grid* são definidas pelo administrador do *grid*. As políticas de *grid* são então traduzidas para políticas de rede através de um "mecanismo de tradução" que opera através de regras.

Após as políticas de rede serem criadas através do mecanismo de tradução, como apresentado na figura 3.1, elas precisam ser novamente traduzidas para ações específicas de configuração dos dispositivos de rede. Dispositivos de rede podem ser, por exemplo, roteadores ou *switches*. A tradução das políticas de rede para ações de configuração é realizado, no modelo proposto, por *Policy Decision Points* (PDPs). PDPs são elementos que fazem parte do sistema de gerenciamento

baseado em políticas (*Policy Based Network Management System - PBNM*) padronizado pelo *Internet Engineering Task Force (IETF)* (WESTERINEN et al., 2001), que será detalhado na primeira seção deste capítulo.

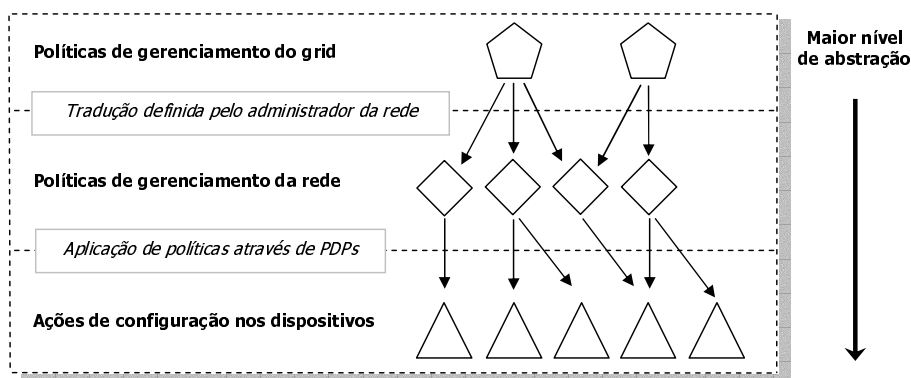


Figura 3.1: Hierarquia para tradução de políticas

Pode-se considerar que a solução de gerenciamento integrado proposta neste trabalho é uma solução de gerenciamento baseada em políticas com hierarquia, pois as políticas de *grid*, em um nível mais alto de abstração, são traduzidas para as políticas de rede, em um menor nível de abstração. Assim, as políticas de *grid* são definidas usando estruturas mais abstratas do que as estruturas utilizadas para definição das políticas de rede. Considera-se que elementos como usuários, recursos e *proxies*, usados na definição das políticas de *grid*, estão em um nível maior de abstração que elementos como dispositivos, interfaces e endereços de rede, usados na definição das políticas de rede (figura 3.2).

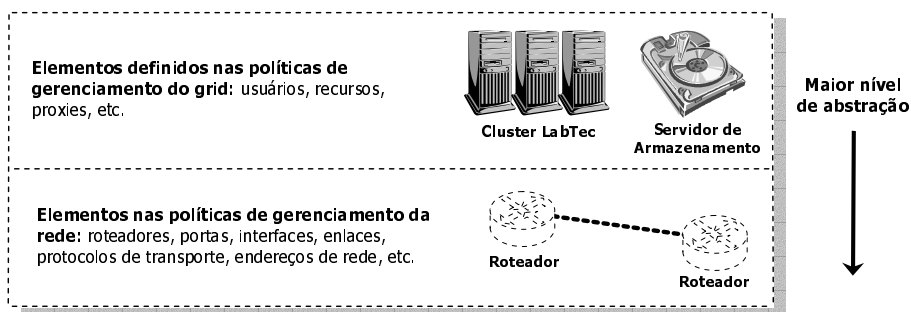


Figura 3.2: Níveis de abstração das políticas de grid e rede

O uso de hierarquias de políticas no gerenciamento de redes já foi abordado por Flegkas et al. (FLEGKAS et al., 2001) para gerenciar uma rede IP com QoS. Na solução de Flegkas et al. foram definidos vários níveis de abstração e as respectivas traduções entre eles. Outro trabalho significativo em hierarquias de políticas foi desenvolvido por Moffett e Sloman (MOFFETT; SLOMAN, 1993), onde os relacionamentos hierárquicos entre políticas de níveis de abstração diferentes foram classificados em cinco tipos, apresentados a seguir:

- **Particionamento de alvos:** o alvo de uma política de baixo nível pode ser um sub-conjunto do alvo da política de mais alto nível;
- **Refinamento de objetivos:** o objetivo de uma política de mais alto nível pode ser refinado em um ou mais objetivos de um nível inferior, referenciando o mesmo alvo;
- **Refinamento arbitrário de objetivos:** nesta forma de refinamento, os alvos e o objetivos de uma política de nível inferior são diferentes dos objetivos e alvos de uma políticas de nível superior;

- Procedimentos: onde uma política pode ser refinada em um conjunto ordenado de políticas de mais baixo nível;
- Delegação de responsabilidade: neste tipo de refinamento, um sujeito delega responsabilidades pelo objetivo para outro sujeito.

No modelo proposto neste trabalho foram definidos dois níveis de abstração (políticas de *grid* e políticas de rede) para integrar o gerenciamento realizado pelos *toolkits* de *grids* e pelos sistemas de gerenciamento de redes. Pode-se classificar, de acordo com Moffet e Sloman, a tradução realizada entre esses dois níveis como um refinamento arbitrário de objetivos. Classifica-se dessa forma pois não existe nenhuma relação direta entre os alvos e objetivos definidos nas políticas de alto nível (*grid*) e nas políticas de baixo nível (rede).

Para representar as políticas de *grid*, as políticas de rede e as regras de tradução que dão suporte ao modelo de gerenciamento integrado optou-se pelo uso duas linguagens. As políticas de *grid* são usadas pelos administradores do *grid* para definir quais recursos que serão acessados pelos usuários do *grid*, quais os *proxies* que serão usados e quais as prioridades desses acessos. As políticas de rede servem para expressar quais os fluxos da rede receberão um tratamento especial em relação a largura de banda disponível. As regras de tradução servem para que os administradores da rede possam determinar como as políticas de *grid* serão traduzidas para as políticas de rede no seu domínio administrativo.

Embora o objetivo deste trabalho não seja o de definir uma nova linguagem para políticas de *grid* e rede, é descrito um conjunto de elementos necessário para tais políticas, através de uma linguagem hipotética, que é baseada nos trabalhos de Sundaram et al. (SUNDARAM; NEBERGALL; TUECKE, 2000) (SUNDARAM; CHAPMAN, 2002) (SUNDARAM; CHAPMAN, 2001), previamente apresentados, e também na arquitetura de gerenciamento baseado em políticas do IETF, a ser apresentada na primeira seção deste capítulo. Apesar deste trabalho utilizar essa linguagem hipotética para definição das políticas, na implementação de ferramentas de gerenciamento completamente funcionais o suporte necessário para representar as políticas poderia ser alcançado através do uso de linguagens de políticas já existentes, tais como Ponder (DAMIANOU et al., 2001) e PDL (LOBO; BHATIA; NAQVI, 1999).

A primeira seção deste capítulo apresenta, como já foi dito, a arquitetura de gerenciamento baseado em políticas do IETF. A segunda seção apresenta a linguagem hipotética usada para representação das políticas e os elementos necessários para definição de políticas de *grid* usando essa linguagem. A terceira seção apresenta os elementos necessários para representação de políticas de rede. Já a seção 3.4 descreve a segunda linguagem definida neste trabalho, que é usada para a definição das regras de tradução. Por fim, a seção 3.5 detalha o modelo de gerenciamento onde as duas linguagens definidas são usadas em conjunto para realizar o gerenciamento integrado de uma infra-estrutura de *grid* e de rede.

### 3.1 Arquitetura de Gerenciamento Baseado em Políticas do IETF

Na arquitetura de gerenciamento baseado em políticas do IETF, apresentada na figura 3.3, os administradores de rede definem e coordenam o uso de políticas através de uma ferramenta de gerenciamento. As políticas, após serem criadas pelos administradores, são armazenadas em um repositório de políticas. A ferramenta de gerenciamento fornece uma interface para criar, editar, excluir ou aplicar as políticas armazenadas no repositório. A aplicação de uma política tem por objetivo fazer com que o comportamento da rede passe a refletir os objetivos expressos nas políticas aplicadas.

Para aplicar uma política os administradores precisam definir em quais elementos da rede essa política será aplicada. Os elementos onde se aplicam as políticas são chamados, segundo a terminologia do IETF, de *Policy Enforcement Points* (PEPs). Cada dispositivo da rede pode possuir vários PEPs. Por exemplo, em um roteador, cada interface de rede pode possuir uma disciplina

de filas de entrada e uma disciplina de filas de saída. Considera-se então que nesse roteador cada fila é um PEP onde uma política pode ser aplicada. Portanto, PEPs são os elementos finais que efetivamente implementarão uma política.

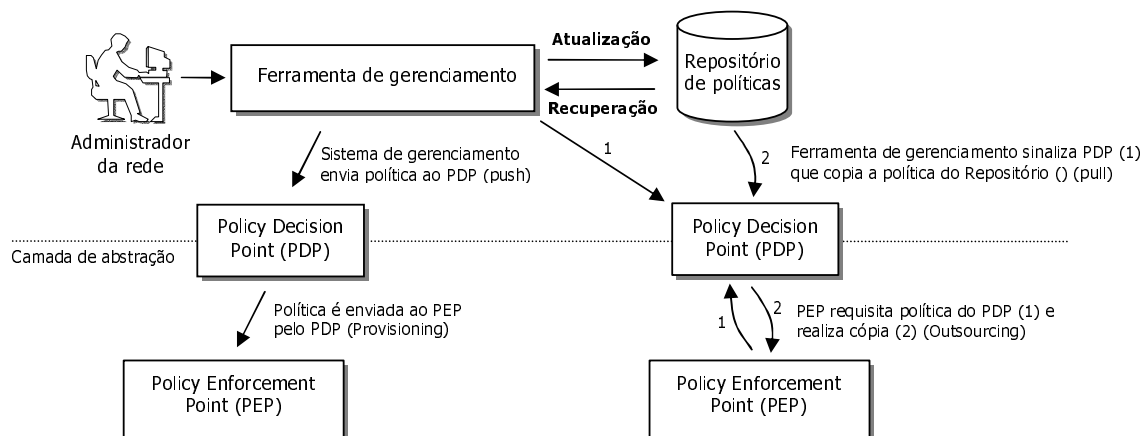


Figura 3.3: Modelo de gerenciamento baseado em políticas do IETF

Além dos PEPs, a arquitetura do IETF define elementos para tradução de políticas, chamados *Policy Decision Points* (PDPs). Os PDPs são responsáveis por traduzir as políticas descritas em linguagem de alto nível para ações de configuração específicas dos dispositivos da rede. Embora o administrador da rede selecione as políticas e PEPs através da ferramenta de gerenciamento, a definição de quais PDPs devem ser contactados para tradução das políticas é feita de forma automática. PDPs se relacionam com os PEPs em um sistema de gerenciamento baseado em políticas assim como os drivers de um sistema operacional se relacionam com os periféricos em um computador. Podem ser implementados PDPs que suportem a configuração de PEPs em dispositivos de diversos tipos, modelos e fabricantes.

O IETF definiu duas abordagens para transferência de políticas do repositório para os PDPs, denominadas *pull* e *push*. No primeiro caso (*pull*), o PDP recebe uma sinalização da ferramenta de gerenciamento para aplicar uma política e essa política é então copiada pelo PDP num acesso direto ao repositório de políticas. No segundo caso (*push*), a ferramenta de gerenciamento recupera a política do repositório e envia a política a ser aplicada acessando diretamente o PDP.

A comunicação entre PDPs e PEPs também pode ser feita, segundo o IETF, através de duas abordagens, denominadas *provisioning* e *outsourcing*. Na abordagem *provisioning*, o PDP inicia a comunicação com o PEP e realiza a configuração de acordo com as definições da política. Já na abordagem *outsourcing*, a requisição da política é iniciada pelo PEP, através de uma consulta sobre uma decisão de gerenciamento que precisa ser tomada pelo PEP. O IETF criou padrões como o *Common Open Policy Service* (COPS) e *COPS Usage for Policy Provisioning* (COPS-PR) para realizar a transferência de políticas entre PDPs e PEPs, embora outros protocolos como SNMP, TELNET e SSH também possam ser usados para esse fim.

O IETF definiu também, em conjunto com o *Distribute Management Task Force* (DMTF), um modelo de informações orientado a objetos que serve como base para representação de políticas chamado de *Policy Core Information Model* (PCIM) (MOORE et al., 2001). O modelo PCIM permite a definição de políticas mas com pouca flexibilidade, o que motivou recentemente a padronização de uma extensão complementar, chamada de *Policy Core Information Model Extensions* (PCIME) (MOORE, 2003). O modelo PCIME foi criado com o objetivo de tornar a definição das políticas mais flexível e extensível do que era permitido com o modelo PCIM.

O modelo de informações PCIME introduziu algumas modificações significativas em relação ao modelo PCIM original. Novos elementos foram incluídos, por exemplo, novas classes para definir políticas de filtros de pacotes de rede, o que permitiu o suporte a especificação de políticas

em áreas não suportadas anteriormente com o PCIM. Além disso, algumas classes e atributos foram retirados e substituídos por outros, com o objetivo de tornar o modelo geral de definição de políticas mais flexível. Apesar dessas mudanças, a interoperabilidade entre os dois modelos foi mantida. Portanto, implementações já existentes em PCIM não foram invalidadas com a criação do PCIME.

O modelo de informações PCIME permite a construção de políticas usando regras compostas por condições e ações no formato *se <condição> então <ação>*. Uma condição é uma expressão lógica usada para expressar os critérios de seleção das regras. Critérios nesse contexto incluem: condições temporais dizendo quando a regra será aplicada, condições de escopo dizendo a que usuários e/ou recursos a regra se aplica e condições de estado dizendo em que circunstâncias a ação deverá ser executada. Quando uma condição resultar em verdadeiro em um recurso gerenciado (PEP), a regra torna-se válida e as ações são executadas. O objetivo das ações é executar uma ou mais operações que afetarão a rede para atingir um estado desejado. Uma ação é representada através de uma atribuição de um valor à uma variável.

A figura 3.4 apresenta o diagrama de classes simplificado do PCIME. Condições e ações são consideradas, respectivamente, expressões lógicas e de atribuição, envolvendo variáveis (*PolicyVariable*) e valores (*PolicyValue*). O modelo de informações PCIME foi desenvolvido de tal forma que novas variáveis e valores em condições e ações podem ser definidos, permitindo facilmente o suporte para definição de políticas relacionadas a tecnologias não previstas originalmente no modelo. Isso é possível através da extensão das classes *PolicyVariable* e *PolicyValue* para que condições e ações envolvendo novos elementos possam ser definidas nas políticas.

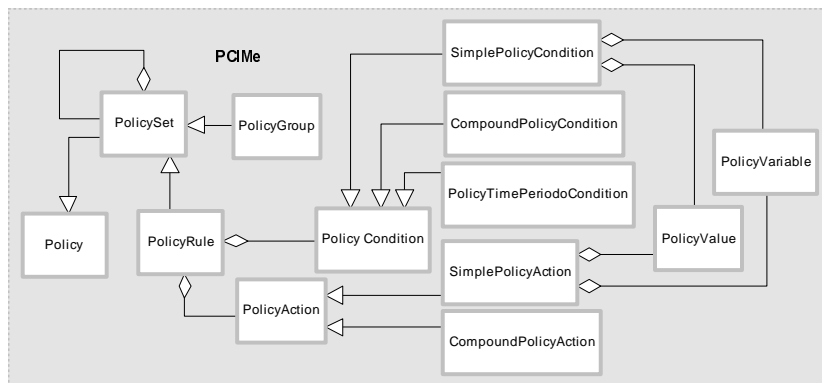


Figura 3.4: Diagrama de classes do modelo de informações do PCIME

## 3.2 Linguagem para Definição de Políticas de Grid

Em comparação às propostas usando políticas no gerenciamento de *grids* estudadas, percebeu-se que novos elementos para definição de políticas são necessários, pois nenhuma das soluções encontradas permite a definição de requisitos de QoS de rede nem expressar políticas considerando a noção de *proxies* (NEISSE et al., 2004). Inicialmente, observando-se as soluções de gerenciamento de *grids* usando políticas propostas por Sundaram et al. (SUNDARAM; NEBERGALL; TUECKE, 2000) (SUNDARAM; CHAPMAN, 2002) (SUNDARAM; CHAPMAN, 2001), verificou-se que as políticas de *grid* precisam ser definidas não somente a partir de usuários e recursos, mas também com base em *proxies* e requisitos de QoS da rede. Partiu-se da suposição de que uma linguagem de políticas de *grid* suporta a definição de *proxies* e QoS da rede a partir de uma extensão do modelo de políticas condição-ação do IETF (PCIME), apresentado na seção anterior.

Como foi constatado na seção 2.2, uma das soluções de gerenciamento de *grid* propostas por Sundaram et al. (SUNDARAM; NEBERGALL; TUECKE, 2000) fornece políticas para controlar a instanciação e tempo de vida de *proxies*, mas não fornece elementos que apoiem a definição de requisitos de QoS de rede para as operações do *grid*. Assim, na abordagem proposta neste trabalho, uma política de *grid* é composta por uma declaração condicional contendo elementos relacionados às restrições temporais, aos usuários do *grid*, aos recursos e aos *proxies*, assim como ações referentes ao controle de acesso, reserva de processador, reserva de disco e reserva de banda da rede.

### Estrutura da linguagem

A listagem de código 3.1 apresenta a BNF usada para a definição das políticas de *grid*. Definiu-se nessa BNF que uma regra é formada por condições, ações e que pode agrupar outras regras internamente. Uma condição é uma única expressão condicional ou uma lista de expressões condicionais que podem ser ligadas pelo operador "e" lógico (and). Cada expressão condicional é uma igualdade (=) ou desigualdade (<, <=, >, >=, !=) entre uma variável e um valor. Uma ação na linguagem é definida como uma atribuição de um valor à uma variável.

```

policy ::= rule
rule ::= if conditions [rule] actions;
conditions ::= condition [and condition]
condition ::= cvariable {=,<,<=,>,>=,!=} cvalue
actions ::= action [; action]
action ::= avariable = avalue

```

Código 3.1: BNF simplificada das políticas de *grid* e rede

As variáveis e valores usados na definição da BNF são mapeamentos diretos das classes `PolicyVariable` e `PolicyValue` definidas no modelo PCIME apresentado na seção anterior. A BNF usada para definir as políticas do *grid* também é usada para definição das políticas de rede, com exceção dos valores aceitos para as variáveis (`cvariable`, `avvariable`) e valores (`cvalue`, `avalue`) das condições e ações. A linguagem para definição das políticas de rede é detalhada na próxima seção.

As variáveis aceitas nas condições (`cvariable`) de uma política de *grid* são: `user`, `proxy`, `resource`, `startTime` e `endTime`, que representam respectivamente um usuário do *grid*, um recurso do *grid* atuando como *proxy*, um recurso do *grid* sendo acessado, o período inicial a partir do qual o acesso será permitido e um período final até quando o acesso será permitido. As variáveis aceitas nas ações (`avvariable`) de uma política de *grid* são `allowAccess`, `maxProcessing`, `maxAllowedStorage` e `networkCoS`, que representam, respectivamente, a permissão de acesso ao recurso, o máximo de processador que poderá ser usado do recurso, o máximo de disco que poderá ser usado do recurso e a qualidade de serviço de rede desejada.

Nas condições e nas ações das políticas de *grid* os valores usados para comparação e atribuição nas variáveis seguem a semântica da variável. Por exemplo, quando a variável `startTime` for usada nas condições o valor comparado deve conter uma informação de tempo. Já para comparação com a variável `proxy` e `resource`, deve ser usada uma identificação de um recurso do *grid*. O mesmo acontece para os valores usados na atribuição das variáveis nas ações da política. Para a variável `maxProcessing`, por exemplo, espera-se a atribuição de um valor percentual, enquanto que para a variável `networkCoS` espera-se a atribuição de um identificador de uma classe de serviço do *grid*.

No exemplo da listagem de código 3.2, duas regras são usadas para definir uma política que permita ao usuário `neisse` acessar, durante o dia 25 de novembro de 2004, um *cluster* do *grid* (`LabTec Cluster`) e, a partir desse *cluster*, no mesmo período, ele também pode acessar o

servidor de armazenamento UFRGS Data Server. Nesse último acesso, o *cluster* LabTec atua como um *proxy* do usuário. Assim, embora o usuário *neisse* não tenha acesso direto ao servidor de armazenamento UFRGS Data Server, ele ainda é capaz de armazenar nesse servidor as informações geradas pelos seus processos executando no *cluster*.

```

if (user == "neisse" and
    resource == "LabTec Cluster" and
    startTime >= "11/25/2004 00:00:00" and
    endTime <= "11/25/2004 23:59:59")
{
    allowAccess = true;
    maxProcessing = 50%;
    networkCoS = remoteProcessControl;
}

if (user == "neisse" and
    proxy == "LabTec Cluster" and
    resource == "UFRGS Data Server" and
    startTime >= "11/25/2004 00:00:00" and
    endTime <= "11/25/2004 23:59:59")
{
    allowAccess = true;
    maxAllowedStorage = 40GB;
    networkCoS = highThroughputDataIntensive;
}

```

### Código 3.2: Exemplo de políticas de grid

Na primeira regra da política apresentada na listagem de código 3.2, as ações permitem ao usuário acessar o *cluster* LabTec e consumir até 50% do processamento total do *cluster*. Além disso, o usuário recebe uma QoS da rede que lhe fornece banda suficiente para controlar remotamente a execução de processos (*remoteProcessControl*). Na segunda regra da política, por sua vez, as ações especificam que o usuário tem acesso ao servidor de armazenamento para consumir até 40GB, e recebe uma QoS da rede para suportar uma transferência intensiva de dados (*highThroughputDataIntensive*). Uma vez que na segunda regra um *proxy* é usado, não é necessário especificar com quais permissões o usuário será autenticado no *cluster*: isso já está especificado na primeira regra para acesso ao *cluster*.

O exemplo de política de *grid* apresentado na listagem de código 3.2 declara a QoS da rede nas ações através da atribuição de uma identificação de classe de serviço (CoS) associada (e.g. *remoteProcessControl* e *highThroughputDataIntensive*) à variável *networkCoS*. Por trás das identificações de CoS encontra-se a definição de um conjunto de parâmetros de QoS relacionados que precisam ser determinados pelo administrador do *grid*. Esse conjunto de parâmetros especifica os requisitos de rede necessários por cada um dos tipos de aplicações executados no *grid*. Considerou-se neste trabalho, a partir de uma análise dos requisitos de aplicações de *grid* apresentados por Foster et. al (FOSTER; KESSELMAN, 1999), que os seguintes parâmetros são necessários para definição de classes de serviço em um *grid*: largura de banda mínima, largura de banda necessária, perda mínima, perda máxima, prioridade, e um marcador indicando se a largura de banda usada pela classe de serviço é compartilhada entre os usuários <sup>1</sup>.

Espera-se que as classes de serviço sejam definidas pelos administradores do *grid* e armazenadas em uma biblioteca de classes de serviço, para serem usadas posteriormente na definição das políticas de *grid*. A decisão de respeitar os parâmetros de QoS definidos nessas classes de serviço na rede depende das regras de tradução definidas pelos administradores da rede em cada domínio que compõe o *grid*, pois é a partir das definições dessas regras que as políticas de rede que fornecem as classes de serviço serão geradas. Assim, embora o administrador do

<sup>1</sup>Outros parâmetros relacionados com a rede podem ser usados na arquitetura de fornecimento de QoS implementada na rede subjacente, mas não precisam ser definidos pela administração do *grid*



*grid* defina nas classes de serviço os requisitos de rede das aplicações do *grid*, a tradução desses parâmetros para políticas de rede correspondentes que efetivamente fornecerão as classes de serviço é de responsabilidade dos administradores da rede em cada domínio administrativo.

### Aninhamento de regras

A linguagem de políticas de *grid* usada nesse trabalho também suporta o aninhamento de regras, isto é, uma regra interna pode ser definida no contexto de outra regra externa. Portanto, é possível formar uma hierarquia de regras. Assim, a definição das duas regras previamente apresentadas (listagem de código 3.2) poderia ser sumarizada em somente uma regra com aninhamento, onde expressa-se uma regra externa relativa ao usuário e período temporal, e duas regras internas relativas aos recursos, *proxies* e ações. Isso é possível pois as informações referentes aos usuários e período temporal se repetem em ambas as políticas (esse conceito de aninhamento é característica do modelo PCIM definido pelo IETF).

A nova política com regras aninhadas, equivalente as duas políticas anteriores, é apresentada na listagem de código 3.3. Nessa política, as regras internas serão consideradas somente se as condições da regra externa tornarem-se válidas, o que otimiza o processo de avaliação da política, já que um número menor de condições precisa ser testado para verificação da validade da regra, considerando que as regras internas só serão avaliadas se as regras externas tornarem-se válidas. Além disso, o agrupamento de regras tornam mais claras para o administrador do *grid* as política, pois políticas relacionadas com um mesmo objetivo podem ser agrupadas e definidas na mesma estrutura, em vez de serem especificadas separadamente.

```

if (user == "neisse" and
    startTime >= "11/25/2004 00:00:00" and
    endTime <= "11/25/2004 23:59:59")
{
  if (resource == "LabTec Cluster") {
    allowAccess = true;
    maxProcessing = 50%;
    networkCoS = remoteProcessControl;
  }
  if (proxy == "LabTec Cluster" and
      resource == "UFRGS Data Server")
  {
    allowAccess = true;
    maxAllowedStorage = 40GB;
    networkCoS = highThroughputDataIntensive;
  }
}

```

Código 3.3: Regras de políticas aninhadas

### Domínios

A linguagem para definição de políticas de *grid* também permite a especificação de recursos e *proxies* através de domínios. Domínios são abstrações usadas em sistemas de gerenciamento de redes para agrupar um conjunto de objetos gerenciáveis, considerando um determinado aspecto ou característica em comum (SLOMAN; MOFFETT, 1989). Um domínio pode agrupar, por exemplo, um conjunto de recursos que possuem funcionalidades semelhantes. No contexto de gerenciamento de *grid*, um domínio poderia ser formado por todos os *clusters* do *grid*. Um domínio mais restrito poderia considerar somente os *clusters* que possuem como característica o suporte à aplicações desenvolvidas usando a biblioteca PVM (*Parallel Virtual Machine*).

Domínios são particularmente importantes na definição de regras de políticas de *grid* onde usuários, *proxies* e recursos não são precisamente definidos, mas somente suas classes. Nas políticas de *grid* previamente apresentadas, uma regra definida pelo administrador poderia expressar que o usuário *neisse* pode acessar um *cluster* do *grid* e dois servidores

de armazenamento, mas sem indicar especificamente qual *cluster* e quais servidores de armazenamento devem ser usados. Isso pode ser especificado através do uso de domínios, como pode ser visto na política da listagem de código 3.4, usando as palavras chave `cluster()` e `storageServer()`.

```

if (user == "neisse" and
    startTime >= "11/25/2004 00:00:00" and
    endTime <= "11/25/2004 23:59:59"
) {
  if (resource == cluster(1)) {
    allowAccess = true;
    maxProcessing = 50%;
    networkCoS = remoteProcessControl;
  }
  if (proxy == cluster(1) and
      resource == storageServer(2)
  ) {
    allowAccess = true;
    maxAllowedStorage = 40GB;
    networkCoS = highThroughputDataIntensive;
  }
}

```

Código 3.4: Políticas de grid com domínios

Nessa política, `cluster` e `storageServer` especificam o domínio dos recursos que serão alocados ao usuário. O número entre parênteses define a quantidade de elementos de cada domínio que serão alocados. Nesse caso, o usuário terá acesso a um único *cluster*, e, usando esse *cluster* como *proxy*, ele ganhará acesso a dois servidores de armazenamento. As noções de domínios também pode ser usada na cláusula `user` para endereçar não somente um único usuário, mas um conjunto deles. Por exemplo, o administrador do *grid* poderia definir uma política que alocasse até dois *clusters* do *grid* para cada usuário de alta prioridade. Nesse caso nem os *clusters* nem os usuários são especificamente definidos, mas somente suas classes: *cluster* e usuário de alta prioridade.

Quando define-se *proxies* e recursos em termos de domínios uma questão importante surge: os recursos são alocados de maneira individual ou compartilhados? Por exemplo, na política previamente apresentada, o limite de 40GB é definido para ambos os servidores ou para cada um deles? No primeiro caso, haveria mais economia de espaço se a soma do espaço total nos dois servidores de armazenamento não exceder o limite de 40GB. No segundo caso, o usuário consumiria 40GB em um servidor de armazenamento, mais 40GB em outro servidor e o limite total seria então 80GB. Embora não seja intenção deste trabalho resolver a questão de compartilhamento de recursos quando domínios são usados para definir as políticas de *grid*, essa questão é crítica quando as políticas de *grid* são traduzidas para políticas de rede <sup>2</sup>

Considerou-se neste trabalho que a alocação de recursos do *grid* é feita de maneira independente, ou seja, os recursos da rede são alocados para cada um dos elementos definidos na política de *grid* de maneira individual. Entretanto, é possível que o administrador do *grid* defina uma classe de serviço (CoS) onde a banda da rede é compartilhada, como já foi apresentado na especificação das CoS de *grid*. Nesse caso, considera-se que a banda da rede alocada para aquelas aplicações do *grid* é compartilhada entre os demais fluxos da mesma aplicação. A especificação do compartilhamento ou não de banda da rede de uma CoS serve para auxiliar o administrador da rede na interpretação dos requisitos do *grid* e definição das regras de tradução e seu uso será detalhado na seção 3.4 sobre definição das regras de tradução.

<sup>2</sup>Apesar deste trabalho focar no compartilhamento de recursos da rede, o compartilhamento genérico de recursos do *grid* é uma característica que precisa ser considerada em uma linguagem de definições de políticas de *grid* operacional não hipotética.

### 3.3 Linguagem para Definição de Políticas de Rede

Para representar as políticas de rede empregou-se neste trabalho, assim como para representação das políticas de *grid*, uma linguagem que se baseia no modelo de políticas condição/ação do IETF, e assim foi usada como base a mesma BNF apresentada na seção anterior (listagem de código 3.1). Nas regras referentes à rede as expressões condicionais avaliam variáveis que dizem respeito aos fluxos da rede. Já as ações das regras especificam parâmetros de QoS como, por exemplo, a banda que será reservada e o atraso desejado dos pacotes dos fluxos selecionados pelas condições das regras nos dispositivos da rede (roteadores/*switches*).

O modelo de informações PCIME já define sub-classes das classes `PolicyVariable` e `PolicyValue` relacionadas com o filtro de pacotes e reserva de banda em uma rede IP (Essas classes são apresentadas na lista abaixo com uma breve descrição). Na definição das ações da política de rede foi usada a classe `QoSPolicyBandwidthAction`, de um modelo de informações auxiliar padronizado pelo IETF para definição de políticas de QoS, chamado de *Policy QoS Information Model* (SNIR et al., 2003).

- `PolicySourceIPv4Variable`: identifica o endereço de origem;
- `PolicyDestinationIPv4Variable`: identifica endereço destino;
- `PolicySourcePortVariable`: identifica a porta origem;
- `PolicyDestinationPortVariable`: identifica a porta destino;
- `PolicyIPProtocolVariable`: identifica o protocolo de transporte;
- `PolicyDSCPVariable`: identifica o campo *Differentiated Services Code Point* (DSCP);

A listagem de código 3.5 apresenta um exemplo de uma política de rede. Essa política declara que o tráfego originado do *host* 143.54.47.242, enviado para o *host* 143.54.47.17, de qualquer porta origem (\*), endereçado para porta HTTP (porta 80), sobre protocolo TCP, e com qualquer valor DSCP (\*) terá uma reserva de no mínimo 7Mbps e no máximo 10Mbps de banda e será marcado com o valor 46 no campo DS (usado pela arquitetura de QoS DiffServ (BLAKE et al., 1998)). Além disso, a política define o período de validade, que nesse caso é durante todo o dia 25 de novembro de 2004. Na especificação da linguagem não foram usados os nomes de variáveis idênticos aos nomes das classes definidos no PCIME. Por questão de simplicidade foram feitas abreviações: por exemplo, DSCP refere-se à classe `PolicyDSCPVariable` e `srcAddress` refere-se à `PolicySourceIPv4Variable`.

```

if (srcAddress == "143.54.47.242" and
    srcPort == "*" and
    dstAddress == "143.54.47.17" and
    dstPort == "80" and
    DSCP == "*" and proto == "TCP" and
    startTime >= "11/25/2004 00:00:00" and
    endTime <= "11/25/2004 23:59:59")
{
    minBandwidth = 7Mbps;
    maxBandwidth = 10Mbps;
    DSCP = 46;
}

```

Código 3.5: Exemplo de política de rede

Num sistema de gerenciamento baseado em políticas tradicional, seguindo os padrões do IETF, o administrador da rede é encarregado de definir as políticas de rede e, além disso, definir em quais dispositivos da rede essa política deve ser aplicada. Existem ferramentas implementadas

que oferecem facilidades para definição desses elementos, por exemplo, através de uma interface gráfica ou mapa da rede. Nessas ferramentas o administrador da rede define a política através de uma interface específica e seleciona os dispositivos onde a política será aplicada.

O objetivo deste trabalho não é implementar uma solução tradicional de gerenciamento baseado em políticas. Tal solução já foi especificada pelo IETF e encontra-se implementada em algumas ferramentas como, por exemplo, o sistema de gerenciamento baseado em políticas QAME (QoS Aware Management Environment) (GRANVILLE et al., 2001), usado como base para o protótipo implementado neste trabalho. Políticas de rede, no contexto deste trabalho, são somente o resultado do processo de tradução de políticas do modelo apresentado que propõe a integração entre o gerenciamento de *grid* e rede.

### 3.4 Linguagem para Tradução de Políticas de Grid em Políticas de Rede

Para integrar o gerenciamento de *grid* com o gerenciamento de rede foi definida uma linguagem de tradução que gera políticas de rede dado políticas de *grid* e especificações de CoS. Para modelar os elementos dessa linguagem procurou-se identificar quais os passos e informações necessárias ao administrador da rede para criação de uma política de rede. Assim, os passos seguidos pelo administrador da rede para especificar uma política são:

1. Identificação dos fluxos e requisitos de QoS;
2. Criação das políticas de rede com condições relacionadas com os fluxos e ações com parâmetros específicos da arquitetura de QoS do seu domínio;
3. Seleção dos PEPs envolvidos e aplicação da política nesses PEPs.

Em um primeiro momento, o administrador da rede precisa identificar os fluxos e os requisitos de QoS desses fluxos. De posse das definições dos fluxos o administrador da rede define as condições das políticas de rede. Após avaliar os requisitos de QoS dos fluxos o administrador define então, nas ações das políticas, os parâmetros de QoS específicos que precisam ser usados na arquitetura de fornecimento de QoS para suprir esses requisitos. Por fim, o administrador precisa identificar quais os PEPs devem ser configurados e aplicar a política nesses PEPs. Uma regra de tradução deve então conter elementos que permitam ao administrador da rede especificar automaticamente cada um desses passos: avaliação dos fluxos e requisitos das políticas de *grid*, criação de políticas de rede e aplicação das políticas criadas em um conjunto de PEPs.

A especificação dos fluxos e requisitos de QoS do *grid* já encontram-se nas políticas de *grid*, mas não de uma maneira facilmente entendida pelo administrador da rede. Seria necessário que o administrador da rede conhecesse a linguagem de políticas de *grid* e as definições de classes de serviço (CoS) para conseguir interpretá-las e descobrir os fluxos e requisitos de QoS do *grid*. Para automatizar esse processo este trabalho propõe um algoritmo, que recebe como entrada as políticas e as CoS do *grid*, e apresenta para o administrador da rede uma lista dos pares de comunicação do *grid* e a QoS requerida em cada um dos pares.

#### Pares de Comunicação

Um par de comunicação é definido como uma associação entre um dispositivo origem, um dispositivo destino, uma definição de QoS e uma definição de agendamento de quando essa comunicação ocorrerá. É possível, considerando um conjunto de políticas de *grid*, que vários pares de comunicação sejam resolvidos.

A listagem de código 3.6 apresenta um exemplo de política de *grid* e na tabela 3.1 o par de comunicação resolvido a partir dessa política. Nesse exemplo, a política de *grid* especifica um usuário e um *cluster*, mas nenhum *proxy* é usado, portanto o objeto *srcResource* referencia o computador do usuário, e o objeto *dstResource* referencia o *cluster*. Se um *proxy* fosse definido então o objeto *srcResource*, em vez de referenciar o computador do usuário, referenciaria o endereço do *proxy*. Se mais de uma origem e destino é especificada na política de *grid* original, não será resolvido apenas um par de comunicação, mas sim uma lista deles.

Tabela 3.1: Pares de comunicação resolvidos da política

Objeto	Valor
<i>srcResource</i>	Máquina do usuário neisse
<i>dstResource</i>	LabTec Cluster
<i>requiredQoS</i>	<i>remoteProcessControl</i>
<i>schedule</i>	<i>startTime</i> >="11/25/2004 00:00:00" <i>endTime</i> <="11/25/2004 23:59:59"

```

if (user == "neisse" AND
    resource == "LabTec Cluster" AND
    startTime >= "11/25/2004 00:00:00" AND
    endTime <= "11/25/2004 23:59:59"
) {
    allowAccess = True;
    networkCoS = remoteProcessControl;
    maxProcessing = 50%;
}

```

### Código 3.6: Política de grid com resolução dos pares de comunicação

O algoritmo de resolução de pares de comunicação avalia todas as políticas de *grid* aplicadas e resolve os pares de comunicação definidos. Os passos do algoritmo de tradução são:

1. Localizar a próxima ação *networkCoS* na política de *grid*;
2. Criar objeto representando um par de comunicação;
3. Associar a QoS requerida do par de comunicação com o *networkCoS* encontrado;
4. Localizar *startTime* e *endTime* associado ao *networkCoS* na política e associar com o agendamento do par de comunicação;
5. Se existir um *proxy* associado ao *networkCoS* na política então definir o *proxy* como dispositivo origem do par de comunicação senão definir o computador do usuário associado ao *networkCoS* como dispositivo origem do par de comunicação;
6. Associar o dispositivo destino do par de comunicação com o *resource* associado ao *networkCoS* na política de *grid*.
7. Se existir próxima ação *networkCoS* na política de *grid* então ir para passo 1.

Considera-se que um elemento está associado à uma ação *networkCoS* quando ele está definido nas condições da regra que especifica a ação, ou se ele está definido em uma regra externa. No exemplo de política de *grid* apresentado na listagem de código 3.7, a ação *networkCoS* possui associada a ela o usuário *neisse* e o recurso *LabTec Cluster*. Se uma ação tiver mais de um elemento *user* associado, será considerado na resolução do par de comunicação o elemento *user* encontrado na regra mais interna.

```

if (user == "neisse" AND
    startTime >= "11/25/2004 00:00:00" AND
    endTime <= "11/25/2004 23:59:59"
) {
  if (resource == "LabTec Cluster") {
    allowAccess = True;
    networkCoS = remoteProcessControl;
    maxProcessing = 50%;
  }
}

```

### Código 3.7: Política de grid

O administrador da rede referencia um par de comunicação dentro de uma regra de tradução através de quatro objetos pré-definidos, denominados: `srcResource`, `dstResource`, `schedule` e `requiredQoS`. Os objetos `srcResource` e `dstResource` (respectivamente passos 5 e 6 do algoritmo) tem propriedades e métodos para identificar os endereços e portas dos dispositivos origem e destino envolvidos na comunicação. O objeto `schedule` (passo 4 do algoritmo) identifica o tempo em que a comunicação ocorrerá e o objeto `requiredQoS` (passo 3 do algoritmo) identifica a classe de serviço do *grid* associada à comunicação.

Um exemplo de política de rede é apresentada na listagem de código 3.8, assim como os pares de comunicação resolvidos a partir dessa política na tabela 3.2. Nessa política, o usuário `mity` tem acesso direto ao cluster LabTec e o usuário `neisse` tem acesso ao cluster LabTec e, usando o *cluster* como *proxy*, tem acesso ao servidor de dados UFRGS. Todos os acessos serão permitidos em um horário comum, durante todo o dia 25 de novembro de 2004, portanto essa condição está na regra mais externa. Como pode-se observar na tabela 3.2, a partir dessas regras foram resolvidos não somente um par de comunicação mas três.

```

if (startTime >= "11/25/2004 00:00:00" and
    endTime <= "11/25/2004 23:59:59")
{
  if (user == "mity" and
      resource == "LabTec Cluster")
  {
    allowAccess = true;
    maxProcessing = 50%;
    networkCoS = remoteProcessControl;
  }
  if (user == "neisse") {
    if (resource == "LabTec Cluster")
    {
      allowAccess = true;
      maxProcessing = 50%;
      networkCoS = remoteProcessControl;
    }
    if (proxy == "LabTec Cluster" and
        resource == "UFRGS Data Server")
    {
      allowAccess = true;
      maxAllowedStorage = 40GB;
      networkCoS = highThroughputDataIntensive;
    }
  }
}

```

### Código 3.8: Exemplo de políticas de grid

Considerando a topologia de rede da figura 3.5, e a política apresentada na listagem de código 3.8, dois caminhos diferentes são usados na aplicação das classes de serviço (`remoteProcessControl` e `highThroughputDataIntensive`) para o usuário `neisse`. Para reservar uma banda mínima para operação da aplicação de controle remoto de processos, os dispositivos intermediários entre o computador do usuário e o *cluster* LabTec

devem ser configurados. No segundo caso, os dispositivos da rede entre o *cluster* LabTec e o UFRGS Data Server devem ser configurados para suportar a aplicação de transferência de dados com grande vazão. É importante notar que nenhuma configuração de dispositivos da rede será executada no caminho entre o computador do usuário e o servidor de armazenamento, uma vez que nenhuma política que liga diretamente o usuário ao servidor de armazenamento é definida.

Tabela 3.2: Pares de comunicação resolvidos da política

	Objeto	Valor
1	srcResource dstResource requiredQoS schedule	Máquina do usuário mity LabTec Cluster remoteProcessControl startTime>="11/25/2004 00:00:00" endTime<="11/25/2004 23:59:59"
2	srcResource dstResource requiredQoS schedule	Máquina do usuário neisse LabTec Cluster remoteProcessControl startTime>="11/25/2004 00:00:00" endTime<="11/25/2004 23:59:59"
3	srcResource dstResource requiredQoS schedule	LabTec Cluster UFRGS Data Server highThroughputDataIntensive startTime>="11/25/2004 00:00:00" endTime<="11/25/2004 23:59:59"

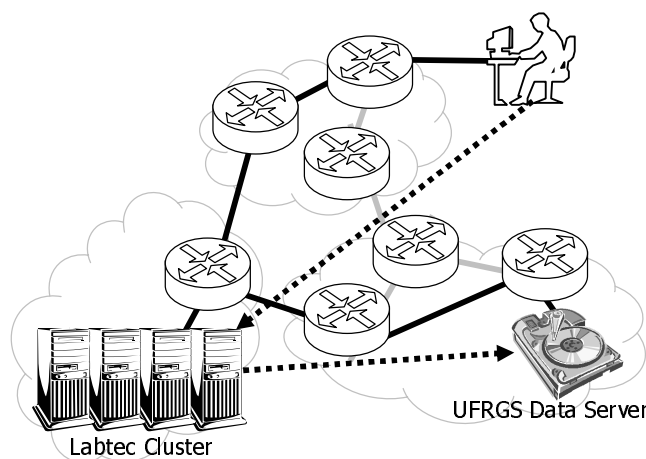


Figura 3.5: Caminhos de reserva de QoS na rede

Os endereços de rede (IP) dos dispositivos envolvidos nos pares de comunicação não são definidos nas políticas de *grid*, somente um identificador dos recursos é usado. Para definir as políticas de rede o administrador da rede precisa dos endereços, protocolos e portas de comunicação desses elementos. Desta forma, para resolver essas informações dos pares de comunicação, é necessária uma consulta ao *toolkit* usado na implementação do *grid*.

Os *toolkits* de *grid*, em geral, implementam serviços para monitoração e descoberta dos recursos compartilhados, através de uma consulta a um serviço. No caso do *toolkit* Globus esse serviço é chamado de *Monitoring and Discovery Service* (MDS). O MDS do *toolkit* Globus é implementado através da tecnologia de Web Services e permite a consulta de propriedades dos

recursos do *grid* (e.g. endereços de rede e protocolos usados) a partir de um nome (chave de consulta).

Algumas vezes, entretanto, endereços e protocolos dos pares de comunicação pode demorar algum tempo para serem resolvidos. Isso acontece porque as informações de endereços e protocolos dos recursos podem não estar disponíveis no *toolkit* no instante de resolução do par de comunicação. Por exemplo, se uma política de *grid* declara que o usuário *neisse* tem o direito de acessar um dos *clusters* do *grid*, o processo de resolução de endereços e protocolos precisará acessar o *toolkit* do *grid* para descobrir informações de endereços e protocolos do computador do usuário e a máquina de acesso (*front-end*) ao *cluster* selecionado. A política de *grid* da listagem de código 3.9 apresenta esse caso.

```

if (user == "neisse" AND
    resource == cluster(2) AND
    startTime >= "11/25/2004 00:00:00" AND
    endTime <= "11/25/2004 23:59:59"
) {
    allowAccess = True;
    networkCoS = remoteProcessControl;
    maxProcessing = 50%;
}

```

### Código 3.9: Política de grid com domínios

Na política da listagem de código 3.9 além de ser necessária a resolução em tempo de avaliação da política do endereço de rede do usuário *neisse* é preciso também determinar quais *clusters* serão alocados. Isso ocorre porque a política utiliza o conceito de domínios na sua definição. A partir dessa política dois pares de comunicação serão criados pois dois clusters serão alocados ao usuário e a informação de quais *clusters* foram alocados somente estará disponível no momento em que o usuário tentar acessar os recursos do *grid*.

Uma situação que pode ocorrer na resolução de endereços é que o usuário *neisse* pode acessar o *grid* de qualquer computador, e o *cluster* a ser usado pode ser determinado somente quando o usuário realizar a requisição de acesso aos recursos do *grid*. A definição do endereço de rede do usuário só é feita quando ele se autentica no *grid*. Por sua vez, a definição do endereço dos *clusters* só é feita quando o usuário requerer o uso dos recursos, pois eles serão alocados dinamicamente, e seu endereço será desconhecido até que a decisão de escalonamento seja tomada pelo *toolkit* do *grid*. Pares de comunicação não resolvidos ficam bloqueados na execução das regras de tradução, aguardando até que seja possível determinar os endereços e protocolos através de uma consulta ao serviço de monitoração e descoberta do *toolkit* do *grid*.

A partir dessas considerações, definiu-se que uma regra de tradução é uma função que recebe como entrada uma definição de par de comunicação e possui elementos que permitem ao administrador da rede definir: expressões condicionais usando as definições dos pares de comunicação, a criação de políticas de rede, a seleção de dispositivos da rede e aplicação das políticas de rede criadas nesses dispositivos. A figura 3.6 apresenta uma visão geral do modelo de tradução de políticas. Nesse modelo as políticas e classes de serviço de *grid* são resolvidas para pares de comunicação que servem de entrada para as regras de tradução. Por sua vez, as regras de tradução são executadas e geram como resultado um conjunto de políticas de rede.

### Definição das Regras de Tradução

Novas regras de tradução são definidas considerando como entrada os quatro objetos que endereçam os pares de comunicação derivados das políticas de *grid*. Uma linguagem orientada a objetos apresentada a seguir é usada para criar as regras de tradução, que são muito similares às políticas de *grid* e rede já apresentadas, com exceção de que nesse caso as regras controlam o processo de tradução. Desse modo, as regras de tradução podem ser consideradas meta-políticas



que governam o processo de tradução de políticas de *grid* para políticas de rede.

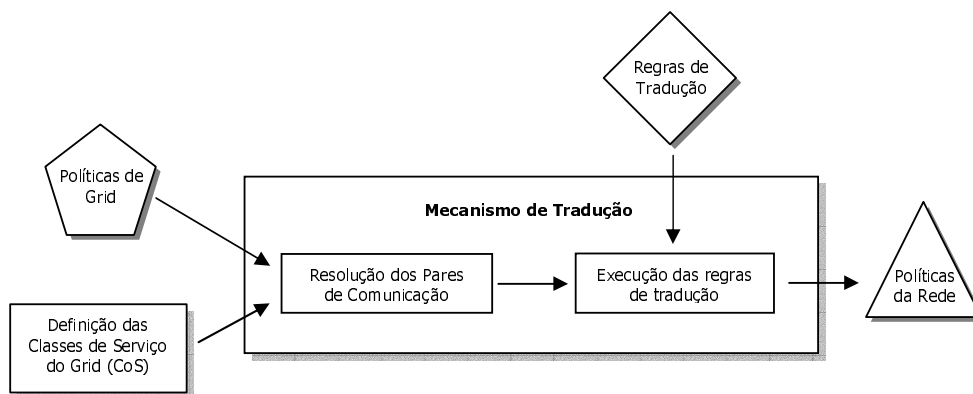


Figura 3.6: Modelo de tradução de políticas

A listagem de código 3.10 apresenta a BNF da linguagem usada para definição das regras de tradução. Definiu-se nessa BNF que uma regra de tradução é formada por condições e ações. Uma condição é uma única expressão condicional ou uma lista de expressões condicionais que podem ser ligadas pelo operador "e" lógico (and). Cada expressão condicional é uma igualdade (=) ou desigualdade (<, <=, >, >=, !=) entre uma variável e um valor. Nas ações de uma regra de tradução pode ser declarada a criação de objeto ou a chamada de um método em um objeto existente. Como estruturas de criação de um objeto é possível instanciar uma classe (objectCreation) ou criar um domínio dinâmico (domain).

```
translationRule ::= if conditions actions;
conditions ::= condition [and condition]
condition ::= trvariable {==,<,<=,>,>=,!} trvalue
actions ::= action [; action]
action ::= {objectCreation, domain, methodCall};
objectCreation ::= object = new ObjectClass()
domain ::= object = select expression
methodCall ::= {object, domain}.method(parameters);
```

Código 3.10: BNF simplificada das regras de tradução

Uma regra de tradução é composta por uma estrutura de condições e ações que manipula os quatro objetos representando um par de comunicação do *grid*, e fornece como resultado um conjunto de políticas de rede aplicadas em um conjunto de PEPs. A criação de políticas de rede é feita nas ações das regras de tradução através de uma biblioteca de classes que permite a criação dinâmica de políticas de rede. O suporte à criação de domínios dinâmicos tem como objetivo fornecer ao administrador da rede uma forma de selecionar PEPs para aplicação das políticas. O suporte à criação de domínios dinâmicos foi desenvolvido por Ceccon (CECCON, 2003), em outra dissertação de mestrado, e foi estendido para atender os requisitos deste trabalho. Detalhes específicos sobre as extensões realizadas serão vistos mais adiante neste capítulo.

A listagem de código 3.11 apresenta um exemplo simples de uma regra de tradução. Nessa regra definiu-se que se existir, nas políticas de *grid*, um par de comunicação onde o endereço do recurso origem está na sub-rede 143.54.47.0/24, seja usado o protocolo de transporte HTTP (port=TCP e protocol=80) e a banda necessária para comunicação seja menor ou igual a 1Mb, então deve-se criar uma política de rede no domínio local para reservar a banda especificada para esse fluxo. Não foi apresentada nessa política o uso dos domínios dinâmicos.

Como pode ser observado no exemplo da listagem de código 3.11, para possibilitar a criação de políticas de rede o administrador precisa primeiramente instanciar um objeto da classe *NetworkPolicy*, e proceder com a manipulação do seu conteúdo para definir as

condições e ações da política de rede. Os métodos `addCondition` e `addAction` da classe `NetworkPolicy` permitem a construção dessa nova política. Uma vez criada através do seu objeto `NetworkPolicy` a política de rede é armazenada num repositório local do domínio administrativo.

```

if (srcResource.address/24 == 143.54.47.0/24 and
    dstResource.port == 80 and
    dstResource.protocol == TCP and
    requiredQoS.bandwidth <= 1Mb)
{
    p1 = new NetworkPolicy();
    p1.addCondition(srcAddress, "=", srcResource.address);
    p1.addCondition(dstPort, "=", dstResource.port);
    p2.addAction(bandwidth, requiredQoS.bandwidth);
}

```

### Código 3.11: Regra de tradução

A listagem de código 3.12 apresenta um exemplo de uma regra de tradução que a partir de um conjunto de condições cria duas novas políticas de rede. Essa regra de tradução define as políticas de rede `p1` e `p2` de forma a marcar pacotes e alocar banda em uma rede subjacente, tipicamente operando com a arquitetura `DiffServ` do IETF (BLAKE et al., 1998). Entretanto, `p1` e `p2` são somente criadas se a política de *grid* original declarar que o recurso origem está localizado na rede local (143.54.47.0/24) e o recurso destino pertence à outra rede, diferente da local. A política de rede `p1` verifica o endereço local e remoto, a porta remota (80), e o protocolo de transporte (TCP) dos pacotes do fluxo e marca o campo DS (*Differentiated Services Field*) dos pacotes com o DSCP 2. A política `p2`, por sua vez, somente verifica o DSCP para garantir a largura de banda requerida determinada na política de *grid* original.

```

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, 46);

    p2 = new NetworkPolicy();
    p2.addCondition(startTime, ">=", schedule.startTime);
    p2.addCondition(endTime, "<=", schedule.endTime);
    p2.addCondition(DSCP, 2);
    p2.addAction(bandwidth, requiredQoS.requiredBandwidth);
}

```

### Código 3.12: Regra de tradução

Uma regra de tradução definida pelo administrador da rede é avaliada pelo mecanismo de tradução, citado no início deste capítulo, que repassa para a regra um único par de comunicação, representado pelos quatro objetos previamente apresentados. Quando uma política de *grid* define mais de um par de comunicação, essa lista de pares é passada ao mecanismo de tradução que executa individualmente todas as regras de tradução definidas para cada par de comunicação resolvido. A listagem de código 3.13 apresenta o algoritmo usado pelo mecanismo de tradução para execução das regras. Para cada política de *grid* aplicada, uma lista de pares de comunicação é resolvida e para cada um dos pares de comunicação todas as regras de tradução são avaliadas.

Após a execução das regras de tradução, o mecanismo de tradução fornece um conjunto de novas políticas de rede para serem aplicadas no domínio.

```
function onDeployGridPolicy (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]);
        }
    }
}
```

Código 3.13: Algoritmo para execução das regras de tradução

### Suporte aos Domínios Dinâmicos para Seleção de PEPs

Em um sistema de gerenciamento baseado em políticas convencional, o administrador da rede é a pessoa responsável por determinar em quais dispositivos (PEPs) da rede gerenciada as políticas serão aplicadas. A seleção desses dispositivos dispara a aplicação da política, apesar de as políticas só serem ativadas durante o período agendado, devido as restrições temporais definidas nas próprias condições das regras da política. Foi necessário então fornecer um suporte para que fosse possível determinar de maneira automática em quais PEPs as políticas criadas nas regras de tradução fossem aplicadas.

Esperar que o administrador da rede selecione os PEPs manualmente para aplicação das políticas não é obviamente uma solução, pois isso implicaria em um bloqueio do processo de tradução aguardando pela intervenção do administrador da rede. Portanto, um mecanismo para suportar a seleção dos PEPs alvo da rede deve ser fornecida para automatizar esse processo. Esse mecanismo foi fornecido pela introdução na linguagem de regras de tradução o suporte à domínios dinâmicos (CECCON et al., 2003). Tais domínios são definidos através de seleções de expressão introduzidas nas regras de tradução, já apresentadas na BNF das regras de tradução. Foi usada nesse trabalho uma linguagem desenvolvida por Ceccon (CECCON, 2003) em sua dissertação de mestrado, denominada Linguagem de Criação de Domínios Dinâmicos. Por questão de simplicidade essa linguagem é também denominada nesse trabalho de "linguagem de seleção de PEPs".

Através dessa linguagem, usando um modelo de informação pré-definido, é possível selecionar um conjunto de elementos gerenciáveis a partir de suas características. A figura 3.7 apresenta o modelo de informações sobre o qual são realizadas as consultas através da linguagem. Foi necessário estender o modelo original desenvolvido pelo autor da linguagem, pois o modelo original não suportava o conceito de PEPs, elemento necessário para aplicação de políticas no contexto deste trabalho. Foi adicionado ao modelo então uma classe denominada "peps", que possui somente o atributo "direction". Além dessa classe foi incluído também um relacionamento da classe "peps" com a classe "interface", onde representou-se que cada interface de um dispositivo pode possuir dois PEPs: um com direção de entrada (in) e outro com direção de saída (out) do fluxo de dados.

A linguagem de seleção, por definição, foi desenvolvida para operar de acordo com um conjunto extensível de classes e atributos. Para estender sua funcionalidade bastou implementar classes adicionais, representando os novos elementos necessários, e incluí-las no modelo de informação da linguagem. Feito isso foi possível usar a linguagem para selecionar qualquer tipo de classes e valores que forem necessários.

A listagem de código 3.15 apresenta a BNF da linguagem de seleção, onde o autor definiu os seguintes elementos: *class*, *attribute*, e *value*. O elemento *class* é usado para identificar uma classe do modelo de informação, por exemplo: *devices*, *interfaces* ou

peps. O elemento `attribute` é usado para identificar um atributo de uma classe do modelo de informações e o elemento `value`, por sua vez, é usado para selecionar os objetos (ocorrências) de uma classe que possuem um determinado atributo com um valor específico.

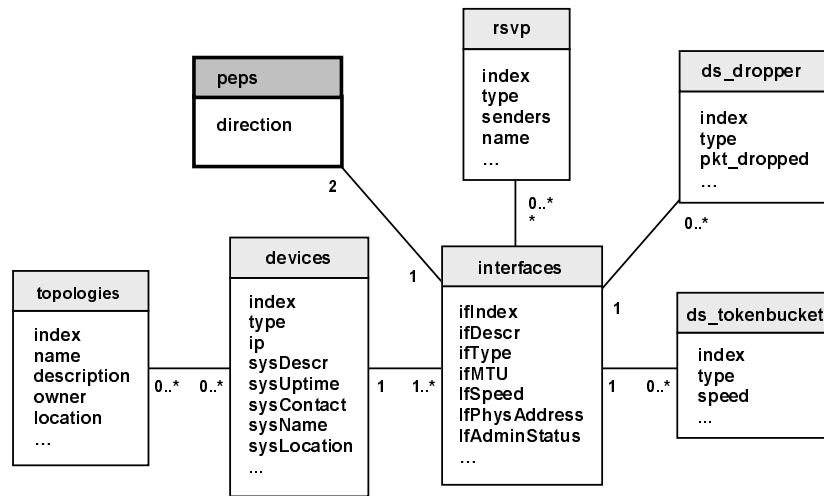


Figura 3.7: Modelo de informação estendido

```

domain ::= select expression
expression ::= term {from term}
term ::= classdata {::classdata}
classdata ::= class { .attribute[value] }

```

Código 3.14: BNF da linguagem de criação de domínios dinâmicos

Para melhor compreensão do uso da linguagem alguns exemplos serão apresentados a seguir, considerando o modelo de informação apresentado na figura 3.7. A primeira expressão (1) cria um domínio dinâmico composto por todos os objetos "peps"(ocorrências) existentes no ambiente gerenciado que possuem a direção de entrada (in). A segunda expressão (2) seleciona todos os objetos "peps"de saída (out) somente das interfaces dos dispositivos que implementam a tecnologia DiffServ (type=DiffServRouter). A última expressão da listagem de código (3) seleciona somente um PEP do ambiente gerenciado, da interface nomeada "eth0"do dispositivo que responde pelo endereço de rede (IP) 143.54.47.1, mas somente se o dispositivo em questão possuir a tecnologia de QoS IntServ (type=IntServRouter).

```

1 - select peps.direction["in"];

2 - select peps.direction["out"]
   from devices.type["DiffServRouter"].interfaces;

3 - select peps.direction["in"]
   from devices.type["IntServRouter"]
      .ip["143.54.47.1"]
      .interfaces["eth0"];

```

Código 3.15: Exemplos de expressões de seleção de PEPs

Além de selecionar um PEP específico é interessante que exista no mecanismo de seleção uma maneira de especificar um conjunto de PEPs no caminho entre dois dispositivos. No exemplo da listagem de código 3.16, a política `p1` é aplicada no PEP de entrada das interfaces do primeiro roteador no caminho entre o dispositivo origem (`srcResource.address`) e o dispositivo

com endereço IP igual a 143.54.47.1. Já a política `p2` é aplicada nos PEPs de saída de todas as interfaces de todos os dispositivos no mesmo caminho entre `srcResource.address` e 143.54.47.1 (incluindo o primeiro dispositivo). As expressões de seleção usam os atributos `type` e `within` do objeto `device` para selecionar os dispositivos com suporte DiffServ no caminho entre os endereços de rede. A seleção dos PEPs de entrada e saída é feita usando o atributo `direction` da classe `pep`.

```

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();
    ...
    inPEPs = select peps.direction["in"]
              from from device.type["DiffServDevice"]
              .within[srcResource.address, 143.54.47.1]
              .interfaces;

    inPEPs[0].deploy(p1);

    p2 = new NetworkPolicy();
    ...
    outPEPs = select peps.direction["out"]
                  from devices.type["DiffServDevice"]
                  .within[srcResource.address, 143.54.47.1]
                  .interfaces;

    outPEPs.deploy(p2);
}

```

Código 3.16: Regra de tradução com aplicação da política de rede

O resultado de uma seleção é um recipiente que possui uma lista de PEPs. O método `deployPolicy` do recipiente aplica uma política em todos os PEPs listados, portanto automatizando o processo de aplicação de políticas na rede. No exemplo da listagem de código 3.16, o recipiente `inPEPs` é usado para aplicar a política `p1` na primeira interface de entrada (`inPEPs[0]`) com o objetivo de marcar os pacotes. O recipiente `outPEPs`, por sua vez, é usado para configurar todas as interfaces de saída do endereço IP origem até o *gateway* local da rede (no exemplo, o endereço IP do *gateway* local é 143.54.47.1).

### 3.5 Modelo para Suporte à Tradução de Políticas de Grid em Políticas de Rede

O modelo para tradução de políticas de *grid* para políticas de rede, que suporta as linguagens de políticas e regras de tradução previamente discutidas, é apresentado na figura 3.8. O modelo é composto por um domínio administrativo de rede e um domínio administrativo de *grid*. O domínio administrativo de *grid* possui um administrador de *grid*, que define as políticas de *grid* usando um editor de políticas baseado na Web, e armazena essas políticas em um repositório. O *toolkit* de *grid* é o software responsável pelo gerenciamento dos recursos compartilhados no *grid*, e recebe como entrada as políticas de *grid* que governam a utilização dos recursos compartilhados no *grid* pelos usuários.

No domínio administrativo de rede existe a figura do administrador da rede, que define, também através de uma interface baseada na Web, as regras de tradução. As regras de tradução são armazenadas em um repositório local de regras. Quando uma política de *grid* é aplicada o mecanismo de tradução é sinalizado e as regras de tradução são executadas. As políticas de rede resultantes da execução das regras de tradução são armazenadas em um repositório de políticas de rede, seguindo os padrões do IETF. A arquitetura de gerenciamento de redes baseado em políticas (PBNM) do IETF, onde foram padronizados PDPs e PEPs, é usada no modelo de tradução de políticas.

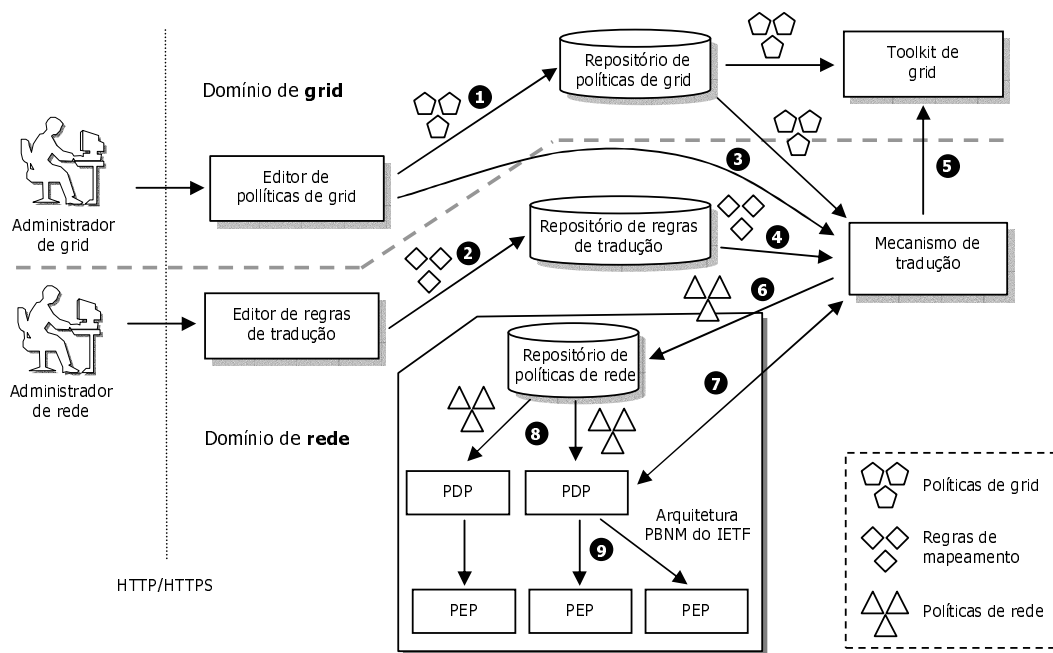


Figura 3.8: Modelo de tradução de políticas

Considera-se que somente um administrador do *grid* é responsável pela criação de políticas usando a linguagem de políticas de *grid* previamente apresentada. Embora a figura apresente somente um administrador da rede, é importante notar que vários administradores da rede podem interagir no modelo para definir as regras de tradução. Regras de tradução de políticas de *grid* para políticas de rede devem ser especificadas para cada domínio participante do *grid*, pois as políticas de rede resultantes são dependentes da tecnologia e também da política de operação da rede de cada domínio (NEISSE et al., 2004b).

Os passos na tradução de uma política de *grid* do modelo da figura 3.8 foram identificados com números de 1 até 9, detalhados a seguir:

1. O administrador do *grid* define as políticas do *grid* e as classes de serviço da rede associadas necessárias através de um editor de políticas do *grid*. As políticas do *grid* e classes de serviço da rede são armazenadas em um repositório global de políticas do *grid*;
2. O administrador da rede de cada domínio administrativo define um conjunto de regras de tradução usando um editor de regras de tradução. Tais regras são então armazenadas em um repositório local de regras;
3. Uma vez que o administrador do *grid* deseje aplicar uma política, o mecanismo de tradução recupera a política do repositório global de políticas;
4. O mecanismo de tradução recupera o conjunto de regras de tradução do repositório local de regras;
5. O mecanismo de tradução traduz as políticas de *grid* usando as regras de tradução e consulta o *toolkit* para descobrir as informações de endereços e protocolos de rede.
6. Quando o mecanismo de tradução constrói novas políticas de rede relacionadas com o domínio local, essas políticas são armazenadas em um repositório local de políticas de rede;
7. Então, o mecanismo de tradução sinaliza um conjunto de PDPs no domínio local para aplicar as políticas de rede criadas em um conjunto de PEPs;

8. Os PDPs sinalizados recuperam as políticas de rede do repositório local;
9. Os PDPs traduzem as políticas de rede para ações de configuração com o objetivo de aplicar tais políticas nos PEPs do domínio local.

## 4 IMPLEMENTAÇÃO DO PROTÓTIPO

A partir do modelo de gerenciamento integrado, apresentado no capítulo anterior, foi implementado um protótipo, que é apresentado neste capítulo. O protótipo é parte integrante do ambiente QAME (*QoS Aware Management Environment*) (GRANVILLE et al., 2001), um sistema de gerenciamento de redes baseado na Web que está sendo desenvolvido no Grupo de Pesquisa em Redes de Computadores da Universidade Federal do Rio Grande do Sul (UFRGS).

A implementação do protótipo foi dividida em dois módulos principais, ambos acessíveis através da Web: um módulo de gerenciamento de *grid* e um módulo de gerenciamento integrado de *grid* e rede. No módulo de gerenciamento de *grid* é fornecida uma interface para o administrador de *grid* definir as classes de serviço e as políticas de operação de *grid*. O módulo de gerenciamento integrado de *grid* e rede permite que o administrador de rede defina as regras de tradução necessárias para configurar o suporte de comunicação requerido pelo *grid* na rede.

A primeira seção deste capítulo apresenta as tecnologias e bibliotecas usadas na implementação do protótipo. A segunda seção apresenta as extensões feitas ao modelo de informação PCIME para implementação do modelo de tradução de políticas proposto usando as linguagens apresentadas no capítulo anterior. Por fim, as duas últimas seções apresentam, respectivamente, os detalhes operacionais do módulo de gerenciamento de *grid* e do módulo de gerenciamento integrado de *grid* e rede.

### 4.1 Detalhes de implementação

A figura 4.1 apresenta as tecnologias usadas na implementação do protótipo de cada um dos elementos do modelo de gerenciamento proposto. O editor de políticas e classes de serviço do *grid*, o editor de regras de tradução de rede e o mecanismo de tradução foram implementados em PHP (PHP, 2004) em um servidor HTTP Apache. Optou-se pela linguagem PHP por questões de compatibilidade, pois PHP foi a linguagem adotada como padrão no desenvolvimento do sistema QAME.

O *toolkit* de *grid*, situado na parte superior direita da figura 4.1, foi implementado como um Web Service, padrão de comunicação usado pela última versão do *toolkit* Globus (GT3). Esse Web service simula o serviço de monitoração e descoberta (MDS - *Monitoring and Discovery Service*) do *toolkit* Globus. Optou-se por esse serviço simulado pois no momento da implementação do protótipo não havia um *toolkit* operacional disponível. Entretanto, espera-se que em uma aplicação real do modelo de gerenciamento as consultas sejam realizadas ao MDS de um *grid* operacional.

O repositório de políticas de *grid*, assim como o repositório de políticas de rede, foi implementado em um servidor OpenLDAP (OPENLDAP, 2004). Optou-se pelo serviço de diretórios LDAP pois é a tecnologia de implementação sugerida pelo IETF para o repositório de políticas. O modelo de informações usado no repositório foi o PCIME, com algumas extensões para representação de elementos das políticas de *grid* não suportados originalmente. As extensões feitas ao modelo PCIME são apresentadas na próxima seção. Por questões de simplicidade na implementação as regras de tradução não foram armazenadas no LDAP, mas em arquivos texto.



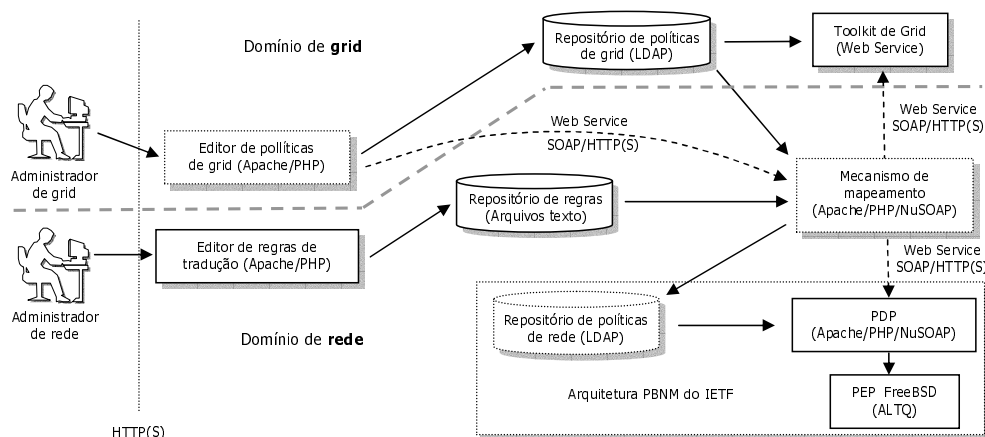


Figura 4.1: Implementação do protótipo

Como pode-se observar na figura 4.1, optou-se pela tecnologia de Web Services para comunicação entre (A) o editor de políticas de *grid* e o mecanismo de tradução, (B) entre o mecanismo de tradução e o *toolkit* de *grid* e (C) entre o mecanismo de tradução e os PDPs. Para suportar o uso de Web Services em PHP na implementação optou-se pela biblioteca NuSOAP (AYALA, 2004). A biblioteca NuSOAP fornece facilidades tanto para implementação de Web Services quanto para fazer chamadas aos Web Services desenvolvidos.

A implementação da arquitetura PBNM do IETF, apresentada na parte inferior da figura 4.1, não faz parte da implementação do protótipo deste trabalho. O sistema QAME já possui implementado os módulos de um sistema de gerenciamento de redes baseado em políticas seguindo os padrões do IETF. Portanto, os PDPs e o repositório de políticas de rede já encontravam-se implementados no sistema QAME, e foram somente reaproveitados na implementação do protótipo deste trabalho.

A figura 4.3 apresenta um exemplo de aplicação do protótipo para o gerenciamento integrado de *grid* e rede. Considerou-se, no exemplo da figura, que o *grid* encontra-se distribuído através de três domínios administrativos de rede independentes. Para que o modelo proposto funcione é necessário que o administrador do *grid* defina as classes de serviço e políticas de operação do *grid* através do módulo de gerenciamento de *grid* e que os administradores de rede definam, em todos os domínios administrativos, regras de tradução usando o módulo de gerenciamento integrado de *grid* e rede.

## 4.2 Modelo de informações e interfaces

Para definição das políticas de *grid* e políticas de rede foram usadas algumas classes definidas no modelo de informação PCIME, assim como algumas extensões, apresentadas a seguir nesta seção. Foram necessárias extensões pois o modelo de informações PCIME não suportava a definição de variáveis e valores usados nas condições e ações das políticas de *grid*, conforme a linguagem de políticas de *grid* apresentada no capítulo anterior. As classes necessárias pela linguagem de definição de regras de tradução e as classes auxiliares usadas no modelo de tradução de políticas também são detalhadas nesta seção.

A figura 4.2 apresenta o diagrama de classes usando a notação UML das extensões feitas ao modelo de informação PCIME para suportar a definição de políticas de *grid*. A classe `PolicyVariable` foi estendida para definição de condições relacionadas aos usuários (`User`), proxies (`Proxy`) e recursos (`Resource`). Foram necessárias extensões para suportar a definição de ações de controle de acesso aos recursos (`AllowAccess`), classes de serviço de rede (`NetworkCoS`), reservas de processador (`MaxProcessing`) e reserva de espaço em disco

(MaxAllowedStorage). Além disso, foi necessário estender a classe PolicyValue para suportar valores de identificadores de recursos do *grid* (GridResourceValue) e valores de definição de classes de serviço da rede (NetworkCoSValue). Para as demais variáveis os valores necessários já encontravam-se definidos no modelo PCIME (BooleanValue e IntegerValue). Foi definida também uma classe GridPolicy que considera as validações necessárias de uma política de *grid* conforme a linguagem de definição de políticas de *grid*.

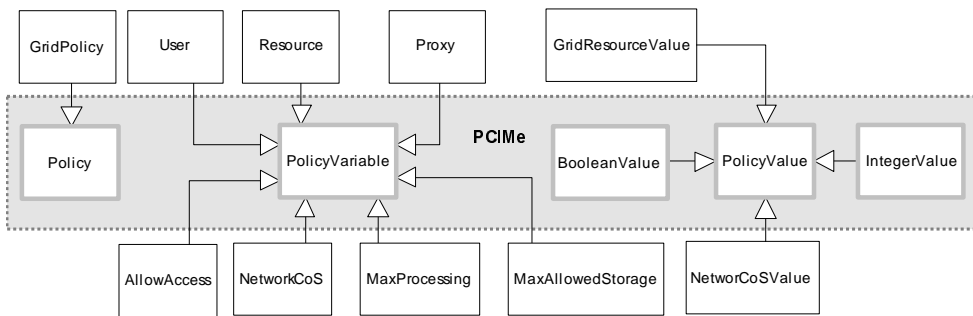


Figura 4.2: Modelo UML das condições e ações das políticas de rede

A figura 4.4 apresenta o diagrama das classes usadas para definição das políticas de rede. Nesse caso somente foi necessário definir a classe NetworkPolicy, que trata das validações das políticas de rede conforme a linguagem de políticas de rede também apresentada no capítulo anterior. As demais variáveis e valores necessários pela linguagem já encontravam-se definidos no modelo PCIME e PQIM (Policy QoS Information Model). As classes definidas até o presente momento para representação de políticas de *grid* e rede foram mapeadas para o repositório de políticas seguindo um esquema LDAP padronizado pelo IETF ??.

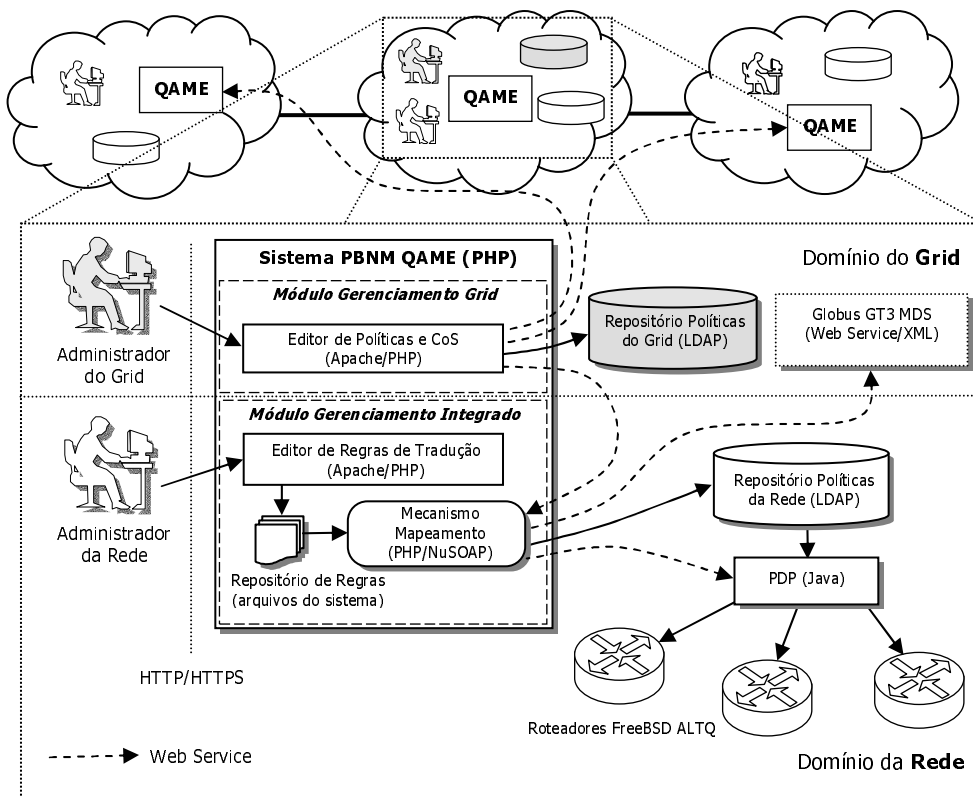


Figura 4.3: Implementação do protótipo

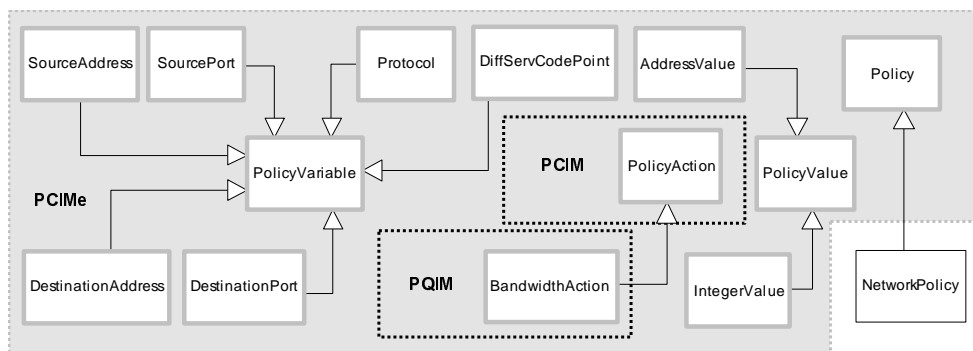


Figura 4.4: Modelo UML das condições e ações das políticas de grid

O diagrama de classes da figura 4.5 apresenta as classes definidas para suportar o modelo de tradução das políticas de *grid* para políticas de rede. São apresentadas nesse modelo algumas classes usadas na definição das políticas de *grid* e de rede (**NetworkCoSValue**, **GridResourceValue** e **PolicyTimePeriodCondition**) relacionados à classe **GridCommunicationPair**, que define os pares de comunicação expressos nas políticas de *grid* (**GridPolicy**). Foram implementadas também classes específicas relacionadas com a definição e execução das regras de tradução (**TranslationEngine**, **TranslationRule** e **PepSelectionExpression**)

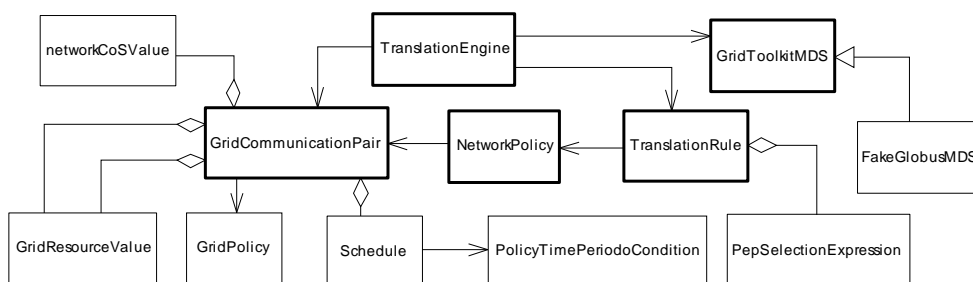


Figura 4.5: Modelo UML das classes responsáveis pela tradução

A interface de comunicação com o toolkit de *grid* é definida no modelo da figura 4.5, em uma classe genérica chamada **GridToolkitMDS**. Essa classe pode ser estendida pra que mais de um tipo de *toolkit* possa ser suportado pelo modelo de gerenciamento proposto. Poderia ser criada uma sub-classe chamada **GlobeMDS**, que fornecesse acesso ao sistema de monitoração e descoberta do toolkit **Globe** (STEEN; HOMBURG; TANENBAUM, 1999). Todas as classes do modelo da figura 4.5, relacionadas com a tradução das políticas de *grid* para políticas de rede, não foram mapeadas para o LDAP, essas classes foram implementadas somente em PHP.

### 4.3 Módulo de Gerenciamento de Grid

O módulo de gerenciamento de *grid* fornece uma interface baseada na Web para que o administrador de *grid* defina as políticas do *grid*, as classes de serviço (CoS) usadas pelas aplicações de *grid* e a localização dos mecanismos de tradução dispersos através dos domínios administrativos de rede participantes do *grid*. Embora o objetivo deste trabalho não seja o de fornecer uma solução específica de gerenciamento de *grid*, foi implementada uma interface simples nesse sentido, como prova de conceito.

A figura 4.6 apresenta o editor de políticas de *grid* implementado. Trata-se de uma página HTML dinâmica, implementada em PHP, que lista as políticas de *grid* existentes no repositório

e fornece ferramentas para criação de novas políticas e edição ou exclusão de uma política já existente. Através da listagem de políticas de *grid* o administrador de *grid* pode também aplicar uma política no *grid* ou removê-la. Convencionou-se no protótipo que uma política que foi aplicada está ativa, enquanto uma que não foi aplicada está inativa. Não foi implementada nenhuma facilidade para seleção de recursos do *grid* onde a política será aplicada. Considera-se nesse caso que a política é aplicada no *grid* em todos os recursos existentes.

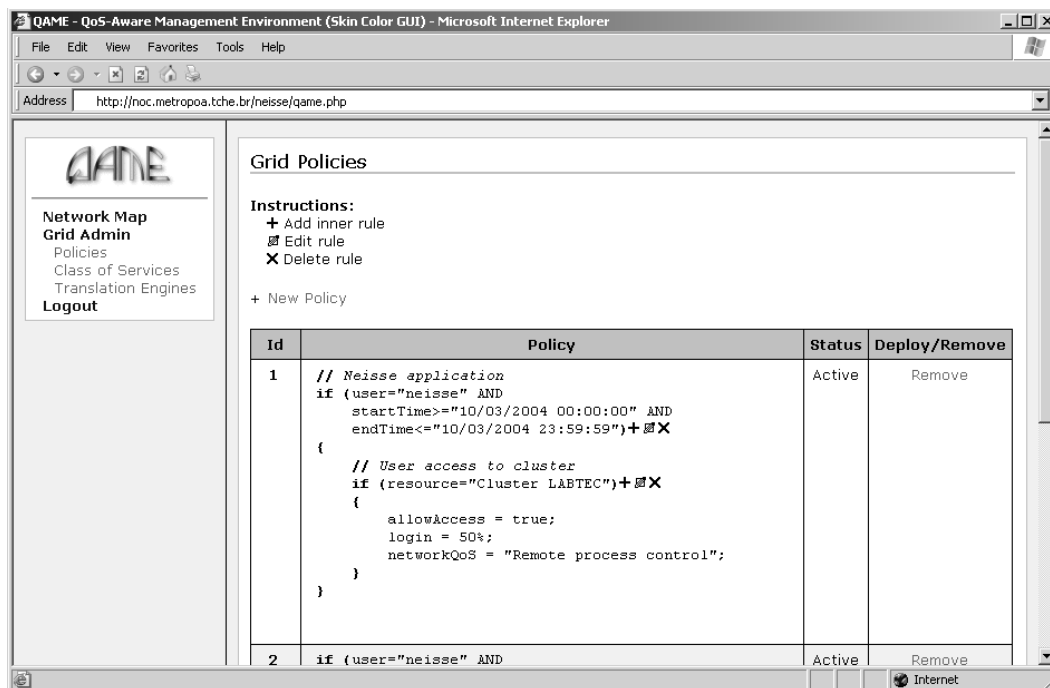


Figura 4.6: Editor de políticas do grid

Além do conceito de ativa e inativa uma política pode ser considerada válida ou inválida pelo sistema. Uma política é válida quando as condições expressas tornarem-se verdadeiras e suas ações são executadas. É possível, por exemplo, que uma política seja definida, esteja válida devido as condições expressas, mas não tenha sido aplicada, portanto, suas ações não são executadas. O conceito de política ativa e válida se aplica tanto para políticas de *grid* quanto para políticas de rede, mas, no que diz respeito as políticas de *grid*, nenhuma consideração nesse sentido é feita na implementação desenvolvida. Espera-se que verificações desse tipo sejam feitas por sistemas de gerenciamento de *grid* baseados em políticas completamente funcionais.

Antes de definir uma política de *grid*, através do editor de políticas de *grid*, o administrador do *grid* deve definir as classes de serviço, que especificam os requisitos básicos dos tipos de aplicação encontrados no *grid*. A definição das classes de serviço serve para criar uma biblioteca de requisitos de rede comuns às aplicações executadas no *grid*. A figura 4.7 apresenta a interface implementada para o gerenciamento das classes de serviço do *grid*.

Através da interface de gerenciamento de classes de serviço o administrador de *grid* pode criar novas classes de serviço e editar ou excluir uma classe de serviço existente. Na criação de uma classe de serviço deve ser especificado um nome, que será usado posteriormente para identificar essa classe de serviço nas políticas, a largura de banda da rede mínima e necessária pela classe, a perda mínima e a máxima aceita, a prioridade, que varia de zero (mínima) a sete (máxima), e um flag indicando se a banda vai ser compartilhada ou não. Os parâmetros definidos para classe de serviço podem ser usados posteriormente pelos administradores de rede de cada domínio na definição das regras de tradução. Essa especificação é importante pois através dela o administrador da rede pode especificar quais os parâmetros locais de sua rede fornecem os parâmetros de qualidade necessários pelos fluxos do *grid*. A classe de serviço é um dos atributos

dos pares de comunicação, derivados das políticas de *grid*, e usados como entrada das regras de tradução.

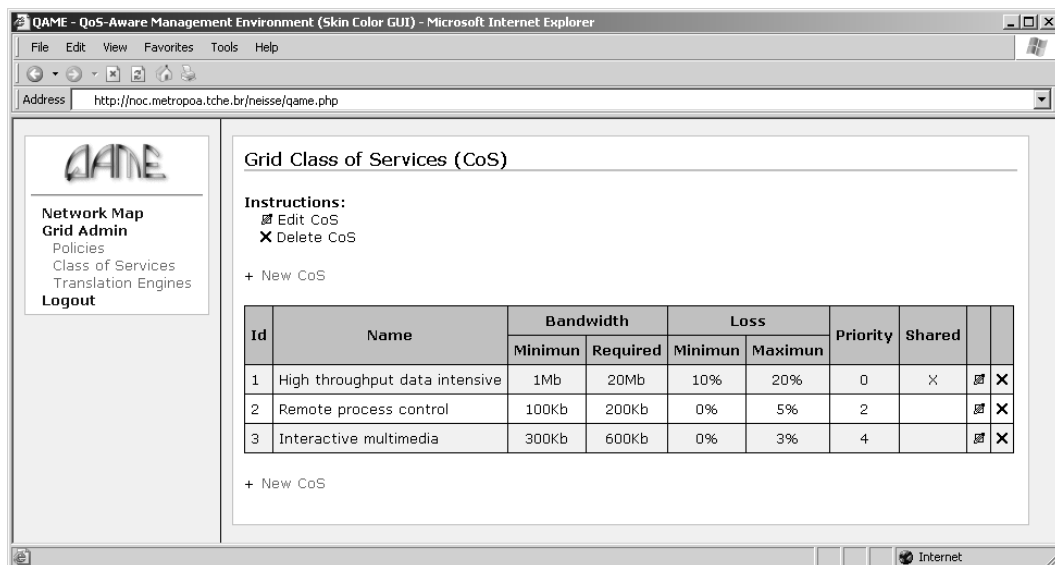


Figura 4.7: Editor de classes de serviço do grid

Como já foi apresentado na primeira seção deste capítulo, quando as políticas de *grid* são criadas e aplicadas o módulo de gerenciamento de *grid* sinaliza todos os mecanismos de tradução que uma nova política de *grid* está disponível para tradução. Para que o módulo de gerenciamento de *grid* faça essa sinalização é necessário configurar a localização de todos os mecanismos de tradução envolvidos. O módulo de gerenciamento de *grid* fornece uma interface para definição dessa localização, apresentada na figura 4.8. O que o administrador de *grid* precisa especificar nessa interface é uma lista de domínios que participam do *grid* e a localização do mecanismos de tradução em cada um desses domínios. Como a interface entre o mecanismos de tradução e o módulo de gerenciamento de *grid* é um Web Service, implementado em SOAP/HTTP, o administrador de *grid* especifica a localização dos mecanismos de tradução informando uma URL HTTP.

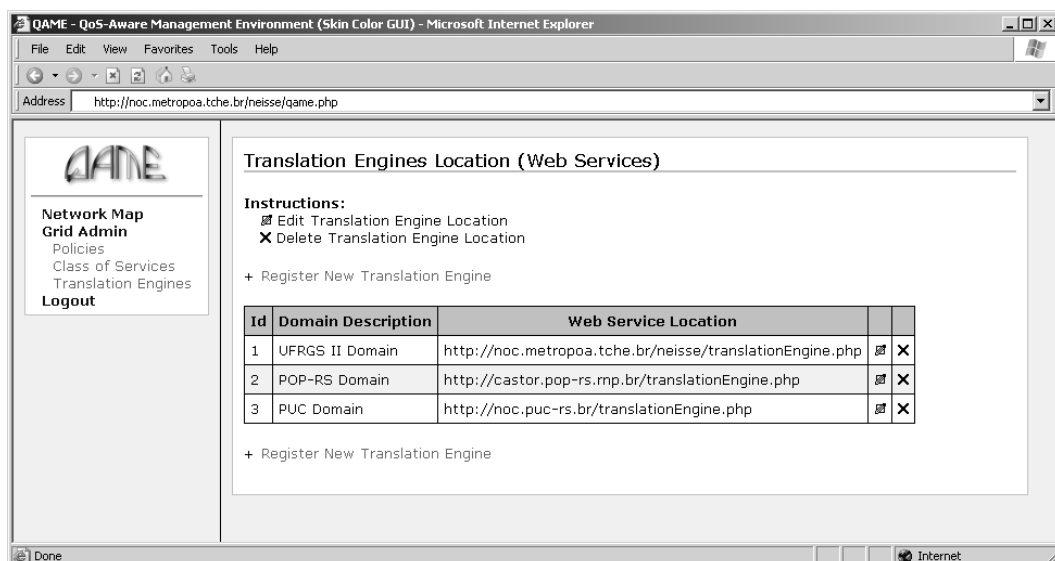


Figura 4.8: Configuração dos mecanismos de tradução

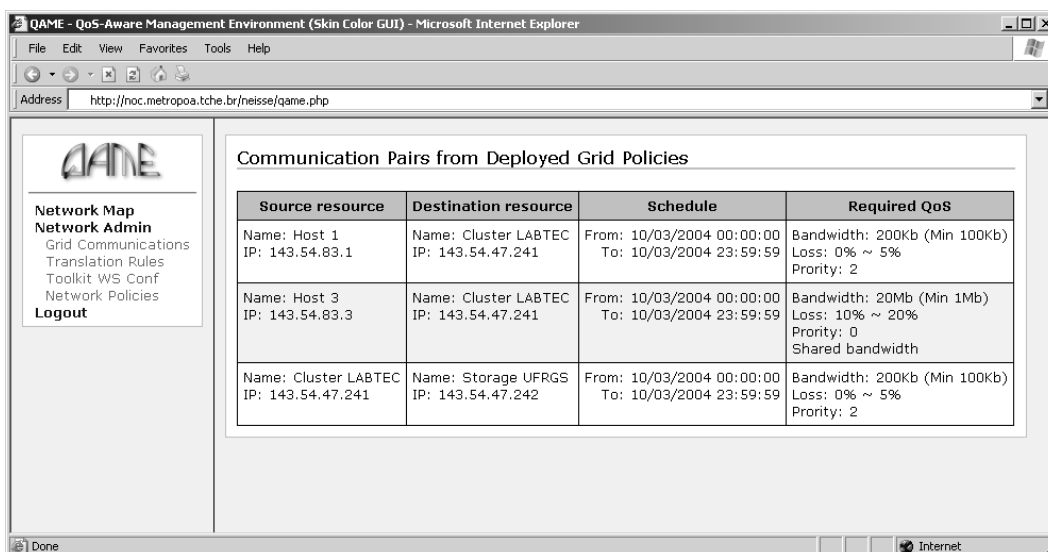
Após a definição da localização dos mecanismos de tradução nos domínios administrativos o administrador do *grid* realiza o gerenciamento da infra-estrutura do *grid* definindo as classes de serviço, definindo políticas de *grid* e aplicando ou removendo as políticas. A lista de mecanismos de tradução só precisa ser alterada se a configuração do *grid* for alterada, por exemplo, com a entrada ou saída de um domínio administrativo da configuração do *grid*.

No protótipo implementado, as políticas e classes de serviço de *grid* servem somente como entrada para que os mecanismos de tradução, dispersos através dos domínios administrativos, configurem a infra-estrutura de rede para operação do *grid*. É interessante que as políticas de *grid* sejam usadas efetivamente para gerenciar outras operações necessárias no *grid*, não relacionadas com a rede, como controle de acesso aos recursos e reservas de disco ou processamento. A linguagem de políticas de *grid* definida neste trabalho suporta essas definições, apesar de que o suporte nos softwares de gerenciamento de *grid* (*toolkits*) para o uso de políticas ainda estar sendo implementado (SUNDARAM; NEBERGALL; TUECKE, 2000).

#### 4.4 Módulo de Gerenciamento Integrado de Grid e Rede

O módulo de gerenciamento integrado de *grid* e rede fornece uma interface baseada na Web para que o administrador de rede defina as regras de tradução de políticas. Além disso, o módulo fornece algumas facilidades para que o administrador da rede visualize o resultado do processo de tradução. As facilidades oferecidas são, por exemplo, uma página com a lista de pares de comunicação definidos nas políticas de *grid* ativas no momento e uma página com a lista de políticas de rede criadas, acompanhadas de uma referência para os pares de comunicação e as regras de tradução que originaram cada uma das políticas de rede.

A figura 4.9 apresenta a interface fornecida no módulo de gerenciamento integrado para que o administrador de rede visualize quais pares de comunicação foram derivados a partir das políticas aplicadas (ativas) no *grid* no momento. Na listagem são apresentados o recurso origem da comunicação, o recurso destino, o horário em que a comunicação está agendada para ocorrer e qual a QoS da rede requerida. Os pares de comunicação são resolvidos a partir das políticas de *grid* aplicadas e são removidos da lista quando a política de *grid* é removida do sistema. Essa lista é interessante pois permite ao administrador da rede visualizar de uma maneira fácil e rápida os pares de comunicação ativos para apoiar a criação das regras de tradução.



The screenshot shows a web browser window titled "QAME - QoS-Aware Management Environment (Skin Color GUI) - Microsoft Internet Explorer". The address bar shows "http://noc.metropoa.tche.br/neisse/qame.php". The main content area displays a table titled "Communication Pairs from Deployed Grid Policies". The table has four columns: Source resource, Destination resource, Schedule, and Required QoS. It lists three communication pairs with their respective source and destination resources, schedules, and QoS requirements.

Source resource	Destination resource	Schedule	Required QoS
Name: Host 1 IP: 143.54.83.1	Name: Cluster LABTEC IP: 143.54.47.241	From: 10/03/2004 00:00:00 To: 10/03/2004 23:59:59	Bandwidth: 200Kb (Min 100Kb) Loss: 0% ~ 5% Priority: 2
Name: Host 3 IP: 143.54.83.3	Name: Cluster LABTEC IP: 143.54.47.241	From: 10/03/2004 00:00:00 To: 10/03/2004 23:59:59	Bandwidth: 20Mb (Min 1Mb) Loss: 10% ~ 20% Priority: 0 Shared bandwidth
Name: Cluster LABTEC IP: 143.54.47.241	Name: Storage UFRGS IP: 143.54.47.242	From: 10/03/2004 00:00:00 To: 10/03/2004 23:59:59	Bandwidth: 200Kb (Min 100Kb) Loss: 0% ~ 5% Priority: 2

Figura 4.9: Visualização dos pares de comunicação do grid

É importante destacar que as informações de endereçamento dos recursos apresentadas na listagem dos pares de comunicação são resolvidas pelo mecanismo de tradução no momento de aplicação da política, através de uma consulta ao *toolkit* de *grid*. Como já foi dito no início desse capítulo, no protótipo implementado essa consulta é realizada em um Web Service que simula o MDS do toolkit Globus (GT3). É possível, entretanto, que no momento de aplicação da política de *grid* o MDS ainda não saiba resolver o endereço de rede do usuário ou dos recursos. Isso pode acontecer em função do usuário não ter feito sua autenticação no *grid* ou devido à política de *grid* estar usando o conceito de domínios para especificação dos recursos. Dessa forma, a listagem de pares de comunicação apresentará uma interrogação (?) no lugar do endereço IP do usuário ou recurso. Além dos endereços o MDS informa também protocolos e portas usadas no acesso aos recursos (ANDREOZZI, 2003).

A resolução dos endereços, protocolos e portas dos pares de comunicação é feita no protótipo pelo mecanismo de tradução através de uma consulta periódica (*polling*) com um intervalo padrão de 5 minutos ao Web Service do *toolkit* (MDS). A configuração do MDS usado para resolução das informações de endereços e do intervalo de *polling* é feita pelo administrador da rede através da interface apresentada na figura 4.10. Nessa interface o administrador da rede informa o endereço do Web Service do MDS que resolve os endereços e protocolos, e também o intervalo de *polling* em minutos. Todas os domínios administrativos de rede que estiverem inseridos no contexto do mesmo *grid* devem usar o mesmo endereço de MDS.

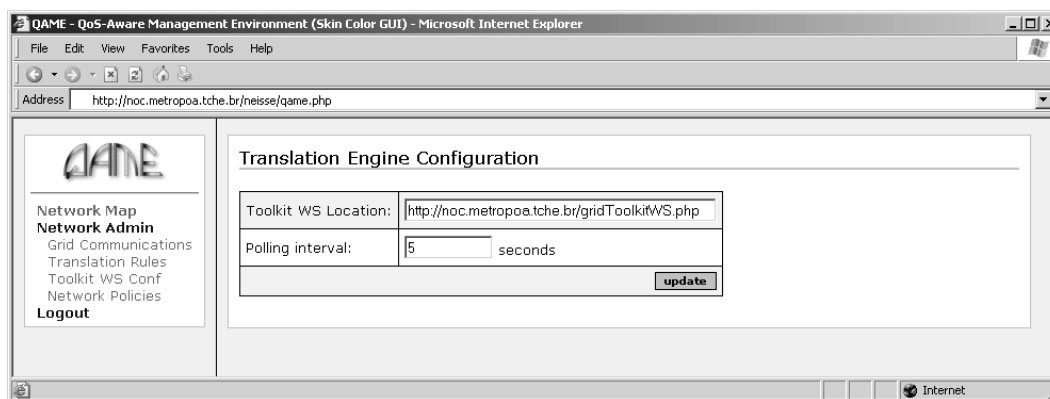


Figura 4.10: Configuração do toolkit no mecanismo de tradução

No protótipo implementado a interface de comunicação entre o mecanismo de tradução e o MDS foi um Web Service, pois o *toolkit* Globus já disponibiliza essa interface de comunicação no seu MDS. Caso seja usado um toolkit onde o MDS seja implementado usando outra interface de comunicação pode-se criar um *gateway* de Web Service (NEISSE et al., 2004) que acesse a informação no MDS usando um protocolo específico. Considerando a criação de *gateways* o protótipo pode ser usado no gerenciamento integrado de *grids* implementados com outros *toolkits* além do Globus.

A definição das regras de tradução é feita no protótipo através da interface ilustrada na figura 4.11. Nessa interface o administrador da rede pode incluir, editar ou excluir uma regra de tradução. A criação de uma regra de tradução é feita a partir da definição de uma regra (estrutura condicional) inicial onde dentro dessa regra podem ser incluídas novas regras, um comando de criação de políticas de rede, um comando de seleção de PEPs e/ou um comando de aplicação de uma política de rede em um conjunto de PEPs previamente selecionados. Para cada uma dessas ações foram definidos assistentes específicos personalizados o que torna mais fácil a operação do protótipo.

As regras de tradução definem dinamicamente através da linguagem de seleção de PEPs o local onde as políticas de rede criadas devem ser aplicadas. Portanto, mesmo que modificações na infra-estrutura da rede sejam feitas as regras de tradução ainda continuam válidas. Para que isso ocorra é necessário que a base de dados de informações de gerenciamento da estrutura da rede, por

exemplo, especificações de enlaces e interfaces dos dispositivos na qual a linguagem de seleção opera, seja mantida atualizada.

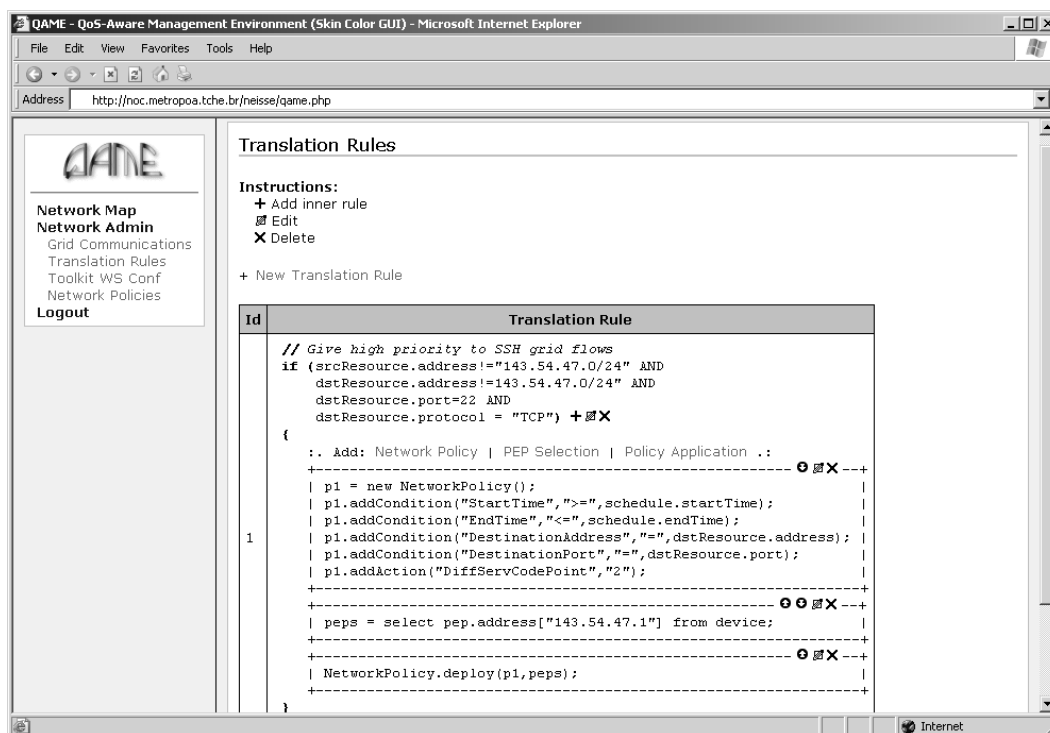


Figura 4.11: Editor de regras de tradução

As regras de tradução são armazenadas em arquivos do sistema operacional através de objetos PHP persistentes. A partir das definições da regra de tradução nesses objetos é criado um script em PHP que especifica a regra de tradução em um sub-conjunto da linguagem PHP. Assim, o mecanismo de execução das regras de tradução é o próprio interpretador PHP que está executando o protótipo desenvolvido. O controle sobre os resultados da execução das regras de tradução é feito através de variáveis estáticas de controle inseridas nas classes implementadas do modelo de informação apresentado na seção 4.2 deste capítulo.

Após a definição das regras de tradução, toda vez que uma política de *grid* for aplicada pelo administrador do *grid*, ela é enviada para todos os mecanismos de tradução através de uma chamada de Web Service. Cada mecanismo de tradução recebe a política de *grid*, verifica os pares de comunicação (endereços, portas e protocolos) definidos, e executa as regras de tradução para cada um desses pares de comunicação. O mecanismo de tradução é um serviço que, após a criação das regras de tradução, não requer nenhuma interação com o administrador da rede para sua operação: trata-se de um processo de gerenciamento autônomo que recebe como entrada as regras de tradução e as políticas de *grid* ativas no sistema e gera como resultado um conjunto de políticas da rede.

Foi implementado no protótipo uma interface, apresentada na figura 4.12, para visualização das políticas de rede criadas pelas regras de tradução. O objetivo dessa interface é permitir que o administrador da rede possa visualizar as políticas de rede criadas e qual a regra de tradução que a originou. Dessa forma, o controle oferecido para que o administrador da rede acompanhe o processo de tradução é bastante detalhado, principalmente para facilitar a depuração das regras de tradução.

Como pode-se observar na figura 4.12, para cada política de rede criada o administrador da rede pode visualizar facilmente além da regra de tradução a política de *grid* e o par de comunicação que gerou a política de rede. Essa informação é interessante para que o administrador da rede possa



analisar o comportamento das suas regras de tradução e verificar quais as políticas de rede estão efetivamente sendo geradas pelo sistema. Sempre que uma política de *grid* ou regra de tradução é removida ou alterada o sistema verifica as políticas de rede relacionadas, ou seja, as políticas de rede criadas em função de uma política de *grid* ou regra de tradução também são alteradas e a lista é atualizada.

Id	Policy	Details	PEP (Status)
1	<pre>if (srcAddress="143.54.47.82/32" AND srcPort="*" AND dstAddress="143.54.47.1/32" AND dstPort="*" AND DSCP="*" AND protocol="TCP" AND startTime="10/03/2004 00:00:00" AND endTime="10/03/2004 23:59:59") {   bandwidth = 10Mbps;   DSCP = 1; }</pre>	Grid Policy Communication Pair Translation Rule	castor.mp.br:eth0/out (active) noc.metroppa.tche.br:sd0/in (failed)
2	<pre>if (srcPort="22" AND dstAddress="143.54.47.1/32" AND dstPort="*" AND protocol="TCP")</pre>		

Figura 4.12: Visualização das políticas de rede criadas

## 5 CONCLUSÕES E TRABALHOS FUTUROS

Este trabalho apresentou um modelo de gerenciamento hierárquico baseado em políticas que traduz políticas de *grid* para políticas de rede através de um mecanismo de tradução. Para operar, esse mecanismo usa regras de tradução, definidas pelos administradores da rede através dos domínios administrativos componentes de um *grid*. Uma vez que cada domínio administrativo tem seu próprio administrador de rede que define as regras de tradução locais e que são diferentes das regras usadas nos outros domínios administrativos, uma única política de *grid* é potencialmente traduzida para diferentes políticas de rede em cada domínio administrativo.

Argumentou-se sobre a definição das políticas do *grid* e um conjunto de elementos foi proposto para permitir uma especificação mais rica dessas políticas. As políticas de *grid* definidas com esses elementos, se comparadas com as políticas encontradas na literatura, permitem expressar regras mais adequadas, principalmente relacionadas com as operações de reservas de recursos da rede. Além disso, tais políticas do *grid* suportam o uso de *proxies* na avaliação dos parâmetros das regras, o que não é encontrado atualmente em outras linguagens de políticas de *grid*.

As regras de tradução, que governam a tradução das políticas de *grid* no modelo podem ser vistas como meta-políticas. Na implementação atual, as regras de tradução são definidas através de um subconjunto da linguagem de *script* PHP. Embora as regras de tradução sejam bastante flexíveis, esta flexibilidade força os administradores da rede a aprender uma nova linguagem para definir traduções mais adequadas. Nesse ponto, assistentes visuais poderiam facilitar a definição de regras de tradução.

A solução de gerenciamento proposta por si só não realiza o gerenciamento integrado do *grid* e da rede. O sucesso na utilização do modelo depende muito da qualidade das regras de tradução definidas. O administrador de rede é livre para definir o conjunto de regras de tradução que ele achar conveniente, o que em muitos casos pode ser pouco eficiente ou não atender os requisitos necessários do *grid*.

Além disso, as políticas de *grid* somente podem ser traduzidas para políticas de rede se o mecanismos de tradução estiverem presentes e devidamente alimentados com as regras de tradução adequadas em todos os domínios administrativos de rede. Esses mecanismos dependem do *toolkit* usado na instalação do *grid*, assim como do sistema de gerenciamento de redes usado pelos administradores da rede. A implementação desenvolvida usa o sistema de gerenciamento de redes QAME e um Web Service que simula o serviço de monitoração e descoberta (MDS) do *toolkit* Globus para resolução dos endereços e protocolos. Se outros sistemas de gerenciamento de redes ou MDSs forem usados a interface de comunicação entre o mecanismo de tradução e os outros elementos do modelo relacionados devem ser respeitadas.

Como resultado deste trabalho foi possível atingir um certo nível de automatização no gerenciamento integrado dos *grids* e da infra-estrutura de rede, pois a rede pode ser configurada sem a intervenção imediata das entidades administrativas. Para que isso ocorra o administrador da rede precisa pre-programar ações que devem ser tomadas quando um par de comunicação for identificado nas políticas de operação do *grid*. Dessa forma ele tem a possibilidade de permitir

determinadas reservas de recursos por parte do *grid* e negar outras, de acordo as políticas locais do domínio.

Como trabalhos futuros, a partir de uma análise do modelo de tradução proposto, pode-se generalizar e definir um modelo de tradução em diversas camadas, o que através de uma modelagem formal representa um novo paradigma de gerenciamento baseado em políticas. Nesse modelo genérico uma camada superior receberia um conjunto de políticas e geraria outro conjunto como resultado, que poderia ser passado como entrada para uma camada inferior. A camada inferior, por sua vez, poderia executar uma nova tradução e produzir um conjunto de políticas para uma camada abaixo, e assim sucessivamente.

Interfaces de usuários mais sofisticadas também podem ser desenvolvidas para reduzir a complexidade na definição das políticas e regras de tradução, por exemplo, através de assistentes. A avaliação do desempenho do modelo também é um ponto que precisa ser verificado. Embora espere-se que o consumo de banda para transferir as políticas de um elemento até outro seja reduzido, uma análise sobre o impacto dessas transferências na rede subjacente precisa ser feita. Mais importante, entretanto, é a avaliação do desempenho do mecanismo de tradução, principalmente considerando a quantidade de políticas de *grid* definidas, os níveis de aninhamentos usados nas políticas, e a quantidade de regras de tradução definidas pelos administradores da rede.

Outro ponto que pode ser abordado no modelo de gerenciamento integrado é a obtenção de informações da infra-estrutura de gerenciamento de rede. Além da configuração da rede em alguns casos também se faz necessária a consulta de parâmetros da rede por parte do *toolkit* de *grid*, para que sejam tomadas decisões de escalonamento com base na carga atual da rede. Na investigação realizada constatou-se que este ponto também não é abordado pelos *toolkits* de gerenciamento de *grids*. Para tornar essa comunicação possível seria necessário definir interfaces padronizadas de comunicação entre as ferramentas de gerenciamento de redes e os *toolkits*, para que as informações necessárias fossem obtidas.

## REFERÊNCIAS

ACCESSGRID. **The Access Grid Project Documentation**. Disponível em: <<http://www.accessgrid.org/documentation/index.html>>. Acesso em: 01 out. 2004.

ANDREOZZI, S. **GLUE Schema implementation for the LDAP model**. Bologna, ITALY: INFN-CNAF, 2003. (I-40127).

AYALA, D. **NuSphere Corporation: nusoap - web services toolkit for php**. Disponível em: <<http://dietrich.ganx4.com/nusoap/>>. Acesso em: 04 out. 2004.

BLAKE, S. et al. **An Architecture for Differentiated Services**: RFC 2475. [S.l.]: Internet Engineering Task Force Network Working Group, 1998.

BRADEN, R.; CLARK, D.; SHENKER, S. **Integrated Services in the Internet Architecture: an overview**: RFC 1633. [S.l.]: Internet Engineering Task Force Network Working Group, 1994.

CECCON, M. B. **Criação e Visualização de Domínios Dinâmicos em Ambientes de Gerenciamento de Redes**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

CECCON, M. B.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Definition and Visualization of Dynamic Domains in Network Management Environments. In: INTERNATIONAL CONFERENCE ON INFORMATION NETWORKING, 2003, Cheju Island, Korea. **Networking Technologies for Enhanced Internet Services**. Berlin: Springer, 2003. p. 828–838.

CHAPIN, S. J.; KATRAMATOS, D.; KARPOVICH, J.; GRIMSHAW, A. The Legion Resource Management System. In: WORKSHOP ON JOB SCHEDULING STRATEGIES FOR PARALLEL PROCESSING, JSSPP, 1999. **Job Scheduling Strategies for Parallel Processing: proceedings**. Berlin: Springer-Verlag, 1999. (Lectures Notes in Computer Science, v. 1659)

CHAPMAN, B.; SUNDARAM, B.; THYAGARAJA, K.; DONEPUDI, H.; KOLATHUR, S.; YAN, N. **EZ-Grid Project**. Disponível em: <<http://www2.cs.uh.edu/ezgrid/>>. Acesso em: 28 set. 2004.

DAMIANOU, N.; DULAY, N.; LUPU, E.; SLOMAN, M. The Ponder Policy Specification Language. In: INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, 2001. **Proceedings...** Berlin: Springer-Verlag, 2001. p.18–38. (Lecture Notes in Computer Science, v.1995)

FERREIRA, L. et al. **Introduction to Grid Computing with Globus**. [S.l.]: IBM Redbooks, 2003.

FLEGKAS, P.; TRIMINTZIOS, P.; PAVLOU, G.; ADRIKOPOULOS, I.; CALVACANTI, C. F. On Policy-Based Extensible Hierarchical Network Management in QoS-Enabled IP Networks. In: INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, 2001. **Proceedings...** Berlin: Springer-Verlag, 2001. p.230–246. (Lecture Notes in Computer Science, v.1995)

FOSTER, I.; FIDLER, M.; ROY, A.; SANDER, V.; WINKLER, L. End-to-End Quality of Service for High-end Applications. **Elsevier Computer Communications Journal**, [S.l.], v. 27, n. 14, p. 1375-1388, 2004.

FOSTER, I.; KESSELMAN, C. **The grid**: blueprint for a new computing infrastructure. [S.l.]: Morgan Kaufmann Publishers Inc., 1999.

FOSTER, I.; KESSELMAN, C.; NICK, J.; TUECKE, S. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. **The Globus Project**. 2002. Disponivel em: <[www.globus.org/research/papers/ogsa.pdf](http://www.globus.org/research/papers/ogsa.pdf)>. Acesso em: 05 nov. 2004.

GLOBUS. **The Globus Alliance - Globus Toolkit**. Disponivel em: <<http://www-unix.globus.org/toolkit/>>. Acesso em: 01 out. 2004.

GRANVILLE, L.; CECCON, M.; TAROUCO, L.; ALMEIDA, M.; CARISSIMI, A. An Approach for Integrated Management of Networks with Quality of Service Support Using QAME. In: INTERNATIONAL WORKSHOP ON DISTRIBUTED SYSTEMS: OPERATIONS AND MANAGEMENT, 12., 2001. **Proceedings...** Amsterdam: Elsevier Science Publishers, 2001. p. 167–178.

HOSCHEK, W.; JAEN-MARTINEZ, J.; SAMAR, A.; STOCKINGER, H.; STOCKINGER, K. Data Management in an International Data Grid Project. In: IEEE/ACM INTERNATIONAL WORKSHOP ON GRID COMPUTING, 1., 2000. **The 1st IEEE/ACM International Workshop on Grid Computing**: proceedings. Berlin: Springer-Verlag, 2000. p.77–90. (Lecture Notes in Computer Science, v. 1971).

IBM. **Grid Computing home page**. Disponivel em:<<http://www.ibm.com/grid>>. Acesso em: 30 set. 2004.

JOHNSTON, W. E.; GANNON, D.; NITZBERG, B. Grids as Production Computing Environments: the engineering aspects of nasa's information power grid. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 8., 1999. **Proceedings...** Washington, DC: IEEE Computer Society, 1999. p. 34.

LOBO, J.; BHATIA, R.; NAQVI, S. A Policy Description Language. In: NATIONAL CONFERENCE ON ARTIFICIAL INTELLIGENCE, AAAI, 1999. **Proceedings...** Menlo Park:AAAI Press, 1999. p.291–298.

MOFFETT, J. D.; SLOMAN, M. S. Policy Hierarchies for Distributed System Management. **IEEE JSAC Special Issue on Network Management**, [S.l.], v.11, n.9, 1993.

MOORE, B. **Policy Core Information Model (PCIM) Extensions**: RFC 3460, Updates RFC 3060. [S.l.]: Internet Engineering Task Force Policy Working Group, 2003.

MOORE, B.; ELLESSON, E.; STRASSNER, J.; WESTERINEN, A. **Policy Core Information Model – Version 1 Specification**: RFC 3060. [S.l.]: Internet Engineering Task Force Policy Working Group, 2001.

NATRAJAN, A.; NGUYEN-TUONG, A.; HUMPHREY, M. A.; GRIMSHAW, A. S. The Legion Grid Portal. **Concurrency and Computation: Practice and Experience**, [S.l.], v. 14, p. 1365–1394, 2002.

NEISSE, R.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Gerenciamento Integrado de Grids e Redes Baseado em Políticas. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 22., 2004, Gramado. **Anais...** Gramado: II/UFRGS, 2004. v. 1, p. 137–140.

NEISSE, R.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Managing Grids Communication Infrastructure through Policy Translations. In: INTERNATIONAL WORKSHOP ON IP OPERATIONS AND MANAGEMENT, 2004. **Proceedings...** Pequim: [s.n.], 2004.

NEISSE, R.; PEREIRA, E. D. V.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. An Hierarchical Policy-Based Architecture for Integrated Management of Grids and Networks. In: INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, 5., 2004, Yorktown Heights, NY. **Proceedings...** Los Alamitos: IEEE, 2004. p.103–106.

NEISSE, R.; VIANNA, R.; GRANVILLE, L. Z.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. Implementation and Bandwidth Consumption Evaluation of Web Services to SNMP Gateways. In: IEEE/IFIP NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, 9., 2004, Seoul, Korea. **Managing next generation convergence networks and services: Technical sessions NOMS 2004**. [S.l.: s.n.], 2004. v.1, p.715–728.

NSF. **National Science Foundation - The TeraGrid project**. Disponível em: <<http://www.teragrid.org>>. Acesso em: 28 set. 2004.

OMG. **Object Management Group CORBA Website**. Disponível em: <<http://www.corba.org>>. Acesso em: 30 set. 2004.

OPENLDAP. **OpenLDAP**: an open source implementation of the lightweight directory access protocol. Disponível em: <<http://www.openldap.org>>. Acesso em: 04 out. 2004.

ORACLE. **Oracle Corporation - Grid Computing Technologies**. Disponível em: <[http://www.oracle.com/technology/products/oracle9i/grid\\_computing/index.html](http://www.oracle.com/technology/products/oracle9i/grid_computing/index.html)>. Acesso em: 01 out. 2004.

PHP. **The PHP Group**: hypertext preprocessor. Disponível em: <<http://www.php.net>>. Acesso em: 04 out. 2004.

SANDER, V.; ADAMSON, W.; FOSTER, I.; ALAIN, R. End-to-End Provision of Policy Information for Network QoS. In: IEEE INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 10., 2001. **Proceedings...** [S.l.]: IEEE Computer Society Press, 2001.

SLOMAN, M.; MOFFETT, J. Domain Management for Distributed Systems. In: IFIP SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT, 1., 1989. **Proceedings...** [S.l.: s.n.], 1989. p. 505–516.

SNIR, Y.; RAMBERG, Y.; STRASSNER, J.; COHEN, R.; MOORE, B. **Policy QoS Information Model**: RFC 3644. [S.l.]: Internet Engineering Task Force Policy Working Group, 2003.

STEEN, M.; HOMBURG, P.; TANENBAUM, A. S. Globe: a wide-area distributed system. **IEEE Concurrency**, [S.l.], p.70–78, Jan. 1999.

SUN. **Sun Microsystems Grid Computing Solutions.** Disponível em: <<http://www.sun.com/software/grid/>>. Acesso em: 30 set. 2004.

SUNDARAM, B.; CHAPMAN, B. M. Policy Engine: a framework for authorization, accounting policy specification and evaluation in grids. In: INTERNATIONAL WORKSHOP ON GRID COMPUTING, 2., 2001. **Grid Computing - GRID 2001, Second International Workshop: proceedings.** Berlin: Springer, 2001. (Lecture Notes in Computer Science, v. 2242).

SUNDARAM, B.; CHAPMAN, B. M. XML-based Policy Framework for Usage Policy Management in Grids. In: INTERNATIONAL WORKSHOP ON GRID COMPUTING, 3., 2002. **Grid Computing - GRID 2002, Third International Workshop: proceedings.** Berlin: Springer, 2002. (Lecture Notes in Computer Science, v. 2536)

SUNDARAM, B.; NEBERGALL, C.; TUECKE, S. Policy specification and restricted delegation in Globus proxies. In: THE INTERNATIONAL CONFERENCE FOR HIGH PERFORMANCE COMPUTING AND COMMUNICATIONS, 2000. **Proceedings...** [S.l.]: IEEE Computer Society, 2000. Disponível em: <<http://www.supercomp.org/sc2000/Proceedings/start.htm>>. Acesso em: 05 nov. 2004.

VERMA, D. C.; SAHU, S.; CALO, S. B.; BEIGI, M.; CHANG, I. A Policy Service for GRID Computing. In: INTERNATIONAL WORKSHOP ON GRID COMPUTING, 3., 2002. **Grid Computing - GRID 2002, Third International Workshop: proceedings.** Berlin: Springer, 2002. p. 243–255. (Lecture Notes in Computer Science, v. 2536)

WESTERINEN, A.; SCHNIZLEIN, J.; STRASSNER, J.; SCHERLING, M.; QUINN, B.; HERZOG, S.; HUYNH, A.; CARLSON, M.; PERRY, J.; WALDBUSSER, S. **Terminology for Policy-Based Management:** RFC 3198. [S.l.]: Internet Engineering Task Force Policy Working Group, 2001.

YANG, K.; GALIS, A.; TODD, C. Policy-based active grid management architecture. In: IEEE INTERNATIONAL CONFERENCE ON NETWORKS, 10., 2002. **Proceedings...** [S.l.]: Kluwer Academic Publishers, 2002. p. 243–248.

YAVATKAR, R.; PENDARAKIS, D.; GUERIN, R. **A Framework for Policy-based Admission Control:** RFC 2753. [S.l.]: Internet Engineering Task Force Policy Working Group, 2000.

## **ANEXO A ARTIGO PUBLICADO**

Neste anexo é apresentado o artigo "Managing Grids Communication Infrastructure through Policy Translations" desenvolvido durante o mestrado. O artigo é resultado do segundo ano e apresenta os problemas investigados nesta dissertação, o modelo proposto para solucionar os problemas, a implementação do protótipo e as conclusões obtidas.

Evento da publicação:

- Nome: 2004 IEEE International Workshop on IP Operations & Management (IPOM 2004)
- URL: <http://www.bupt.edu.cn/quick/link/ipom/>
- Data: De 11 a 13 de outubro de 2004
- Local: Beijing Friendship Hotel, Pequim, China



# Managing Grids Communication Infrastructure through Policy Translations

Ricardo Neisse, Lisandro Z. Granville, Maria Janilce B. Almeida and Liane Margarida R. Tarouco  
 Institute of Informatics - Federal University of Rio Grande do Sul  
 Av. Bento Gonçalves, 9500 - Porto Alegre, RS - Brazil  
 Email: {neisse, granville, janilce, liane}@inf.ufrgs.br

*Abstract*—Computing grids require the underlying network infrastructure to be properly configured in order to have appropriate communications among the grids' nodes. The management of networks and the management of grids are currently executed by different tools operated by different administrative personnel. Eventually, the grid communication requirements will need corresponding support from the network management tools, but such requirements are fulfilled only when grid administrators manually asks network administrators for 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 rules are used by the proposed translation mechanism to generate the necessary underlying network configuration policies.

## I. INTRODUCTION

In order to manage a grid infrastructure, the management of the underlying network that provides the communication support is also a requirement. 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 [1] [2] [3] 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.

A commonly required network configuration in a conference grid, implemented for instance with the AccessGrid toolkit [3], 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 [1], through its Globus Architecture for Reservation and Allocation (GARA) [4]. 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. [5] [6]. 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. [7] define a management service where global grid policies are combined with local domain policies. The local policies have high priority, which means that if a global policy defines a 20GB disk allocation in a server, but the local policy allows only 10GB, the local policy is chosen and the 10GB limit is allocated. Here, potential conflicts of interest between the grid and the local 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 grid administrator are supposed to have some kind of agreement regarding the grid and network resources on the local domain.

Another proposals using policies for network configuration aiming grid support are presented by Yang et al. [8] and Sander et al. [9]. The solution from Yang et al. specifies an architecture divided in a policy-based management layer (that follows the IETF definitions of PEPs and PDPs [10]), and a layer that uses the concept of programable networks (active networks represented by a middleware) to automatically configure the network devices. Sander et al. [9] propose a policy-based architecture to configure the network QoS of different administrative domains members of a grid using an inter-domain signaling protocol, that sequentially configures an end-to-end communication path (e.g. a user accessing a server). Although both solutions are based on policies, they do not specify how grid and network policies are defined neither present any facility to allow the integration with the grid toolkits mentioned before.

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.

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 where 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. The remainder of this paper is organized as follow: Section 2 presents the proposed hierarchical policy-based management architecture; Section 3 shows the prototype developed based on such architecture; Section 4 finishes the paper with conclusions and future work.

## II. 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. After the creation of the grid policies 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 (PDPs) [10] of a regular policy-based management system.

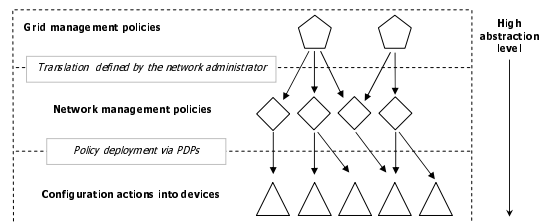


Fig. 1. Hierarchy for policy translation

To define grid policies we used an hypothetical language, which is based in previous work on grid policies [5] [6]. 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 [11], 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 == "neisse" and
    proxy == "LabTec Cluster" and
    resource == "UFRGS 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 rule presented in the Listing 2 the user *neisse* is able to access, during a specific period of time, a storage server (*UFRGS Data Server*) through a grid cluster (*LabTec Cluster*). 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. The QoS requirement (*highThroughputDataIntensive*) is defined in the policy actions.

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. *highThroughputDataIntensive*). Behind this identification, 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: (1) the grid administrator defines grid policies and the 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 and (4) 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 and (9) translate the network policies to configuration actions in order to deploy such policies in the local domain PEPs.

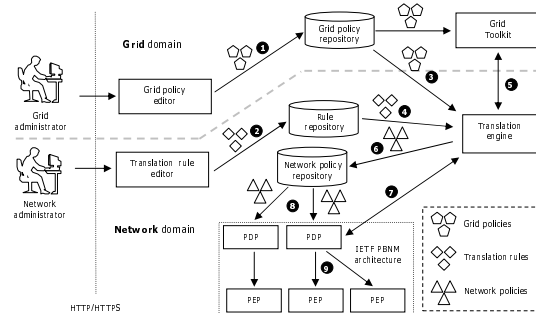


Fig. 2. Policy Translation Architecture

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.

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 4 presents an example of a translation rule that creates two new network policies from a single grid policy.

```

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(dstAddress,"=",dstResource.address);
  p1.addCondition(dstPort,"=",dstResource.port);
  p1.addCondition(dstProtocol,"=", "tcp");
  p1.addAction(DSCP,2);
  inPEPs = select pep.direction["in"]
    .within[srcResource.address, 143.54.47.1]
    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(bandwidth,requiredQoS.requiredBandwidth);
  outPEPs = select pep.direction["out"]
    .within[srcResource.address, 143.54.47.1]
    from device.type["DiffServDevice"];
  outPEPs.deployPolicy(p2);
}

```

Listing 4: Translation rule examples with network policy deployment

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 destination addresses, the destination 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 [12]. Such

domains are defined through selection expressions introduced in the translation rules. In the example from Listing 4, 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).

### III. 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 and 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.

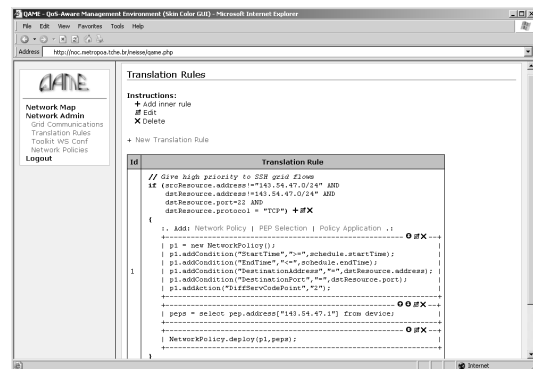


Fig. 3. Grid Translation Rule and Policy Editor

Figure 3 shows the translation rule editor interface. 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 Policy Core Information Model Extension (PCIME) schema [13] and implemented policies, policy conditions and policy actions for grid and network management. We modeled 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.

The distributed operation of the prototype is presented in figure 5. The figures shows the grid and network administrator

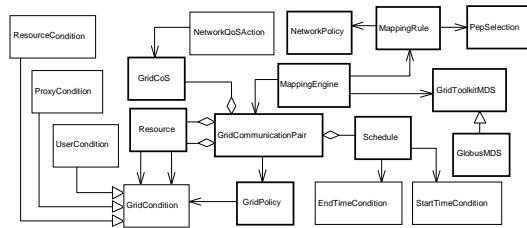


Fig. 4. Information model for policy translation (UML)

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 PCIME. 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.

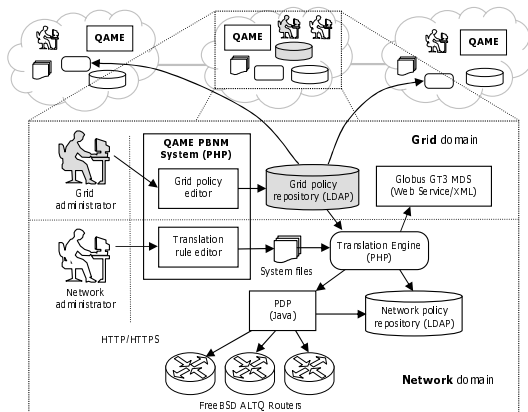


Fig. 5. Prototype implementation

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 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.

#### IV. CONCLUSIONS AND FUTURE WORK

We presented in this paper an architecture that translates grid policies to network policies through a translation enginebased on translation rules. 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

- [1] "The globus project," 2003, <http://www.globus.org>.
- [2] M. Steen, P. Homburg, and A. S. Tanenbaum, "Globe: A wide-area distributed system," *IEEE Concurrency*, pp. 70–78, Jan. 1999.
- [3] "The access grid (ag) user documentation," 2003, <http://www-fp.mcs.anl.gov/ft/accessgrid>.
- [4] I. Foster et al., "End-to-end quality of service for high-end applications," *IEEE Computer Communications Special Issue on Network Support for Grid Computing*, 2002.
- [5] B. Sundaram, C. Nebergall, and S. Tuecke, "Policy specification and restricted delegation in globus proxies," in *SuperComputing 2000*.
- [6] 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.
- [7] D. C. Verma et al., "A policy service for grid computing," in *Grid Computing - GRID 2002, Third International Workshop*, ser. Lecture Notes in Computer Science, Nov. 2002, pp. 243–255.
- [8] K. Yang, A. Galis, and C. Todd, "Policy-based active grid management architecture," *10th IEEE International Conference on Networks*, 2002.
- [9] 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.
- [10] A. Westerinen et al., "Terminology for policy-based management," Request for Comments: 3198, Nov. 2001, IETF.
- [11] R. Yavatkar et al., "A framework for policy-based admission control," Request for Comments: 2753, Jan. 2000, IETF.
- [12] M. B. Ceccon et al., "Definition and visualization of dynamic domains in network management environments," *Lecture Notes in Computer Science*, vol. 2662. LNCS, 2003, pp. 828–838.
- [13] B. Moore, "Policy core information model (pcim) extensions," Request for Comments: 3460, Updates RFC 3060, Jan. 2003, IETF.