Criação e Visualização de Domínios Dinâmicos em Ambientes de Gerenciamento de Redes

Márcio Bartz Ceccon, Lisandro Zambenedetti Granville, Maria Janilce Bosquiroli Almeida, Liane Margarida Rockenbach Tarouco

Universidade Federal do Rio Grande do Sul - UFRGS
Instituto de Informática
Caixa Postal 15.064, CEP: 91.501-970
Porto Alegre, RS

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

Abstract. *Dynamic domains are domains that need be quickly created, used and discarded. Today, there are no facilities available to support dynamic domains in most network management systems. This paper introduces two new languages to deal with dynamic domains. The first language is used to create new domains through the selection of managed objects. The second language, on its turn, is used to visualize the dynamic domains created by the first language. Both languages are explained through examples and implementations details are presented.*

Resumo. *Domínios dinâmicos são domínios que necessitam ser rapidamente criados, utilizados e descartados. Atualmente, na maioria dos sistemas de gerenciamento de redes não existem facilidades disponíveis para suportar o conceito de domínios dinâmicos. Este artigo apresenta duas novas linguagens relacionadas à manipulação de domínios dinâmicos. A primeira linguagem é utilizada para criar novos domínios a partir da seleção de objetos gerenciáveis. A segunda linguagem, por sua vez, é utilizada para visualizar os domínios dinâmicos criados através da primeira linguagem. Ambas as linguagens são explicadas através de exemplos. Além disso, detalhes de implementação são apresentados.*

1. Introdução

Em sistemas de gerenciamento de redes, domínios são recursos utilizados para agrupar objetos gerenciáveis [Sloman e Moffett 1989]. Os mapas de rede utilizados por estes sistemas são exemplos bastante comuns do uso de domínios. Estes mapas, geralmente, agrupam os dispositivos que pertencem a um mesmo segmento de rede, mas poderiam agrupar os dispositivos segundo outros aspectos. Por exemplo, domínios podem agrupar dispositivos que possuem funcionalidades ou características semelhantes, como servidores de correio eletrônico ou roteadores. Domínios são importantes porque as ações de gerenciamento podem ser aplicadas a todos os dispositivos membros de um domínio ao mesmo tempo. As ações de gerenciamento aplicadas aos domínios são automaticamente repassadas aos membros do mesmo, não sendo necessário, então, repetir a mesma ação em cada dispositivo gerenciável um a um.

A criação de domínios pode ser realizada com o auxílio de algumas facilidades. Por exemplo, nos sistemas de gerenciamento padrão os mapas de rede que contém os dispositivos, enlaces e sub-redes da rede gerenciada, podem ser gerados através de mecanismos de descoberta de topologia. Por outro lado, o administrador de rede pode também criar os seus próprios domínios de forma manual, por exemplo, arrastando os dispositivos gerenciáveis desejados para novos mapas através de facilidades de interface de usuário. Após serem criados, os domínios normalmente são armazenados em bases de dados para uso posterior. Porém, existem algumas situações em que determinados domínios precisam ser rapidamente criados, utilizados e descartados. Por exemplo, em um sistema de gerenciamento de QoS baseado em políticas (*Policy-Based Network Management* - PBNM) [Sloman 1994] um administrador de redes deveria executar as seguintes ações de forma a priorizar o tráfego de uma sessão de videoconferência entre os *hosts* A e B, sendo os roteadores intermediários desconhecidos:

1. Criar uma política que define a QoS esperada para a sessão em questão, e armazenar a política criada em um repositório de políticas;
2. Descobrir os roteadores entre A e B. Os roteadores identificados devem ser adicionados ao domínio da sessão citada. Quando o último roteador entre A e B for descoberto, o caminho inverso, de B para A, deve também ser verificado, já que os caminhos de ida e volta podem ser diferentes. Todos os novos roteadores descobertos devem ser adicionados ao domínio da sessão;
3. Aplicar a política de QoS, armazenada anteriormente, no domínio da sessão, ao invés de aplicar a política em cada roteador da sessão entre A e B um a um;
4. Descartar o domínio da sessão após o término da videoconferência e o fechamento da sessão entre A e B.

O cenário apresentado demonstra duas características interessantes de domínios:

- Algumas vezes, a criação de domínios pode ser lenta, principalmente quando é feita manualmente pelos administradores de rede;
- Alguns domínios têm tempo de vida pequeno, pois são utilizados para um determinado propósito e depois são descartados.

Com relação à primeira característica, o tempo necessário para a criação de um domínio deve ser menor que o tempo de vida deste mesmo domínio. Por exemplo, a criação do domínio da sessão da videoconferência comentada anteriormente deve ser mais rápida que o tempo de vida da videoconferência, senão a política de QoS definida e armazenada será aplicada nos roteadores após o término da videoconferência. Domínios que precisam ser rapidamente criados, utilizados e descartados serão referenciados neste trabalho através do termo “domínios dinâmicos”.

Em relação aos aspectos de visualização, a apresentação visual de domínios deve ser realizada de forma adequada, visto que, atualmente, as GUIs estão presentes na maioria dos sistemas de gerenciamento. Geralmente, mapas visuais são usados na apresentação de dispositivos, enlaces e sub-redes. Entretanto, outras formas visuais de apresentação de domínios também podem ser utilizadas, como por exemplo, tabelas, gráficos 3D ou matrizes. Os processos de visualização de domínios utilizados pelos sistemas de gerenciamento atuais apresentam limitações em relação à configuração de

características visuais dos domínios apresentados. Estas características são estáticas, não permitindo ao usuário, desta forma, escolher como deseja visualizar um determinado domínio. Por exemplo, os dispositivos que apresentam problemas, geralmente são destacados nos mapas de rede com a cor vermelha. Porém, o usuário não consegue alterar esta cor ou escolher uma outra cor para destacar os dispositivos que possuem suporte a *DiffServ* (*Differentiated Services*) [Bernet et al. 2002], por exemplo.

Neste contexto, dois principais problemas podem ser identificados: aplicações de gerenciamento de redes convencionais possuem pouco ou nenhum suporte ao conceito de domínios dinâmicos e; o suporte fornecido à visualização de domínios é restrito a poucas opções de configuração. Assim, neste artigo são apresentadas duas novas linguagens que tem por objetivo a criação e a visualização de domínios dinâmicos. A primeira linguagem é utilizada para criar, de forma automatizada, novos domínios dinâmicos. A segunda linguagem, por sua vez, é usada para configurar características visuais dos domínios dinâmicos criados através da primeira linguagem. O artigo apresenta, também, um protótipo desenvolvido para suportar as linguagens definidas. Este protótipo é baseado na Web e foi desenvolvido utilizando-se PRECCX, PHP4, MySQL e SNMP.

O restante do artigo está assim dividido: na seção 2 são apresentados os trabalhos relacionados. A seção 3 apresenta o ambiente gerenciado e o modelo de informação utilizados no desenvolvimento do trabalho, e nos quais as linguagens de criação e visualização de domínios dinâmicos apresentadas na seção 4 são aplicadas. Na seção 5 é apresentado o protótipo desenvolvido para validar as linguagens definidas. Por fim, na seção 6, são apresentadas as conclusões e os trabalhos futuros.

2. Domínios e Trabalhos Relacionados

Objetos gerenciáveis são abstrações de recursos que podem ser gerenciados em um sistema de gerenciamento de redes de computadores. Interfaces de rede, tabelas de roteamento e processos de conformação de tráfego são exemplos de objetos gerenciáveis. Estes objetos podem ser inspecionados através de operações de leitura (por exemplo, a obtenção da tabela de roteamento de um determinado roteador) ou configurados através de operações de escrita (por exemplo, a configuração do processo de conformação de um roteador de borda). Objetos gerenciáveis tornam-se membros de um domínio ao serem agrupados pelo mesmo. O domínio que agrupa objetos é chamado de domínio pai de tais objetos. Como domínios também são objetos gerenciáveis e podem fazer parte de outros domínios, acaba sendo possível a existência de uma hierarquia de domínios [Sloman e Moffett 1989]. Um domínio, ao ser membro de um outro domínio, é chamado de subdomínio do domínio pai. Sob o ponto de vista do usuário, domínios contêm os objetos gerenciáveis que agrupam. No entanto, domínios possuem apenas referências a tais objetos. Como conseqüência, um objeto gerenciável pode ser membro de diversos domínios ao mesmo tempo, possuindo assim, diferentes domínios pai.

Diversas pesquisas relacionadas a domínios têm sido desenvolvidas ao longo dos anos. Sloman et al, além de introduzir os conceitos básicos de domínios apresentados anteriormente [Sloman e Moffett 1989], demonstrou também como domínios podem ser utilizados no gerenciamento de redes celulares [Sloman et al. 1993] e aplicados no gerenciamento baseado em políticas (PBNM) [Daminou et al. 2002]. Kar, Keller e Calo

utilizaram a noção de domínios no gerenciamento fim-a-fim de serviços de hospedagem de aplicações fornecidos por ISPs [Kar et al. 2000]. Já Miranda, Nogueira e Machado usaram domínios no gerenciamento de redes de telecomunicações [Miranda et al. 2002].

Normalmente, domínios são armazenados em repositórios para serem utilizados posteriormente. Entretanto, existem algumas situações em que domínios tornam-se obsoletos muito rapidamente, fazendo com que o seu armazenamento seja desnecessário. Outro aspecto importante de domínios está relacionado ao tempo necessário para criá-los. Caso o tempo gasto para identificar e adicionar os objetos desejados a um domínio seja elevado, este domínio pode tornar-se obsoleto antes mesmo de ser utilizado. Os trabalhos citados anteriormente são baseados no armazenamento de domínios em repositórios e, também, na criação de domínios através de processos lentos, como por exemplo, a inclusão manual de objetos em mapas de rede, estruturas semelhantes a diretórios ou em árvores hiperbólicas [Daminou et al. 2002]. Neste contexto, é possível verificar que as soluções atuais possuem pouco ou nenhum suporte ao conceito de domínios dinâmicos.

Além de serem importantes porque manipulam e agrupam os objetos gerenciáveis, os domínios também devem ser apresentados pelos sistemas de gerenciamento de redes de forma adequada. A visualização de um domínio depende da natureza de seus membros. Mapas de rede, por exemplo, podem ser visualizados de diversas maneiras. Mapas 2D, como os utilizados pelos *softwares* SNMPc [CastleRock Computing 2002] e HP OpenView [Hewlard Packard 2002], podem não ser uma solução satisfatória para visualizar segmentos de rede com número muito elevado de dispositivos e enlaces [Eick 1996]. Mapas em 3D, utilizados em *softwares* como CA Unicenter [Strum 1998] e Mapnet [CAIDA Web Site 2002], podem solucionar o problema da aglomeração de dispositivos e enlaces. Entretanto, os administradores de rede possuem preferência por mapas 2D, pois os mesmos são mais fáceis de serem manipulados. Árvores hiperbólicas, propostas recentemente por Damianou et al. [Daminou et al. 2002], são de fácil manipulação e fornecem visualizações adequadas de domínios.

De modo geral, as facilidades existentes relacionadas à visualização de domínios são limitadas de alguma forma. Ainda que alguns *softwares* possuam ricos conjuntos de opções associadas à personalização das características visuais dos domínios apresentados, o administrador de rede não tem a possibilidade de escolher uma alternativa distinta das impostas por estes *softwares*. Por exemplo, no *software* Mapnet [CAIDA Web Site 2002] o administrador de rede tem a opção de escolher entre dois tipos diferentes de coloração a serem usados nos nodos e enlaces dos *backbones* visualizados: por ISPs ou por largura de banda. Porém, ele não tem a possibilidade de escolher as cores utilizadas em cada uma destas opções, pois as cores são definidas pelo *software*.

Com o objetivo de solucionar os problemas apresentados acima, nas duas próximas seções serão apresentadas duas novas linguagens: uma linguagem para a criação de domínios dinâmicos, e outra linguagem para a configuração das características visuais dos domínios que forem criados através da primeira linguagem.

3. Ambiente Gerenciado e Modelo de Informação para Domínios Dinâmicos

Administradores de rede esperam obter uma visão da rede gerenciada ao acessar um sistema de gerenciamento. Esta visão é composta por um conjunto de objetos gerenciáveis. Em alguns casos, diversas visões da mesma rede podem coexistir em um mesmo sistema de gerenciamento e cada uma destas visões pode ser acessada por diferentes administradores [Palma e Rodrigues 2001]. Os objetos gerenciáveis, além de poderem ser membros de uma visão, podem também fazer parte de domínios (e.g. mapas de rede). A utilização de domínios facilita o acesso e a manipulação dos objetos gerenciáveis. Estes objetos podem estar localizados em dispositivos de rede (e.g. interfaces de um roteador), distribuídos pela rede (e.g. sessões RSVP) ou em um repositório (e.g. registros de usuários da rede). Além disso, o acesso a diferentes objetos gerenciáveis requer a utilização de protocolos distintos, como por exemplo, SNMP – *Simple Network Management Protocol* [Case et al. 1990] (e.g. acesso às interfaces de um roteador), COPS – *Common Open Policy Services* [Durham et al. 2000] (e.g. acesso à sessões RSVP) ou LDAP – *Lightweight Directory Access Protocol* [Wahl et al. 1997] (e.g. acesso aos registros de usuários da rede).

O gerenciamento de uma rede com todas estas complexidades e o fornecimento de facilidades para a criação de domínios dinâmicos requer a definição de três elementos distintos:

- O ambiente gerenciado no qual os domínios dinâmicos serão criados e utilizados;
- O modelo de informação responsável por descrever como as informações do ambiente gerenciado estão organizadas;
- A linguagem utilizada para criar novos domínios dinâmicos de acordo com as informações disponíveis no ambiente gerenciado.

A próxima subseção apresenta o ambiente gerenciado e o modelo de informação que foram utilizados durante a criação da linguagem de criação de domínios dinâmicos. A seção 4, por sua vez, abordará as linguagens em si.

3.1. Ambiente Gerenciado

O primeiro passo para definir uma linguagem de criação de domínios dinâmicos é descrever o ambiente de rede utilizado. Tal ambiente é apresentado na figura 1. Inicialmente, este ambiente foi considerado como um conjunto de dispositivos heterogêneos. Além destes dispositivos, alguns serviços também foram levados em conta, como por exemplo, os serviços de diretórios (e.g. LDAP) e de rede (e.g. DNS) apresentados na figura 1 à esquerda. Acima de todos estes elementos está localizada a estação de gerenciamento de rede (NMS). Neste exemplo, apenas uma estação central de gerenciamento foi considerada. Entretanto, em um ambiente de gerenciamento distribuído, outros elementos de gerência poderiam ser usados, como por exemplo, MLMs (*Mid-Level Managers*), BBs (*Bandwith Brokers*) e PDPs (*Policy Decision Points*).

Visto que os recursos de rede (*host*, roteadores, diretórios, etc.) são heterogêneos, o acesso aos mesmos requer o uso de diferentes protocolos. Porém, os diferentes métodos de acesso existentes não devem contribuir para o aumento da

complexidade das atividades relacionadas ao gerenciamento. Conseqüentemente, definiu-se que os recursos de rede, apesar de possuírem diferentes métodos de acesso, são "vistos" pela estação de gerenciamento de uma forma padronizada. Para possibilitar isto, a estação de gerenciamento possui uma camada intermediária de *softwares* implementados de forma a abstrair os detalhes de acesso aos objetos gerenciáveis (figura 1 à direita). Mais precisamente, os detalhes de baixo nível dos objetos gerenciáveis constituem o modelo de dados (definidos, por exemplo, através do SMIv1, SMIv2 [McCloghrie et al. 1999] ou SPPI [McCloghrie et al. 2001]), enquanto que os detalhes de alto nível destes objetos compõem o modelo de informação de gerenciamento. Este modelo é responsável por abstrair os detalhes do modelo de dados. Uma explicação completa sobre as diferenças entre o modelo de dados e o modelo de informação, no contexto do gerenciamento de redes, pode ser verificada em um *draft* do IETF desenvolvido pelo Grupo de Pesquisa de Gerenciamento de Redes (*Network Management Research Group* - NMRG) [Internet Research Task Force 2002] do IRTF [Pras e Shoenwaelder 2002].

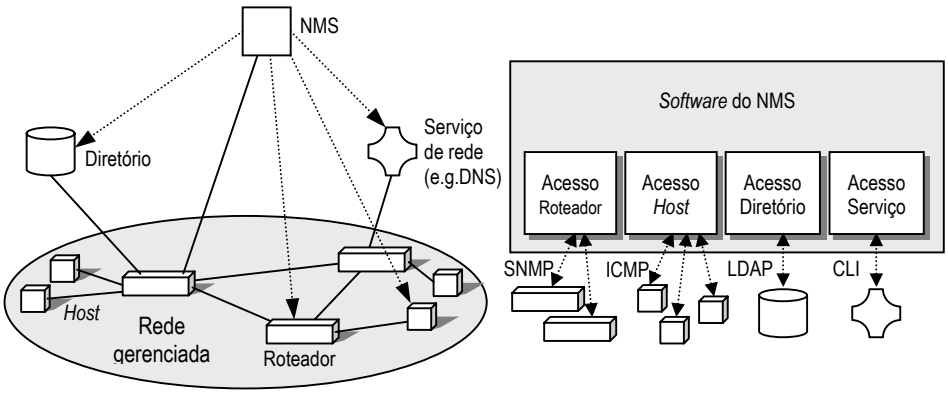

Figura 1. Ambiente gerenciado

O *software* da estação de gerenciamento responsável por manipular o modelo de informação dos objetos gerenciáveis e domínios é apresentado na figura 1 à direita. A próxima subseção aborda o modelo de informação utilizado.

3.2. Modelo de Informação

As informações do ambiente gerenciado manipuladas pelo *software* apresentado na figura 1 estão organizadas em um modelo de informação. A figura 2 apresenta, em notação UML, o modelo utilizado no desenvolvimento deste trabalho. Este modelo é formado por classes (e.g. *topologies* e *devices*) que possuem atributos (e.g. o atributo *owner* da classe *topologies* e o atributo ip da classe *devices*) e por relacionamentos com cardinalidades entre as classes. Neste modelo, a classe *topologies* é responsável por armazenar informações das topologias do ambiente gerenciado. Cada topologia possui um conjunto de dispositivos e cada dispositivo pode ser membro de várias topologias. As informações dos dispositivos do ambiente gerenciado são armazenadas na classe *devices*. Os dispositivos podem possuir diversas interfaces de rede, porém, uma determinada interface pode pertencer a somente um dispositivo. As informações das interfaces do ambiente gerenciado são armazenadas na classe *interfaces*. Cada interface, por sua vez, pode possuir vários *token buckets DiffServ,* vários *droppers DiffServ* e

várias sessões RSVP. Cada *token bucket* e cada *dropper* podem estar associados à somente uma interface, enquanto que, cada sessão RSVP pode estar associada a diferentes interfaces de dispositivos distintos.

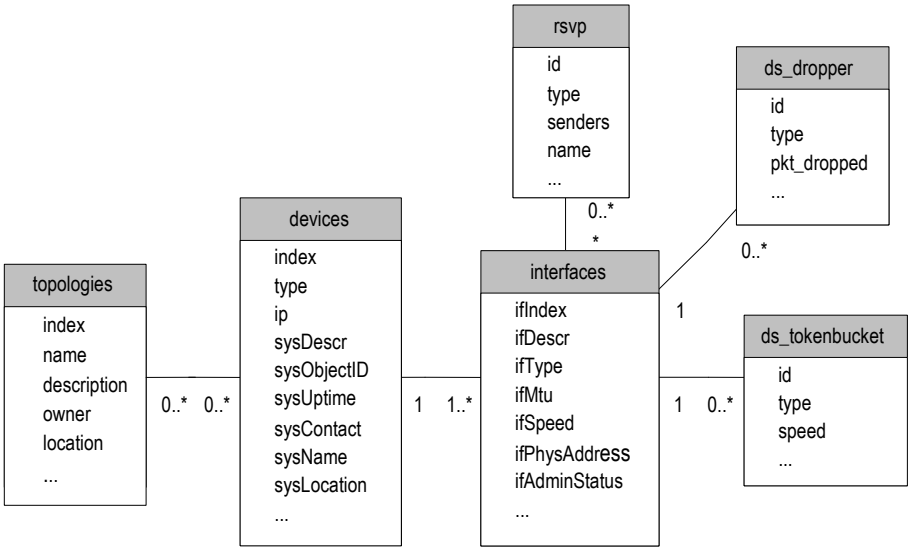

Figura 2. Modelo de informação

Este modelo de informação, utilizado de forma auxiliar na criação das linguagens, obviamente só representa um conjunto limitado de redes de computadores, pois a quantidade de informações de gerenciamento é reduzida. No entanto, para possibilitar que a linguagem de criação de domínios dinâmicos não fique limitada a operar em uma única rede de computadores, assumiu-se que o modelo é dinâmico e pode ser estendido para acomodar novos objetos gerenciáveis. Por exemplo, se uma nova MIB for compilada no ambiente de gerência, uma nova classe pode ser adicionada ao modelo de informação para que os objetos gerenciáveis descritos em tal MIB sejam suportados. Além disso, classes existentes no modelo também podem ser removidas. O modo como este modelo de informação dinâmico foi implementado é comentado na seção 5. A seguir, é apresentada a forma como as linguagens propostas consultam as informações armazenadas no modelo de informação para criar e visualizar novos domínios dinâmicos.

4. Linguagens de Criação e Visualização de Domínios Dinâmicos

Considerando o modelo de informações e o ambiente gerenciado apresentados na seção anterior, apresentamos aqui as duas linguagens para criação e visualização de domínios dinâmicos.

4.1. Linguagem de Criação de Domínios Dinâmicos

As seguintes características foram consideradas na definição da linguagem de criação de domínios dinâmicos apresentada nesta subseção.

- A linguagem deve operar de acordo com um modelo de informação dinâmico, no qual novas classes possam ser adicionadas e outras classes possam ser removidas;

- A linguagem deve possuir uma sintaxe simples e fácil de usar, pois os administradores de rede não estão habituados a usar linguagens, por exemplo, como o SQL;
- A linguagem deve fornecer um mecanismo capaz de selecionar objetos gerenciáveis e agrupá-los em domínios dinâmicos. A manipulação destes objetos através de operações de escrita não deve fazer parte do escopo da linguagem, e deve ser realizada por facilidades externas à linguagem.

Assim, a definição de domínios dinâmicos é realizada a partir da elaboração de expressões de seleção que seguem a seguinte BNF (*Backus-Naur Form*):

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

Nesta BNF, o elemento `class` é utilizado para identificar uma classe de um modelo de informação (por exemplo, o modelo de informação apresentado na figura 2), elemento `attribute` é utilizado para identificar um atributo de uma classe de um modelo de informação e o elemento `value` é utilizado para selecionar os objetos (ocorrências) de uma classe que possuem um determinado atributo com um valor específico.

A linguagem de criação de domínios dinâmicos definida por meio desta BNF não é limitada a operar de acordo com um conjunto específico de classes e atributos. Conseqüentemente, a linguagem pode ser utilizada com diferentes modelos de informação. Para melhor compreensão da linguagem, alguns exemplos serão apresentados a seguir, considerando o modelo de informação abordado na figura 2.

```
1) select topologies
2) select topologies.owner["secAdm"]
3) select topologies.owner["secAdm"].location["admBuilding"]
```

A expressão de seleção apresentada no exemplo 1 criará um domínio dinâmico formado por todas as topologias existentes no ambiente gerenciado. Na segunda expressão apresentada, o domínio dinâmico criado será formado somente pelas topologias gerenciadas pelo administrador de segurança (`secAdm`). A expressão 3, por sua vez, selecionará apenas as topologias gerenciadas pelo administrador de segurança que estiverem localizadas no prédio da administração (`admBuilding`).

Na implementação atual, os domínios dinâmicos criados através desta linguagem são formados por conjuntos de valores dos atributos que são identificadores únicos (oid) dos objetos selecionados de uma classe. Por exemplo, nas expressões de seleção 1, 2 e 3, os domínios dinâmicos serão formados por conjuntos de valores do atributo `index` da classe `topologies`. As expressões apresentadas a seguir demonstram o uso do conector "`::`". Este conector permite que atributos de uma classe sejam utilizados para selecionar objetos de uma outra classe.

```
4) select topologies::devices.type["DSRouter"]
5) select topologies::devices::interfaces.ifType["ATM"]
6) select topologies::devices["DSRouter"]::interfaces["ATM"]
```

A expressão de seleção apresentada no exemplo 4 criará um domínio dinâmico formado por todas as topologias do ambiente gerenciado que possuírem dispositivos do tipo roteador *DiffServ* (`DSRouter`). Na expressão 5, o domínio dinâmico criado será formado por todas as topologias que possuírem dispositivos com interfaces de rede do tipo ATM.

As classes do modelo de informação devem possuir atributos padrão. Um atributo padrão é utilizado quando expressões semelhantes à apresentada no exemplo 6 são avaliadas. Nesta expressão de seleção, o domínio dinâmico criado será formado por todas as topologias que possuírem dispositivos do tipo roteador *DiffServ* com interfaces do tipo ATM. Esta expressão possui dois atributos padrão: o atributo `type` da classe `devices` e o atributo `ifType` da classe `interfaces`.

A primeira classe após a cláusula `select` identifica a classe responsável por fornecer os valores (identificadores únicos) que serão armazenados no domínio dinâmico criado. A cláusula `from` é utilizada para alterar a localização da primeira classe de uma expressão de seleção sem removê-la desta expressão.

```
7) select devices from topologies["admBuilding"]
8) select interfaces["ATM"] from devices.sysName["CampusGateway"]
```

Na expressão de seleção 7, o domínio dinâmico criado será formado por todos os dispositivos das topologias localizadas no prédio da administração (neste exemplo, `location` é o atributo padrão da classe `topologies`). Já a expressão de seleção do exemplo 8, criará um domínio dinâmico formado por todas as interfaces de rede do tipo ATM dos dispositivos do *gateway* da universidade (`CapusGatway`).

Domínios mais complexos podem ser criados através da elaboração de expressões de seleção que combinem os elementos `from`, "`::`" e "`.`". Após serem criados, os domínios dinâmicos são armazenados temporariamente em uma área de memória para que possam ser utilizados pela linguagem de visualização (apresentada na próxima subseção). Assim que forem utilizados, os domínios são descartados e a área de memória é liberada.

4.2. Linguagem de Visualização de Domínios Dinâmicos

O objetivo da linguagem apresentada nesta subseção é possibilitar visualizações personalizadas dos domínios criados através da linguagem de criação de domínios dinâmicos. O processo de personalização da visualização destes domínios é baseado na configuração de atributos da facilidade visual (e.g. mapas, gráficos, tabelas) utilizada. A definição de uma nova visualização personalizada deve seguir os seguintes passos:

- Criação dos domínios dinâmicos desejados através da linguagem de criação de domínios dinâmicos;
- Escolha da facilidade visual responsável por apresentar os domínios dinâmicos criados. Uma facilidade visual é o recurso utilizado para gerar uma determinada apresentação gráfica. Tabelas, mapas topológicos, gráficos e matrizes são exemplos de tais facilidades. Espera-se que o ambiente gerenciado possua um conjunto de facilidades visuais disponíveis, e que o administrador de rede precise apenas especificar qual destas facilidades será utilizada para apresentar os domínios criados por ele;
- Escolha, entre os domínios dinâmicos criados, daqueles que serão apresentados através da facilidade visual escolhida. Normalmente, apenas um domínio é apresentado através de uma facilidade visual. Entretanto, vários domínios podem ser apresentados ao mesmo tempo através da mesma visualização;
- Configuração dos atributos da facilidade visual escolhida conforme os domínios dinâmicos criados.

Os últimos três passos apresentados são suportados pela linguagem de visualização de domínios dinâmicos. Esta linguagem é definida conforme a seguinte BNF:

```
visualization ::= [selection] show dmview {and dmview}
selection     ::= using visualfacility [customization]
dmview        ::= domain [customization]
customization ::= with attribute=value {, attribute=value}
```

Nesta BNF, o elemento `domain` representa um domínio criado com a linguagem de criação de domínios dinâmicos, o elemento `visualfacility` indica a facilidade visual a ser usada para apresentar os domínios dinâmicos, o elemento `attribute` identifica um determinado atributo da facilidade visual escolhida e o elemento `value` associa um valor específico ao atributo identificado pelo elemento `attribute`.

Primeiramente, para explicar a linguagem de visualização de domínios dinâmicos através de exemplos, alguns domínios precisam ser definidos.

```
DiffServRouters = select devices.type["DSRouter"]
IntServRouters  = select devices.type["ISRouter"]
RSVP33Routers   = select devices.type["Router"]::interfaces::
                                                  rsvp.index[33]
```

Na linguagem de visualização de domínios dinâmicos, a cláusula `using` é utilizada para escolher a facilidade visual desejada, enquanto que, a cláusula `show` é usada para indicar os domínios dinâmicos que serão apresentados.

```
1) using table show DiffServRouters
2) using topology show IntServRouters and RSVP33Routers
```

As expressões de visualização apresentadas acima utilizam duas facilidades visuais distintas: `table` e `topology`. Na primeira expressão, a facilidade `table` é utilizada para apresentar visualmente os membros do domínio `DiffServRouters` através de uma tabela. Já a segunda expressão utiliza a facilidade `topology` para apresentar os membros dos domínios `IntServRouters` e `RSVP33Routers` através de um mapa topológico. A cláusula `and` utilizada no exemplo 2 é necessária para que se possa apresentar mais de um domínio (e.g. `IntServRouters` e `RSVP33Routers`) na mesma visualização. As facilidades visuais não suportam a apresentação de domínios que possuam objetos de classes distintas. Para que diferentes domínios possam ser apresentados ao mesmo tempo em uma visualização, os mesmos devem possuir elementos da mesma classe do modelo de informação utilizado. Por exemplo, diferentes domínios formados por dispositivos podem ser apresentados através da facilidade `topology`. Entretanto, domínios distintos formados por dispositivos e por interfaces não podem ser apresentados através da facilidade `table`.

As facilidades visuais possuem atributos próprios que podem ser configurados para modificar algumas características gráficas das visualizações que serão apresentadas. A habilidade para configurar estes atributos é a principal característica da linguagem de visualização de domínios dinâmicos. A cláusula `with` desta linguagem é o elemento responsável por possibilitar a personalização das visualizações geradas por uma facilidade visual. Os atributos configurados por esta cláusula podem ser aplicados a todos os domínios de uma visualização como um todo ou a cada domínio em separado.

```
3) using table with cellcolor=blue show IntServRouters and
                                                RSVP33Routers

4) using table show IntServRouters with cellcolor=yellow and
                          DiffServRouters with cellcolor=green

5) using table with cellcolor=blue show RSVP33Routers and
                          IntServRouters with cellcolor=yellow and
                          DiffServRouters with
                          cellcolor=green
```

A expressão do exemplo 3 apresentará os membros dos domínios `IntServRouters` e `RSVP33Routers` em uma tabela com as células azuis. No entanto, na expressão do exemplo 4, os membros do domínio `IntServRouters` serão apresentados em células amarelas, enquanto que, os membros do domínio `DiffServRouters` serão apresentados em células verdes. A cláusula `with` quando situada após um domínio específico é utilizada para configurar atributos que sejam válidos somente para tal domínio (como demonstrado no exemplo 4). Este tipo de configuração sobrepõe as configurações globais (como demonstrado no exemplo 3). Por exemplo, na expressão 5, embora a cor das células da tabela tenham sido configuradas para azul, os membros do domínio `IntServRouters` são apresentados em células amarelas e os membros do domínio `DiffServRouters` são exibidos em células verdes.

```
6) show IntServRouters and DiffServRouters
```

As facilidades visuais disponíveis em um ambiente gerenciado possuem vários atributos (por exemplo, o atributo `cellcolor` da facilidade `table`). Cada um destes atributos possui um valor padrão associado que será utilizado nas visualizações se

outros valores não forem atribuídos através da linguagem de visualização. De maneira similar, cada classe do modelo de informação utilizado possui uma facilidade visual padrão. Como conseqüência, a cláusula `using` responsável por indicar a facilidade visual utilizada para apresentar determinados domínios pode ser omitida da expressão de visualização. Por exemplo, assumindo que `topology` é a facilidade visual padrão da classe `devices`, a expressão demonstrada no exemplo 6 apresentará os membros dos domínios `IntServRouters` e `DiffServRouters` através de um mapa topológico. A próxima seção apresenta como as linguagens de criação e visualização de domínios dinâmicos são suportadas em um protótipo desenvolvido.

5. Implementação

O suporte às linguagens de criação e visualização de domínios dinâmicos foi implementado como um subsistema do ambiente QAME [Granville et al. 2001]. As próximas duas subseções apresentam como cada linguagem é suportada no QAME.

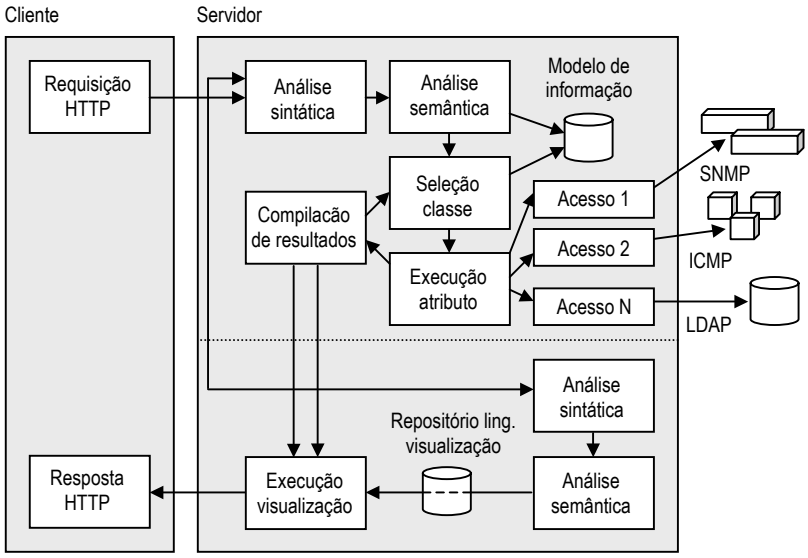

Figura 3. Passos para a criação e visualização de domínios dinâmicos

5.1. Suporte à Criação de Domínios Dinâmicos

As expressões de criação de novos domínios dinâmicos são passadas para o protótipo desenvolvido a partir de um conjunto de caixas de texto de um formulário HTML do ambiente QAME. As expressões de visualização de domínios dinâmicos, por sua vez, são passadas para este protótipo através de uma única caixa de texto do mesmo formulário HTML (o suporte as expressões de visualização é apresentado na próxima subseção). Um *script* PHP4 é então executado para avaliar as expressões passadas e criar os domínios solicitados. Este processo é realizado de acordo com os passos apresentados na parte superior da figura 3.

Neste exemplo, uma única requisição HTTP envia ao servidor duas expressões de criação e uma expressão de visualização de domínios dinâmicos. Para cada expressão de criação de domínios recebida pelo servidor, os seguintes passos são executados. Primeiramente, a análise sintática da expressão é executada. Esta análise é realizada através de um programa escrito na linguagem C gerado pelo *software* PRECCX [Breuer

e Bowen 1993]. Após, o processo responsável por executar a análise semântica acessa a descrição do modelo de informação para verificar se a expressão é valida. Na implementação atual, a descrição do modelo de informação está armazenada em uma base dados MySQL. Caso a sintaxe utilizada esteja correta e a expressão esteja consistente com o modelo de informação armazenado na base MySQL, o ciclo de execução é iniciado.

Classe	Atributo	Script	Função
topologies	owner	topologies.php	get_owner
topologies	location	topologies.php	get_location
...	...	...	...
devices	type	devices.php	get_type
devices	sysName	devices.php	get_sysName
devices	sysLocation	devices.php	get_sysLocation
...	...	...	...
interfaces	ifType	interfaces.php	get_ifType
...	...	...	...

Figura 4. Informações complementares armazenadas junto ao modelo de informação

Os passos seleção de classe e execução de atributo apresentados na figura 3 são responsáveis por definir os *drivers* de acesso que serão utilizados no processamento de uma expressão de criação. Estas escolhas são realizadas a partir do acesso à tabela de informações de *drivers* do ambiente gerenciado. Um *driver* de acesso é uma função de um *script* PHP4 que acessa os objetos gerenciáveis necessários através do protocolo apropriado e busca os valores solicitados. Por exemplo, considerando a tabela de informações de *drivers* da figura 4, é possível notar que no processamento da expressão `devices.sysLocation["LabCom"]`, o arquivo PHP4 `devices.php` será acessado e a função `get_sysLocation` será invocada. O código simplificado da função `get_sysLocation` (o código atual é mais complexo) é apresentado na figura 5. É importante mencionar que nesta função o protocolo SNMP foi utilizado para obter-se a localização dos objetos gerenciáveis. Porém, outros protocolos podem ser utilizados, pois isto depende somente da implementação da função de acesso utilizada.

```
01 list get_sysLocation (list devices, string value) {
02    list ret, string tmp, int i;
03    if (!devices) devices=AllDevices;
04    for(i=0; i<devices.number; i++) {
05       tmp = snmpget (devices[j], "sysLocation");
06       if(tmp == value) ret.add (tmp);
07    }
08    return ret;
09 }
```

Figura 5. Código simplificado da função `get_sysLocation`

Os passos seleção de classe e execução de atributo (parte superior da figura 3) são aplicados a cada segmento de uma expressão passada. Os segmentos de uma expressão são separados pelo conector "`::`" ou pela cláusula `from`. Por exemplo, a expressão `select devices["DSRouter"]::interfaces["ATM"] from topologies["LabCom"]` possui três segmentos. O segmento `devices["DSRouter"]`

retorna todos os roteadores *DiffServ*. O segmento interfaces, por sua vez, fornece todas as interfaces de rede ATM e o segmento `topologies["LabCom"]` obtém todas as topologias localizadas no "LabCom". Cada segmento de uma expressão é avaliado em separado e retorna um resultado intermediário. Visto que, uma expressão pode possuir vários segmentos, diversos resultados intermediários serão retornados. No entanto, o passo compilação de resultados apresentado na figura 3 processa os resultados intermediários toda vez que um novo resultado é retornado e prossegue então para uma nova seleção de classe e execução de atributo. Após, o último segmento de uma expressão ser avaliado, o resultado intermediário retornado é computado em um resultado final. Este resultado é então enviado para a facilidade visual em questão. Por exemplo, o resultado final da expressão apresentada acima será o conjunto dos roteadores *DiffServ* localizados no "LabCom" que possuem interfaces ATM. De acordo com o exemplo da figura 3, dois domínios dinâmicos serão passados para a facilidade visual. A avaliação da expressão de visualização é detalhada na próxima subseção.

5.2. Suporte à Visualização de Domínios Dinâmicos

Na parte inferior da figura 3 são apresentados os passos para avaliar e executar uma expressão de visualização. Primeiramente, a análise sintática da expressão passada é realizada. Esta análise também é feita através de um programa escrito na linguagem C gerado pelo software PRECCX. Após, a análise semântica desta expressão é executada. Esta análise é realizada a partir do acesso ao repositório de informações das facilidades visuais. Este repositório descreve as facilidades visuais disponíveis no ambiente gerenciado e os atributos que cada facilidade suporta.

Assim que a expressão de visualização for validada, o processo responsável por executar a análise semântica invoca a facilidade visual indicada nesta expressão. Cada facilidade visual disponível no ambiente gerenciado foi implementada em um *script* PHP4. Estes *scripts* são responsáveis por apresentar graficamente os domínios dinâmicos criados através dos processos apresentados na parte superior da figura 3. Após a execução da expressão, a facilidade visual envia a visualização dos domínios criados para o navegador Web do usuário. Além das páginas HTML padrão, cada facilidade visual, obviamente, pode utilizar outras técnicas de apresentação gráfica e outros recursos. Por exemplo, na implementação atual, a facilidade table gera tabelas em HTML. Porém, a facilidade topologies gera mapas topológicos na tecnologia FLASH.

```
dev1= select devices.sysName["noc"] from topologies["LabCom"]
dev2= select devices from topologies["LabCom"]
```

Os resultados gerados a partir da execução das expressões acima são apresentados na figura 6 à esquerda através da facilidade table (`using table with columns={type, ip, sysName, sysLocation} show dev1 with cellcolor=gray and dev2`). A figura 6 à direita apresenta os mesmos resultados, porém, através da facilidade topology (`using topology with opaque=0.5 show dev1 with opaque=1 and dev2`).

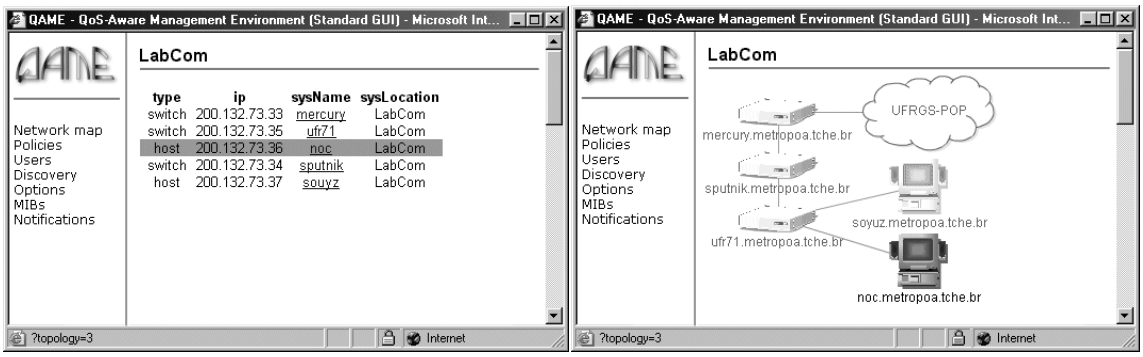

Figure 6. Exemplos da visualização de domínios dinâmicos

6. Conclusão e Trabalhos Futuros

Neste artigo foram apresentadas duas linguagens: uma linguagem para a criação de domínios dinâmicos e outra linguagem para visualização dos domínios criados através da primeira linguagem. A implementação de um subsistema para suportar as linguagens no ambiente QAME também foi apresentado. Além disso, no decorrer do texto foram demonstrados exemplos práticos da utilização das linguagens definidas.

Acredita-se que objetivo principal deste trabalho foi alcançado, visto que, um suporte inicial a domínios dinâmicos foi desenvolvido. Com relação ao processo de desenvolvimento do protótipo, outras contribuições foram constatadas. Primeiramente, como a criação de domínios dinâmicos é baseada em uma visão comum dos objetos gerenciáveis, foi implementado um único e simples modelo de informação ao invés de diferentes modelos de dados. A obtenção das informações organizadas no modelo de informação é fornecida pela utilização dos *drivers* de acesso. Na implementação atual, estes *drivers* são funções em scripts PHP4. Além de possibilitar a operação adequada da linguagem de criação de domínios dinâmicos, o modelo de informação desenvolvido faz com que o administrador de rede não precise se preocupar com os protocolos específicos necessários para acessar os objetos gerenciáveis desejados.

O suporte as linguagens de criação e visualização de domínios dinâmicos foi implementado no sistema QAME. Este sistema tem por objetivo o gerenciamento de redes com suporte a QoS. Uma operação normal neste ambiente é a seleção de dispositivos, nos quais, políticas de rede devem ser aplicadas. As linguagens de criação e visualização, no contexto do QAME, facilitaram o processo de aplicação de políticas.

O desenvolvimento do protótipo apresentado neste artigo ainda continua. Visto que, a pesquisa está voltada para domínios dinâmicos, facilidades de armazenamento devem estar também disponíveis (neste caso, os domínios dinâmicos irão operar como domínios padrão). Como trabalhos futuros, as facilidades de armazenamento devem gravar não somente os domínios criados a partir da linguagem de criação, mas também as expressões das linguagens de criação e visualização de domínios dinâmicos, e a visualização final fornecida. Por fim, acreditamos que embora as linguagens baseadas em texto sejam fáceis de usar, a utilização de assistentes gráficos para auxiliar o usuário na elaboração das expressões aumentariam a habilidade de criação e visualização dos domínios dinâmicos.

Referências

Bernet, Y., Blake, S., Grossman, D. and Smith, A. (2002) "An Informal Management Model for DiffServ Routers", IETF RFC 3290.

Breuer, P. and Bowen, J. (1993) "The PRECC Compiler-Compiler", In E. Davies and A. Findlay (eds). Proc. UKUUG/SUKUG Joint New Year.

CAIDA Web Site. (2002) "Mapnet", http://www.caida.org.

Case, J., Fedor, M. Schoffstal, M. and Davin, J. (1990) "A Simple Network Management Protocol (SNMP)", IETF RFC 2205.

CastleRock Computing. (2002) "SNMPc", http://www.snmpc.co.uk.

Daminou, N., Dulay, N., Lupu, E., Sloman, M. and Tonouchi, T. (2002) "Tools for Domain-based Policy Management of Distributed Systems", IEEE/IFIP NOMS.

Durham, D., Boyle, J., Cohen, R., Herzog, S., Rajan, R. and Sastry, A. (2000) "The COPS (Common Open Policy Service Protocol)", IETF RFC 2748.

Eick, S. (1996) "Aspects of Network Visualization", Computer Graphics and Applications, v.6, n.2.

Granville, L., Ceccon, M., Tarouco, L. and Almeida, M. (2001) "An Approach for Integrated Management of Network with Quality of Service Support Using QAME", IFIP/IEEE DSOM.

Hewlard Packard. (2002) "HP OpenView", http://www.hp.com/openviwew.

Internet Research Task Force – IRTF. (2002) "Network Management Research Group - NMRG", http://www.irtf.org.

Kar, G., Keller, A. and Calo, S. (2000) "Managing Application Services over Service Provider Networks: Architecture and Dependency Analysis", IEEE/IFIP NOMS.

McCloghrie, K., Fine, M., Seligson, J., Chan, K., Hahn, S., Sahita, R., Smith, A. and Richmeyer, F. (2001) "Structure of Policy Provisioning Information (SPPI)", IETF RFC 3159.

McCloghrie, K., Perkins, D., Schoenwaelder, J., Case, J., Rose, M. and Waldbusser, S. (1999) "Structure of Management Informatin Version 2 (SMIv2)", IETF RFC 2578.

Miranda, S., Nogueira, J. and Machado, C. (2002) "Event Analysis for Telecommunication Management: Building a Special System and its Use", IEEE/IFIP NOMS.

Palma, C. and Rodrigues, L. (2001) "Supporting Views in Network Management Systems", IEEE/IFIP DSOM.

Pras, A. and Shoenwaelder, J. (2002) "On the Difference between Informartion Models and Data Models", IETF draft <draft-irtf-nmrg-im-dm-00.txt> (work in progress).

Sloman, M. (1994) "Policy Driven Management for Distributed Systems", Jornal of Network and Systems Management, Plenum Press, v.2, n.4.

Sloman, M., Valey, B., Moffet, J. and Twidle, K. (1993) "Domain Management and Accounting in an International Cellular Network", IM.

Strum, R. (1998) "Working with Unicenter TNG", ISBN 0-7897-1765-4.

Wahl, M. et al. (1997) "Lightweight Directory Access Protocol (v3)", IETF RFC 2251.