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

LEANDRO VAGUETTI

Uma Solução baseada em políticas para Gerenciamento Integrado de QoS e Multicast em Redes IP

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

Prof. Dr. Lisandro Zambenedetti Granville
Orientador

Porto Alegre, dezembro de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Vaguetti, Leandro

Uma Solução baseada em políticas para Gerenciamento Integrado de QoS e Multicast em Redes IP / Leandro Vaguetti. – Porto Alegre: Programa de Pós-Graduação em Computação, 2003.

103 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientador: Lisandro Zambenedetti Granville.

1. Gerenciamento integrado de Redes de Computadores. 2. PBNM. 3. Gerenciamento de Multicast. 4. Gerenciamento de QoS. I. Granville, Lisandro Zambenedetti. II. Título.

AGRADECIMENTOS

Este, com certeza, é o momento em que todos os mestrandos passam um filme de suas vidas, acredito que é a parte da dissertação que dá mais prazer em escrever.

Bom, muitas pessoas merecem meus agradecimentos, algumas me acompanham a muito tempo, outras nem tanto, algumas desde meu nascimento, outras desde seus nascimentos, enfim todas essas pessoas fazem parte de minha vida e com certeza são muito importantes.

Gostaria de começar agradecendo o apoio de meu orientador, Prof. Dr. Lisandro Granville, agradeço até pelos puxões de orelha. Agradeço também o apoio de meus colegas do LabCom (em especial ao Neisse, Sérgio, Evandro e Michelle), e a todos aqueles que por aqui passaram durante esses dois anos, incluindo os professores.

Agradeço a dedicação de minha noiva Thaís por me acompanhar e apoiar, agradeço também ao meu filho Carlos Henrique, que apesar do pouco tempo que temos para ficar juntos, sempre me recebe com um sorriso lindo.

E como não poderiam faltar, agradeço de coração aos meus pais Ocelino e Laura, que apesar de não poderem me ajudar financeiramente, me dotaram daquilo que considero mais importante para alcançar a felicidade: coragem e vontade de vencer.

Nossa, esta parecendo uma oração !!!

Com certeza, apenas com palavras não consigo demonstrar a gratidão que tenho para com essas pessoas. Mas sempre estarei disposto a dizer "Obrigado!!!!".

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

QoS	*Quality of Service*
IETF	*Internet Engineering Task Force*
RSVP	*Resource ReSerVation Protocol*
IntServ	*Integrated Service*
DiffServ	*Differentiated Service*
IGMP	*Internet Group Manager Protocol*
MIB	*Management Information Base*
RFC	*Request for Comments*
PMT	*Policy Management Tool*
PDP	*Policy Decision Point*
PR	*Policy Repository*
PEP	*Policy Enforcement Point*
COPS	*Common Open Policy Service*
LDAP	*Lightweight Directory Access Protocol*
QAME	*QoS-Aware Management Environment*
Mbone	*Multicast BackBone*
DVMRP	*Distance Vector Multicast Routing*
MOSPF	*Multicast Open Shortest Path First*
PIM	*Protocol Independent Multicasting*
PBNM	*Policy-Based Network Management*
CIM	*Core Information Model*
PCIM	*Policy Core Information Model*
PCIMe	*Policy Core Information Model Extensions*
QPIM	*QoS Policy Information Model*
DMTF	*Distribiuted Management Task Force*
RNP2	Rede Nacional de Pesquisa 2

UDP	*User Datagram Protocol*
TCP	*Transmission Control Protocol*
SMI	*Structure of Management Information*
XML	*eXtensible Markup Language*
FSM	*Finite State Machine*
IP	*Internet Protocol*
SNMP	*Simple Network Management Protocol*
OID	*Object IDentifier*
DSCP	*DiffServ Code Point*
HTTP	*HyperText Transfer Protocol*
BW	*Bandwidth*
LDAP	*Lightweight Directory Access Protocol*
PHP	*Pre Hypertext Processor*
HTML	*HyperText Markup Language*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Aplicações como videoconferência, vídeo sob-demanda, aplicações de ensino a distância, entre outras, utilizam-se das redes de computadores como infra-estrutura de apoio. Mas para que tal uso seja efetivo, as redes de computadores, por sua vez, devem fornecer algumas facilidades especiais para atender às necessidades dessas aplicações. Dentre as facilidades que devem ser fornecidas estão os suportes à qualidade de serviço (QoS - *Quality of Service*) e as transmissões multicast.

Além do suporte a QoS e multicast nas redes, é necessário fornecer um gerenciamento da rede adequado às expectativas de tais aplicações. Soluções que fornecem gerenciamento de forma individual para tais facilidades, já foram propostas e implementadas. Entretanto, estas soluções não conseguem agir de modo integrado, o que torna a tarefa do gerente da rede extremamente complexa e difícil de ser executada, pois possibilitam um fornecimento não adequado das facilidades desejadas às aplicações.

Nesta dissertação é apresentada uma solução para gerenciamento integrado de QoS e multicast. Fazem parte da solução: a definição e implementação de uma arquitetura para gerenciamento integrado de QoS e multicast, utilizando gerenciamento baseado em políticas (PBNM - *Policy-Based Network Management*), além da validação da solução proposta através da implementação de um protótipo. Um ambiente, condições de teste, e análise dos resultados obtidos, também são apresentados durante a dissertação.

Palavras-chave: Gerenciamento integrado de Redes de Computadores, PBNM, Gerenciamento de Multicast, Gerenciamento de QoS.

A Policy-based solution for Integrated QoS and Multicast Management in IP Netwoks

ABSTRACT

Advanced applications, such as video conference, video on-demand, distance learning, among others, make use of computer networks as infrastructure support. However, in order to achieve an effective usage, the computer networks, on their turn, must offer some special capabilities to handle the applications needs. Quality of service (QoS) and multicast transmission support are some of the capabilities that must be supported.

Besides the QoS and multicast support, it is necessary to supply a network management suitable to the applications expectations. Proposals that provide an individual solution for each of the capabilities have already been implemented. However, these solutions are not integrated, which makes the task of network management difficult and complex, due to the limitations inherent to non integrated systems.

This Master thesis presents a solution for integrated QoS and multicast management. The solution is composed by: the definition and implementation of an architecture for integrated QoS and multicast management, using Policy-Based Network Management (PBNM), besides the validation of the solution proposal through the implementation of a prototype. A test environment and conditions, as well as the analysis of the experimental results, are also presented throughout the thesis.

Keywords: Integrated Network Management, PBNM, Multicast Management and QoS Management.

1 INTRODUÇÃO

As novas redes de computadores, como a Internet2 e a RNP2, viabilizam a utilização de aplicações avançadas como videoconferência, vídeo interativo, bibliotecas digitais e laboratórios virtuais (LEOPOLDINO; SANTOS MOREIRA, 2001). Tais aplicações exigem da rede utilizada um conjunto mais amplo de serviços, com complexidade superior ao serviço de melhor esforço (*best-effort*) oferecido atualmente. Tal conjunto de serviços deve cuidar para que o tráfego gerado mantenha-se de acordo com os níveis de qualidade esperados pelas aplicações. A inserção da noção de qualidade de serviço (QoS - *Quality of Service*) é extremamente importante nos ambientes de rede para que as aplicações possam operar com um nível de garantia mínimo, ausente em redes "convencionais"como as redes baseadas no protocolo IP (MCDYSAN, 1999). A capacidade de garantir o nível de qualidade da rede é um dos requisitos que devem ser encontrados nas novas redes de computadores.

Além de QoS, uma capacidade comumente requisitada pelas novas aplicações é a capacidade de transmissão *multicast*. A multicast tem a função de endereçar um grupo de um ou mais dispositivos de rede, oferecendo vantagens para aplicações que necessitem de otimização quanto ao uso da rede e utilizem-se de grupos multicast com diversos participantes recebendo dados simultaneamente. As aplicações mais críticas normalmente necessitam trocar informações com várias estações simultaneamente, além de esperar um comportamento adequado da rede de acordo com suas necessidades (e.g. largura mínima de banda garantida). Portanto, a disponibilidade de QoS e multicast nas redes apresenta-se como uma facilidade realmente necessária das novas aplicações.

Supondo que as facilidades necessárias para que as aplicações executem da forma esperada pelos usuários sejam oferecidas pela rede, espera-se que as aplicações possam operar adequadamente. Entretanto, podem ocorrer eventos na rede que acabem por degradar o serviço, oferecido inicialmente e, por conseqüência, diminuir o desempenho da aplicação. Torna-se, assim, necessário o uso de um mecanismo de gerenciamento que possa identificar e controlar os eventos ocorridos, assim como, de que forma as facilidades utilizadas estão sendo disponibilizadas às aplicações.

O gerenciamento de QoS e multicast atualmente pode ser realizado através de várias ferramentas, porém a QoS é gerenciada através de ferramentas específicas para o gerenciamento de QoS, levando em conta apenas aspectos relacionados à QoS, de forma única (EDER; NAG, 2001). Podem-se citar alguns esforços para facilitar o gerenciamento de QoS: IntServ-MIB (BAKER; KRAWCZYK; SASTRY, 1997a), RSVP-MIB (BAKER; KRAWCZYK; SASTRY, 1997b), DiffServ-MIB (BAKER; CHAN; SMITH, 2002), etc. Da mesma forma, uma rede multicast é gerenciada através de ferramentas específicas ao multicast: IPMroute-MIB (MCCLOGHRIE; FARINACCI; THALER, 2000a), IGMP-MIB (MCCLOGHRIE; FARINACCI; THALER, 2000b) e PIM-MIB (MCCLOGHRIE

et al., 2000). Entretanto, as aplicações avançadas necessitam das duas facilidades de forma conjunta, e conseqüentemente necessitam de um mecanismo de gerenciamento que opere também de forma única, o que não ocorre nas ferramentas de gerenciamento atuais, onde o gerenciamento de QoS e multicast é tratado de forma não integrada.

O principal objetivo desta dissertação é fornecer uma solução de gerenciamento integrado para QoS e multicast em redes IP. Optou-se pelo uso do gerenciamento baseado em políticas (PBNM - *Policy-based Network Management*) (SLOMAN, 1994) como elemento facilitador da integração. Nesse contexto, é necessário definir políticas para gerenciamento integrado de QoS e multicast, onde a definição das políticas seguem as recomendações do *Policy Group* do IETF (*Internet Engineering Task Force*) (POLICY, 2003). Além disso, o IETF propõe uma arquitetura geral de gerenciamento baseado em políticas, definindo quem são e como operam cada um de seus elementos (e.g. PDP's e PEP's) (WESTERINEN et al., 2001), porém, baseado nas definições estudadas, torna-se necessário acrescentar novos elementos à arquitetura inicial, para que as políticas definidas por esse trabalho sejam aplicadas adequadamente (e.g. monitores que sinalizem eventos ocorridos na rede). Num segundo momento é apresentado o protótipo implementado, baseado na arquitetura definida anteriormente. São apresentados também os testes realizados para validar a solução proposta e o protótipo.

Este trabalho está organizado da seguinte forma: o capítulo 2 apresenta os trabalhos relacionados ao gerenciamento de QoS e multicast. O capítulo 3 apresenta uma proposta de integração do gerenciamento dos serviços de QoS e multicast usando gerenciamento baseado em políticas. No capítulo 4 é apresentada a implementação do protótipo baseada nas definições propostas no capítulo 3. O capítulo 5 apresenta o ambiente de testes utilizado para validar a solução proposta e análise do resultados obtidos durante a fase de testes. Por fim, o capítulo 6 encerra esta dissertação apresentando as conclusões.

2 TRABALHOS RELACIONADOS

Em se tratando de gerenciamento de QoS e multicast, diversas soluções podem ser encontradas, dentre elas estão MIB's definidas, tanto para gerenciamento de QoS quanto para gerenciamento de multicast, algumas ferramentas, ambientes desenvolvidos para dar suporte a tais gerenciamentos e alguns esforços para padronizar tecnologias de gerenciamento emergentes. Neste capítulo serão apresentadas as soluções encontradas atualmente para o fornecimento de gerenciamento de QoS e multicast, além de uma discussão sobre o gerenciamento integrado das duas tecnologias.

2.1 Gerenciamento de QoS

O gerenciamento de QoS pode ser alcançado através de várias ferramentas já existentes no mercado, ferramentas estas que normalmente possuem sua implementação baseada no padrão de gerenciamento Internet (SNMP- *Simple Network Management Protocol*) (CASE et al., 1990) e que podem adicionar ao seu conjunto de objetos gerenciáveis os objetos gerenciáveis encontrados nas MIB's definidas, para padronizar o gerenciamento de arquiteturas para fornecimento de QoS como Serviços integrados (IntServ) (BRADEN; CLARK; SHENKER, 1994) e Serviços diferenciados (DiffServ) (BLAKE et al., 1998). Além disso, alguns protocolos definidos para possibilitar a implementação de tais arquiteturas, como é o caso do RSVP (BRADEN et al., 1997), também podem ser gerenciados através de MIB's. O gerenciamento de QoS também pode ser alcançado através de ambientes que não se utilizam apenas de SNMP para prover gerenciamento. Nestes ambientes, como é o caso do ambiente de gerenciamento de QoS QAME (GRANVILLE; TAROUCO, 2001), utilizam-se de formas alternativas (e.g PBNM) para fornecer o gerenciamento necessário.

A arquitetura para fornecimento de QoS IntServ pode ser gerenciada através de duas MIB's, uma delas fornece a possibilidade de gerenciar a arquitetura de forma genérica e a outra se preocupa com a garantia dos serviços solicitados. A IntServ-MIB está definida na RFC 2213 (BAKER; KRAWCZYK; SASTRY, 1997a), sob o título de "*Integrated Services Management Information Base using SMIv2*". Esta MIB é composta de duas estruturas tabulares, onde a estrutura tabular "intSrvIfAttribTable"define os objetos gerenciáveis de uma determinada interface, como por exemplo, o número de bits por segundo alocados ou o tamanho do buffer necessário para todos os fluxos em rajada. A segunda estrutura tabular pertencente à MIB ("intSrvFlowTable") descreve os fluxos que passam em determinada interface. Neste caso, como exemplo, parâmetros utilizados para determinar o número de filas usadas ou o peso usado para priorizar um tráfego, podem ser configurados. A MIB que complementa a MIB anterior é definida na RFC 2214 (BAKER; KRAWCZYK; SASTRY, 1997c), sob o título de "*Integrated*

Services Management Information Base Guaranteed Service Extensions using SMIv2". Esta MIB apresenta apenas uma estrutura tabular denominada "intSrvGuaranteedIfTable", que oferece objetos para que possam ser configurados parâmetros que determinem a degradação máxima sofrida por um determinado tráfego, como o parâmetro setado para determinar o máximo de pacotes transferidos com atraso.

Normalmente a arquitetura IntServ é associada ao protocolo de sinalização RSVP, no entanto, o gerenciamento do protocolo RSVP não pode ser feito através da IntServ-MIB. Assim, o protocolo RSVP também possui uma MIB para facilitar seu gerenciamento. A RSVP-MIB é definida na RFC 2206 (BAKER; KRAWCZYK; SASTRY, 1997b), sob o título de "*RSVP Management Information Base using SMIv2*". Esta RFC define quatro objetos gerenciáveis e sete estruturas tabulares. Os quatro objetos serão citados a seguir e as estruturas tabulares comentadas posteriormente:

- rsvpBadPackets - Contabiliza os pacotes recebidos que devem ser descartados;
- rsvpSenderNewIndex - Objeto usado para nomear os valores para rsvpSenderNumber;
- rsvpResvNewIndex - Objeto usado para nomear os valores para rsvpResvNumber;
- rsvpResvFwdNewIndex - Objeto usado para nomear os valores para rsvpResvFwdNumber;

As sete estruturas tabulares definidas na RSVP-MIB são descritas na Tabela 2.1.

Tabela 2.1: Estruturas tabulares que compõem a RSVP-MIB

Estrutura Tabular	Descrição
rsvpSessionTable	possui informações relativas às sessões RSVP
rsvpSenderTable	possui informações das trocas de mensagens *Sender*
rsvpResvTable	possui informações das trocas de mensagens *Resv*
rsvpResvFwdTable	possui informações das trocas de mensagens *ResvFwd*
rsvpSenderOutInterfaceTable	guarda informações sobre o status interface *SenderOut*
rsvpIfTable	possui informações sobre as interfaces RSVP
rsvpNbrTable	possui informações das interfaces vizinhas às interfaces RSVP

Entretanto, a MIB que mais oferece objetos a serem gerenciados é a DiffServ-MIB que está definida na RFC 3289 (BAKER; CHAN; SMITH, 2002), sob o título de "*Management Information Base for differentiated Services Architecture*". Esta MIB fornece suporte ao gerenciamento dos Serviços diferenciados. Ao todo são quinze estruturas tabulares definidas na MIB, cada uma das quinze estruturas vai ser descrita através da Tabela 2.2.

O fato de serem fornecidas MIB's para auxiliar no gerenciamento de QoS, por si só, não garantem um bom nível de gerenciamento. Gerenciar uma rede que possui garantias

Tabela 2.2: Estruturas tabulares que compõem a DiffServ-MIB

Estrutura Tabular	Descrição
diffServDataPathTable	Ponteiros para o primeiro caminho de dados funcional, indexados por interface e pelo sentido nessa interface
diffServClfrTable	Elementos classificados para identificar um tráfego específico como parte de um ambiente agregado
diffServClfrElementTable	Permite especificar filtros para classificar elementos em campos multivalorados
diffServMultiFieldClfrTable	Número de fluxos com reserva ativos na interface
diffServMeterTable	Especifica o status de um determinado caminho de dados funcional. Opções : "*succeed*", "*fail*"e "*specific*"
diffServTBParamTable	Configura os parâmetros para definir um *Token Bucket*
diffServActionTable	Identifica a seqüência de ações que devem ser aplicadas a um pacote
diffServCountActTable	Contabiliza as ações pertinentes ao determinado tráfego para gerações de estatíticas
diffServDscpMarkActTable	Possibilita identificar os pacotes que devem ser marcados com requisito DSCP
diffServRandomDropTable	Identifica o algoritmo de descarte para determinada fila
diffServAlgDropTable	Usada para definir parâmetros mínimos e máximos para descartes randômicos
diffServQTable	Permite a construção de uma simples fila ou classe de serviço
diffServSchedulerTable	Permite a construção de simples fila até a mais complexa hierarquia de filas
diffServMinRateTable	Define a taxa de saída mínima em uma fila
diffServMaxRateTable	Define a taxa de saída máxima em uma fila

sobre QoS envolve a análise de um volume de informações maior que o encontrado na gerência de redes sem QoS. A gerência tradicional não é capaz de gerenciar uma rede desse tipo, porque o tempo despendido pelo gerente de rede na análise das informações de gerência obtidas é proibitivo. Logo, é necessário utilizar um ambiente de gerência onde o grande volume de dados possa ser reduzido a níveis aceitáveis, de forma a permitir ao gerente da rede resolver os problemas encontrados em tempo hábil.

O ambiente de gerência denominado QAME (GRANVILLE; TAROUCO, 2001) (GRANVILLE et al., 2000) procura fornecer facilidades para que o gerente seja capaz de aplicar a gerência de *QoS* de forma mais efetiva. QAME suporta a execução de seis principais tarefas de gerência de *QoS*: implantação, monitoração, manutenção, descoberta, visualização e análise de *QoS*. Para que as tarefas de gerência citadas anteriormente possam ser efetivamente realizadas, a arquitetura do QAME conta com

módulos de software disjuntos, porém relacionados, que podem estar localizados em vários pontos críticos distintos da rede.

Além das ferramentas e tecnologias apresentadas anteriormente, existe uma que tem se destacado como tendência no gerenciamento de *QoS*. Trata-se do gerenciamento baseado em políticas (PBNM), tecnologia de gerenciamento emergente que visa fornecer ao gerente facilidades de gerenciamento, através da definição de políticas de alto nível na rede. Tais políticas devem ser definidas de forma abstrata, evitando a descrição de como os recursos gerenciados devem ser tratados com relação a uma tecnologia específica (e.g DiffServ). Após as políticas serem definidas e armazenadas, o ambiente de gerenciamento se encarrega de executá-las da forma adequada, onde o ambiente deve traduzir as políticas de alto nível para ações, nos dispositivos da rede. Para que essas políticas possam ser armazenadas de forma padronizada, o IETF tem reunido esforços para definir e organizar métodos de armazenamento de políticas genéricas e em especial aquelas que tratam do gerenciamento de QoS.

Dois modelos de informação têm sido discutidos no IETF, com relação ao gerenciamento de QoS. Atualmente os dois se encontram em forma de *draft* e propõem mecanismos para definição de políticas para gerenciamento de QoS, sob dois focos distintos. O QPIM (*Policy QoS Information Model*) (SNIR et al., 2003) propõe a definição de políticas mais abstratas, independentes da tecnologia utilizada nos dispositivos aos quais as políticas serão aplicadas, preocupando-se somente em armazenar os parâmetros que ditam o comportamento esperado da rede, com relação à QoS exigida pelas aplicações. Neste modelo as políticas podem ser definidas para aplicação em diversos dispositivos ou domínios administrativos, sem modificação da política. O outro foco abordado pelo IETF é o armazenamento de políticas orientadas a um dispositivo específico, definindo a QoS de acordo com a tecnologia oferecida por cada dispositivo. Esta proposta é chamada pelo IETF de *Information Model for describing Network Device QoS Datapath Mechanisms* (MOORE et al., 2003).

De forma geral, o gerenciamento de *QoS* está bem estruturado com relação ao fornecimento de MIB's, onde podem ser encontradas definições e objetos gerenciáveis para as principais arquiteturas e protocolos. Entretanto, o simples gerenciamento destas estruturas tem causado grande dificuldade aos gerentes de redes, devido à quantidade de informações e detalhes técnicos que devem ser considerados. Visando diminuir tal complexidade o gerenciamento de *QoS* baseado em políticas vem oferecendo facilidades de gerenciamento, onde a quantidade de informações tratadas pelos gerentes é tão grande a ponto de impossibilitar o gerenciamento de forma adequada. No entanto, as definições e padronizações relativas ao gerenciamento de *QoS* baseado em políticas ainda estão sob discussão, o que torna o efetivo gerenciamento de *QoS* muito complexo atualmente.

2.2 Gerenciamento de Multicast

Com o surgimento de infra-estruturas de serviços e protocolos de multidifusão, tornou-se necessária a disponibilização de recursos de apoio à gerência dos mesmos. Dentre os recursos disponibilizados estão principalmente MIB's, que abrangem de aspectos gerais até aspectos específicos dos protocolos utilizados em transmissões multicast na internet. O gerenciamento de multicast também pode ser feito através de ferramentas que utilizam ou não as MIB's, definidas para esta tarefa. Nesta seção serão apresentadas as MIB's disponíveis para gerenciamento de multicast, além de algumas ferramentas que também auxiliam no gerenciamento.

Em se tratando de MIB's, uma das mais importantes para o gerenciamento multicast é a MIB de roteamento multicast (também denominada IPMRoute-MIB). Esta é proposta na RFC 2932 (MCCLOGHRIE; FARINACCI; THALER, 2000a), tendo como objetivo definir objetos gerenciáveis que auxiliem na gerência do roteamento multicast de forma genérica, onde os objetos gerenciáveis dependentes de algum protocolo de roteamento multicast devem ser definidos nas MIBs específicas a cada protocolo. A IPMRoute-MIB define um objeto gerenciável e quatro estruturas tabulares. O objeto "ipMRouteEnable"é um objeto do tipo inteiro, onde o valor 1 indica roteamento multicast habilitado e 2 indica o mesmo desabilitado. As demais estruturas tabulares componentes da MIB serão descritas na Tabela 2.3.

Tabela 2.3: Estruturas tabulares que compõem a IPMRoute-MIB

Estrutura Tabular	Descrição
ipMRouteTable	representa os ramos ascendentes das árvores de distribuição na perspectiva do roteador gerenciado
ipMRouteNextHopTable	representa os ramos descendentes das árvores de distribuição na perspectiva do roteador gerenciado
ipMRouteInterfaceTable	informações sobre as interfaces usadas no roteamento multicast
ipMRouteBoundaryTable	informações sobre limites de endereços multicast

Para tornar o gerenciamento de roteamento multicast completo, foram definidas MIB's que oferecem objetos gerenciáveis para cada protocolo de roteamento multicast de forma específica, visando fornecer o gerenciamento de aspectos relacionados apenas a um determinado protocolo. A seguir serão apresentadas as MIB's dos protocolos de roteamento multicast DVMRP (DEERING; WAITZMAN; PARTRIDGE, 1988) e PIM (DEERING et al., 1995).

A PIM-MIB (*Protocol Independent Multicast MIB for IPv4*) (MCCLOGHRIE et al., 2000) tem por objetivo definir objetos gerenciados que permitam monitorar a operação dos protocolos de roteamento PIM-SM e PIM-DM em roteadores que utilizam esses protocolos. Nesta seção, "PIM"é usado para referenciar situações que se aplicam tanto ao protocolo PIM-SM como ao protocolo PIM-DM. A PIM-MIB define um objeto simples chamado "pimJoinPruneInterval", que informa o intervalo de tempo (em segundos) padrão para o roteador gerenciado a ser considerado entre as transmissões periódicas de mensagens PIM-SM dos tipos: *Join* e *Prune*. As estruturas tabulares pertencentes à MIB são apresentadas, na Tabela 2.4.

Outra MIB's proposta para protocolos de roteamento é a *Distance-Vector Multicast Routing Protocol MIB* (DVMRP-MIB). Ainda que de forma experimental, esta MIB é aplicável em roteadores multicast IPv4 que implementam DVMRP. Está MIB não suporta gerenciamento de DVMRP para outras famílias de endereços, incluindo IPv6. A DVMRP-MIB controla todos os aspectos do DVMRP protocol. Consiste de 6 grupos de objetos e um grupo de notificações. Os grupos de objetos são apresentados na Tabela 2.5.

Além disso, a DVMRP-MIB define três objetos gerenciáveis isolados: o "dvmrpVersionString"(que contém informações da versão DVMRP utilizada), o

Tabela 2.4: Estruturas tabulares que compõem a PIM-MIB

Estrutura Tabular	Descrição
pimInterfaceTable	lista as interfaces nas quais ambos os protocolos PIM e IGMP estão habilitados
pimNeighborTable	lista os roteadores vizinhos pelo protocolo PIM
pimRPSetTable	lista informações sobre os roteadores candidatos a RPs (pontos de encontro ou *Rendezvous Points* na terminologia do protocolo PIM) para grupos de multidifusão IP
pimCandidateRPTable	lista os grupos para os quais o roteador gerenciado é candidato
pimComponentTable	lista informações sobre os domínios PIM aos quais o roteador gerenciado está conectado

Tabela 2.5: Grupos de objetos da DVMRP-MIB

Ferramenta	Descrição
dvmrpGeneralGroup	usado para descrever informações de configuração de forma geral
dvmrpInterfaceGroup	usado para descrever configurações de interfaces, estatísticas e argumentos da tabela de roteamento multicast
dvmrpNeighborGroup	usado para descrever configurações de estatísticas relativas aos roteadores vizinhos
dvmrpTreeGroup	usado para descrever o estado da árvore e construção DVMRP
dvmrpSecurityGroup	usado para gerenciar chaves de segurança DVMRP

objeto "dvmrpNumRoutes"(que contém o número de entradas na tabela de rotas) e "dvmrpReachableRoutes"(que contém o número de entradas na tabela de roteamento com métricas finitas). Além dos três objetos citados acima, a DVMRP-MIB ainda define cinco estruturas tabulares, descritas na Tabela 2.6.

Um protocolo, bastante citado anteriormente, é o protocolo que controla os grupos multicast (IGMP), que também possui uma MIB definida para fornecer seu gerenciamento. A *Internet Group Management Protocol* MIB (IGMP-MIB) define objetos gerenciados, que permitem monitorar a operação do protocolo IGMP em *hosts* e roteadores. Uma implementação desta MIB para roteadores deve fornecer todos os objetos por ela definidos, enquanto uma implementação para *hosts* deve fornecer um sub-conjunto determinado dos mesmos. Basicamente, esta MIB define duas estruturas tabulares, que devem ser implementadas tanto em *hosts* como roteadores. Alguns campos ou colunas são exigidos apenas em roteadores. As estruturas tabulares da IGMP-MIB são descritas na Tabela 2.7.

Além das MIB's, algumas ferramentas podem ser apresentadas para ajudar na tarefa de gerenciamento de multicast, tais ferramentas são bastante simples, no entanto, muito úteis, com relação à monitoração e depuração do tráfego de pacotes multicast, pois auxiliam

Tabela 2.6: Estruturas tabulares que compõem a DVMRP-MIB

Estrutura Tabular	Descrição
dvmrpInterfaceTable	lista as interfaces nas quais ambos os protocolos DVMRP e IGMP estão habilitados
dvmrpNeighborTable	lista os roteadores vizinhos pelo protocolo DVMRP
dvmrpRouteTable	contém as informações de roteamento multicast usadas pelo DVMRP
dvmrpRouteNextHoptable	contém as informações para os próximos *hops*, que devem ser alcançados através do roteamento IP multicat
dvmrpPruneTable	apresenta uma lista de roteadores acima na árvore em estado de *prune*

Tabela 2.7: Estruturas tabulares que compõem a IGMP-MIB

Estrutura Tabular	Descrição
igmpInterfaceTable	apresenta informações sobre a interface na qual o IGMP está habilitado
igmpCacheTable	apresenta informações sobre o grupo com membros em uma interface específica

no gerenciamento de grupos, principalmente aqueles que fazem parte do MBone. As ferramentas e um breve comentário de como elas podem ser utilizadas, serão apresentados na Tabela 2.8.

Tabela 2.8: Ferramentas auxiliares no gerenciamento multicast

Ferramenta	Descrição
mrinfo	as informações fornecidas consistem na versão do programa *mrouted*, como o roteador está conectado a outros roteadores via túneis ou de forma nativa, e a especificação dos túneis (se existirem)
mtrace	apresenta o caminho de difusão seletiva seguido pelos datagramas de uma origem para um determinado destino, coletando estatísticas sobre o tráfego
map-mbone	apresenta a topologia da rede de todos roteadores multicast atingíveis de um dado roteador no MBone

Considerações finais podem ser feitas com relação ao estado atual do gerenciamento de transmissões multicast na internet. O gerenciamento de multicast atualmente está bem estruturado e definido para trabalhar sobre uma rede "*best-effort*", possui bases de informação de gerenciamento para quase todos protocolos de roteamento, assim como bases de informação sobre os nodos que fazem parte de grupos multicast. Entretanto, ainda existem alguns protocolos de roteamento que não possuem bases de informação (MIBs) definidas, como por exemplo, o protocolo MOSPF. E outros que estão em fase

experimental, como por exemplo, a DVMRP-MIB. Além disso, o gerenciamento tem sido feito através da utilização de diversas ferramentas, fator que dificulta o trabalho dos gerentes de rede, pois, além do exigido conhecimento das tecnologias utilizadas pela rede, ainda é necessário dominar cada uma das ferramentas e agrupar as informações coletadas, de forma não-otimizada.

2.3 Gerenciamento de QoS e Multicast

Nas seções anteriores foram apresentadas várias soluções para gerenciamento de QoS e multicast de forma não-integrada. Cada uma destas várias soluções fornece suporte a algum aspecto do gerenciamento de forma isolada para QoS ou para multicast. Neste contexto, a falta de integração das ferramentas de gerenciamento podem trazer dificuldades ao gerente da rede na execução de suas tarefas. Nesta seção serão discutidas algumas das dificuldades encontradas pelos gerentes de rede em gerenciar QoS e multicast de forma conjunta, utilizando-se as soluções encontradas atualmente e apontando-se quais seriam as possíveis soluções para o fornecimento do gerenciamento de QoS e multicast de forma integrada.

Algumas das dificuldades encontradas pelos gerentes em utilizar diversas ferramentas para executar o gerenciamento, de modo geral, serão listadas a seguir:

- O gerente necessita dominar a operação de cada uma das ferramentas utilizadas no gerenciamento;
- Cada ferramenta utilizada possui características e configurações diferenciadas;
- Os resultados (relatórios) apresentados pelas diversas ferramentas utilizadas apresentam formatos não compatíveis entre si ou não seguem o mesmo padrão, dificultando o cruzamento das informações coletadas;
- Fornecimento dos mesmos resultados, diferenciando-se apenas por alguma informação fornecida, que não pode ser obtida através das outras ferramentas encontradas;
- Alguma ação executada em uma ferramenta de gerenciamento deve disparar uma ação em outra ferramenta utilizada. Esta necessidade atualmente deve ser percebida pelo gerente da rede, enquanto num ambiente integrado poderia ser automatizada;
- Os fluxos são analisados e coletados de forma não-integrada, dificultando alguma ação eficaz numa situação onde o mau comportamento de um fluxo pode afetar o comportamento de outro fluxo na rede;

Portanto, além do conhecimento da tecnologia utilizada na rede, ainda é necessário operar diversas ferramentas para construir um resultado significativo. Esta dificuldade acaba por prejudicar o gerenciamento, pois os gerentes acabam por não agregar alguma informação importante ao sistema, por evitar o uso de diversas ferramentas.

No caso do gerenciamento de QoS e multicast, em especial, as dificuldades encontradas podem ser ainda maiores, pois novas informações devem ser consideradas, principalmente com relação aos fluxos de dados existentes. Tal dificuldade se deve à dinamicidade das topologias formadas pelos grupos multicast, que fazem com que o gerenciamento tradicional, fornecido apenas para redes de topologias e domínios

estáticos, tenham de se adaptar a uma topologia totalmente dinâmica. Conseqüentemente, as definições de qualidade de serviço orientadas aos fluxos multicast, também têm de se adaptar às modificações topológicas, dificultando o gerenciamento, principalmente quanto a aspectos de configuração e controle de tráfego de *QoS*. Para completar a lista anterior, será apresentada a seguir uma lista complementar, com os principais problemas encontrados, quando o gerenciamento de QoS e multicast deve ser fornecido:

- Determinar a reserva de recursos de QoS para fluxos multicast, onde a formação dos grupos é dinâmica;
- Gerenciar a distribuição dos fluxos, onde vários receptores recebem o mesmo fluxo simultaneamente;
- Determinar a liberação dos recursos, quando o fluxo multicast deixar de trafegar por determinado dispositivo;
- Determinar quais os ramos da árvore multicast estão recebendo a *QoS* exigida, e quais as políticas adotadas nestas situações.

Como acontece em relação ao gerenciamento específico de *QoS*, a tecnologia que possui maiores vantagens para fornecer o gerenciamento integrado de *QoS* e multicast, é o gerenciamento baseado em políticas, pois utilizando tal tecnologia, é possível definir políticas de forma única, que atendam as necessidades de *QoS* e multicast na rede. Entretanto, o uso do gerenciamento baseado em políticas para gerenciar *QoS* e multicast, assim como para gerenciamento de *QoS*, não está padronizado e ainda existem muitos focos de pesquisa na área. Por isso, o gerenciamento QoS de multicast atualmente não é feito de modo integrado, o que torna o funcionamento das redes que se utilizam das duas tecnologias de forma conjunta, não-confiável.

3 GERÊNCIA INTEGRADA DE QOS E MULTICAST

Este capítulo inicialmente irá apresentar o gerenciamento baseado em políticas (PBNM -*Policy-Based Network Management*) como estratégia de gerenciamento adotada para desenvolver a solução proposta neste trabalho. Em seguida, serão apresentados os problemas que podem ser percebidos quando da utilização de PBNM, em sua forma tradicional, para desenvolver tal solução. Por isso, a implementação do trabalho requer adaptações no modelo de PBNM tradicional. Assim, as modificações sugeridas no modelo atual, visando ao desenvolvimento da solução, também são descritas. Por fim, este capítulo apresenta a solução proposta para fornecimento de gerenciamento integrado de QoS e multicast.

3.1 PBNM(*Policy-Based Network Management*)

Nos capítulos anteriores, em vários momentos, comentou-se sobre PBNM, de modo a introduzir o objetivo do gerenciamento baseado em políticas. Nesta seção, será apresentado o modelo de gerenciamento baseado em políticas do IETF, com um nível maior de detalhes, apresentando sua arquitetura geral e elementos mais importantes, o estado atual do desenvolvimento de pesquisas relevantes na área, e comentários sobre como algumas empresas desenvolvedoras de tecnologia para gerenciamento de rede têm demonstrado interesse por tal tecnologia.

O gerenciamento baseado em políticas foi inicialmente proposto por Sloman (SLOMAN, 1994). O autor descreveu uma metodologia para especificar parâmetros de comportamento, mais abstratos, a serem cumpridos por cada elemento da rede, da melhor maneira possível, levando-se em conta as características de cada elemento. Em um sistema heterogêneo, constituído por diversos equipamentos de diversos fabricantes, essas políticas poderão assumir valores distintos, em função das características particulares de cada elemento gerenciado. Em outras palavras, o PBNM procura introduzir um nível de abstração de informações de gerenciamento, para facilitar o cumprimento das tarefas executadas pelo administrador da rede. Dois exemplos de política com alto nível de abstração são apresentados a seguir:

(1) "O tráfego Web deve receber ao menos 50% dos recursos disponíveis"
(2) "Em qualquer interface que essa política for aplicada, garantir ao menos 30% da largura de banda para fluxos UDP e, no mínimo, 40% da largura disponível para fluxos TCP, entre 10 e 11 horas da manhã"

Quando o administrador da rede passa a lidar com muitas estruturas diferentes, o

gerenciamento passa a ser complexo e, conseqüentemente, mais difícil de ser mantido. Com a abstração fornecida, o administrador da rede preocupa-se em determinar as políticas de gerência a serem usadas. Os sistemas de gerência, por sua vez, preocupam-se em interpretar essas políticas e implantá-las na rede. Abaixo são apresentadas as duas políticas abstratas citadas anteriormente, que devem ser interpretadas pelos ambientes de gerência, através de uma linguagem mais formal:

(1) `If <protocol == HTTP> then <minimum BW = 50%>`
(2) `If( <timeofDay >= 10:00am> and <timeofday <= 11:00am> ) then (`
`If <IP protocol is UDP> then <guarantee 30% of avaiable BW>`
`and If <IP protocol is TCP> then <guarantee 40% of avaiable BW> )`

A arquitetura de um sistema de gerenciamento baseado em políticas foi proposta pelo IETF (LORENZ; ORDA, 1998) (VASILAKOS et al., 1998) e é mostrada na Figura 3.1. O PMT (*Policy Management Tool*) é a interface com o administrador da rede, que especifica as políticas. PR (*Policy Repository*) é o banco de dados de políticas. PDP (*Policy Decision Point*) é o componente responsável pela interpretação da política e pela decisão de configuração dos equipamentos. Sua função é adaptar a política para cada dispositivo que está sendo gerenciado. A tecnologia sugerida para comunicação entre o PR e o PDP é o protocolo LDAP (*Lightweight Directory Access Protocol*) (WAHL et al., 1997). PEP (*Policy Enforcement Point*) são entidades lógicas que executam as ações determinadas pelo PDP. A tecnologia sugerida para comunicação com o PEP é realizada através do protocolo COPS (*Common Open Policy Service*) (LI; NAHRSTEDT, 1999). Entretanto, protocolos como SNMP, telnet, entre outros, podem ser utilizados da mesma forma.

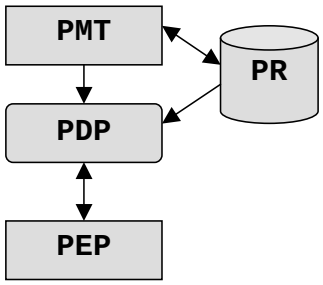

Figura 3.1: Arquitetura de gerenciamento baseado em políticas

O elemento chave desta arquitetura é o PDP, que executa grande parte do processamento de controle do sistema. É nesse elemento que as políticas de alto nível são traduzidas para ações compreendidas pelos elementos alvo na rede e, posteriormente, aplicadas nos PEPs. Portanto, a implementação do PDP deve conhecer em detalhe cada um dos PEPs existentes na rede e a melhor forma de comunicação com eles. Além disso, a existência de mais de um PDP aumenta a complexidade da arquitetura, porque as decisões passam a ser distribuídas. Por outro lado, tem-se um aumento na robustez da solução, porque se um PDP deixar de funcionar, os outros podem assumir suas funções. Quando da existência de mais de um PDP no ambiente de gerenciamento, a implementação do PMT também torna-se mais complexa, pois é através dele que serão definidas quais políticas serão aplicadas e quais os PDPs distribuídos na rede terão de implantar tais políticas. O PMT ainda deve servir como elemento centralizador para controle das notificações de falhas, no caso de políticas aplicadas de forma distribuída.

Inicialmente, o IETF apresentou os elementos e a terminologia (WESTERINEN et al., 2001) para a definição de uma arquitetura geral para PBNM. Esta definição foi proposta através de uma *draft* que não teve continuidade. Atualmente, o *Policy Framework Working Group* do IETF tem suas atenções voltadas em definir como as políticas devem ser organizadas através de um modelo de informação que seja capaz de suportar qualquer tipo de política definida. Os primeiros resultados foram obtidos através da padronização do *Policy Core Information Model* (MOORE et al., 2001), que apresenta a visão do IETF de como as políticas devem ser organizadas e definidas.

De acordo com o IETF, as políticas devem ser aplicadas segundo um conjunto de condições que disparam um conjunto de ações. Tal relacionamento entre condições e ações é chamado de regra. Nesse contexto, uma política deve ser composta por uma ou mais regras. Outro resultado apresentado foi o *Policy Core Information Model Extensions* (MOORE, 2003), que define novas classes para definição de políticas mais detalhadas, além de modificações na proposta anterior, no sentido de tornar a definição de políticas mais flexível.

O IETF também aposta nos serviços de diretórios para armazenar as políticas, por isso vem propondo, através de *drafts* (STRASSNER et al., 2002), o mapeamento dos modelos de informação para estruturas de armazenamento em serviços de diretórios LDAP. Outras pesquisas relevantes na área são propostas. O *toolkit* Ponder (DAMIANOU et al., 2002) (DAMIANOU et al., 2001), projeto coordenado por Morris Sloman, oferece um conjunto de ferramentas para definição e aplicação de políticas. Diferentemente do IETF, Ponder trabalha baseado em eventos. Tais eventos são percebidos pelo ambiente de gerência, que aplica as políticas adequadas de acordo com o evento percebido. Segundo o IETF, as regras que percebem a declaração de tais eventos, são definidas implicitamente nos modelos de informação PCIM e PCIMe.

Além de grupos de pesquisa, empresas desenvolvedoras de tecnologia para redes têm se mostrado bastante interessadas no aprimoramento, padronização e desenvolvimento do PBMN. Empresas de grande porte como CISCO, IBM, HP têm participado ativamente das tentativas de padronização de definições envolvendo PBNM. Algumas empresas como a HP também possuem ferramentas que utilizam PBNM (e.g. HP Openview PolicyXpert) (HP, 2001) como modelo de gerenciamento. Mas, de modo geral, o gerenciamento baseado em políticas necessita evoluir. Alguns tópicos do gerenciamento têm sido abordados em detalhes, como QoS e segurança. Entretanto, outras áreas funcionais do gerenciamento (e.g. gerenciamento de falhas e configuração) deveriam receber uma maior atenção e suporte no PBNM, devido às suas características e importância, dentro do gerenciamento de redes como um todo.

3.2 Problemas com políticas tradicionais e gerência integrada de QoS e multicast

De acordo com as definições apresentadas pelo IETF, dois modelos de organização de políticas com baixo nível de abstração podem ser usados. O primeiro modelo, conhecido como *Police Core Information Model - PCIM* (MOORE et al., 2001), é o modelo tradicional para organização de políticas, e tem sido utilizado como referência atualmente. O segundo modelo, chamado *Police Core Information Model Extensions - PCIMe* (MOORE, 2003), estende o modelo tradicional, adicionando novas classes e atributos, tornando a organização de políticas, através deste modelo, mais detalhada e flexível. O novo modelo foi padronizado, recentemente, e ainda não possui muitas

implementações disponíveis, apesar de seu potencial com relação à organização de políticas.

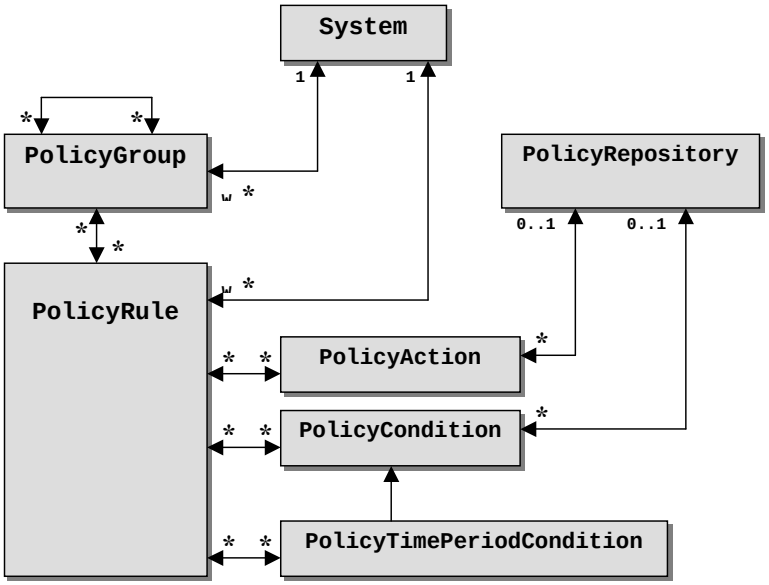

Figura 3.2: Policy Core Information Model

O modelo de dados representado no PCIM (Figura 3.2) possibilita a definição de políticas limitadas com relação à QoS e multicast, pois cada política pode ser composta apenas por um conjunto de ações e condições, um conjunto de regras (que por sua vez são compostas de ações e condições) e um conjunto de grupos de regras. Tais regras são definidas linearmente, o que acaba limitando a construção das políticas com relação ao aspecto hierárquico das mesmas. Por outro lado, o único mecanismo para definir o esquema de execução de uma política é um atributo de prioridade, vinculado às regras, de modo a determinar qual a prioridade de execução entre regras de uma mesma política. Assim, como comentado anteriormente, políticas para QoS e multicast têm uma característica especial de execução devido a dinâminicidade de localização dos fluxos, fator que acaba tornando inviável a definição de tais políticas, utilizando o PCIM como modelo. Outra alternativa é a adoção de um mecanismo auxiliar para definição políticas em conjunto com o PCIM. Um exemplo de política para QoS e multicast, seguindo unicamente o PCIM, é apresentada na Figura 3.3.

A política apresentada na Figura 3.3 é aplicada da seguinte maneira. No momento especificado nas políticas, as seguintes regras são aplicadas: `EnableMulticastSupport` e `QoSforMulticastGroup`. Quando a regra `QoSforMulticastGroup` é aplicada, os recursos necessários para QoS são alocados. As configurações necessárias para tráfego de fluxo multicast são realizadas no momento em que a regra `EnableMulticastSupport` também é aplicada. No momento em que um novo membro entra no grupo multicast, os recursos necessários com relação à QoS já estão alocados. Quando qualquer membro do grupo multicast deixa de fazer para o grupo multicast, os recursos relativos à QoS continuam alocados desnecessariamente.

Em qualquer uma das arquiteturas de fornecimento de QoS, onde políticas como a apresentada na Figura 3.3 são aplicadas para fornecimento de gerenciamento de QoS e multicast, o desperdício de recurso é inaceitável, devido à utilização desnecessária de memória dos dispositivos intermediários, e principalmente se disciplinas

```
Policy: Policy for QoS and multicast
Rule:   EnableMulticastSupport
        if (timeOfDay >= 4pm) and (timeOfDay <= 6pm)
        then
            MroutedEnabled = true
            QueryInterval = 1000 ms
            MRoutingProtocol = DVMRP
Rule:   QoSforMulticastGroup
        if (timeOfDay >= 4pm) and (timeOfDay <= 6pm)
            and (DestAddr == 224.0.0.214)
        then
            Bandwidth = 300 Kbps
            MaxJitter = 10%
            MaxLoss = 30%
```

Figura 3.3: Política para gerenciamento de QoS e multicast usando PCIM

de enfileiramento de pacotes como *Custom Queued* (Cisco, 1999) estiverem sendo utilizadas.

Neste contexto, considerando a forma ideal de aplicação de uma política para QoS e multicast de forma integrada, o modelo sugerido no PCIM não suporta a definição de políticas que expressem o comportamento esperado da rede nessas situações. Não existem no PCIM mecanismos que possibilitem a definição da estratégia de execução das regras de forma dinâmica. Sendo assim, é necessário agregar elementos externos à política para determinar as estratégias de execução. Uma das formas de definição desta estratégia é utilizar FSM (*Finite State Machine*) (GRANVILLE et al., 2002) (VAGUETTI et al., 2003) (GRANVILLE et al., 2003). Utilizando-se FSM como recurso auxiliar, a política é definida como apresentado na Figura 3.4.

No exemplo apresentado na Figura 3.4 tem-se uma política que deve ser aplicada dentro de um determinado período de tempo (no caso, entre 4 e 6 horas da tarde). Abstraindo os aspectos temporais da política, existem algumas regras que devem ser aplicadas em resposta a alguns eventos reportados pela rede. Tais regras são disparadas através de *scripts*, que por sua vez estão associados aos estados da FSM. A transição dos estados da regra está diretamente ligada aos eventos. No exemplo, tem-se 2 regras, uma que configura aspectos relativos à multicast (*Rule1*) e outra que configura os aspectos relativos à QoS (*Rule2*). Além disso, a política possui três scripts, onde S1 aplica a *Rule1*, S2 aplica a *Rule2* e S3 remove a *Rule2*. De acordo com a FSM definida, quando a política é aplicada, o estado 1 é disparado (nesse caso, o *script* S1 é executado) configurando o multicast da rede, se o evento correspondente à entrada de um novo membro no grupo especificado, é percebido, então, a FSM passa ao estado 2 (S2) onde a QoS é configurada. Se é percebida a saída de todos os membros do grupo, então o estado 3 (S3) é executado, liberando os recursos de QoS. Entretanto, se algum membro voltar a fazer parte do grupo, os recursos de QoS devem ser novamente configurados, nesse caso, a FSM volta ao estado 2 (S2). Quando a política não for mais válida, temporalmente todas as regras são removidas.

Além do PCIM, existe outro modelo proposto, recentemente, para organizar a definição de políticas. Utilizando-se do modelo apresentado no PCIMe (Figura 3.5), é possível definir políticas integradas para QoS e multicast, sem a necessidade de uso de um mecanismo auxiliar para definir a estratégia de execução das regras em uma política. A principal diferença entre PCIMe e PCIM, com relação à estratégia de

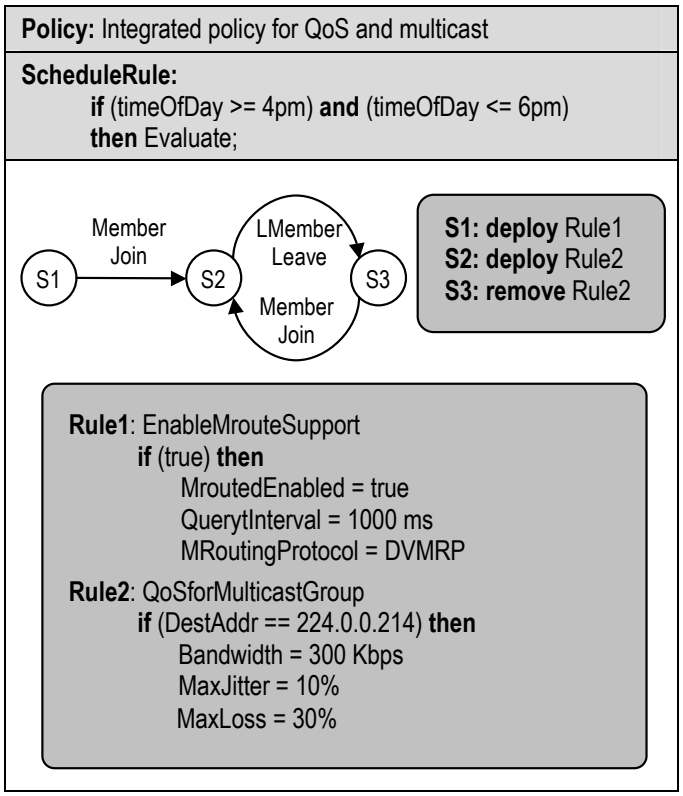

Figura 3.4: Política para gerenciamento integrado de QoS e multicast usando PCIM e FSM

execução, é a criação de uma classe intermediária chamada `PolicySet`, que possui relacionamentos com outras classes que permitem as regras serem definidas com um número ilimitado de níveis hierárquicos. Além disso, a mesma classe ainda define um atributo (`PolicyDecisionStrategy`) usado para definir a estratégia de prioridade dentre um conjunto de regras organizadas de forma hierárquica. Outra modificação importante foi a inclusão de um novo atributo (`ExecutionStrategy`) na classe `PolicyRule`. Tal atributo permite definir qual a estratégia de execução de cada regra. Através desse, é possível configurar se uma regra deve ser aplicada durante todo o tempo de execução da regra, se deve ser aplicada até o momento que falhar a primeira vez, ou se deve ser aplicada sempre que tiver sucesso de execução. Uma política, com o mesmo objetivo de aplicação dos modelos anteriores, pode ser definida usando o novo modelo, como apresentado na Figura 3.6).

A forma como a política definida, usando PCIMe, é aplicada, é descrita a seguir. No momento em que a política é aplicada a `Rule1` é verificada, nesse caso, é verificada a condição de tempo de aplicação da regra. Também na `Rule1` são definidos dois atributos. O atributo `ExecutionStrategy` é setado de forma que a regra seja verificada durante todo o tempo especificado como condição; já o atributo `DecisionStrategy` define que todas as ações componentes da regra devem ser executadas conjuntamente. Num nível hierárquico inferior está a `Rule2` (que faz parte da lista de ações da `Rule1`). Tal regra habilita o suporte à multicast, determina que tais valores devem ser mantidos até que a regra imediatamente superior (`Rule1`) deixe de executar. Além disso, a `Rule2` possui duas ações que disparam duas novas regras (`Rule3` e `Rule4`) e os atributos `DecisionStrategy` e `ExecutionStrategy`. No caso das regras `Rule3` e `Rule4`, uma

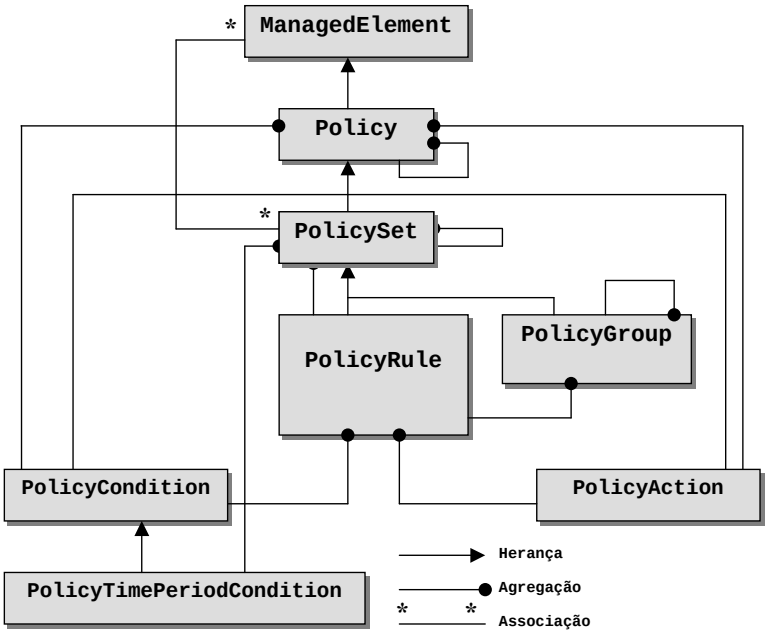

Figura 3.5: Policy Core Information Model Extensions

é contrária à outra, ou seja, se o fluxo multicast para o grupo 224.0.0.214 for percebido, a `Rule3` torna-se válida e suas ações (configurações relativas à QoS) são executadas; caso contrário, se a fluxo multicast, em questão, não for percebido, a `Rule4` torna-se válida, retirando as configurações setadas pela `Rule3`. O elemento que torna possível esse tipo de execução é o atributo `ExecutionStrategy`, que define que as duas regras devem sempre ser aplicadas, independentemente de falharem ou não, enquanto as regras superiores forem válidas.

Nesta seção foram apresentados os problemas encontrados quando da definição de políticas para gerenciamento integrado de QoS e multicast, usando o modelo tradicional de organização e as formas possíveis de solucionar tais problemas, tendo como base as orientações do IETF. Entretanto, nas duas formas de organização possíveis, modificações na arquitetura tradicional de PBNM são necessárias. No caso da adoção de PCIM como modelo, a arquitetura deve ser complementada com um mecanismo, como por exemplo FSM, para definir a estratégia de execução. Utilizando o PCIMe, não é necessário um mecanismo auxiliar, entretanto, elementos que verifiquem constantemente a validade das condições das regras, devem existir para dar suporte ao que é definido no atributo `ExecutionStrategy`. A forma de organização de políticas adotada, nesse trabalho, e a nova arquitetura de PBNM proposta para suportar tal modelo de organização, serão discutidas na próxima seção.

3.3 Solução Proposta

Duas formas de organização de políticas foram apresentadas na seção anterior: uma forma de organização usando o modelo PCIM e outra usando o PCIMe. Neste trabalho, a forma de organização de políticas adotada é o PCIMe. Tal modelo foi escolhido, pois, além de herdar todas as facilidades de organização de políticas contidas em PCIM (e.g `PolicyCondition` e `PolicyAction`), ainda oferece elementos que possibilitam a declaração de políticas, sem que seja necessário acrescentar novos elementos na estrutura

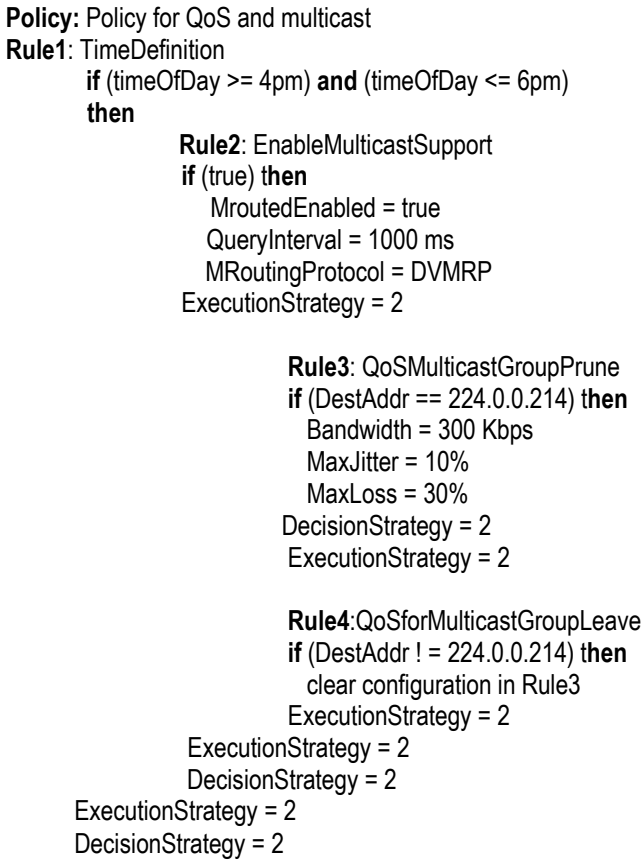

Figura 3.6: Política para gerenciamento integrado de QoS e Multicast usando PCIMe

básica de organização das políticas para determinar seu funcionamento.

Dentre os novos atributos oferecidos em PCIMe, estão atributos que possibilitam a definição da forma como as regras devem ser aplicadas internamente à política ((e.g. `pcimDecisionStrategy` e `pcimExecutionStrategy`), e um conjunto básico de classes e atributos que definem variáveis e valores (e.g. `pcimVariable` e `pcimValue`), que podem ser utilizados na definição das condições e ações, de uma política individualmente. Entretanto, as classes e atributos oferecidos inicialmente em PCIMe atendem apenas às necessidades de definição de políticas básicas (e.g. políticas para controle de fluxo e políticas de filtragem de pacotes). Assim, se políticas mais específicas a uma determinada tecnologia necessitam ser definidas, torna-se necessária a criação de atributos e classes específicos para declaração de tais políticas (e.g. atributos e classes para definição de variáveis e valores relacionados à QoS e multicast). Porém, PCIMe já possui suporte à declaração de novas classes e atributos, permitindo que novas variáveis e valores sejam agregados à PCIMe, de forma a permitir a declaração de condições e ações não-previstas anteriormente. A modelagem apresentada em PCIMe permite que a organização dos dados torne-se extensível (e.g. `pmcastGroupIPv4VariableAuxClass`, classe definida de forma a extender o PCIMe para que variáveis que definam endereços de grupos multicast possam ser declarados em políticas de multicast).

Como comentado anteriormente, além de PCIMe, outros modelos de informação são necessários para declaração de políticas integradas de QoS e multicast. Assim, a declaração de modelos auxiliares é necessária, pois atributos diretamente relacionados à QoS e multicast não são encontrados diretamente em PCIM e PCIMe. Entretanto, tais

políticas não podem ser declaradas, sem a utilização de tais elementos (e.g. `qpMaxDelay`, variável definida em um modelo de informação auxiliar para que o atraso máximo de um pacote durante uma troca de mensagens pudesse ser definida em políticas para QoS). Para declarar os atributos relativos à QoS, este trabalho utiliza-se de um modelo de informação também definido pelo IETF: trata-se do PQIM (*Policy QoS Information Model*). No caso da declaração dos atributos relativos à multicast, torna-se necessária a criação de um modelo auxiliar que suporte a declaração de variáveis e valores específicos para multicast, pois o IETF ainda não oferece nenhum tipo de suporte à declaração de tais políticas.

```
Policy: Policy for QoS and multicast
Rule1: TimeDefinition
        if (pcimTPCtimeOfDay >= 8am) and (pcimTPCtimeOfDay <= 11am)
        then
                Rule2: EnableMulticastSupport
                if (pmcastMroutedEnabled == false) then
                    pmcastMroutedEnabled = true
                pcimExecutionStrategy = 2

                        Rule3: QoSMulticastGroupPrune
                        if (pmcastGroupIPv4 == 224.0.0.13)
                        then
                            qpBandwidthUnits = 0
                            qpMinBandwidth = 300 Kbps
                            qpMaxBandwidth = 450 Kbps
                        pcimDecisionStrategy = 2
                        pcimExecutionStrategy = 2

                        Rule4:QoSforMulticastGroupLeave
                        if (pmcastGroupIPv4 ! = 224.0.0.13)
                        then
                            clear configuration in Rule3
                        pcimExecutionStrategy = 2
                pcimExecutionStrategy = 2
                pcimDecisionStrategy = 2
        pcimExecutionStrategy = 2
        pcimDecisionStrategy = 2
```

Figura 3.7: Exemplo de Política para gerenciamento de QoS e Multicast usando PCIMe

A Figura 3.7 apresenta um exemplo de política integrada para QoS e multicast, utilizando-se do PCIMe e alguns modelos auxiliares que permitem a definição de tal política. A política apresenta um conjunto de regras que permitem o gerenciamento de um fluxo multicast que necessita de garantias de QoS. A forma como as regras estão dispostas internamente à política, e as condições e ações utilizadas em tais regras determinam como a política vai ser aplicada no decorrer do tempo válido para funcionamento da mesma. A política exemplo possui quatro regras, sendo que algumas destas regras estão declaradas internamente a uma regra posicionada de forma superior na hierarquia de regras. A regra mais superior à política (`Rule1`) define o período de tempo de aplicação da regra como condição, e dentre suas ações, é declarada uma nova regra (`Rule2`), que controla o suporte a fluxos multicast. Dentre as ações da `Rule2` estão definidas duas novas regras (`Rule3` e `Rule4`) que, de acordo com um determinado fluxo multicast, atendem ou não às necessidades de QoS exigidas. Na declaração da política é possível também observar a utilização de atributos definidos em PCIMe (e.g. `pcimDecisionStrategy`), QPIM (e.g. `qpBandwithUnits` e variáveis e atributos definidos neste trabalho para

declaração de condições e ações relacionadas à multicast (e.g. `pmcastMRouteEnable` e `pmcastGroupIPv4`). Uma apresentação mais detalhada da política é feita no final da seção.

Além dos atributos citados anteriormente, um importante atributo, definido no PCIMe e utilizado na política exemplo, é o atributo `ExecutionStrategy`. Tal atributo exige da arquitetura tradicional de PBNM um controle que não seria necessário utilizando PCIM, pelo menos de forma explícita. Entretanto, para que a política exemplo ou qualquer política definida, baseada na organização fornecida no PCIMe, seja aplicada corretamente em uma arquitetura de PBNM, é necessário que a arquitetura tenha suporte a esse tipo de controle. A Figura 3.1 apresentou a arquitetura tradicional de PBNM, definida antes da padronização do PCIMe. No entanto, nenhum dos elementos apresentados na Figura 3.1 fornece alguma forma de controle de como as políticas estão sendo executadas. Após a definição do PCIMe, algum elemento na arquitetura deve ser responsável por desempenhar essa função vital, sob pena de que políticas que se utilizem de PCIMe não possam ser aplicadas. O elemento mais adequado para desenvolver este tipo trabalho na arquitetura tradicional seria o PDP, porém, segundo literatura estudada, a função do PDP é de tradução de políticas de alto nível para políticas mais específicas, tornando-as possíveis de serem aplicadas nos PEPs. Uma nova arquitetura foi definida neste trabalho. Tal arquitetura busca fornecer o suporte necessário para utilização de políticas baseadas no PCIMe, e apresenta com maior nível de detalhes os elementos necessários para aplicação de políticas integradas de QoS e multicast utilizando-se do PCIMe (Figura 3.8).

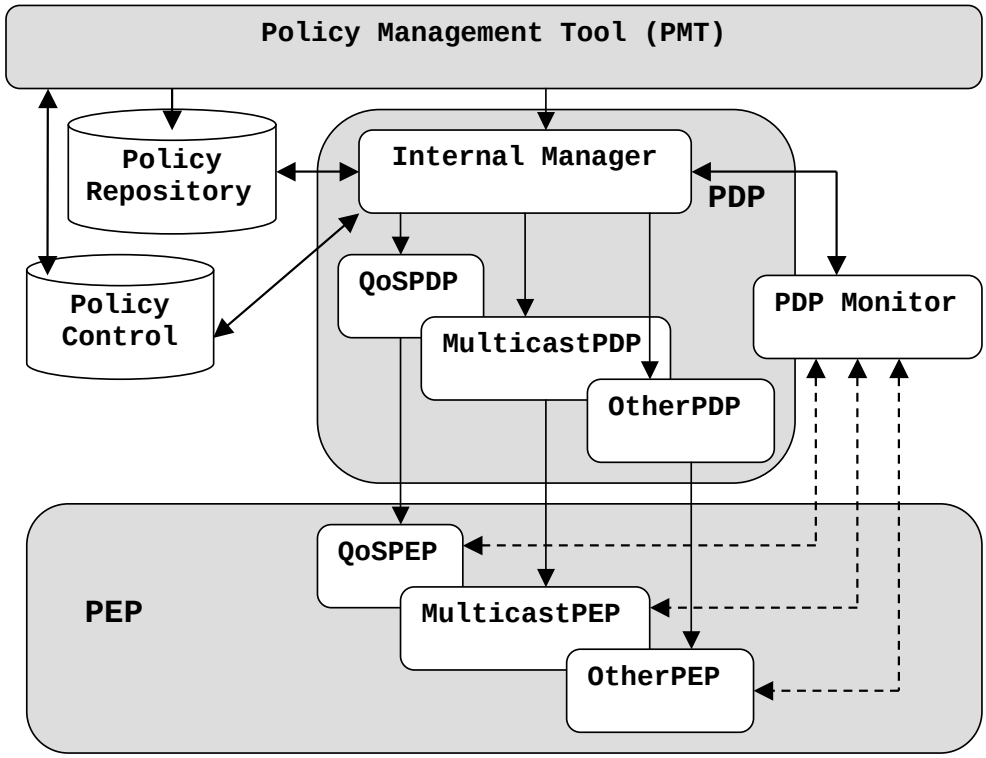

Figura 3.8: Novos elementos na Arquitetura PBNM

A Figura 3.8 apresenta a nova arquitetura e um conjunto de novos elementos adicionados à arquitetura tradicional, com objetivo de prover suporte ao PCIMe. Os elementos PMT, PDP, PEP e PR já foram explicados anteriormente, porém novos elementos são destacados internamente aos elementos anteriores, e estes serão

comentados a seguir.

Inicialmente são divididas as funções do PDP em duas funcionalidades diferentes. A primeira funcionalidade é a de controle de aplicação das políticas. Tal funcionalidade controla a execução de cada regra e onde cada política vai ser aplicada no ambiente de gerenciamento. Seus elementos principais são o `Policy Control`, que armazena informações que ditam em quais PDPs e PEPs determinada política deve ser aplicada, além do status de aplicação de cada política, com relação aos elementos (e.g. PEPs). Tal elemento é atualizado quando necessário pelo `Internal Manager`. `Internal Manager` é o elemento que controla a execução de uma política e conseqüentemente cada regra dentro da política. Uma de suas funções é verificar a validade de cada política com relação a cada PEP associado, e atualizar o `Policy Control` para que, a partir destas informações, o `PMT` possa executar as ações necessárias com relação à política. Outra função do `Internal Manager` é verificar questões temporais relacionadas à política, como por exemplo, se uma política deve ser aplicada entre quatro e cinco horas da manhã. Além das funções anteriores, o `Internal Manager` ainda é responsável por verificar a relação hierárquica de execução de uma política (e.g. quando as regras filhas tornam-se inválidas, a regra pai deve tornar-se inválida também, e vice-versa) e verificar o status de execução das regras dentro da política, de acordo com o que está definido no atributo `ExecutionStrategy`. Para que o `Internal Manager` possa fazer a verificação do status de execução das regras, um outro elemento definido é utilizado para auxiliá-lo. Trata-se do `PDPMonitor`.

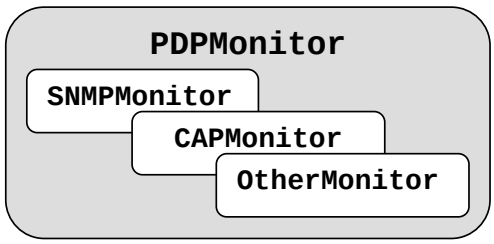

Figura 3.9: PDPMonitor

O `PDPMonitor` pode ser considerado uma extensão do PDP ou um novo elemento na arquitetura. Neste trabalho ele é considerado uma extensão do PDP, pois sua função é dar suporte às decisões do `Internal Manager`, que está situado internamente ao PDP. O `PDPMonitor` tem como função principal verificar o *status* das condições definidas em cada regra (e.g. verificar se a existência de fluxo multicast para o grupo 224.0.0.13, de acordo com a `Rule3` da política exemplo). Tais condições devem ser notificadas ao o `PDPMonitor` pelo `Internal Manager` que, por sua vez, verifica as condições atribuídas para verificação. Diversas verificações de condições podem ser solicitadas ao `PDPMonitor`. Por isso é necessário que ele suporte várias formas de monitoração (e.g. através de snmp ou através de captura de pacotes). A Figura 3.9 apresenta o `PDPMonitor` com um nível maior de detalhe, destacando-se alguns tipos de monitoração que podem ser suportadas pelo elemento (e.g. SNMPMonitor e CAPMonitor).

A segunda funcionalidade do PDP é a tradução das políticas definidas num nível mais abstrato para políticas de mais baixo nível, objetivando a aplicação direta a um PEP do sistema. Neste contexto, a arquitetura utiliza-se de elementos tradutores específicos para cada tecnologia aplicada (e.g. `QoSPDP` e `MulticastPDP`). Tais elementos são acionados pelo `Internal Manager` no momento que as regras relacionadas a cada tecnologia tornam-se válidas. Por exemplo, quando um fluxo multicast para o grupo

224.0.0.13 é percebido, as ações definidas na Rule3 são aplicadas. Para facilitar a aplicação, PEPs específicos para cada tecnologia também são propostos (e.g QoSPEP e MulticastPEP). No exemplo citado anteriormente, os elementos internos QoSPDP e QoSPEP são utilizados para aplicação da regra; em outro contexto, poderiam ser usados outros elementos tradutores e aplicadores.

A seguir, será descrito como se comporta a aplicação da política apresentada, na Figura 3.7, na nova arquitetura apresentada, na Figura 3.8. Inicialmente a política é definida e armazenada no Policy Repository, neste momento a definição de quais PDPs e PEPs aplicarão a política, também é armazenada no Policy Control. Todas estas definições são feitas através do PMT, que também tem a função de notificar ao PDP que uma nova política está armazenada no PR. Quando tal notificação acontece, o PDP, através do Internal Manager, baixa as políticas para seu modulo de controle e inicia o processo de controle das regras da política. Como as regras das políticas definidas, usando PCIMe podem ter vários níveis hierárquicos, o Internal Manager preocupa-se também com a hierarquia de execução. A primeira regra a ser verificada na política exemplo é a Rule1. Esta regra apresenta uma condição temporal, definindo que tal regra e suas ações devem ser aplicadas entre oito e onze horas da manhã. No momento que esta regra se torna válida, suas ações são aplicadas, neste caso, é disparada a Rule2. Os atributos pcimExecutionStrategy e pcimDecisionStrategy também são setados, garantindo que regra deve ser verificada, até que a condição de tempo seja inválida.

Num segundo momento, a Rule2 é verificada, neste caso, o elemento PDPMonitor é utilizado para verificar se a capacidade de rotear fluxo multicast é válida nos PEPs em questão. Em caso negativo, as ações determinadas na regra devem ser aplicadas. Dentre as ações, destaca-se a habilitação do roteamento multicast, através do atributo pmcastMRoutedEnabled (utilizando os elementos PDPMulticast e PEPMulticast) e duas novas regras: Rule3 e Rule4. A verificação da validade dessa regra é determinada para que deixe de ser verificada somente quando a Rule1 torna-se inválida. As regras Rule3 e pcimRule4 passam a ser verificadas utilizando novamente o PDPMonitor. Neste caso as condições das regras são contrárias, uma verifica se existe fluxo multicast para o grupo 224.0.0.13, e a outra verifica se não existe tal fluxo. Se o fluxo existir, a Rule3 é aplicada configurando-se os parâmetros relativos à QoS usando os elementos PDPQoS e PEPQoS; caso contrário, a Rule4 torna-se ativa, desconfigurando o QoS aplicado pela Rule3. As duas regras são verificadas até que a regra imediatamente superior (Rule2) torne-se inválida, porém as duas regras são constantemente verificadas, devido à utilização do atributo pcimExecutionStrategy, na definição da política.

4 IMPLEMENTAÇÃO DO PROTÓTIPO

Neste capítulo será apresentada a implementação do protótipo responsável por declarar e aplicar políticas integradas de QoS e multicast em Redes IP. Inicialmente será demonstrado como as políticas podem ser criadas num ambiente baseado na Web, e como tais políticas são controladas através do mesmo ambiente. Após a demonstração inicial, será demonstrado como os elementos da arquitetura implementada se comunicam, quais tecnologias são utilizadas para tal comunicação, e quais as tecnologias utilizadas na implementação de cada elemento da arquitetura. No final do capítulo, serão apresentados os detalhes internos da implementação.

4.1 Operações e funcionamento do protótipo

O protótipo implementado está integrado ao ambiente de gerenciamento QAME (*QoS-Aware Management Environment*), e é a partir do ambiente gráfico oferecido pelo sistema, que serão demonstradas as operações e o funcionamento do protótipo.

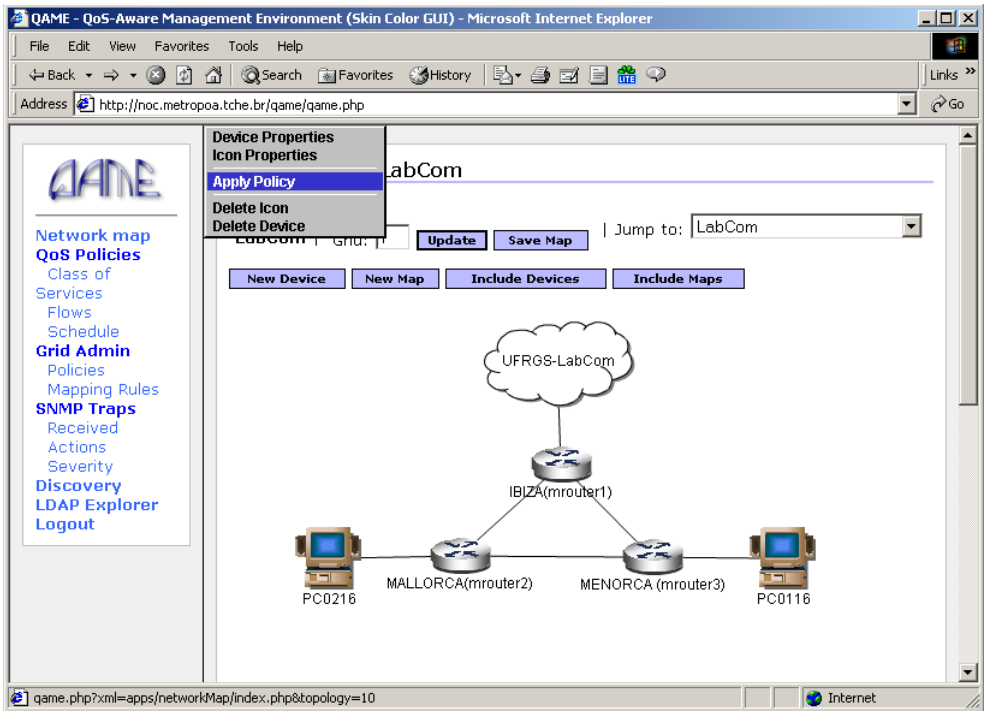

Figura 4.1: Módulo de associação de políticas e PDPs aos PEDs

Os primeiros módulos apresentam o funcionamento da operação mais executada por

parte do administrador da rede, que se resume ao fato de atribuir políticas previamente cadastradas aos PEPs, nos quais as políticas serão aplicadas. Para que as políticas sejam aplicadas, o sistema deve possuir informações prévias das próprias políticas, dos PEPs, onde as políticas serão aplicadas, e dos PDPs que irão aplicar tais políticas no momento oportuno. Os módulos que tratam do cadastro de políticas, PEPs e PDPs, serão apresentados posteriormente.

Inicialmente é possível determinar, através do ambiente gráfico, quais políticas são aplicadas a cada PEP e quais PEPs vão ser associados a cada PDP existente no sistema. Para que isso seja possível, é necessário selecionar o elemento dentro de uma determinada topologia, abrir um pequeno menu associado a cada elemento, através de uma combinação de teclas, e selecionar a opção "Apply Policy"(Figura 4.1). O sistema identificará se tal elemento foi registrado como PEP ou PDP no sistema. Se o elemento foi cadastrado como PEP, uma tela, com a opção de associar políticas ao elemento selecionado, será apresentada ao usuário. Através desse módulo de associação, várias políticas podem ser associadas ao elemento selecionado. Ainda é possível definir um nome, uma breve descrição, além de editar e deletar tal associação, se for necessário (Figura 4.2). As Figuras 4.1 e 4.2 demonstram o elemento `IBIZA` sendo selecionado através do ambiente, e sendo associado às políticas denominadas: `Policy1`, `QoSMulticastPolicy` e `QoSPolicy`.

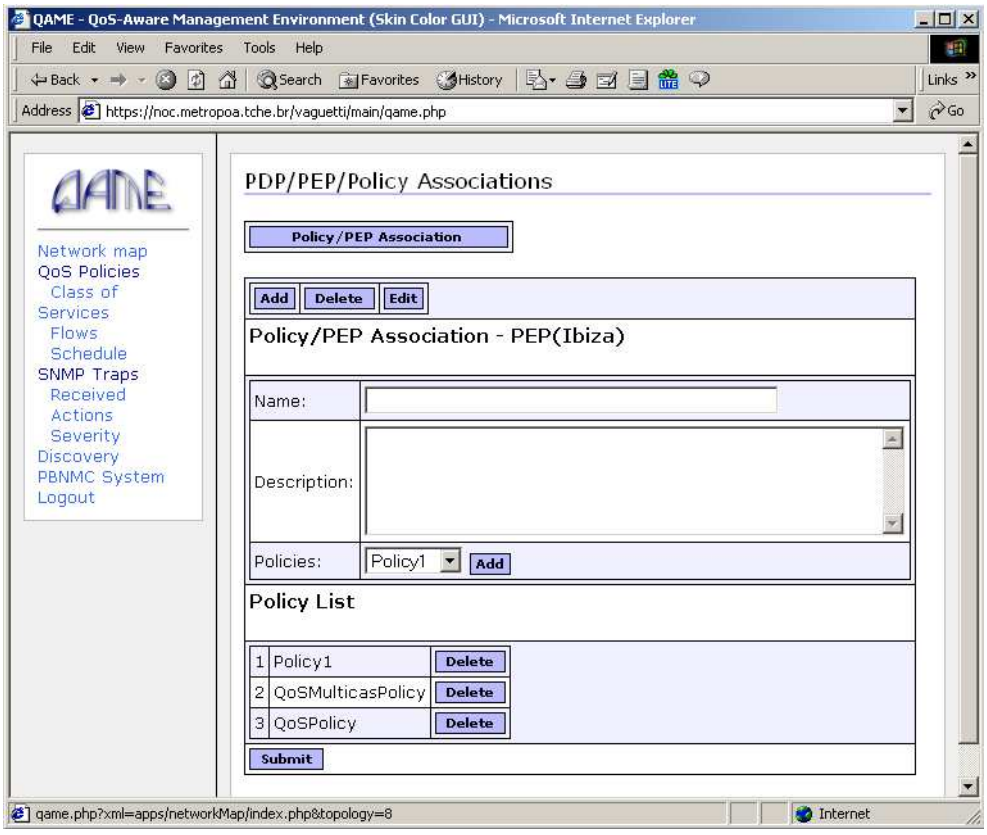

Figura 4.2: Módulo de Associação Policy/PEP

Se o elemento selecionado foi cadastrado como PDP, então uma opção de cadastro diferente da apresentada aos PEPs é fornecida. Neste caso é possível associar o PDP selecionado aos PEPs existentes no sistema. Quando um PEP é associado a um PDP, tal PDP é responsável por aplicar todas as políticas vinculadas ao PEP associado. O

sistema automaticamente faz a associação entre as políticas que devem ser aplicadas a cada PEP ao PDP, ao qual os PEPs foram associados. Vários PEPs podem ser associados a um mesmo PDP (Figura 4.3). A Figura 4.3 demonstra o PDP identificado através do elemento `MALLORCA`, sendo associado aos PEPs `IBIZA` e `PC0216`. Neste caso as políticas associadas a tais PEPs serão automaticamente vinculadas ao PDP (`MALLORCA`). Este módulo também apresenta as possibilidades de edição de deleção das associações.

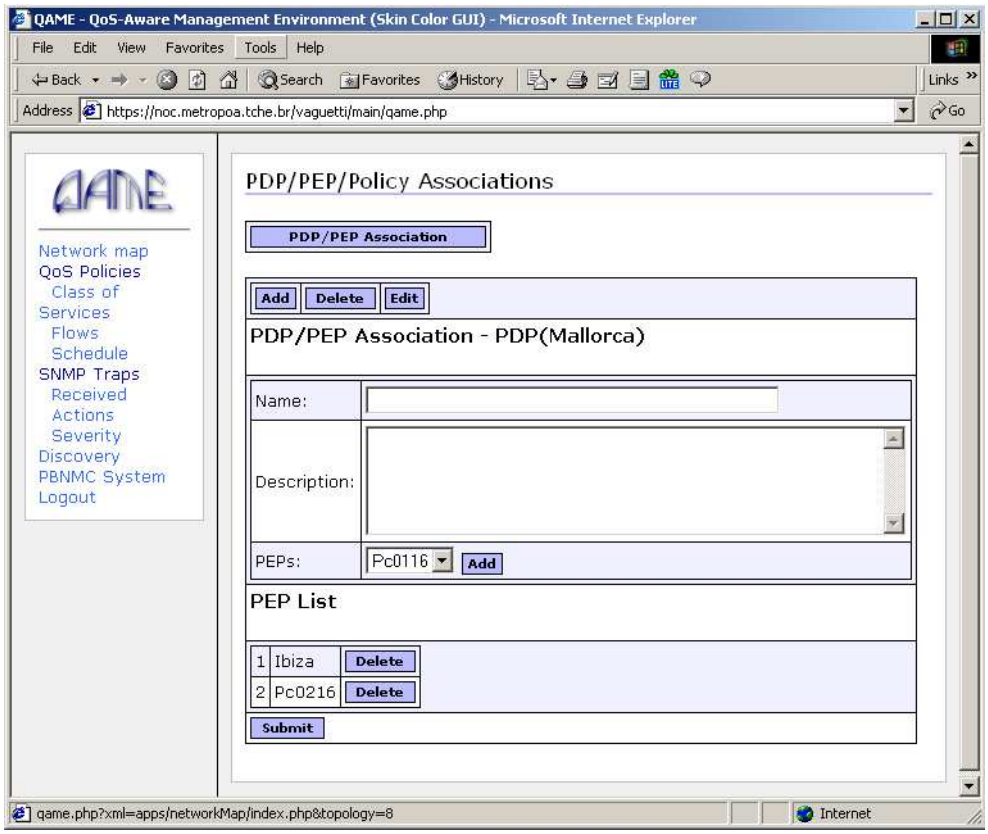

Figura 4.3: Módulo de Associação PDP/PEP

Os módulos apresentados anteriormente permitem que todas as associações necessárias ao sistema sejam criadas. Porém, para que a ferramenta possa operar corretamente, ou até mesmo as associações anteriores possam ser feitas, uma seqüência de passos deve ser executada previamente. O requisito mínimo para que o módulo de associações torne-se disponível é que o sistema possua ao menos uma política definida, um PEP e um PDP cadastrados. Por isso, os próximos módulos apresentados serão os módulos de cadastro de PDPs, PEPs, e Políticas. O cadastro de itens em tais módulos faz parte da seqüência de passos citada anteriormente.

A ferramenta possui um módulo de controle, onde é possível cadastrar todos os elementos necessários para que o módulo de associações possa ser usado. O módulo chamado PBNMC System possui várias opções de controle, dentre elas estão: Controle de PDPs, Controle de PEPs, Controle de Regras, Controle de Políticas e Controle de políticas através da ferramenta LDAPExplorer. A Figura 4.4 apresenta a opção de controle de PDPs. Através desta opção é possível cadastrar novos PDPs e associá-los a hosts na rede, onde os hosts da rede são previamente cadastrados, através de módulos já existentes na ferramenta QAME. A Figura 4.4 mostra o elemento `PC0116` sendo identificado como um PDP pelo sistema. Opções para deletar, editar e listar PDPs cadastrados também são

oferecidas neste módulo.

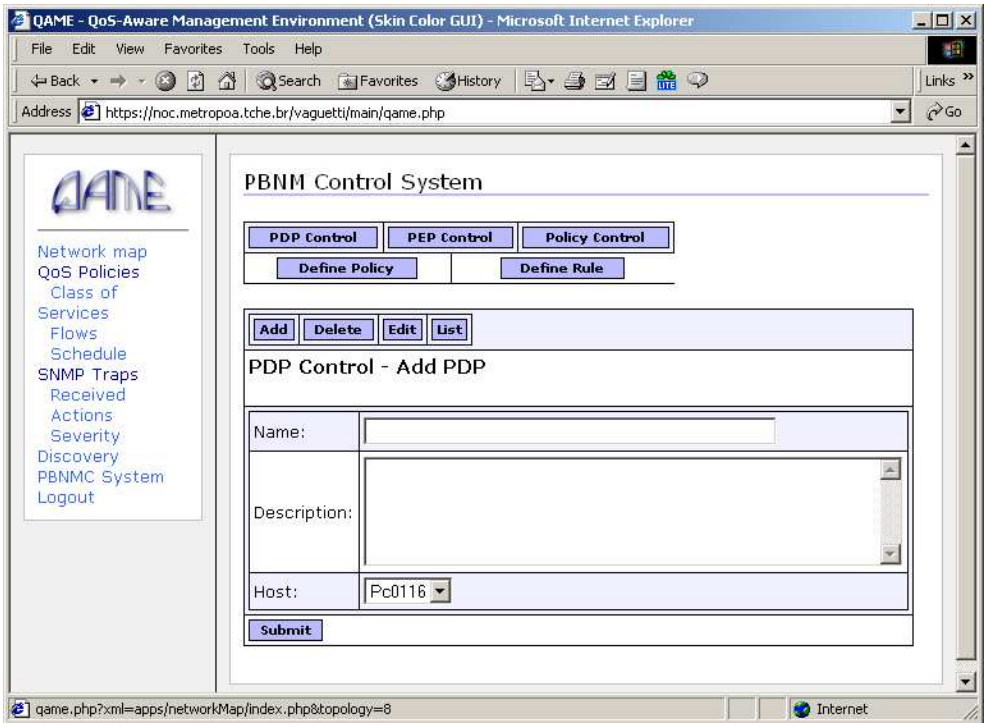

Figura 4.4: Módulo de controle de PDPs

Assim como existe um módulo para controle de PDPs, existe um módulo semelhante para controle de PEPs, e este é apresentado através da Figura 4.5. Neste módulo também é possível vincular host identificados na rede a PEPs.

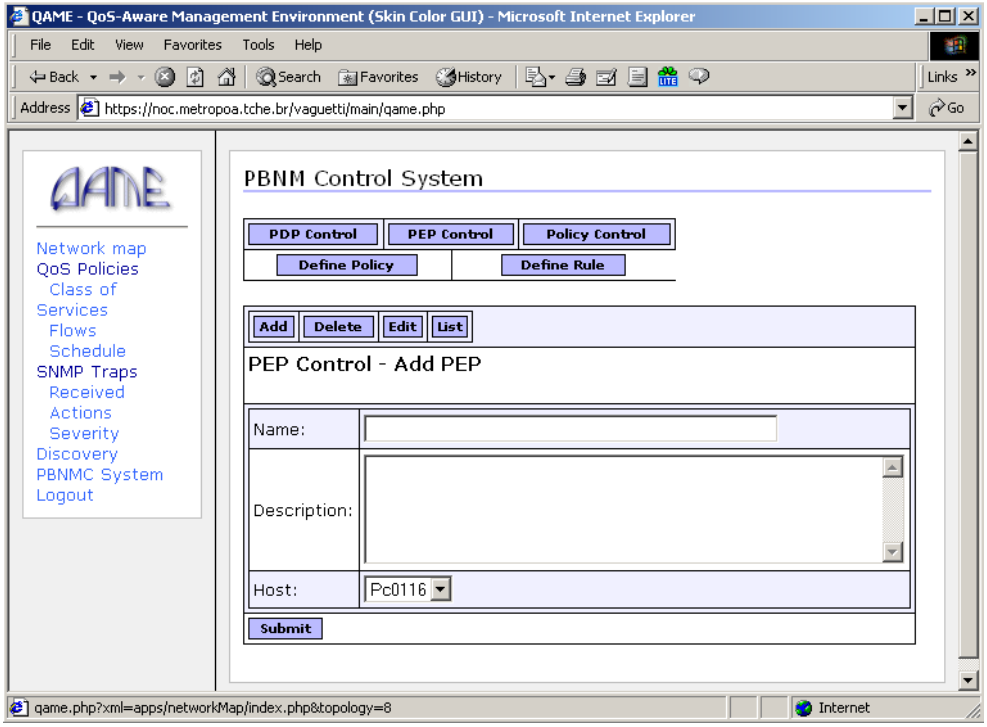

Figura 4.5: Módulo de controle de PEPs

O módulo de controle também permite que políticas sejam declaradas no sistema, mas para que as políticas sejam declaradas, é necessário que as regras desta política sejam criadas previamente.

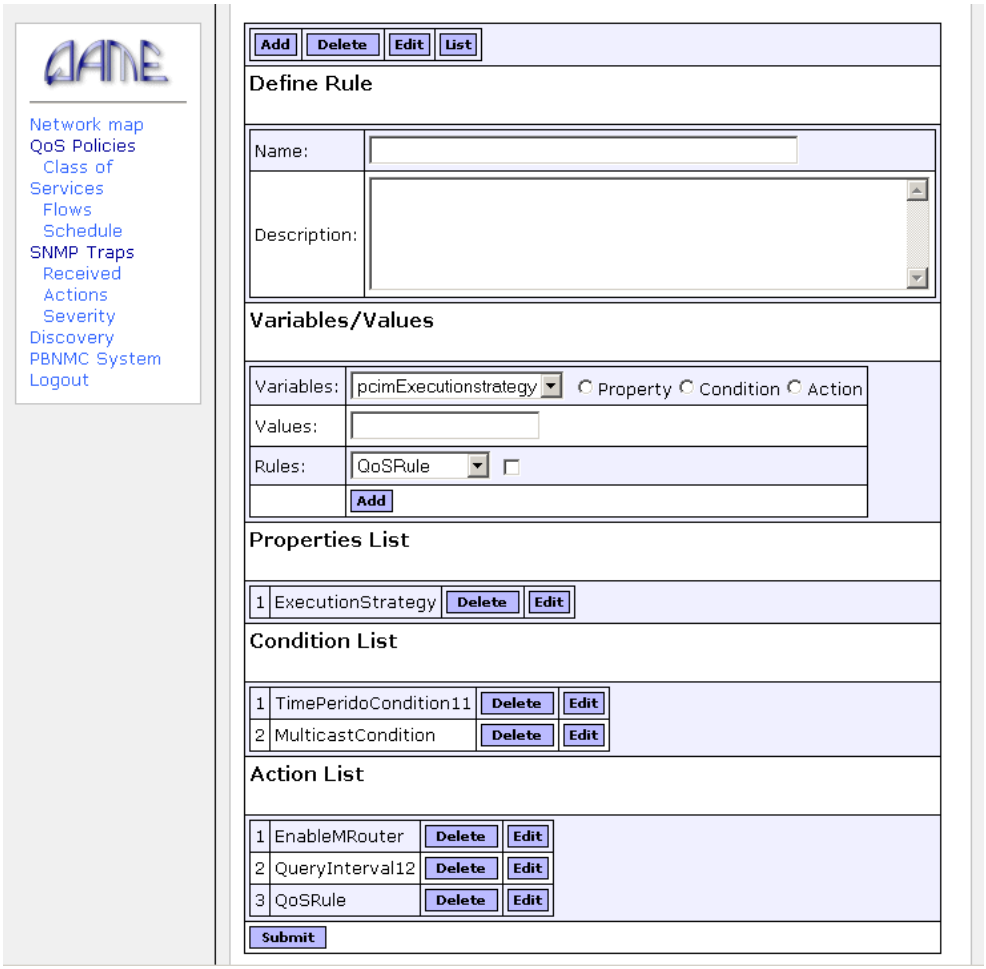

Figura 4.6: Módulo de controle de Regras

A Figura 4.6 demonstra o módulo que permite que regras sejam criadas. Tais regras podem possuir propriedades, condições e ações. A definição de como as regras podem ser declaradas está descrita no PCIMe. E para suportar tais definições, o protótipo ainda permite que regras sejam associadas a ações de uma regra de nível superior. Neste contexto, é permitida a criação de uma hierarquia de regras.

Após declaradas as regras uma política pode ser finalmente criada. É importante salientar que uma política só pode ser criada se uma regra existir, ou seja, todas as políticas devem possuir ao menos uma regra associada. A Figura 4.6 apresenta uma regra sendo definida com os seguintes itens: uma propriedade `ExecutionStrategy`; duas condições, uma que define as questões temporais da regra (`TimePeriodCondition11`) e outra que apresenta condições para fluxos multicast (`MulticastCondition`); e três ações, onde a primeira (`EnableMRouter`) habilita o roteamento multicast, a segunda (`QueryInterval12`) instancia a freqüência de consultas IGMP, e a terceira dispara a execução de uma nova regra, neste caso (`QoSRule`).

A Figura 4.7 apresenta o módulo que demonstra como as políticas podem ser declaradas. Tal módulo permite que diversas regras sejam associadas a uma política. No exemplo está sendo definida uma política que possuirá as seguintes regras associadas:

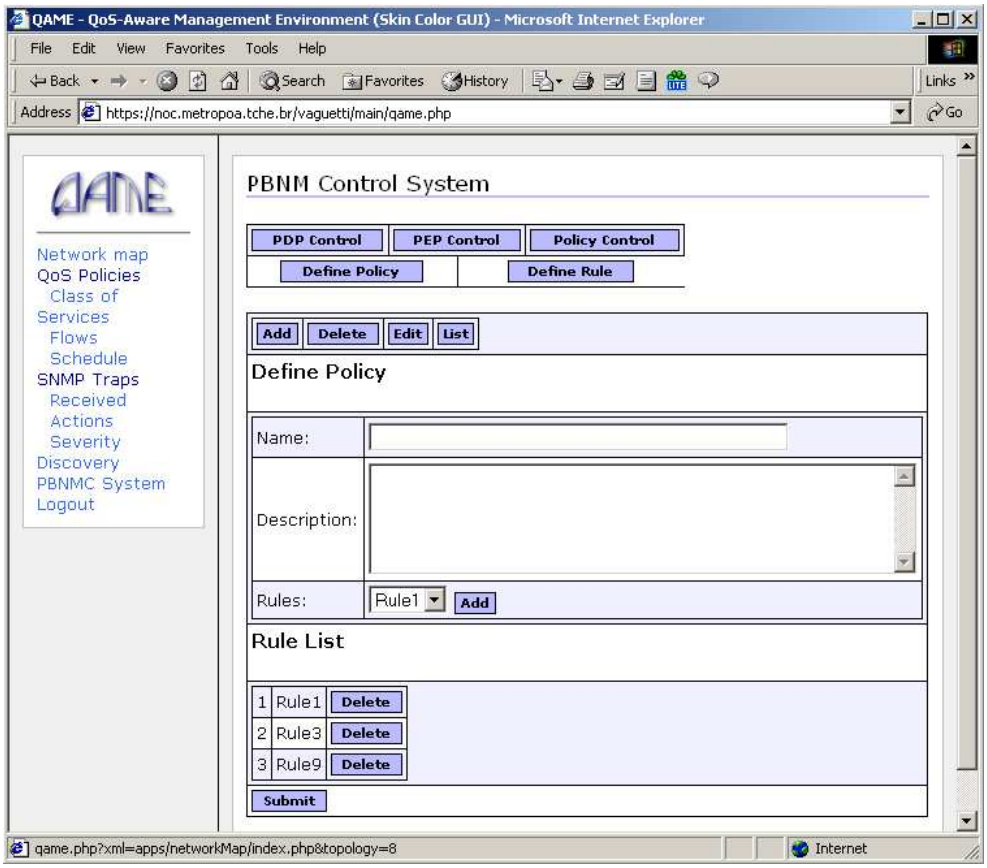

Figura 4.7: Módulo de controle de Políticas

`Rule1`, `Rule3` e `Rule9`. Todas as opções de controle de cadastro também são oferecidas nos módulos de controle de regras e políticas.

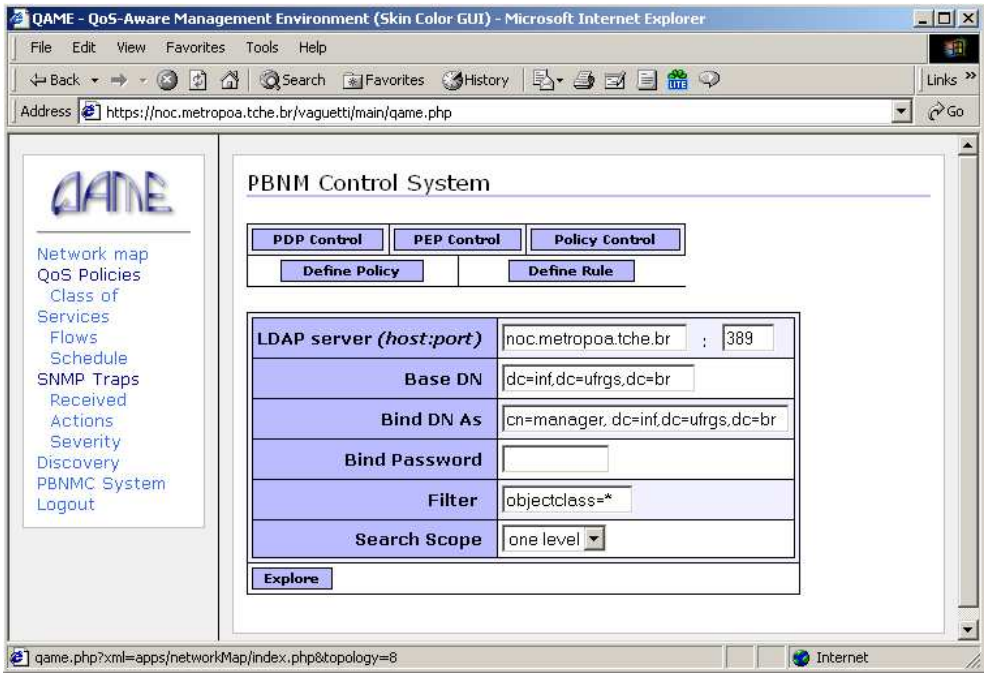

Figura 4.8: Módulo de controle de políticas através do LDAPExplorer

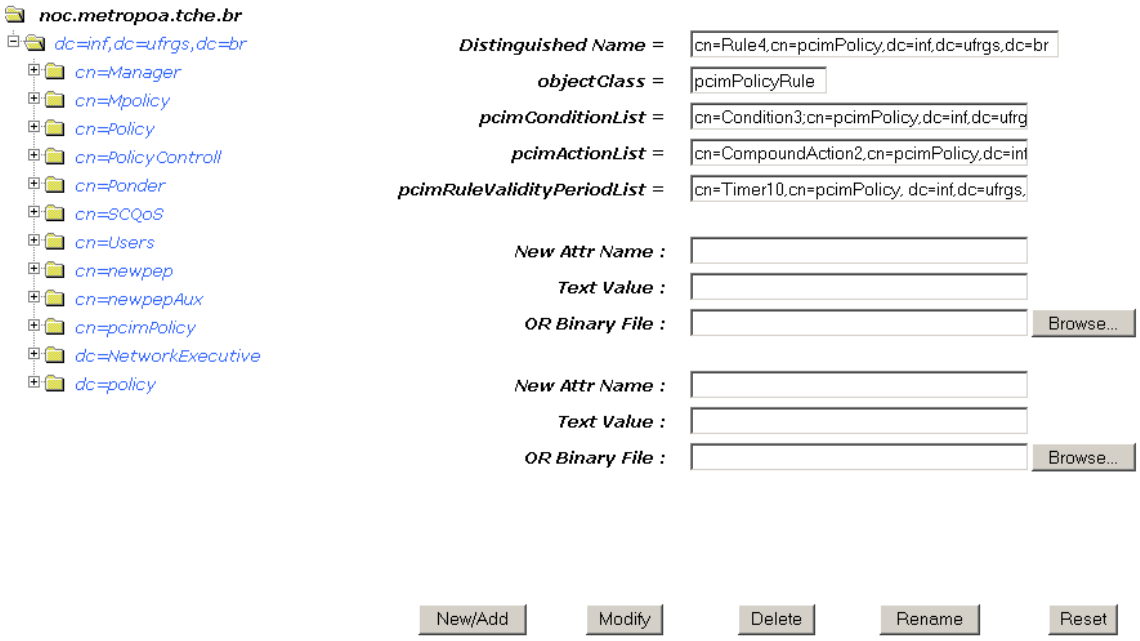

Figura 4.9: Ferramenta LDAPExplorer integrada ao protótipo

Ainda é possível controlar a criação de políticas de forma detalhada através da opção `Policy Control`, que permite que as políticas declaradas através do sistema possam ser visualizadas e, se necessário, modificadas diretamente no Repositório de Políticas (PR). Isto torna-se possível, através de uma ferramenta agregada ao sistema chamada "LDAPExplorer", que permite a visualização através de um *webBrowser* como exatamente as políticas foram armazenadas no serviço de diretórios LDAP (Figura 4.8 e 4.9). A partir do uso de tal ferramenta, já é possível perceber uma das tecnologias utilizadas na implementação do protótipo, neste caso o LDAP. Mais detalhes sobre as tecnologias utilizadas na implementação serão apresentados na próxima seção.

4.2 Modelo funcional e tecnologias

Nesta seção serão apresentados os elementos da arquitetura descrita no capítulo 3, com um nível maior de detalhe, apresentando a forma de comunicação entre os elementos, o modelo funcional e as tecnologias utilizadas na implementação do protótipo. Inicialmente os elementos que compõem a arquitetura foram divididos em elementos de baixo e alto nível, de acordo com suas funções. Basicamente, os elementos de alto nível (*High level elements*) são integrantes da parte de controle de aplicação das políticas, enquanto os elementos de baixo nível (*Low level elements*) são integrantes da parte de aplicação efetiva das políticas nos elementos da rede.

A Figura 4.10 apresenta os elementos de alto nível da arquitetura. O elemento denominado PMT, apresentado na seção anterior, é um software baseado na web, que permite que políticas e seus devidos controles sejam adicionados ao sistema. O PMT foi desenvolvido com tecnologias com PHP4, HTML e XML. As políticas declaradas através do PMT são armazenadas em um serviço de diretórios LDAP (*LDAP Directory Services*), de acordo com o modelo definido no PCIM e PCIMe.

Além das políticas, a parte de controle é definida através do PMT e também armazenada no serviço de diretórios LDAP. A base de controle armazena informações

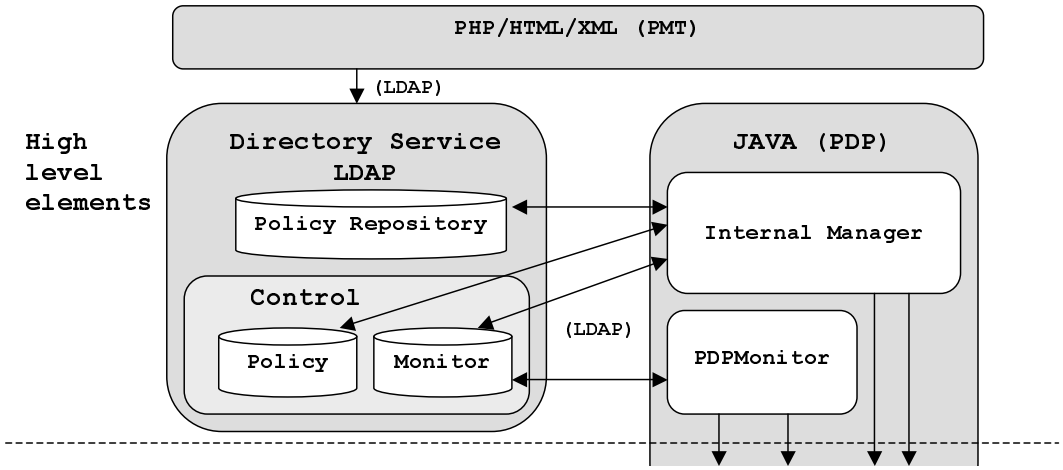

Figura 4.10: Elementos de alto nível da arquitetura PBNM

de relacionamento entre políticas que devem ser aplicadas nos PEPs e os PDPs que deverão efetivamente aplicar tais políticas, além de informação de *status* de aplicação das políticas.

Uma vez declarada, a política e seus relacionamentos de controle se tornam disponíveis ao PDP. Tal controle é feito definindo elementos no serviço de diretórios LDAP, que indicam que uma nova política deve ser aplicada por determinado PDP do sistema, sendo função dos PDPs verificarem constantemente a existência de novas políticas a serem baixadas e aplicadas. O elemento interno ao PDP, chamado `Internal Manager`, tem como uma de suas funções verificar a existência de novas políticas e baixá-las quando elas forem disponibilizadas. Tais políticas são buscadas no `Policy Repository`. A partir desse momento, o `Internal Manager` passa a controlar as questões temporais de aplicação das políticas e cada regra vinculada a política, se houverem. As políticas são armazenadas internamente ao `Internal Manager`, em forma de objetos hierárquicos, que são capazes de representar toda a hierarquia de objetos proposta em CIM, PCIM e PCIMe, assim como partes de alguns modelos específicos (e.g QPIM). O elemento `Internal Manager` é responsável por outras funções internas ao PDP, porém tais funções serão comentadas posteriormente nesta seção.

Para que o controle de execução seja efetuado, é necessário que existam elementos na rede que verifiquem as condições impostas através das regras na política. O papel de verificação e validação é executado pelo elemento denominado `PDPMonitor`. O `PDPMonitor` é um elemento adicionado à arquitetura tradicional de PBNM, que se encarrega de verificar o estado das condições impostas nas regras. Este elemento é fundamental tanto para o controle de execução das regras, quanto para o controle hierárquico existente entre as regras de uma mesma política, pois uma vez que uma regra de nível superior falhe (este evento deve ser percebido através do controle de execução), todas as regras imediatamente inferiores (ativas ou não) se tornam inválidas, juntamente com os monitores condicionais (`PDPMonitor`) de tais regras.

O `PDPMonitor` pode ser implementado junto ao PDP, como também pode ser implementado como um elemento separado, mas controlado e associado ao PDP. Na implementação do protótipo apresentada, o `PDPMonitor` foi implementado como um elemento separado, que é controlado e fornece informações ao PDP, através do serviço de diretórios LDAP. Neste contexto, o `Internal Manager` define quais são as variáveis e valores que devem ser monitorados através dos `PDPMonitor` distribuídos na rede. Neste

caso, o PDPMonitor deve verificar a validade da variável e valores associados. Uma vez obtido através da verificação, o valor é armazenado no serviço de diretórios LDAP (o valor obtido é uma representação de verdadeiro ou falso). Portanto, a comunicação entre o PDPMonitor e o Internal Manager é feita através do protocolo LDAP.

Mantendo-se esta forma de comunicação, é possível obter-se um controle mais detalhado sobre o *status* de aplicação das políticas nos PEPs, possibilitando o gerenciamento do status das políticas ao nível de suas regras individuais. Por exemplo, é possível identificar qual regra de uma política falhou e em qual ponto da rede está acontecendo tal falha (através da identificação e localização do PEP que gerou tal falha). Informações tão detalhadas são importantes para identificar problemas na aplicação de políticas críticas, que devem ser aplicadas, possibilitando, assim, uma tentativa de solucionar os problemas identificados. O armazenamento de tais informações foi implementado no protótipo, e são utilizadas pelo Internal Manager, para controle de execução e controle hierárquico das regras. Entretanto, não foi implementado nenhum tipo de gerenciamento a nível de PMT, com o intuito de corrigir possíveis problemas identificados na aplicação das regras.

Todos os elementos comentados anteriormente fazem parte dos chamados elementos de alto nível da arquitetura. Neste nível, os elementos não têm nenhum tipo de preocupação com o tipo de equipamentos ou tecnologias usadas para a efetiva aplicação das políticas nos PEPs.

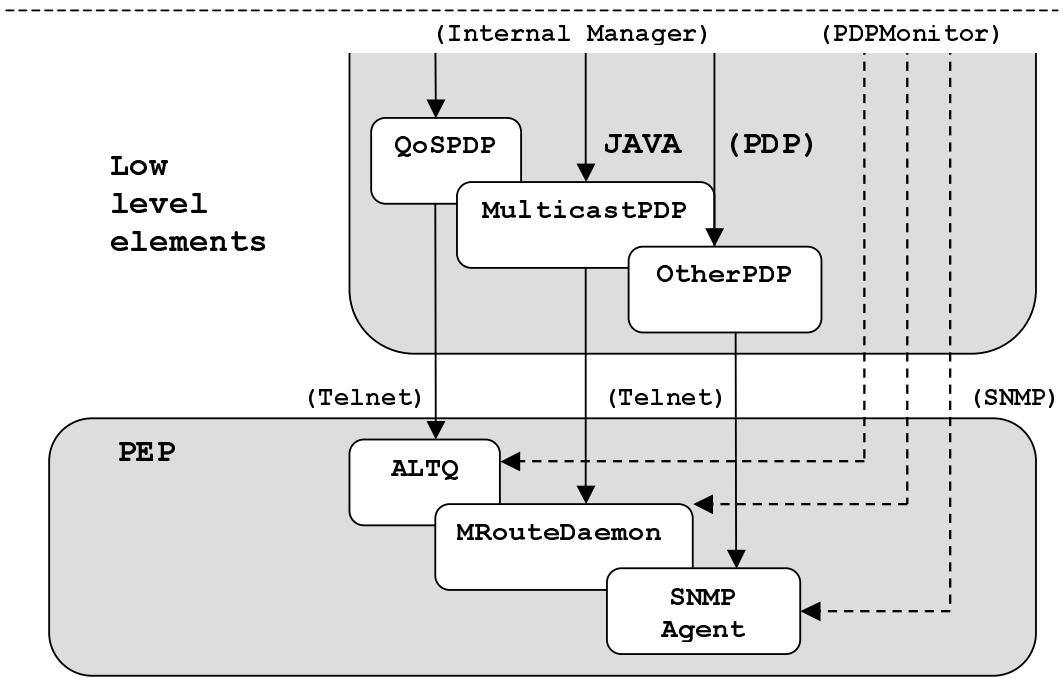

Figura 4.11: Elementos de baixo nível da arquitetura PBNM

Fazem parte dos chamados elementos de baixo nível da arquitetura os elementos direcionados ao controle de tecnologias específicas, porém, de igual importância na aplicação das políticas (Figura 4.11). Trata-se dos módulos do sistema, chamados de PDPs específicos. Tais PDPs específicos têm a função de traduzir as políticas de alto nível, repassadas a eles através do Internal Manager, para que possam ser interpretadas, de acordo com cada dispositivo ou tecnologia de rede a cuja política está vinculada. O vínculo entre as políticas e as tecnologias é identificado através das variáveis que

descrevem as ações e condições (e.g. a variável `qpBandwith` está vinculada à QoS).

O protótipo implementa dois tipos de PDPs específicos, um PDP direcionado à tradução de políticas para ações de configuração de QoS (`QoSPDP`), e outro direcionado à tradução de políticas para ações relativas ao controle de fluxos multicast (`MulticastPDP`). Outros PDPs específicos podem ser adicionados ao sistema, pois o protótipo fornece algumas interfaces java que possibilitam tal implementação (mais detalhes sobre as interfaces oferecidas pelo protótipo serão apresentadas na próxima seção). A decisão sobre qual PDP específico vai traduzir e aplicar cada regra é tomada pelo `Internal Manager`. Uma única política, que possua várias regras, pode ter suas regras individualmente traduzidas por PDPs específicos diferentes, pois cada um deles só consegue atender às necessidades de tradução de uma tecnologia.

Além dos PDPs específicos, também são apresentados como elementos de baixo nível na arquitetura os PEPs. Os PEPs são os elementos capazes de configurar as políticas traduzidas pelos PDPs nos dispositivos. A Figura 4.11 apresenta três PEPs utilizados na implementação do protótipo. O PEP denominado `ALTQ` é capaz de configurar políticas de QoS utilizando o software de controle de QoS, Altq. A comunicação entre o `QoSPDP` e o PDP `ALTQ` é feita através do protocolo telnet. Outro PEP apresentado é o `MRouteDaemon`, que é capaz de aplicar políticas traduzidas pelo MulticastPDP em roteadores multicast, que se utilizam do `Mrouted` como software de controle de fluxo multicast. A comunicação entre o `MulticastPDP` e o `MRouteDaemon` também é feita através de telnet. Além dos PEPs já apresentados anteriormente, um outro tipo de PEP pode ser utilizado, trata-se do PEP capaz de aplicar as políticas, através do protocolo SNMP. Neste caso, o PDP tradutor específico deve ser implementado de acordo com a necessidade de configuração.

4.3 Detalhes de implementação

Nesta seção serão apresentados alguns detalhes da implementação do protótipo como: serviço de diretórios utilizado, bem como os schemas utilizados ou criados para armazenar as políticas e informações de controle do sistema; detalhes sobre a implementação do PDP, como os pacotes java, desenvolvidos e utilizados na implementação, com o intuito de organizar o desenvolvimento, além de interfaces java, criadas para possibilitar que novas tecnologias pudessem ser utilizadas no futuro, utilizando a mesma infra-estrutura básica; exemplos de implementação possíveis aos PDPMonitors, e quais foram desenvolvidos no protótipo; pacotes java adicionais, utilizados no desenvolvimento.

O serviço de diretório utilizado é baseado em software livre, e trata-se do OpenLDAP, sendo que a versão utilizada é `v.1.4.8.6`. Além dos schemas, fornecidos pelo próprio serviço de diretórios, alguns schemas adicionais tiveram de ser incluídos ou desenvolvidos juntamente com a implementação do protótipo. Os principais schemas LDAP utilizados no protótipo serão listados a seguir e apresentados na íntegra, no Anexo A:

- *Policy Core LDAP Schema* <draft-ietf-policy-core-schema-16.txt>, que mapeia para um schema LDAP a RFC 3060 (*Policy Core Information Model*), anexo A.1;

- *Policy Core Extensions LDAP Schema* <draft-reyes-policy-core-ext-schema-03.txt>, que mapeia para um schema LDAP a RFC 3460 (*Policy Core Extensions Information Model*), anexo A.2;

- Mapeamento para LDAP de parte do *Policy QoS Information Model)*, anexo A.3;
- Criação de schema adicionando ao PCIMe variáveis relacionadas à multicast, anexo A.3;
- Criação de schema para armazenar informações de controle de políticas, PEPs, PDPs e PDPMonitors, anexo A.4.

O mapeamento dos modelos de informação para LDAP seguiram as sugestões fornecidas pelo DMTF (*Distribiuted Management Task Force*).

Para que as políticas armazenadas no serviço de diretórios LDAP pudessem ser armazenadas internamente ao PDP, optou-se por desenvolver um pacote java que fosse capaz de mapear todos detalhes contidos na política armazenada (inclusive seus aspectos hierárquicos) para objetos que devem ser instanciados em tempo de execução, à medida que novas políticas são definidas. Neste contexto, foi desenvolvido um pacote java que implementa todas as classes definidas nos modelos de informação CIM, PCIM, PCIMe, além de classes capazes de suportar os objetos relacionados à QoS e multicast, adicionados ao sistema.

Um exemplo de como as informações buscadas no serviço de diretórios LDAP são armazenadas em forma de objetos, é descrito abaixo. Se uma política armazenada, no LDAP, é composta por uma regra e, nesta regra é definida uma condição associada a uma ação, além das propriedades de controle da regra como `ExecutionStrategy` e `RuleEnable`, basta instanciar um novo objeto a partir da classe pcimRule, que foi adicionada ao pacote desenvolvido no protótipo, e utilizar os métodos listados abaixo para carregar as informações internamente ao objeto.

- `setVarpcimeRuleExecutionStrategy()` - armazena o valor associado à propriedade `ExecutionStrategy`;
- `setVarpcimeRuleRuleEnable()` - armazena o valor associado à propriedade `RuleEnable`;
- `setVarpcimeRuleConditionListType()` - armazena o conjunto de condições;
- `setVarpcimeRuleSequencedActions()` - armazena a seqüência de ações.

Uma vez carregadas as informações internamente ao objeto, e aplicada agora, utilizando o objeto como fonte de informação das políticas, todas as classes implementadas fornecem métodos para carregar informações nos objetos instanciados, assim como métodos para extrair tais informações e poder manipulá-las. No exemplo apresentado, os métodos necessários para extrair ou manipular as informações contidas no objeto instanciado são listadas a seguir:

- `getVarpcimeRuleExecutionStrategy()` - extrai o valor associado à propriedade `ExecutionStrategy`;
- `getVarpcimeRuleRuleEnable()` - extrai o valor associado à propriedade `RuleEnable`;
- `getVarpcimeRuleConditionListType()` - extrai o conjunto de condições;

- `getVarpcimeRuleSequencedActions()` - extrai a seqüência de ações.

Algumas classes que compõem o pacote, inclusive a classe pcimRule, comentada anteriormente, são apresentadas na Figuras 4.12. Nesta figura é possível observar a hierarquia de classes e identificar classes definidas nos modelos CIM do DMTF (e.g. class cimManagedElement), e também classes definidas nos modelos PCIM (e.g. class pcimPolicy) e PCIMe (e.g. class pcimeSimplePAction) do IETF. O conjunto de classes destinadas a armazenar as informações relacionadas às políticas e implementadas neste protótipo, é apresentado na íntegra, no anexo B.

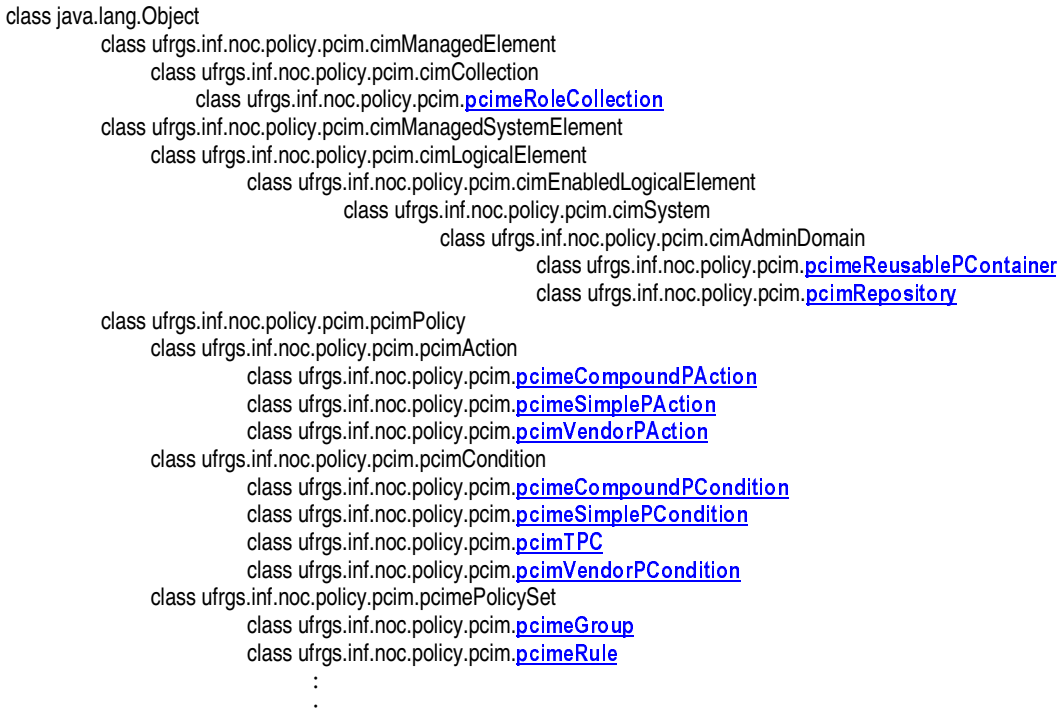

Figura 4.12: Classes implementadas para carregar as políticas internamente ao PDP

Além do pacote desenvolvido para armazenamento das políticas internamente ao PDP denominado `Policy Container`, outros pacotes são fornecidos. Tais pacotes possuem interfaces e implementações de tais interfaces, que desta forma possibilitam que as implementações feitas no protótipo possam ser substituídas por novas implementações das interfaces, sem que o modo de funcionamento do PDP seja modificado. O pacote denominado `PDPDaemon` fornece classes que são implementações não-substituíveis, como as classes apresentadas na Figura 4.13.

Dentre as classes apresentadas na figura, a principal classe é a `pdpDaemon`, que controla o funcionamento de todo o PDP e utiliza a classe `pdpIManager`, que implementa o elemento `Internal Manager`, citado na arquitetura PBNM.

O pacote PDPDaemon ainda fornece duas interfaces e implementações para tais interfaces. A interface `pdpPTManager` é responsável pela comunicação e busca das políticas no repositório de políticas. Como o repositório de políticas adotado no protótipo é o serviço de diretórios LDAP, foi implementada uma classe que implementa a interface `pdpPTManager` chamada `pdpPTMLdap`, que é responsável por buscar as políticas no serviço de diretórios LDAP e armazená-las em forma de objetos, utilizando o pacote `Policy Container`. A outra interface fornecida pelo pacote PDPDaemon é a interface `pdpCManager`, responsável por armazenar as informações de controle relacionadas às

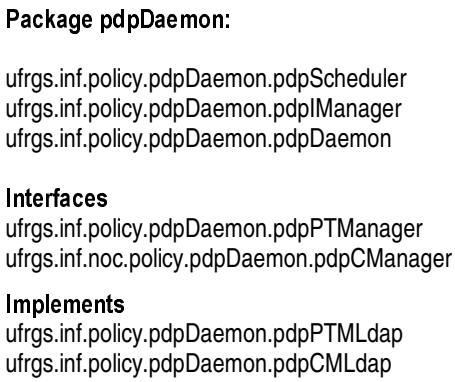

Figura 4.13: Pacote PDPDaemon

políticas (e.g. em quais PEPs será aplicada determinada política). Esta interface foi implementada no protótipo pela classe `pdpCMLdap`, pois as informações de controle de políticas também são armazenadas no serviço de diretórios LDAP no protótipo.

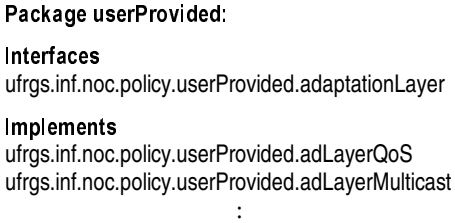

Figura 4.14: Pacote userProvided

Além dos pacotes `Policy Container` e `PDPdaemon`, o pacote `userProvided` também foi desenvolvido e utilizado no protótipo. Este pacote, que é apresentado na Figura 4.14, fornece apenas uma interface (`adaptationLayer`), que é responsável por fornecer a estrutura básica para desenvolvimentos dos PDPs específicos, citados na seção anterior. Juntamente com a interface, o pacote possui duas implementações para tal interface, trata-se das classes `adLayerQoS` (que traduz as políticas de QoS abstratas para políticas de QoS de baixo nível) e `adLayerMulticast` (que traduz as políticas de multicast abstratas para políticas de multicast de mais baixo nível).

Para finalizar os detalhes da implementação, é necessário comentar como o elemento `PDPMonitor` foi implementado. Primeiramente, é necessário destacar que o `PDPMonitor` não foi implementado junto ao software do PDP, e sim, como sendo um elemento individual na arquitetura, porém controlado pelo PDP. Como já havia sido comentado anteriormente, o `PDPMonitor` comunica-se com o PDP através do LDAP, portanto, é necessário que todos os `PDPMonitores` implementados possuam suporte a LDAP, mesmo que a forma de monitoração utilizada seja diferente para cada novo `PDPMonitor`.

Nesta implementação dois `PDPMonitores` foram implementados (Figura 4.15), um deles capaz de capturar pacotes na subrede onde se encontra, além de identificar se o host de destino faz parte de algum grupo multicast citado em alguma política (identificando assim o fluxo multicast). Um pacote java externo foi utilizado para fazer a captura dos pacotes no `PDPMonitor`, trata-se do pacote `JPCAP`. O outro `PDPMonitor` é um pouco mais genérico e possibilita a monitoração através de SNMP. Neste caso, são passados através do LDAP os OID´s do objetos gerenciáveis que se pretende monitorar ou verificar

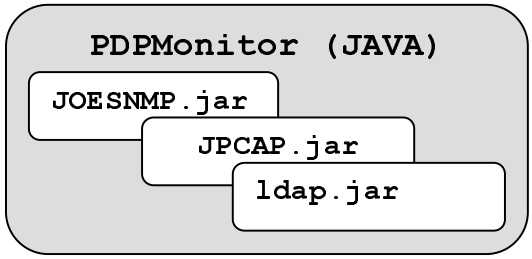

Figura 4.15: Bibliotecas utilizadas na implementação do PDPMonitor

o valor associado. Através deste `PDPMonitor` é possível consultar a IGMP-MIB e verificar quais são os fluxos multicast que estão passando por determinado roteador na rede. O `PDPMonitor` com suporte a SNMP também utilizou um pacote externo para fazer a monitoração, trata-se do pacote `JOESNMP`.

5 AMBIENTE DE TESTES E RESULTADOS OBTIDOS

Este capítulo apresenta os resultados obtidos através da utilização do protótipo implementado. Inicialmente será apresentado o ambiente no qual é possível aplicar políticas de teste e validar o protótipo. Num segundo momento, os resultados observados durante as aplicações das políticas no ambiente de testes são apresentados e analisados.

5.1 Ambiente e testes

O ambiente de testes utilizado é apresentado na Figura 5.1. A figura apresenta uma rede composta de três roteadores implementados usando o sistema operacional FreeBSD 4.8. Cada um destes roteadores possui três interfaces de rede, sendo que cada roteador comunica-se diretamente com os outros roteadores e com um host comum na subrede. Os roteadores ainda possuem suporte à QoS e multicast, através dos softwares AltQd (permite configuração de QoS em roteadores FreeBSD) e MRouted (permite roteamento de fluxos multicast), respectivamente. Não está descrito na figura, mas o roteamento entre os roteadores que compõem a rede de teste é dinâmico, utilizando-se o protocolo RIP, implementado através do *daemon* Routed no FreeBSD.

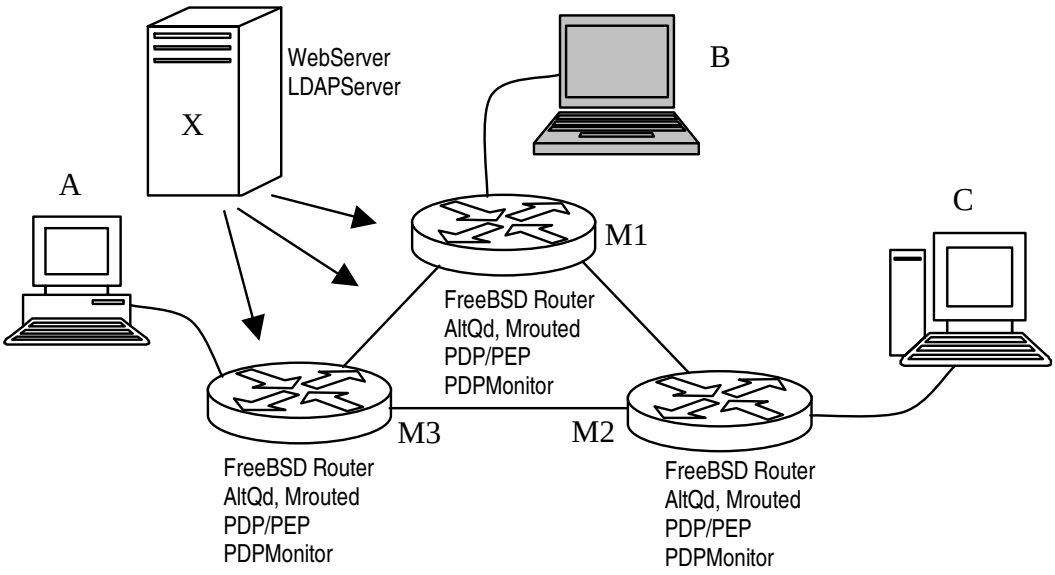

Figura 5.1: Ambiente de rede utilizado para testar o protótipo

Detalhes sobre capacidade de processamento, memória e configuração das interfaces de rede, de cada um dos roteadores são apresentados nas tabelas: Tabela 5.1 (M1-MRouter1), Tabela 5.2 (M2-MRouter2), Tabela 5.3 (M3-MRouter3).

Tabela 5.1: Configuração do MRourter1

Componente	Descrição
Processador	Pentium 300 Mhz - Celeron
Memória RAM	128MB
Interface1	xl0 - 143.54.47.56
Interface2	ed0 - 192.168.1.1
Interface3	rl0 - 192.168.2.254

Tabela 5.2: Configuração do MRourter2

Componente	Descrição
Processador	Pentium 300 Mhz - Celeron
Memória RAM	128MB
Interface1	xl0 - 192.168.16.1
Interface2	ed0 - 192.168.2.1
Interface3	ed1 - 192.168.3.254

Tabela 5.3: Configuração do MRourter3

Componente	Descrição
Processador	Pentium 300 Mhz - Celeron
Memória RAM	48MB
Interface1	fxp0 - 192.168.16.130
Interface2	ed0 - 192.168.3.1
Interface3	ed1 - 192.168.1.254

Tomaremos como exemplo a política integrada de QoS e multicast apresentada, na seção 3.3 (Figura 3.7), com algumas modificações relacionadas à aspectos temporais (Figura 5.2), para listar os passos executados no protótipo para aplicar a política. Os passos listados a seguir supõem que a política já está armazenada internamente ao PDP.

- No momento em que a política é aplicada, o mrouted é ativado em todos os roteadores da rede, possibilitando fluxo multicast nativo entre os roteadores do ambiente de testes;

- No mesmo instante, são ativados os PDPMonitores que estão localizados nos roteadores para identificar o fluxo multicast especificado na política (tal PDPMonitor foi implementado utilizando o pacote JPCAP);

- No momento em que o PDPMonitor percebe o fluxo, o PDP é notificado através do LDAP e a política de QoS (através do AltQ) é configurada no roteador ou roteadores, que perceberem o fluxo (o PDPMonitor só aciona o PDP se o fluxo for percebido em mais de uma interface do roteador);

- Quando o fluxo multicast deixar de ser percebido, o PDP é novamente notificado e então a regra é retirada do AltQ, liberando os recursos da rede.

```
Policy: Policy for QoS and multicast
Rule1: TimeDefinition
        if ( time >= 10) and ( time <= 100)
         then
                Rule2: EnableMulticastSupport
                 if (pmcastMroutedEnabled == false) then
                    pmcastMroutedEnabled = true
                 pcimExecutionStrategy = 2

                        Rule3: QoSMulticastGroupPrune
                        if (pmcastGroupIPv4 == 224.2.55.22)
                        then
                            qpBandwidthUnits = 0
                           qpMaxBandwidth = 5000 Kbps
                         pcimDecisionStrategy = 2
                       pcimExecutionStrategy = 2

                        Rule4:QoSforMulticastGroupLeave
                         if (pmcastGroupIPv4 ! = 224.2.55.22)
                          then
                            clear configuration in Rule3
                         pcimExecutionStrategy = 2
                  pcimExecutionStrategy = 2
                  pcimDecisionStrategy = 2
          pcimExecutionStrategy = 2
          pcimDecisionStrategy = 2
```

Figura 5.2: Política aplicada no ambiente de testes

5.2 Análise dos resultados

```
Class 0 on Interface rl0: tcp_class
      priority: 1 depth: 0 offtime: 24125 [us] wrr_allot: 534 bytes
      nsPerByte: 8000 (1.00Mbps),     Measured: 985.87K [bps]
      pkts: 15046,    bytes: 21286747
      overs: 4992,    overactions: 4991
      borrows: 0,     delays: 4991
      drops: 0,       drop_bytes: 0
      QCount: 4,      (qmax: 30)
      AvgIdle: -60 [us],      (maxidle: 1593 minidle: -12000 [us])
Class 1 on Interface rl0: video_class
      priority: 1 depth: 0 offtime: 53886 [us] wrr_allot: 240 bytes
      nsPerByte: 17777        (450.02Kbps),   Measured: 128.31K [bps]
      pkts: 3947,     bytes: 2974595
      overs: 154,     overactions: 149
      borrows: 0,     delays: 149
      drops: 0,       drop_bytes: 0
      QCount: 0,      (qmax: 30)
      AvgIdle: 433 [us],      (maxidle: 3541 minidle: -26665 [us])
```

Figura 5.3: Observação de fluxos através do altqstat

Antes de iniciar a análise dos resultados, é necessário demonstrar o desperdício de recurso que pode ser observado em algumas configurações de QoS. No ambiente de testes foi utilizado o software AltQ para fornecer a qualidade de serviço do ambiente. Tornou-se possível então, efetuar alguns testes para observar o possível desperdício de banda quando se deseja reservar recursos a um determinado fluxo. Um simples teste foi efetuado, duas classes foram criadas para priorizar dois fluxos. A primeira classe (*class 0*), reserva 1Mbps de banda para fluxos TCP, enquanto a classe (*class 1*), reserva 450kbps de banda para fluxos multicast com grupo destino igual 224.2.55.22 (Figura 5.3). As classes foram criadas utilizando CBQ. O software para monitorar a utilização dos fluxos é o altqstat.

Pode-se observar então, através da figura, fluxos TCP passando pela classe 0, utilizando todo o recurso disponível para tal classe (1000-985.87kbps). Também é possível observar recursos de banda livre na classe 1 (450-128.31kbps). Relacionando com o problema enfrentado neste trabalho, pode-se concluir que, se os recursos relativos à QoS forem alocados de forma estática para fluxos multicast, em muitos momentos o desperdício de recurso será expressivo. No entanto, se os recursos forem alocados dinamicamente, o percentual de recurso de rede desperdiçado pode diminuir consideravelmente, desde que sua alocação seja definida corretamente.

A seguir, serão apresentados gráficos obtidos através da monitoração de fluxos de dados e utilização de memória, em cada roteador do ambiente de testes. As informações foram capturadas durante a aplicação da política descrita na Figura 5.2. Basicamente, a política apresentada, na figura, determina que os MRouters (roteadores para fluxo multicast), sejam ativados no tempo 10 de execução. Segundo a política, a reserva de recurso deve ser feita sob-demanda entre os tempos 10 e 100. Através dos gráficos podemos observar o momento em que os fluxos multicast passam pelas interfaces dos roteadores, e as mudanças ocorridas no uso da memória.

Para entender os resultados obtidos no gráficos é necessário descrever como ocorreram os fluxos durante os testes. Inicialmente um *host* da rede 143.54.47.0\25 iniciou a transmissão de um fluxo multicast através do grupo 224.2.55.22. Num segundo momento um *host* da rede 192.168.16.0\25 passou a fazer parte do mesmo grupo (o fluxo de dados multicast neste momento, trafega pelos Mrouters M1 e M2) e finalmente um *host* da rede 192.168.16.128\25 também torna-se integrante do grupo 224.2.55.22 (fluxo multicast passando pelos Mrouters M1 e M3). Os tempos de entrada e saída de cada um dos *hosts* podem ser percebidos através dos gráficos apresentados.

Observando os gráficos construídos a partir de monitoração ao MRouter1, podemos verificar alguns detalhes. Na Figura 5.4, pode-se observar picos na utilização de memória ocorridos no tempo 10 (habilitação do mrouted), tempo 40 (habilitação do altqd) e no tempo 90 (altqd desabilitado). É possível perceber que o altqd foi habilitado no tempo 40, pois um fluxo multicast de saída, passa a ser percebido no interface rl0 (Figura 5.6), significando que existe um novo membro no grupo 224.2.55.22 a partir da rede 192.168.16.0\25. Segundo o gráfico da figura 5.6, o fluxo multicast foi percebido entre os tempos 40 e 60. É apresentado também, um gráfico demonstrando o fluxo de entrada na interface xl0 do MRouter1, demonstrando a duração da transmissão multicast (Figura 5.5).

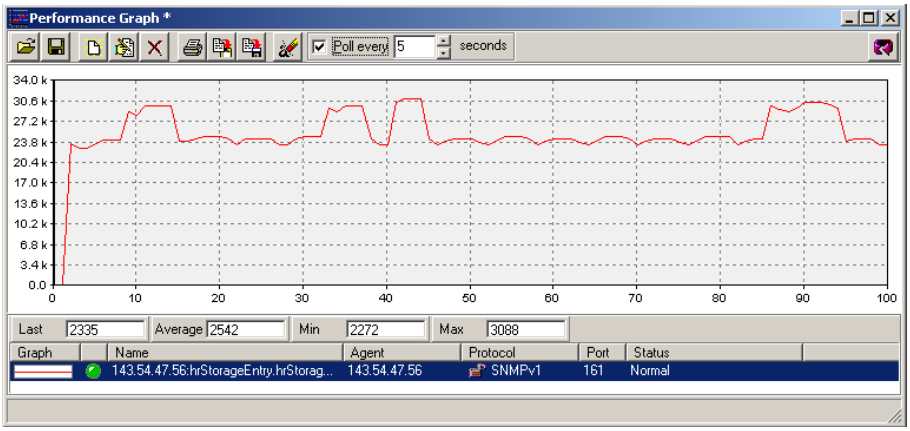

Figura 5.4: Memória RAM utilizado no Mrouter1 durante a aplicação da política

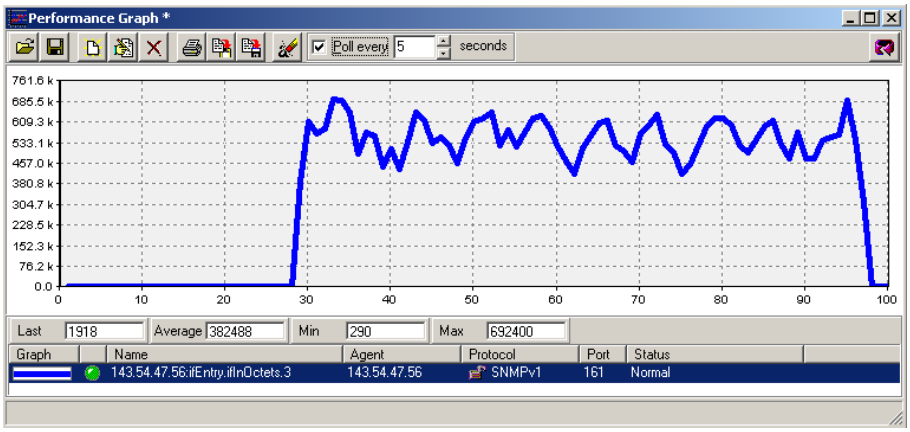

Figura 5.5: Octetos entrando no Mrouter1, interface xl0

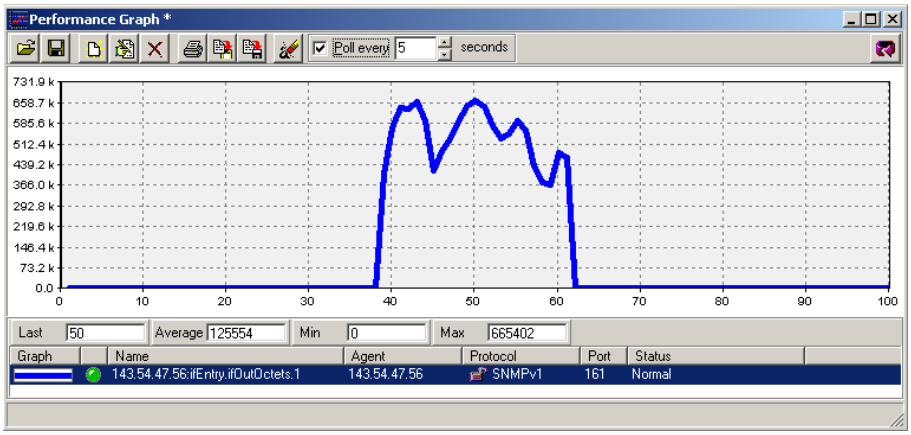

Figura 5.6: Octetos saindo do Mrouter1, interface rl0

Nas Figuras 5.7 e 5.8, pode-se observar picos na utilização de memória no tempo 10 (habilitação do mrouted), tempo 40 (habilitação do altqd) e tempo 60 (altqd desabilitado). O fluxo de saída observado na interface xl0 (Mrouter2) é muito semelhante ao tráfego observado na interface de saído da interface rl0 (MRouter1), pois tais interfaces passaram a perceber o fluxo nos mesmos momentos iniciais e finais.

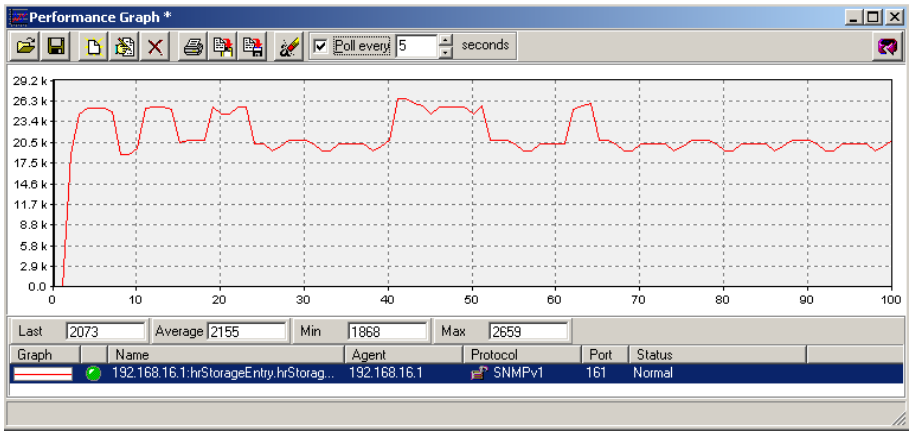

Figura 5.7: Memória RAM utilizado no Mrouter2 durante a aplicação da política

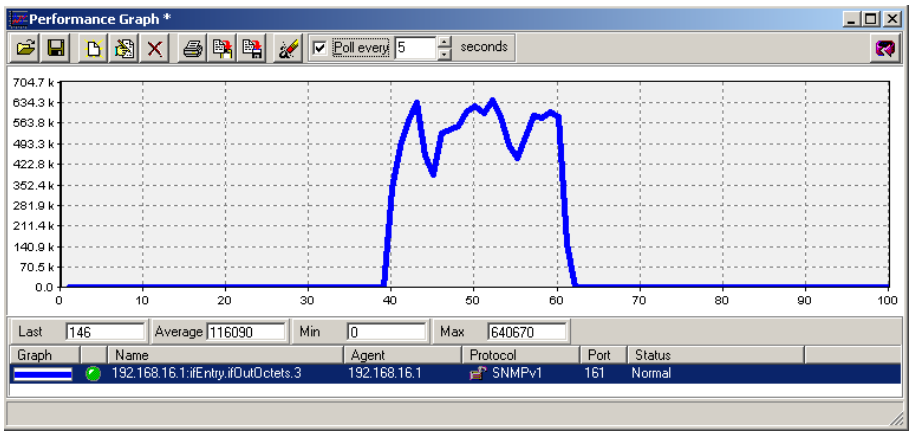

Figura 5.8: Octetos saindo do Mrouter2, interface xl0

Nas Figuras 5.9 e 5.10, pode-se observar picos na utilização de memória no tempo 10 (habilitação do mrouted), tempo 50 (habilitação do altqd) e tempo 90 (altqd desabilitado). O fluxo de saída observado na interface fxp0 (Mrouter3), indica que um novo membro, iniciado a partir da rede 192.168.16.128\25, passou a fazer parte do grupo 224.2.55.22. Comparando todos gráficos é possível perceber que o altqd foi habilitado no MRouter1 quando o fluxo gerado a partir da rede 192.168.16.0\25 foi iniciado, e desabilitado somente quando o fluxo gerado pela rede 192.168.16.128\25 deixou de ser percebido.

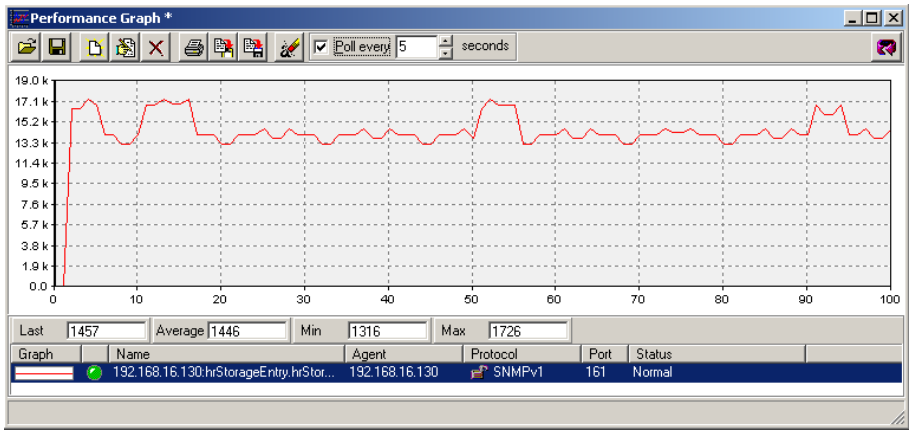

Figura 5.9: Memória RAM utilizado no Mrouter3 durante a aplicação da política

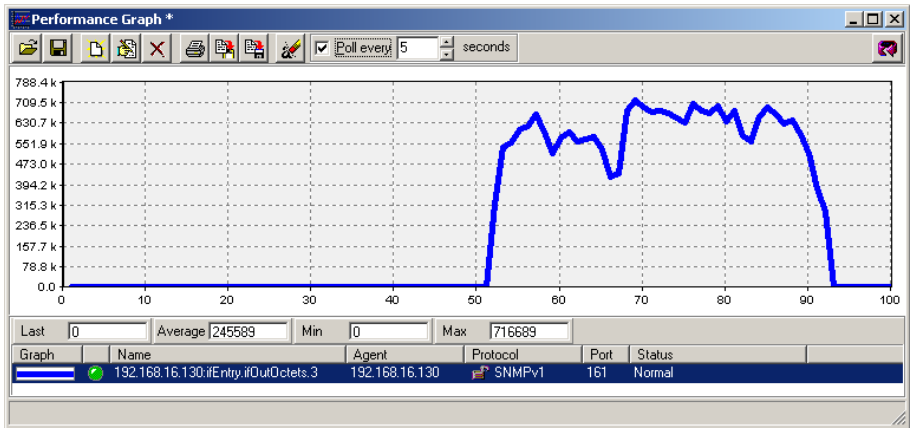

Figura 5.10: Octetos saindo do Mrouter3, interface fxp0

Não foi possível observar a partir dos gráficos e do ambiente de testes um desperdício de recurso no que diz respeito a utilização da memória, pois, mesmo com os testes feitos reservando os recursos estaticamente (tais gráficos não foram apresentados na dissertação), apenas picos de utilização de memória foram observados.

6 CONCLUSÕES E TRABALHOS FUTUROS

Esta dissertação descreve a pesquisa, definições e implementações feitas com o objetivo de se fornecer um mecanismo de gerenciamento capaz de integrar o gerenciamento do QoS e multicast. Após a conclusão de todas as fases da dissertação, algumas conclusões foram alcançadas. Este capítulo será utilizado para descrever as conclusões alcançadas durante este trabalho, assim como apresentar um reflexão sobre trabalhos futuros que podem ser desenvolvidos na área de pesquisa.

Algumas conclusões alcançadas são refletidas diretamente no problema abordado, mais especificamente gerenciamento integrado de QoS e multicast, porém, outras conclusões foram alcançadas, mas estas, com uma característica mais genérica, refletindo problemas encontrados quanto à integração do gerenciamento de redes de modo amplo. Primeiramente serão apresentadas as conclusões diretamente vinculadas ao gerenciamento de QoS e multicast; num segundo momento, as conclusões sobre integração de gerenciamento de redes serão comentadas.

Uma das conclusões e motivações para o desenvolvimento do trabalho foi a dificuldade encontrada para fornecer um adequado gerenciamento de QoS e multicast de forma tradicional, onde são usadas diversas ferramentas, sendo que cada uma executa uma tarefa de gerenciamento e pouca ou nenhuma integração existe entre elas. Entretanto, pode-se concluir também que existem mecanismos emergentes no contexto de gerenciamento de redes, capazes facilitar a integração de qualquer tipo de gerenciamento, e neste caso específico, integração do gerenciamento de QoS e Multicast. O mecanismo de gerenciamento emergente encontrado, e já comentado anteriormente, é gerenciamento baseado em políticas (PBNM).

No entanto, durante a pesquisa, constatou-se que o gerenciamento baseado em políticas com relação à QoS já existia, e estava dando os primeiros passos no sentido de padronizar tal gerenciamento. Portanto, o gerenciamento baseado em políticas para QoS é mais maduro. Maduro, principalmente com relação ao gerenciamento baseado em políticas para multicast, onde não foram encontrados esforços no sentido de melhor definir tal gerenciamento. Neste contexto o gerenciamento baseado em políticas para multicast necessita de maior investigação, pois esse trabalho levantou apenas alguns tópicos relativos ao gerenciamento de multicast, levando-se em conta apenas as necessidades de multicast encontradas nas aplicações avançadas, citadas no início desta dissertação.

A principal e mais esperada conclusão do trabalho é que, através da arquitetura definida, aliada às decisões de opção por tecnologias a serem utilizadas na implementação do protótipo, é possível fornecer gerenciamento integrado de QoS e multicast. Tal conclusão foi obtida através da própria implementação do protótipo, assim como através dos testes realizados. Cabe lembrar que os testes realizados foram feitos sobre um ambiente de testes real, porém limitado, mas capaz de validar a implementação. Outras

conclusões foram alcançadas, estas, porém, com uma abrangência maior com relação à integração de gerenciamento.

Dentre as conclusões alcançadas, pode-se destacar que o mecanismo de gerenciamento baseado em políticas é capaz de integrar diversas formas de gerenciamento de rede, que até o momento vinham sendo realizados de forma individual e com a mínima integração. Entretanto, contatou-se que o gerenciamento baseado em políticas, ainda não definiu como deve ser seu funcionamento com relação a diversas áreas funcionais de gerenciamento. Muitos pesquisadores e até mesmo empresas de desenvolvimento de software de gerenciamento têm se dedicado ao PBNM, porém, em muitos aspectos as definições que podem levar um produto padronizado ao mercado, ainda não existem. Diversas linguagens para definição de políticas de alto nível são apresentadas, porém cada uma delas aborda um aspecto de gerenciamento diferente. Arquiteturas para a aplicação das políticas também têm sido apresentadas, mas poucas definições de como devem se comportar cada um dos elementos das arquiteturas são documentados. O IETF tem focado suas pesquisas na definição de modelos de informação para padronizar a organização e armazenamento das políticas, porém, poucos modelos de informação foram definidos e tomados como padrão até o momento (e.g PCIM e PCIMe), além disso, os modelos até agora apresentados representam as necessidades de organização de políticas de apenas parte de algumas áreas funcionais de gerenciamento (e.g. QPIM com relação ao gerenciamento de tráfego).

Com relação a trabalhos futuros também é possível dividi-los em áreas de abrangências, assim como as conclusões alcançadas, portanto serão apresentados os trabalhos futuros ligados ao problema solucionado na dissertação, e em seguida, trabalhos futuros que podem trazer um benefício indireto ao problema tratado. Os trabalhos futuros serão listados a seguir:

- Na implementação do protótipo foram implementados uma quantidade limitada de PDPs específicos, assim como PDPMonitores, suficientes para validar a proposta. Portanto, um trabalho futuro seria a implementação de novos PDPs específicos e PDPMonitores capazes de aplicar políticas integradas de QoS e multicast em outros tipos de arquiteturas de QoS, utilizando-se outros protocolos de roteamento multicast (e.g. arquitetura DiffServ e o protocolo PIM);

- Os PDPMonitores foram implementados no protótipo de forma bem particular, sendo possível a implementação de um *framework* para desenvolvimento de PDPMonitores, já que estes necessitam monitorar diversos elementos, através de diversas tecnologias;

- Utilizando o PMT do protótipo, é possível controlar a aplicação das políticas de forma individual a cada elemento da rede, porém uma das possibilidades definidas nas padronizações já existentes, seria a aplicação de políticas a um conjunto de elementos da rede com os mesmos papéis (*Roles*), esta seria uma sugestão de implementação futura;

- Durante o desenvolvimento do trabalho, um pequeno modelo de informação foi definido diretamente no LDAP, tal modelo de informação se preocupou apenas com as necessidades da implementação do protótipo, mas a definição de um modelo de informação para armazenamento de políticas de multicast, que suporte todos os aspectos relacionados à muticast, torna-se necessário;

- Com relação ao PBNM de forma geral, alguns trabalhos importantes poderiam ser desenvolvidos, entre eles está a definição de uma linguagem de alto nível, que possibilite a definição de políticas que integrem todas as áreas funcionais do gerenciamento de redes;
- Para a definição da linguagem sugerida no item anterior, é necessário que modelos de informação que atendam aos requisitos básicos de cada área funcional do gerenciamento de redes, também devam ser criados;
- A última sugestão de trabalho futuro seria a definição de uma arquitetura ou definição detalhada dos elementos da arquitetura tradicional de PBNM, com relação ao seu funcionamento e papel dentro do gerenciamento baseado em políticas de forma geral.

REFERÊNCIAS

BAKER, F.; CHAN, K.; SMITH, A. **Management Information Base for the Differentiated Services Architecture**: rfc 3289. [S.l.]: IETF, 2002.

BAKER, F.; KRAWCZYK, J.; SASTRY, A. **Integrated Services Management Information Base using SMIv2**: rfc 2213. [S.l.]: IETF, 1997.

BAKER, F.; KRAWCZYK, J.; SASTRY, A. **RSVP Management Information Base using SMIv2**: rfc 2206. [S.l.]: IETF, 1997.

BAKER, F.; KRAWCZYK, J.; SASTRY, A. **Integrated Services Management Information Base Guaranteed Service Extensions using SMIv2**: rfc 2214. [S.l.]: IETF, 1997.

BLAKE, S. et al. **An Architecture for Differentiated Services**: rfc 2475. [S.l.]: IETF, 1998.

BRADEN, R.; CLARK, D.; SHENKER, S. **Integrated Services in the Internet Architecture**: rfc 1633. [S.l.]: IETF, 1994.

BRADEN, R. et al. **Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification**: rfc 2205. [S.l.]: IETF, 1997.

CASE, J. et al. **A Simple Network Management Protocol (SNMP)**: rfc 1157 (obsoletes: rfc 1098). [S.l.]: IETF, 1990.

DAMIANOU, N. et al. The Ponder Policy Specification Language. In: WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, POLICY, 2., 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.18–22. (Lecture Notes in Computer Science, v.1995).

DAMIANOU, N. et al. Tools for Domain-based Policy Management of Distributed Systems. In: NETWORK OPERATIONS AND MANAGEMENT SYMPOSIUM, NOMS, 15., 2002, Florence, Italy. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.15–19.

DEERING, S. et al. **Protocol independent multicast (PIM)**: motivation and architecture. [S.l.]: IETF, 1995. Expired Internet Draft <draft-ietf-idmr-pim-arch-01.txt>.

DEERING, S.; WAITZMAN, D.; PARTRIDGE, C. **Distance Vector Multicast Routing Protocol**: rfc 1075. [S.l.]: IETF, 1988.

EDER, M.; NAG, S. **Service Management Architectures Issues and Review**: rfc 3052. [S.l.]: IETF, 2001.

GRANVILLE, L. Z.; COELHO, G.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. PoP - An Automated Policy Replacement Architecture for PBNM. In: WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, POLICY, 3., 2002. **Proceedings...** Los Alamitos: IEEE Computer Society, 2002. p.140–145.

GRANVILLE, L. Z. et al. Management of Networks with End-to-End Differentiated Service QoS Capabilities. In: INTERNATIONAL CONFERENCE ADVANCES IN INFORMATION SYSTEMS, ADVIS, 1., 2000, Izmir, Turkey. **Proceedings...** Heidelberg: Springer-Verlag, 2000. p.147–158.

GRANVILLE, L. Z.; TAROUCO, L. M. R. QAME - QoS-Aware Management Environment. In: ANNUAL INTERNATIONAL COMPUTER SOFTWARE AND APPLICATIONS CONFERENCE, COMPSAC, 25., 2001. **Proceedings...** Los Alamitos: IEEE Computer Society, 2001. p.269.

GRANVILLE, L. Z.; VAGUETTI, L.; ALMEIDA, M. J. B.; TAROUCO, L. M. R. A PBNM System for Integrated QoS and Multicast Management. In: INTERNATIONAL WORKSHOP ON POLICIES FOR DISTRIBUTED SYSTEMS AND NETWORKS, POLICY, 4., 2003. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.243–246.

HP, C. **HP Open View PolicyXpert - User's Guide**: manufacturing part. Disponível em: <http://www.openview.hp.com>. Acesso em: 05 nov. 2003.

LEOPOLDINO, G. M.; SANTOS MOREIRA, E. dos. **Modelos de Comunicação para Videoconferência**. [S.l.]: RNP NewsGeneration, 2001. v.5, n.3.

LI, B.; NAHRSTEDT, K. A Control-Based Middleware Framework for Quality of Service Adaptations. **IEEE Journal of Selected Areas in Communications**, [S.l.], v.17, n.9, p.1632–1650, Sept. 1999.

LORENZ, D. H.; ORDA, A. QoS routing in networks with uncertain parameters. **IEEE/ACM Transactions on Networking**, [S.l.], v.6, n.6, p.768–778, 1998.

MCCLOGHRIE, K. et al. **Protocol Independent Multicast MIB for IPv4**: rfc 2934. [S.l.]: IETF, 2000.

MCCLOGHRIE, K.; FARINACCI, D.; THALER, D. **IPv4 Multicast Routing MIB**: rfc 2932. [S.l.]: IETF, 2000.

MCCLOGHRIE, K.; FARINACCI, D.; THALER, D. **Internet Group Management Protocol MIB**: rfc 2933. [S.l.]: IETF, 2000.

MCDYSAN, D. **QoS and Traffic Management in IP and ATM Netwoks**. [S.l.]: McGraw-Hill Osborne Media, 1999. n.1.

MOORE, B. **Policy Core Information Model (PCIM) Extensions**: rfc 3460 (updates rfc 3060). [S.l.]: IETF, 2003.

MOORE, B. et al. **Policy Core Information Model – Version 1 Specification**: rfc 3060. [S.l.]: IETF, 2001.

MOORE, B. et al. **Information Model for Describing Network Device QoS Datapath Mechanisms**. [S.l.]: IETF, 2003. internet-draft <draft-ietf-policy-qos-device-info-model-10.txt>.

POLICY, W. G. **IETF Working Groups**: policy framework. Disponível em: <http://www.ietf.org>. Acesso em: 08 ago. 2003.

SLOMAN, M. Policy Driven Management for Distributed Systems. **Journal of Network and Systems Management**, [S.l.], v.2, n.4, p.333–360, 1994.

SNIR, Y. et al. **Policy QoS Information Model**. [S.l.]: IETF, 2003. Internet-Draft <draft-ietf-policy-qos-info-model-05.txt>.

STRASSNER, J. et al. **Policy Core LDAP Schema**. [S.l.]: IETF, 2002. Internet-Draft <draft-ietf-policy-core-schema-16.txt>.

VAGUETTI, L. et al. An architecture for integrated policy-based management of QoS and multicast-enabled networks. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 21., 2003. **Anais...** Natal : UFRN/DIMAp, 2003. v.1, p.217–232.

VASILAKOS, A. V. et al. Evolutionary-Fuzzy Prediction for Strategic Inter-Domain Routing: architecture and mechanisms. In: IEEE WORLD CONFERENCE ON COMPUTATIONAL INTELLIGENCE, WCCI, 16., 1998, Anchorage, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 1998. p.4–9.

WAHL, M. et al. **Lightweight Directory Access Protocol (v3)**: rfc 2252. [S.l.]: IETF, 1997.

WESTERINEN, A. et al. **Terminology for Policy-Based Management**: rfc 3198. [S.l.]: IETF, 2001.

ANEXO A LDAP SCHEMAS

Modelos de informação utilizados para definição de políticas mapeados para LDAP, e utilizados no serviço de diretório OpenLDAP (v.1.4.8.6).

A.1 Policy Core Information Model Schema

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.3
        NAME 'pcimKeywords'
        DESC 'A set of keywords to assist directory clients in
        locating the policy objects applicable to them.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.4
        NAME 'pcimGroupName'
        DESC 'The user-friendly name of this policy group.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
        )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.5
        NAME 'pcimRuleName'
        DESC 'The user-friendly name of this policy rule.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
        )
```

attributetype (1.3.6.1.4.1.12619.1.3.2.6
NAME ’pcimRuleEnabled’
DESC ’An integer indicating whether a policy rule is administratively enabled (value=1), disabled (value=2), or enabled for debug (value=3).’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.7
NAME ’pcimRuleConditionListType’
DESC ’A value of 1 means that this policy rule is in disjunctive normal form; a value of 2 means that this policy rule is in conjunctive normal form.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.8
NAME ’pcimRuleConditionList’
DESC ’Unordered set of DNs of pcimRuleConditionAssociation entries representing associations between this policy rule and its conditions.’
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)

attributetype (1.3.6.1.4.1.12619.1.3.2.9
NAME ’pcimRuleActionList’
DESC ’Unordered set of DNs of pcimRuleActionAssociation entries representing associations between this policy rule and its actions.’
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)

attributetype (1.3.6.1.4.1.12619.1.3.2.10
NAME ’pcimRuleValidityPeriodList’
DESC ’Unordered set of DNs of pcimRuleValidityAssociation entries that determine when the pcimRule is scheduled to be active or inactive.’
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)

attributetype (1.3.6.1.4.1.12619.1.3.2.11
NAME ’pcimRuleUsage’
DESC ’This attribute is a free-form sting providing guidelines on how this policy should be used.’
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.12
NAME ’pcimRulePriority’
DESC ’A non-negative integer for prioritizing this pcim-Rule relative to other pcimRules. A larger value indicates a higher priority.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.13
NAME ’pcimRuleMandatory’
DESC ’If TRUE, indicates that for this policy rule, the evaluation of its conditions and execution of its actions (if the condition is satisfied) is required.’
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.14
NAME ’pcimRuleSequencedActions’
DESC ’An integer enumeration indicating that the ordering of actions defined by the pcimActionOrder attribute is mandatory(1), recommended(2), or dontCare(3).’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.15
NAME ’pcimRoles’
DESC ’Each value of this attribute represents a role-combination.’
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.16
        NAME 'pcimConditionGroupNumber'
        DESC 'The number of the group to which a policy
        condition belongs. This is used to form the DNF or CNF
        expression associated with a policy rule.'
        EQUALITY numericStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
        SINGLE-VALUE
        )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.17
        NAME 'pcimConditionNegated'
        DESC 'If TRUE (FALSE), it indicates that a policy
        condition IS (IS NOT) negated in the DNF or CNF
        expression associated with a policy rule.'
        EQUALITY booleanMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE
        )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.18
        NAME 'pcimConditionName'
        DESC 'A user-friendly name for a policy condition.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
        )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.19
        NAME 'pcimConditionDN'
        DESC 'A DN that references an instance of a reusable
        policy condition.'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        SINGLE-VALUE
        )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.20
        NAME 'pcimValidityConditionName'
        DESC 'A user-friendly name for identifying an instance of
        pcimRuleValidityAssociation entry.'
        EQUALITY caseIgnoreMatch
        ORDERING caseIgnoreOrderingMatch
        SUBSTR caseIgnoreSubstringsMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
        SINGLE-VALUE
        )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.21
              NAME 'pcimTimePeriodConditionDN'
              DESC 'A reference to a reusable policy time period
              condition.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.22
              NAME 'pcimActionName'
              DESC 'A user-friendly name for a policy action.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.23
              NAME 'pcimActionOrder'
              DESC 'An integer indicating the relative order of an action
              in the context of a policy rule.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.24
              NAME 'pcimActionDN'
              DESC 'A DN that references a reusable policy action.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.25
              NAME 'pcimTPCTime'
              DESC 'The start and end times on which a policy rule is
              valid.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
              SINGLE-VALUE
              )
```

attributetype (1.3.6.1.4.1.12619.1.3.2.26
NAME ’pcimTPCMonthOfYearMask’
DESC ’This identifies the valid months of the year for a policy rule using a 12-bit string that represents the months of the year from January through December.’
EQUALITY bitStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.27
NAME ’pcimTPCDayOfMonthMask’
DESC ’This identifies the valid days of the month for a policy rule using a 62-bit string. The first 31 positions represent the days of the month in ascending order, and the next 31 positions represent the days of the month in descending order.’
EQUALITY bitStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.28
NAME ’pcimTPCDayOfWeekMask’
DESC ’This identifies the valid days of the week for a policy rule using a 7-bit string. This represents the days of the week from Sunday through Saturday.’
EQUALITY bitStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.6
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.29
NAME ’pcimTPCTimeOfDayMask’
DESC ’This identifies the valid range of times for a policy using the format Thhmmss/Thhmmss.’
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.44
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.30
NAME ’pcimTPCLocalOrUtcTime’
DESC ’This defines whether the times in this instance represent local (value=1) times or UTC (value=2) times.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.31
NAME ’pcimVendorConstraintData’
DESC ’Mechanism for representing constraints that have not been modeled as specific attributes. Their format is identified by the OID stored in the attribute pcimVendor-ConstraintEncoding.’
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
)

attributetype (1.3.6.1.4.1.12619.1.3.2.32
NAME ’pcimVendorConstraintEncoding’
DESC ’An OID identifying the format and semantics for the pcimVendorConstraintData for this instance.’
EQUALITY objectIdentifierMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.33
NAME ’pcimVendorActionData’
DESC ’ Mechanism for representing policy actions that have not been modeled as specific attributes. Their format is identified by the OID stored in the attribute pcimVendorActionEncoding.’
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
)

attributetype (1.3.6.1.4.1.12619.1.3.2.34
NAME ’pcimVendorActionEncoding’
DESC ’An OID identifying the format and semantics for the pcimVendorActionData attribute of this instance.’
EQUALITY objectIdentifierMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.35
NAME ’pcimPolicyInstanceName’
DESC ’The user-friendly name of this policy instance.’
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.36
              NAME 'pcimRepositoryName'
              DESC 'The use-friendly name of this policy repository.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.37
              NAME 'pcimSubtreesAuxContainedSet'
              DESC 'DNs of objects that serve as roots for DIT subtrees
              containing policy-related objects.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.38
              NAME 'pcimGroupsAuxContainedSet'
              DESC 'DNs of pcimGroups associated in some way with
              the instance to which this attribute has been appended.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.39
              NAME 'pcimRulesAuxContainedSet'
              DESC 'DNs of pcimRules associated in some way with the
              instance to which this attribute has been appended.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )

objectClass   ( 1.3.6.1.4.1.12619.1.3.1.1
              NAME 'pcimPolicy'
              DESC 'An abstract class that is the base class for all classes
              that describe policy-related instances.'
              SUP top
              ABSTRACT
              MAY ( cn $ pcimKeywords )
              )

objectClass   ( 1.3.6.1.4.1.12619.1.3.1.2
              NAME 'pcimGroup'
              DESC 'A container for a set of related pcimRules and/or a
              set of related pcimGroups.'
              SUP pcimPolicy
              ABSTRACT
              MAY ( pcimGroupName )
              )
```

objectClass (1.3.6.1.4.1.12619.1.3.1.3
NAME ’pcimGroupAuxClass’
DESC ’An auxiliary class that collects a set of related pcimRule and/or pcimGroup entries.’
SUP pcimGroup
AUXILIARY
)

objectClass (1.3.6.1.4.1.12619.1.3.1.4
NAME ’pcimGroupInstance’
DESC ’A structural class that collects a set of related pcimRule and/or pcimGroup entries.’
SUP pcimGroup
STRUCTURAL
)

objectClass (1.3.6.1.4.1.12619.1.3.1.5
NAME ’pcimRule’
DESC ’The base class for representing the "If Condition then Action"semantics associated with a policy rule.’
SUP pcimPolicy
ABSTRACT
MAY (pcimRuleName $ pcimRuleEnabled $ pcimRuleConditionListType $ pcimRuleConditionList $ pcimRuleActionList $ pcimRuleValidityPeriodList $ pcimRuleUsage $ pcimRulePriority $ pcimRuleMandatory $ pcimRuleSequencedActions $ pcimRoles)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.6
NAME ’pcimRuleAuxClass’
DESC ’An auxiliary class for representing the "If Condition then Action"semantics associated with a policy rule.’
SUP pcimRule
AUXILIARY
)

objectClass (1.3.6.1.4.1.12619.1.3.1.7
NAME ’pcimRuleInstance’
DESC ’A structural class for representing the "If Condition then Action"semantics associated with a policy rule.’
SUP pcimRule
STRUCTURAL
)

objectClass (1.3.6.1.4.1.12619.1.3.1.8
NAME ’pcimRuleConditionAssociation’
DESC ’This class contains attributes characterizing the relationship between a policy rule and one of its policy conditions.’
SUP pcimPolicy
MUST (pcimConditionGroupNumber $ pcimConditionNegated)
MAY (pcimConditionName $ pcimConditionDN)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.9
NAME ’pcimRuleValidityAssociation’
DESC ’This defines the scheduled activation or deactivation of a policy rule.’
SUP pcimPolicy
STRUCTURAL
MAY (pcimValidityConditionName $ pcimTimePeriodConditionDN)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.10
NAME ’pcimRuleActionAssociation’
DESC ’This class contains attributes characterizing the relationship between a policy rule and one of its policy actions.’
SUP pcimPolicy
MUST (pcimActionOrder)
MAY (pcimActionName $ pcimActionDN)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.11
NAME ’pcimConditionAuxClass’
DESC ’A class representing a condition to be evaluated in conjunction with a policy rule.’
SUP top
AUXILIARY
)

objectClass (1.3.6.1.4.1.12619.1.3.1.12
NAME ’pcimTPCAuxClass’
DESC ’This provides the capability of enabling or disabling a policy rule according to a predetermined schedule.’
SUP pcimConditionAuxClass
AUXILIARY
MAY (pcimTPCTime $ pcimTPCMonthOfYearMask $ pcimTPCDayOfMonthMask $ pcimTPCDayOfWeekMask $ pcimTPCTimeOfDayMask $ pcimTPCLocalOrUtcTime)
)

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.13
               NAME 'pcimConditionVendorAuxClass'
               DESC 'A class that defines a registered means to describe a
               policy condition.'
               SUP pcimConditionAuxClass
               AUXILIARY
               MAY ( pcimVendorConstraintData $ pcimVendorCon-
               straintEncoding )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.14
               NAME 'pcimActionAuxClass'
               DESC 'A class representing an action to be performed as a
               result of a policy rule.'
               SUP top
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.15
               NAME 'pcimActionVendorAuxClass'
               DESC 'A class that defines a registered means to describe a
               policy action.'
               SUP pcimActionAuxClass
               AUXILIARY
               MAY ( pcimVendorActionData $ pcimVendorActionEn-
               coding )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.16
               NAME 'pcimPolicyInstance'
               DESC 'A structural class to which aux classes containing
               reusable policy information can be attached.'
               SUP pcimPolicy
               MAY ( pcimPolicyInstanceName )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.17
               NAME 'pcimElementAuxClass'
               DESC 'An auxiliary class used to tag instances of classes
               defined outside the realm of policy as relevant to a particular
               policy specification.'
               SUP pcimPolicy
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.18
               NAME 'pcimRepository'
               DESC 'A container for reusable policy information.'
               SUP top
               ABSTRACT
               MAY ( pcimRepositoryName )
               )
```

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.19
               NAME ’pcimRepositoryAuxClass’
               DESC ’An auxiliary class that can be used to aggregate
               reusable policy information.’
               SUP pcimRepository
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.20
               NAME ’pcimRepositoryInstance’
               DESC ’A structural class that can be used to aggregate
               reusable policy information.’
               SUP pcimRepository
               STRUCTURAL
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.21
               NAME ’pcimSubtreesPtrAuxClass’
               DESC ’An auxiliary class providing DN references to roots
               of DIT subtrees containing policy-related objects.’
               SUP top
               AUXILIARY
               MAY ( pcimSubtreesAuxContainedSet )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.22
               NAME ’pcimGroupContainmentAuxClass’
               DESC ’An auxiliary class used to bind pcimGroups to an
               appropriate container object.’
               SUP top
               AUXILIARY
               MAY ( pcimGroupsAuxContainedSet )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.23
               NAME ’pcimRuleContainmentAuxClass’
               DESC ’An auxiliary class used to bind pcimRules to an
               appropriate container object.’
               SUP top
               AUXILIARY
               MAY ( pcimRulesAuxContainedSet )
               )
```

A.2 Policy Core Information Model Extensions Schema

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.40
              NAME 'pcimPolicySetName'
              DESC 'The user-friendly name of a policy set.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.41
              NAME 'pcimDecisionStrategy'
              DESC 'The evaluation method used for the components of
              a in the pcimPolicySet. Valid values: 1 [FirstMatching], 2
              [AllMatching]'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.42
              NAME 'pcimPolicySetList'
              DESC 'List of DN references to pcimPolicySetAssociation
              entries used to aggregate policy sets.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.43
              NAME 'pcimPriority'
              DESC 'Policy priority.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.44
              NAME 'pcimPolicySetDN'
              DESC 'DN reference to a pcimPolicySet entry.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              SINGLE-VALUE
              )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.45
        NAME 'pcimConditionListType'
        DESC 'a value of 1 means that this policy rule is in
        disjunctive normal form; a value of 2 means that this policy
        rule is in conjunctive normal form.'
        EQUALITY numericStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
        SINGLE-VALUE
        )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.46
        NAME 'pcimConditionList'
        DESC 'unordered set of DN references to pcimCondi-
        tionAssociation entries used to aggregate policy condi-
        tions.'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.47
        NAME 'pcimActionList'
        DESC 'Unordered set of DN references to pcimActionAs-
        sociation entries used to aggregate policy actions.'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.48
        NAME 'pcimSequencedActions'
        DESC 'Indicates whether the ordered execution of actions
        in an aggregate is Mandatory, Recommended, or DontCare.'
        EQUALITY numericStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
        SINGLE-VALUE
        )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.49
        NAME 'pcimExecutionStrategy'
        DESC 'Indicates the execution strategy to be used upon an
        action aggregate. VALUES: 1 [Do until success]; 2 [Do all];
        3 [do until failure]. Default value = 2.'
        EQUALITY numericStringMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
        SINGLE-VALUE
        )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.50
        NAME 'pcimVariableDN'
        DESC 'DN reference to a pcimVariable entry.'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        SINGLE-VALUE
        )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.51
              NAME 'pcimValueDN'
              DESC 'DN reference to a pcimValue entry.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.52
              NAME 'pcimIsMirrored'
              DESC 'Indicates whether traffic that mirrors the specified
              filter is to be treated as matching the filter.'
              EQUALITY booleanMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.53
              NAME 'pcimVariableName'
              DESC 'The user-friendly name of a variable.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.54
              NAME 'pcimExpectedValueList'
              DESC 'List of DN references to pcimValueAuxClass
              entries that represent the acceptable values.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.55
              NAME 'pcimVariableModelClass'
              DESC 'Specifies a CIM class name or oid.'
              EQUALITY caseIgnoreMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.56
              NAME 'pcimVariableModelProperty'
              DESC 'Specifies a CIM property name or oid.'
              EQUALITY caseIgnoreMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.57
              NAME 'pcimExpectedValueTypes'
              DESC 'List of object class names or oids of subclasses of
              pcimValueAuxClass that define acceptable value types.'
              EQUALITY caseIgnoreMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.58
              NAME 'pcimValueName'
              DESC 'The user-friendly name of a value.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.59
              NAME 'pcimIPv4AddrList'
              DESC 'List of IPv4 address values, ranges or hosts.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.60
              NAME 'pcimIPv6AddrList'
              DESC 'List of IPv6 address values, ranges or hosts.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.61
              NAME 'pcimMACAddrList'
              DESC 'List of MAC address values or ranges.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.62
              NAME 'pcimStringList'
              DESC 'List of strings or wildcarded strings.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.63
              NAME ’pcimBitStringList’
              DESC ’List of bit strings or masked bit strings.’
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.64
              NAME ’pcimIntegerList’
              DESC ’List of integers or integer ramges.’
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.65
              NAME ’pcimBoolean’
              DESC ’A boolean value.’
              EQUALITY booleanMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.66
              NAME ’pcimReusableContainerName’
              DESC ’The user-friendly name of a reusable policy
              container.’
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.67
              NAME ’pcimReusableContainerList’
              DESC ’List of DN references to pcimReusableContainer
              entries.’
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.68
              NAME 'pcimRole'
              DESC 'String representing a role.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.69
              NAME 'pcimRoleCollectionName'
              DESC 'The user-friendly name of a role collection.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.70
              NAME 'pcimElementList'
              DESC 'List of DN references to entries representing
              managed elements.'
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.71
              NAME 'pcimFilterName'
              DESC 'The user-friendly name of a filter.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.72
              NAME 'pcimFilterIsNegated'
              DESC 'If TRUE, indicates that the filter matches all but the
              messages or packets that conform to the specified criteria.
              Default: FALSE.'
              EQUALITY booleanMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
              SINGLE-VALUE
              )
```

```
attributetype  ( 1.3.6.1.4.1.12619.1.3.2.73
               NAME ’pcimIPHdrVersion’
               DESC ’The IP version.’
               EQUALITY numericStringMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
               SINGLE-VALUE
               )

attributetype  ( 1.3.6.1.4.1.12619.1.3.2.74
               NAME ’pcimIPHdrSourceAddress’
               DESC ’The IP source address.’
               EQUALITY octetStringMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
               SINGLE-VALUE
               )

attributetype  ( 1.3.6.1.4.1.12619.1.3.2.75
               NAME ’pcimIPHdrSourceAddressEndOfRange’
               DESC ’The end or address range for the IP source address.’
               EQUALITY octetStringMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
               SINGLE-VALUE
               )

attributetype  ( 1.3.6.1.4.1.12619.1.3.2.76
               NAME ’pcimIPHdrSourceMask’
               DESC ’The address mask for the IP source address.’
               EQUALITY octetStringMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
               SINGLE-VALUE
               )

attributetype  ( 1.3.6.1.4.1.12619.1.3.2.77
               NAME ’pcimIPHdrDestAddress’
               DESC ’The IP destination address.’
               EQUALITY octetStringMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
               SINGLE-VALUE
               )

attributetype  ( 1.3.6.1.4.1.12619.1.3.2.78
               NAME ’pcimIPHdrDestAddressEndOfRange’
               DESC ’The end of address range for the IP destination
               address.’
               EQUALITY octetStringMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
               SINGLE-VALUE
               )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.79
              NAME 'pcimIPHdrDestMask'
              DESC 'The address mask for the IP destination address.'
              EQUALITY octetStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.80
              NAME 'pcimIPHdrProtocolID'
              DESC 'The IP protocol type.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.81
              NAME 'pcimIPHdrSourcePortStart'
              DESC 'The start of the source port range.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.82
              NAME 'pcimIPHdrSourcePortEnd'
              DESC 'The end of the source port range.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.83
              NAME 'pcimIPHdrDestPortStart'
              DESC 'The start of the destination port range.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.84
              NAME 'pcimIPHdrDestPortEnd'
              DESC 'The end of the destination port range.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.85
              NAME 'pcimIPHdrDSCPList'
              DESC 'The DSCP values.'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              )
```

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.86
              NAME ’pcimIPHdrFlowLabel’
              DESC ’The IP flow label.’
              EQUALITY octetStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.87
              NAME ’pcim8021HdrSourceMACAddress’
              DESC ’The source MAC address.’
              EQUALITY octetStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.88
              NAME ’pcim8021HdrSourceMACMask’
              DESC ’The source MAC address mask.’
              EQUALITY octetStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.89
              NAME ’pcim8021HdrDestMACAddress’
              DESC ’The destination MAC address.’
              EQUALITY octetStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.90
              NAME ’pcim8021HdrDestMACMask’
              DESC ’The destination MAC address mask.’
              EQUALITY octetStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.91
              NAME ’pcim8021HdrProtocolID’
              DESC ’The 802.1 protocol ID.’
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.92
              NAME ’pcim8021HdrPriority’
              DESC ’The 802.1 priority.’
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              )
```

attributetype (1.3.6.1.4.1.12619.1.3.2.93
NAME ’pcim8021HdrVLANID’
DESC ’The 802.1 VLAN ID.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
)

attributetype (1.3.6.1.4.1.12619.1.3.2.94
NAME ’pcimFilterListName’
DESC ’The user-friendly name of a filter list.’
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.95
NAME ’pcimFilterDirection’
DESC ’The direction of the packets or messages to which this filter is to be applied.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
)

attributetype (1.3.6.1.4.1.12619.1.3.2.96
NAME ’pcimFilterEntryList’
DESC ’List of DN references to pcimFilterEntry entries.’
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)

attributetype (1.3.6.1.4.1.12619.1.3.2.97
NAME ’pcimVendorVariableData’
DESC ’Mechanism for representing variables that have not been modeled as specific attributes. Their format is identified by the OID stored in the attribute pcimVendor-VariableEncoding.’
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
)

attributetype (1.3.6.1.4.1.12619.1.3.2.98
NAME ’pcimVendorVariableEncoding’
DESC ’An OID identifying the format and semantics for the pcimVendorVariableData for this instance.’
EQUALITY objectIdentifierMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.99
NAME ’pcimVendorValueData’
DESC ’Mechanism for representing values that have not been modeled as specific attributes. Their format is identified by the OID stored in the attribute pcimVendor-ValueEncoding.’
EQUALITY octetStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
)

attributetype (1.3.6.1.4.1.12619.1.3.2.100
NAME ’pcimVendorValueEncoding’
DESC ’An OID identifying the format and semantics for the pcimVendorValueData for this instance.’
EQUALITY objectIdentifierMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.38
SINGLE-VALUE
)

objectClass (1.3.6.1.4.1.12619.1.3.1.24
NAME ’pcimPolicySet’
DESC ’Abstract class that represents a collection of policies that form a coherent set.’
SUP pcimPolicy
ABSTRACT
MAY (pcimPolicySetName $ pcimDecisionStrategy $ pcimRoles $ pcimPolicySetList)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.25
NAME ’pcimPolicySetAssociation’
DESC ’Structural class that contains attributes character-izing the relationship between a policy set and one of its components.’
SUP pcimPolicy
STRUCTURAL
MUST (pcimPriority)
MAY (pcimPolicySetName $ pcimPolicySetDN)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.26
NAME ’pcimeGroup’
DESC ’A container for a set of related pcimPolicyRule entries and/or a set of related pcimGroup entries.’
SUP pcimPolicySet
ABSTRACT
MAY (pcimGroupName)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.27
NAME ’pcimPolicyRule’
DESC ’The base class for representing the "If Condition then Action"semantics associated with a Policy Rule’
SUP pcimPolicySet
ABSTRACT
MAY (pcimRuleName $ pcimRuleEnabled $ pcimConditionListType $ pcimConditionList $ pcimActionList $ pcimRuleValidityPeriodList $ pcimRuleUsage $ pcimRuleMandatory $ pcimSequencedActions $ pcimExecutionStrategy)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.28
NAME ’pcimPolicyRuleAuxClass’
DESC ’An auxiliary class for representing the "If Condition then Action"semantics associated with a policy rule.’
SUP pcimPolicyRule
AUXILIARY
)

objectClass (1.3.6.1.4.1.12619.1.3.1.29
NAME ’pcimPolicyRuleInstance’
DESC ’A structural class for representing the "If Condition then Action"semantics associated with a policy rule.’
SUP pcimPolicyRule
STRUCTURAL
)

objectClass (1.3.6.1.4.1.12619.1.3.1.30
NAME ’pcimConditionAssociation’
DESC ’This class contains attributes characterizing the relationship between a policy condition and one of its aggregators: pcimPolicyRule or pcimCompoundConditionAuxClass. It is used in the realization of a policy condition structure.’
SUP pcimPolicy
STRUCTURAL
MUST (pcimConditionGroupNumber $ pcimConditionNegated)
MAY (pcimConditionName $ pcimConditionDN)
)

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.31
               NAME 'pcimActionAssociation'
               DESC 'This class contains attributes characterizing the
               relationship between a policy action and one of its
               aggregators. It is used in the realization of a policy action
               structure.'
               SUP pcimPolicy
               STRUCTURAL
               MUST ( pcimActionOrder )
               MAY ( pcimActionName $ pcimActionDN )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.32
               NAME 'pcimSimpleConditionAuxClass'
               DESC 'An auxiliary class that evaluate the matching
               between a value and a variable.'
               SUP pcimConditionAuxClass
               AUXILIARY
               MAY ( pcimVariableDN $ pcimValueDN )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.33
               NAME 'pcimCompoundConditionAuxClass'
               DESC 'An auxiliary class that represents a boolean
               combination of simpler conditions.'
               SUP pcimConditionAuxClass
               AUXILIARY
               MAY ( pcimConditionListType $ pcimConditionList )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.34
               NAME 'pcimCompoundFilterAuxClass'
               DESC 'A compound condition with mirroring capabilities
               for traffic caracterization.'
               SUP pcimCompoundConditionAuxClass
               AUXILIARY
               MAY ( pcimIsMirrored )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.35
               NAME 'pcimSimpleActionAuxClass'
               DESC 'This class contains attributes characterizing the
               relationship between a Simple PolicyAction and one
               variable and one value.'
               SUP pcimActionAuxClass
               AUXILIARY
               MAY ( pcimVariableDN $ pcimValueDN )
               )
```

objectClass (1.3.6.1.4.1.12619.1.3.1.36
NAME ’pcimCompoundActionAuxClass’
DESC ’A class that aggregates simpler actions in a sequence with specific execution strategy.’
SUP pcimActionAuxClass
AUXILIARY
MAY (pcimActionList $ pcimSequencedActions $ pcimExecutionStrategy)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.37
NAME ’pcimVariable’
DESC ’Base class for representing a variable whose actual value can be matched against or set to a specific value.’
SUP top
ABSTRACT
MAY (pcimVariableName $ pcimExpectedValueList)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.38
NAME ’pcimExplicitVariableAuxClass’
DESC ’Explicitly defined policy variable evaluated within the context of the CIM Schema.’
SUP pcimVariable
AUXILIARY
MUST (pcimVariableModelClass $ pcimVariableModel-Property)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.39
NAME ’pcimImplicitVariableAuxClass’
DESC ’Implicitly defined policy variables whose evaluation depends on the usage context. Subclasses specify the data type and semantics of the variables.’
SUP pcimVariable
AUXILIARY
MUST (pcimExpectedValueTypes)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.40
NAME ’pcimSourceIPv4VariableAuxClass’
DESC ’Source IP v4 address’
SUP pcimImplicitVariableAuxClass
AUXILIARY
)

objectClass (1.3.6.1.4.1.12619.1.3.1.41
NAME ’pcimSourceIPv6VariableAuxClass’
DESC ’Source IP v6 address’
SUP pcimImplicitVariableAuxClass
AUXILIARY
)

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.42
               NAME ’pcimDestinationIPv4VariableAuxClass’
               DESC ’Destination IP v4 address’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.43
               NAME ’pcimDestinationIPv6VariableAuxClass’
               DESC ’Destination IP v6 address’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.44
               NAME ’pcimSourcePortVariableAuxClass’
               DESC ’Source port’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.45
               NAME ’pcimDestinationPortVariableAuxClass’
               DESC ’Destination port’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.46
               NAME ’pcimIPProtocolVariableAuxClass’
               DESC ’IP protocol number’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.47
               NAME ’pcimIPVersionVariableAuxClass’
               DESC ’IP version number’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.48
               NAME ’pcimIPToSVariableAuxClass’
               DESC ’IP ToS’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.49
               NAME ’pcimDSCPVariableAuxClass’
               DESC ’DiffServ code point’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
```

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.50
               NAME ’pcimFlowIdVariableAuxClass’
               DESC ’Flow Identifier’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.51
               NAME ’pcimSourceMACVariableAuxClass’
               DESC ’Source MAC address’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.52
               NAME ’pcimDestinationMACVariableAuxClass’
               DESC ’Destination MAC address’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.53
               NAME ’pcimVLANVariableAuxClass’
               DESC ’VLAN’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.54
               NAME ’pcimCoSVariableAuxClass’
               DESC ’Class of service’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.55
               NAME ’pcimEthertypeVariableAuxClass’
               DESC ’Ethertype’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.56
               NAME ’pcimSourceSAPVariableAuxClass’
               DESC ’Source SAP’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.57
               NAME ’pcimDestinationSAPVariableAuxClass’
               DESC ’Destination SAP’
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
```

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.58
               NAME 'pcimSNAPOUIVariableAuxClass'
               DESC 'SNAP OUI'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.59
               NAME 'pcimSNAPTypeVariableAuxClass'
               DESC 'SNAP type'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.60
               NAME 'pcimFlowDirectionVariableAuxClass'
               DESC 'Flow direction'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.61
               NAME 'pcimValueAuxClass'
               DESC 'Base class for representing a value that can be
               matched against or set for a specific variable.'
               SUP top
               AUXILIARY
               MAY ( pcimValueName )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.62
               NAME 'pcimIPv4AddrValueAuxClass'
               DESC 'IP v4 address value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimIPv4AddrList )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.63
               NAME 'pcimIPv6AddrValueAuxClass'
               DESC 'IP v6 address value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimIPv6AddrList )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.64
               NAME 'pcimMACAddrValueAuxClass'
               DESC 'MAC address value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimMACAddrList )
               )
```

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.65
               NAME 'pcimStringValueAuxClass'
               DESC 'String value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimStringList )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.66
               NAME 'pcimBitStringValueAuxClass'
               DESC 'Bit string value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimBitStringList )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.67
               NAME 'pcimIntegerValueAuxClass'
               DESC 'Integer value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimIntegerList )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.68
               NAME 'pcimBooleanValueAuxClass'
               DESC 'Boolean value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimBoolean )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.69
               NAME 'pcimReusableContainer'
               DESC 'A container for reusable policy information.'
               SUP top
               ABSTRACT
               MAY ( pcimReusableContainerName $ pcimReusableCon-
               tainerList )
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.70
               NAME 'pcimReusableContainerAuxClass'
               DESC 'An auxiliary class that can be used to aggregate
               reusable policy information.'
               SUP pcimReusableContainer
               AUXILIARY
               )
```

objectClass (1.3.6.1.4.1.12619.1.3.1.71
NAME ’pcimReusableContainerInstance’
DESC ’A structural class that can be used to aggregate reusable policy information.’
SUP pcimReusableContainer
STRUCTURAL
)

objectClass (1.3.6.1.4.1.12619.1.3.1.72
NAME ’pcimRoleCollection’
DESC ’This class is used to group together entries that share a same role.’
SUP pcimPolicy
STRUCTURAL
MUST (pcimRole)
MAY (pcimRoleCollectionName $ pcimElementList)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.73
NAME ’pcimFilterEntry’
DESC ’This class is used as a base class for representing message or packet filters.’
SUP pcimPolicy
ABSTRACT
MAY (pcimFilterName $ pcimFilterIsNegated)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.74
NAME ’pcimIPHeaders’
DESC ’This class defines an IP header filter.’
SUP pcimFilterEntry
STRUCTURAL
MAY (pcimIPHdrVersion $ pcimIPHdrSourceAddress $ pcimIPHdrSourceAddressEndOfRange $ pcimIPHdrSourceMask $ pcimIPHdrDestAddress $ pcimIPHdrDestAddressEndOfRange $ pcimIPHdrDestMask $ pcimIPHdrProtocolID $ pcimIPHdrSourcePortStart $ pcimIPHdrSourcePortEnd $ pcimIPHdrDestPortStart $ pcimIPHdrDestPortEnd $ pcimIPHdrDSCPList $ pcimIPHdrFlowLabel)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.75
NAME ’pcim8021Headers’
DESC ’This class defines an 802.1 header filter.’
SUP pcimFilterEntry
STRUCTURAL
MAY (pcim8021HdrSourceMACAddress $ pcim8021HdrSourceMACMask $ pcim8021HdrDestMACAddress $ pcim8021HdrDestMACMask $ pcim8021HdrProtocolID $ pcim8021HdrPriority $ pcim8021HdrVLANID)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.76
NAME ’pcimVendorVariableAuxClass’
DESC ’A class that defines a registered means to describe a policy variable.’
SUP pcimVariable
AUXILIARY
MAY (pcimVendorVariableData $ pcimVendorVariableEncoding)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.77
NAME ’pcimVendorValueAuxClass’
DESC ’A class that defines a registered means to describe a policy value.’
SUP pcimValueAuxClass
AUXILIARY
MAY (pcimVendorValueData $ pcimVendorValueEncoding)
)

A.3 QoS e multicast Schemas

attributetype (1.3.6.1.4.1.12619.1.3.2.101
NAME ’qpMaxPacketSize’
DESC ’This action controls the Per-Hop-Behavior provided to behavior aggregates’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.102
NAME ’qpForwardingPriority’
DESC ’Must be non-negative. The value 0 means that pre-emptive forwarding is not required. A positive value indicates the priority that is to be assigned for this (set of) flow(s). Larger values represent higher priorities.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.103
NAME ’qpBandwidthUnits’
DESC ’Two values are possible. The value of 0 is used to specify units of bits/sec, while the value of 1 is used to specify units as a percentage of the available bandwidth. If this property indicates that the bandwidth units are percentages, then each of the bandwidth properties expresses a whole- number percentage, and hence its maximum value is 100.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.104
NAME ’qpMinBandwidth’
DESC ’The value must be greater than 0. If the property qpMaxBandwidth is defined, then the value of qpMinBandwidth must be less than or equal to the value of qpMaxBandwidth.’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.105
NAME ’qpMaxBandwidth’
DESC ’This action controls the Per-Hop-Behavior provided to behavior aggregates’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.106
NAME ’qpMaxDelay’
DESC ’The value must be greater than 0.(microseconds)’
EQUALITY numericStringMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
SINGLE-VALUE
)

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.107
              NAME 'qpMaxJitter'
              DESC 'The value must be greater than 0.(microseconds)'
              EQUALITY numericStringMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.108
              NAME 'qpFairness'
              DESC 'The value of FALSE means that fair queuing is not
              required for this class of traffic, while the value of TRUE
              means that fair queuing is required for this class of traffic.'
              EQUALITY booleanMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
              SINGLE-VALUE
              )

attributetype ( 1.3.6.1.4.1.12619.1.3.2.109
              NAME 'pcimFloatList'
              DESC 'List of floats or floats ramges.'
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              )

objectClass   ( 1.3.6.1.4.1.12619.1.3.1.78
              NAME 'pmcastGroupIPv4VariableAuxClass'
              DESC 'Multicast Group IPv4 address'
              SUP pcimImplicitVariableAuxClass
              AUXILIARY
              )

objectClass   ( 1.3.6.1.4.1.12619.1.3.1.79
              NAME 'pmcastGroupIPv6VariableAuxClass'
              DESC 'Multicast Group IPv6 address'
              SUP pcimImplicitVariableAuxClass
              AUXILIARY
              )

objectClass   ( 1.3.6.1.4.1.12619.1.3.1.80
              NAME 'pmcastIGMPProtocolVariableAuxClass'
              DESC 'IGMP Protocol'
              SUP pcimImplicitVariableAuxClass
              AUXILIARY
              )
```

```
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.81
               NAME    'pmcastigmpInterfaceQueryIntervalVariableAux-
               Class'
               DESC 'The IGMP query interval in seconds'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.82
               NAME 'pmcastMRouteEnableVariableAuxClass'
               DESC 'The mulicast route enable'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.83
               NAME 'pmcastRouterProtocolVariableAuxClass'
               DESC 'The multicast router protocol'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.84
               NAME 'pmcastJoinPruneIntervalVariableAuxClass'
               DESC 'Join and Prune Interval messages'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.85
               NAME 'pmcastpimInterfaceModeVariableAuxClass'
               DESC 'Dense or Sparce type modes'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.86
               NAME 'pmcastdvmrpInterfaceMetricVariableAuxClass'
               DESC 'The distance metric used by protocol'
               SUP pcimImplicitVariableAuxClass
               AUXILIARY
               )

objectClass    ( 1.3.6.1.4.1.12619.1.3.1.87
               NAME 'pcimFloatValueAuxClass'
               DESC 'Float value.'
               SUP pcimValueAuxClass
               AUXILIARY
               MUST ( pcimFloatList )
               )
```

objectClass (1.3.6.1.4.1.12619.1.3.1.88
NAME ’QoSPolicyPHBAction’
DESC ’This action controls the Per-Hop-Behavior provided to behavior aggregates.’
SUP pcimActionAuxClass
AUXILIARY
MAY (qpMaxPacketSize)
)

objectClass (1.3.6.1.4.1.12619.1.3.1.89
NAME ’QoSPolicyBandwidthAction’
DESC ’This action controls the bandwidth, delay, and forwarding characteristics of the PHB.’
SUP QoSPolicyPHBAction
AUXILIARY
MAY (qpForwardingPriority $ qpBandwidthUnits $ qp-MinBandwidth $ qpMaxBandwidth $ qpMaxDelay $ qpMaxJitter $ qpFairness)
)

A.4 Policy Controll Schema

attributetype (1.3.6.1.4.1.12619.1.3.2.110
NAME ’pdpID’
DESC ’The system pdpID ’
EQUALITY caseIgnoreMatch
ORDERING caseIgnoreOrderingMatch
SUBSTR caseIgnoreSubstringsMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.111
NAME ’PolicyDN’
DESC ’Policy Distinguish Name’
EQUALITY distinguishedNameMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
SINGLE-VALUE
)

attributetype (1.3.6.1.4.1.12619.1.3.2.112
NAME ’polCOperationStatus’
DESC ’if OperationStatus is TRUE the policy is applied, if OperationStatus is FALSE the policy isn´t applied’
EQUALITY booleanMatch
SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
SINGLE-VALUE
)

```
attributetype ( 1.3.6.1.4.1.12619.1.3.2.113
              NAME ’polCValidity’
              DESC ’if PolicyValidity is TRUE the policy is valid in PDP,
              if PolicyValidity is FALSE the policy isn´t valid in PDP’
              EQUALITY booleanMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.114
              NAME ’pdpPepAssociationList’
              DESC ’List of PEP and PDP association in this Policy’
              EQUALITY distinguishedNameMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.115
              NAME ’pepID’
              DESC ’The system pepID ’
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.116
              NAME ’pepCValidityStatusList’
              DESC ’if ValidityStatus is TRUE the policy is valid in PEP,
              if ValidityStatus is FALSE the policy isn´t valid in PEP’
              EQUALITY booleanMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.117
              NAME ’pepCIPAddress’
              DESC ’The IP Address of PEP’
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )
attributetype ( 1.3.6.1.4.1.12619.1.3.2.118
              NAME ’pdpCIPAddress’
              DESC ’The IP Address of PDP’
              EQUALITY caseIgnoreMatch
              ORDERING caseIgnoreOrderingMatch
              SUBSTR caseIgnoreSubstringsMatch
              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
              SINGLE-VALUE
              )
```

```
attributetype  ( 1.3.6.1.4.1.12619.1.3.2.119
               NAME ’pdpControllDN’
               DESC ’DN for the PDP association’
               EQUALITY distinguishedNameMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
               SINGLE-VALUE
               )
attributetype  ( 1.3.6.1.4.1.12619.1.3.2.120
               NAME ’pepControllList’
               DESC ’List of PEP aggregated in this PDP and Policy’
               EQUALITY distinguishedNameMatch
               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.90
               NAME ’policyController’
               DESC ’Information about Policy status ’
               SUP Top
               MUST ( policyDN $ polCOperationStatus)
               MAY ( polCValidity $ pdpPepAssociationList)
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.91
               NAME ’pdpController’
               DESC ’Information about Policy status in PDP’
               SUP Top
               MUST ( pdpID )
               MAY ( pdpCIPAddress )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.92
               NAME ’pepController’
               DESC ’Information about Policy status in PEP’
               SUP Top
               MUST ( pepID )
               MAY ( pepCIPAddress )
               )
objectClass    ( 1.3.6.1.4.1.12619.1.3.1.93
               NAME ’pdpPepAssociation’
               DESC ’Association between PDPs and PEPs’
               SUP Top
               MUST ( pdpControllDN $ pepControllList )
               MAY ( pepCValidityStatusList )
               )
```

ANEXO B PACKAGE POLICY CONTAINER

class java.lang.Object
- class ufrgs.inf.noc.policy.pcim.cimComponent
  - class ufrgs.inf.noc.policy.pcim.cimSystemComponent
    - class ufrgs.inf.noc.policy.pcim.pcimeContainedDomain
    - class ufrgs.inf.noc.policy.pcim.pcimRepositoryInpcimRepository
- class ufrgs.inf.noc.policy.pcim.cimDependency
  - class ufrgs.inf.noc.policy.pcim.pcimeExceptedValueForVariable
  - class ufrgs.inf.noc.policy.pcim.pcimeRoleCollectionIncimSystem
  - class ufrgs.inf.noc.policy.pcim.pcimPolicyIncimSystem
    - class ufrgs.inf.noc.policy.pcim.pcimActionInpcimRepository
    - class ufrgs.inf.noc.policy.pcim.pcimConditionInpcimRepository
    - class ufrgs.inf.noc.policy.pcim.pcimePolicySetIncimSystem
      - class ufrgs.inf.noc.policy.pcim.pcimeGroupIncimSystem
      - class ufrgs.inf.noc.policy.pcim.pcimeRuleIncimSystem
    - class ufrgs.inf.noc.policy.pcim.pcimeReusablePolicy
    - class ufrgs.inf.noc.policy.pcim.pcimGroupIncimSystem
    - class ufrgs.inf.noc.policy.pcim.pcimRuleIncimSystem
- class ufrgs.inf.noc.policy.pcim.cimManagedElement
  - class ufrgs.inf.noc.policy.pcim.cimCollection
    - class ufrgs.inf.noc.policy.pcim.pcimeRoleCollection
- class ufrgs.inf.noc.policy.pcim.cimManagedSystemElement
  - class ufrgs.inf.noc.policy.pcim.cimLogicalElement
    - class ufrgs.inf.noc.policy.pcim.cimEnabledLogicalElement
      - class ufrgs.inf.noc.policy.pcim.cimSystem
        - class ufrgs.inf.noc.policy.pcim.cimAdminDomain
          - class ufrgs.inf.noc.policy.pcim.pcimeReusablePContainer
          - class ufrgs.inf.noc.policy.pcim.pcimRepository
- class ufrgs.inf.noc.policy.pcim.pcimPolicy
  - class ufrgs.inf.noc.policy.pcim.pcimAction
    - class ufrgs.inf.noc.policy.pcim.pcimeCompoundPAction
    - class ufrgs.inf.noc.policy.pcim.pcimeSimplePAction
    - class ufrgs.inf.noc.policy.pcim.pcimVendorPAction
  - class ufrgs.inf.noc.policy.pcim.pcimCondition
    - class ufrgs.inf.noc.policy.pcim.pcimeCompoundPCondition
    - class ufrgs.inf.noc.policy.pcim.pcimeSimplePCondition
    - class ufrgs.inf.noc.policy.pcim.pcimTPC
    - class ufrgs.inf.noc.policy.pcim.pcimVendorPCondition

class ufrgs.inf.noc.policy.pcim.pcimePolicySet
class ufrgs.inf.noc.policy.pcim.pcimeGroup
class ufrgs.inf.noc.policy.pcim.pcimeRule
class ufrgs.inf.noc.policy.pcim.pcimeValue
class ufrgs.inf.noc.policy.pcim.pcimeBitStringValue
class ufrgs.inf.noc.policy.pcim.pcimeBooleanValue
class ufrgs.inf.noc.policy.pcim.pcimeIntegerValue
class ufrgs.inf.noc.policy.pcim.pcimeIPv4AddrValue
class ufrgs.inf.noc.policy.pcim.pcimeIPv6AddrValue
class ufrgs.inf.noc.policy.pcim.pcimeMACAddrValue
class ufrgs.inf.noc.policy.pcim.pcimeStringValue
class ufrgs.inf.noc.policy.pcim.pcimeVariable
class ufrgs.inf.noc.policy.pcim.pcimeExplicitVariable
class ufrgs.inf.noc.policy.pcim.pcimeImplicitVariable
class ufrgs.inf.noc.policy.pcim.pcimeCoSVariable
class ufrgs.inf.noc.policy.pcim.pcimeDestinationIPv4Variable
class ufrgs.inf.noc.policy.pcim.pcimeDestinationIPv6Variable
class ufrgs.inf.noc.policy.pcim.pcimeDestinationMACVariable
class ufrgs.inf.noc.policy.pcim.pcimeDestinationPortVariable
class ufrgs.inf.noc.policy.pcim.pcimeDestinationSAPVariable
class ufrgs.inf.noc.policy.pcim.pcimeDSCPVariable
class ufrgs.inf.noc.policy.pcim.pcimeEthertypeVariable
class ufrgs.inf.noc.policy.pcim.pcimeFlowDirectionVariable
class ufrgs.inf.noc.policy.pcim.pcimeFlowIdVariable
class ufrgs.inf.noc.policy.pcim.pcimeIPProtocolVariable
class ufrgs.inf.noc.policy.pcim.pcimeIPToSVariable
class ufrgs.inf.noc.policy.pcim.pcimeIPVersionVariable
class ufrgs.inf.noc.policy.pcim.pcimeSNAPOUIVariable
class ufrgs.inf.noc.policy.pcim.pcimeSNAPTypeVariable
class ufrgs.inf.noc.policy.pcim.pcimeSourceIPv4Variable
class ufrgs.inf.noc.policy.pcim.pcimeSourceIPv6Variable
class ufrgs.inf.noc.policy.pcim.pcimeSourceMACVariable
class ufrgs.inf.noc.policy.pcim.pcimeSourcePortVariable
class ufrgs.inf.noc.policy.pcim.pcimeSourceSAPVariable
class ufrgs.inf.noc.policy.pcim.pcimeVLANVariable
class ufrgs.inf.noc.policy.pcim.pcimGroup
class ufrgs.inf.noc.policy.pcim.pcimRule
class ufrgs.inf.noc.policy.pcim.cimMemberOfCollection
class ufrgs.inf.noc.policy.pcim.ElementInpcimeRoleCollection
class ufrgs.inf.noc.policy.pcim.pcimComponent
class ufrgs.inf.noc.policy.pcim.pcimActionInpcimRule
class ufrgs.inf.noc.policy.pcim.pcimConditionInpcimRule
class ufrgs.inf.noc.policy.pcim.pcimeActionStructure
class ufrgs.inf.noc.policy.pcim.pcimActionInpcimAction
class ufrgs.inf.noc.policy.pcim.pcimActionInpcimeRule
class ufrgs.inf.noc.policy.pcim.pcimeConditionStructure
class ufrgs.inf.noc.policy.pcim.pcimConditionInpcimCondition
class ufrgs.inf.noc.policy.pcim.pcimConditionInpcimeRule

class ufrgs.inf.noc.policy.pcim.pcimeSetComponent
class ufrgs.inf.noc.policy.pcim.pcimeValueInpcimeSimplePAction
class ufrgs.inf.noc.policy.pcim.pcimeValueInpcimeSimplePCondition
class ufrgs.inf.noc.policy.pcim.pcimeVariableInpcimeSimplePAction
class ufrgs.inf.noc.policy.pcim.pcimeVariableInpcimeSimplePCondition
class ufrgs.inf.noc.policy.pcim.pcimGroupInpcimGroup
class ufrgs.inf.noc.policy.pcim.pcimRuleInpcimGroup
class ufrgs.inf.noc.policy.pcim.pcimRuleValidityPeriod