NetTracker: Uma Arquitetura Operacional Extensível para Ferramentas de Gerenciamento de Redes.

Rogério Borges Mouro, Fábio Teruo Morishita e Edson S. Moreira.

(mouro, teruo, edson@icmsc.sc.usp.br)

Instituto de Ciências Matemáticas de São Carlos - ICMSC/USP

Po Box 668 - CEP 13560-970 - São Carlos - SP - Brasil

Resumo

O avanço tecnológico em muito tem dificultado a atualização das ferramentas de gerenciamento de rede, principalmente por não existirem propostas concretas para uma arquitetura operacional. Neste artigo é apresentado uma avançada arquitetura operacional, para sistemas de gerenciamento de redes, que possui como características a facilidade e flexibilidade de incorporação de novas tecnologias e adequação a diferentes plataformas. Ao final são apresentadas resultados preliminares obtidos na aplicação destes conceitos no sistema de gerenciamento NetTracker, em termos de capacidade de atualização, funcionalidade e portabilidade.

Abstract

Technological advances have contributed to make difficult upgrades of network management tools, mainly due to the fact that there are not concrete proposals for operational architecture in this field. In this article we present an advanced operational architecture for network management systems, that has as main features the easiness and flexibility for the incorporation of new technologies and the porting to different platforms. Finally we present some preliminary results obtained on the application of this approach on the development of the NetTracker, a network management system, in terms of upgrading capability, functionality and portability.

1. Introdução

Inovações e evoluções tecnológicas refletem as necessidades organizacionais e estruturais dos ambientes computacionais atuais. A velocidade com que tais transformações vêm ocorrendo surpreende mesmo as organizações que contribuem diretamente com esta escalada tecnológica, deixando-as constantemente sujeitas a mobilizarem esforços no sentido de manterem atualizados seus produtos.

Nos últimos anos, muito tem se discutido sobre gerenciamento de redes, e várias foram as contribuições dadas por pesquisadores em todo mundo para se tentar sanar deficiências na área. Arquiteturas de gerenciamento foram propostas, como a *OSI Management Architecture* (ANS87) e a *Internet-standard Network Management Architecture* (CAS90). Contudo, estas contribuições permaneceram restritas ao campo funcional, estabelecendo os requisitos desejáveis para eventuais ferramentas de auxílio às atividades de gerenciamento de redes. Muito pouco tem sido discutido ou proposto no sentido de se estabelecer arquiteturas operacionais tecnologicamente superiores, e que tragam como características centrais a facilidade e flexibilidade de atualização tecnológica, bem como a capacidade de adequação a diferentes plataformas.

Inicialmente, neste artigo, são estabelecidos os requisitos essenciais a uma arquitetura operacional avançada para ferramentas de gerenciamento de redes. Esta arquitetura, aliada aos requisitos funcionais definidos pela proposta de gerenciamento ISO/OSI (KLE88), dispõe de grande flexibilidade em termos de capacidade de atualização e independência tecnológica. A seguir é apresentado o protótipo da ferramenta de gerenciamento de redes NetTracker, desenvolvida adotando-se os conceitos da nova arquitetura proposta. Ao final, são apresentados os resultados preliminares obtidos por esta ferramenta, em termos de capacidade de atualização, funcionalidade e portabilidade.

2. Definição de Requisitos

Dentro da necessidade evolutiva e mediante um criterioso levantamento em busca de eventuais inadequações tecnológicas nas ferramentas de gerenciamento existentes, chegou-se a conclusão de que os efeitos depredatórios da evolução, sobre uma dada ferramenta, têm como principal causa a ineficácia das arquiteturas operacionais existentes, inibindo e coibindo, de sobremaneira, a capacidade de absorção de novas tecnologias.

Neste contexto e objetivando subsídios para a elaboração de uma nova e moderna arquitetura operacional, foram definidos os requisitos básicos e essenciais a serem seguidos.

Tais subsídios foram elaborados mediante uma extensiva prospecção tecnológica, tendo como critério fundamental a adesão às especificações de padrões abertos internacionais, como os propostos pelas organizações ISO (*International Standard Organization*), ITU (*International Telecommunication Union*), OMG (*Object Management Group*) IAB (*Internet Architecture Board*) e W3C (*World Wide Web Consortium*). Outros elementos contribuintes para a concepção dos requisitos vieram da avaliação de produtos em áreas de tecnologias afins, tais como Jigsaw© (W3C, MIT, INRIA), Netscape Server's Architecture (Netscape™, Inc), SystemView® (IBM™) e Jeeves© (Sun Microsystems™).

Centrado nas premissas: uma arquitetura tecnologicamente superior e capacidade de atualização fácil, foram elaborados os requisitos que se seguem.

- Uma ferramenta de gerenciamento deve ser facilmente portável para diferentes plataformas de sistemas operacionais e arquiteturas de hardware, exigindo-se um mínimo de esforço neste sentido;
- Uma ferramenta de gerenciamento deve ser amplamente extensível em relação às suas capacidades, possibilitando a inserção de novas funcionalidades sem a necessidade de re-codificações das partes já existentes. Além do mais, deve apresentar como característica desejável a capacidade de extensão dinâmica, sem a necessidade de interrupções de seu funcionamento;
- Uma ferramenta de gerenciamento deve possuir capacidades modernas e eficientes de comunicação com o administrador da rede, provendo-o de total controle sobre os recursos disponíveis, independentemente da sua localização geodésica e do grau de segurança do meio de comunicação utilizado. Com tal intuito, protocolos confiáveis e seguros de comunicação deverão ser empregados;
- Uma ferramenta de gerenciamento deve utilizar-se das tecnologias e especificações abertas complacentes, devidamente padronizadas por entidades a contento, e de reconhecida superioridade funcional.

O direcionamento destes requisitos teve como resultado o estabelecimento de uma nova e poderosa arquitetura, baseada em tecnologias emergentes (Java, CORBA, HTTP, SSL, SNMPv2), e que serviu como base para a implementação do protótipo inicial da ferramenta de gerenciamento de redes NetTracker, apresentada ao longo deste trabalho. Antes, entretanto, são apresentadas as tecnologias utilizadas em sua implementação.

A cada dia, surgem novas e modernas tecnologias que possuem como característica comum a generalização de suas aplicações nas mais variadas áreas de interesse. No segmento de Tecnologia da Informação, este comportamento pode ser amplamente observado; diga-se de passagem a integração de áreas como Hipertextos, Protocolos de Comunicação, Criptografia de Dados e Multimídia, que culminou em um dos mais extraordinários sistemas de informação de nossa era: a Interntet e, mais recentemente, sua malha de navegação *World Wide Web*. Seguem-se algumas das tecnologias que integram a arquitetura da ferramenta de gerenciamento de redes NetTracker.

Java - Desenvolvida pela Sun Microsystems, a linguagem de programação Java (GOS96) apresenta uma série de características únicas que tornaram-na um marco na história da informática moderna. Tais características incluem e não limitam-se às que se seguem:

- Java é um linguagem fortemente baseada em tipos e que integra todos os conceitos de orientação a objetos (BOO91);
- Resolução de dependências em tempo de execução. Esta característica é de extrema importância pois permite não apenas a carga dinâmica de classes como também a utilização de referências cruzadas nas definições das mesmas (i.é. duas classes distintas podem possuir referências entre si).
- Independência de plataforma. Característica importante da linguagem Java, foi a principal responsável pela rápida e maciça aceitação da mesma. O código gerado pelo compilador (conhecido por *bytecode*) não é direcionado a nenhuma arquitetura específica de processador, mas sim à especificação de uma máquina virtual conhecida como *Java Virtual Machine* (LIN97).
- Código Transportável. Característica advinda das demais, permite o envio e recebimento de código, possibilitando sua posterior execução após uma completa checagem de segurança. Tais códigos são conhecidos por *Java Applets* e são comumente executados nas máquinas virtuais atreladas aos ambientes de navegação (*Browsers*).

CORBA - O aumento e a rápida proliferação do número de hardwares e softwares disponíveis impulsionaram o desenvolvimento em busca de uma arquitetura que permitisse a interoperabilidade entre dispositivos de rede. Como resposta a esta necessidade, o OMG (*Object Management Group*) propôs a especificação CORBA (*Common Object Request Broker Architecture*) (BEN95), que de uma forma simples, permite a comunicação entre objetos de aplicações, independente de suas localidades. Junto à CORBA, o OMG também definiu a IDL (*Interface Definition Language*) e as APIs (*Application Programming Interfaces*), utilizados para o desenvolvimento de aplicações distribuídas.

O relacionamento entre os objetos distribuídos é realizado através de uma arquitetura cliente-servidor, onde há a presença do ORB (*Object Request Broker*) como agente intermediário. Através do ORB um cliente pode executar um método de um objeto local ou remoto qualquer. De fato, o ORB intercepta a chamada a um método, localiza o objeto que o implementa, executa o método e por fim retorna os resultados. Tudo isso acontece de forma transparente, sem a necessidade do cliente saber onde o objeto destino encontra-se, a linguagem de programação, o sistema operacional ou qualquer outro aspecto relacionado à interface com o objeto. Neste cenário, a IDL tem como proposta simplificar o desenvolvimento destas aplicações, pois permite descrever detalhadamente as interfaces dos métodos de cada objeto que compõe uma aplicação distribuída.

HTTP - O *HyperText Transfer Protocol* (DAV97) é um protocolo aberto, responsável por permitir o transporte de informações de tipos variados sobre uma rede de arquitetura TCP/IP (TAN89). Este protocolo é comumente empregado no tráfego de códigos hipertextos HTML (FIE97), executáveis bytecode Java, multimídia, textos planares ASCII, entre outros.

SSL - O padrão de segurança de informação SSL (Secure Sockets Layer), desenvolvido inicialmente pela Netscape™ (FRE96), especifica uma camada de segurança entre a aplicação e o Socket (STE90) de comunicação, protegendo o conteúdo da informação mediante sua criptografia. Tal especificação encontra-se atualmente em fase de padronização internacional, devendo emergir como uma alternativa definitiva para o transporte de informações de caráter restritivo. O protocolo em si é baseado no algoritmo de criptografia de dados RSA (RIV78) e provê suporte à certificação de autenticidade, tanto do servidor quanto do cliente envolvidos na troca das informações.

SNMPv2 - De acordo com o escopo do gerenciamento Internet, um ambiente gerenciado é constituído de entidades agentes que provêem acesso às informações de gerenciamento e no mínimo de uma entidade gerente atuando no papel de centralizador e coordenador de ações. A troca de informações de gerenciamento é efetuada mediante a utilização de um protocolo específico, através do qual se definem as formas de autenticação, autorização e controle de acesso.

O SNMP (*Simple Network Management Protocol*) surgiu como uma proposta rápida às necessidades do mercado por um protocolo de gerenciamento em ambientes TCP/IP (ROS91). Em princípio o SNMP deveria ser apenas um protocolo provisório, sendo substituído pelo CMIP (*Common Management Information Protocol*). Devido ao atraso no desenvolvimento do CMIP, entre outros fatores, o SNMP tornou-se o principal padrão de gerenciamento. Entretanto, devido a seu caráter provisório, apresentava deficiências relacionadas a segurança e baixo desempenho em ambientes grandes.

Com o objetivo de solucionar tais problemas foi proposto uma nova versão do protocolo SNMP (STA93)(CAS93a). O SNMPv2 corrige problemas de segurança, além de adicionar novas funcionalidades, onde destaca-se a capacidade de gerenciamento distribuído, via primitivas de comunicação gerente-gerente (CAS93b), e as formas de tratamento e transporte dos dados.

3. Arquitetura do Sistema

A pesar da estrutura de gerenciamento Internet (ROS90) mostrar-se desatualizada frente às novas proposta tecnológicas (DER96), ainda sim apresenta-se como o principal padrão existente, com funcionalidade comprovada. Sendo assim, é evidente a necessidade de sua adoção como elemento de adequação a realidade atual.

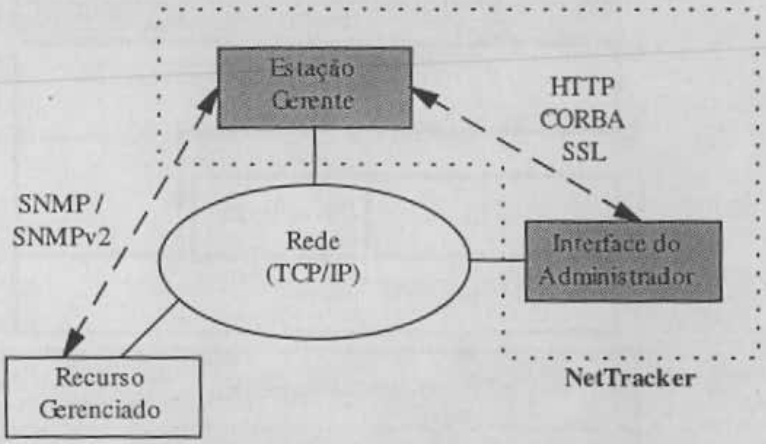

Figura 3.1 - Estrutura de Comunicação intramódulos

A Figura 3.1 sintetiza o relacionamento entre os elementos da estrutura de gerenciamento que integram o ambiente gerenciado pela ferramenta NetTracker, e que seguem de perto a estrutura do gerenciamento Internet. Tais elementos são descritos a seguir:

- **Recurso Gerenciado** - representa todos os elementos que compõem a rede e que sejam providos de capacidades de gerenciamento Internet (i.é, *Routers, Bridges, Hosts, etc*), comunicando-se diretamente com a estação de gerenciamento, via protocolo SNMP e SNMPv2.
- **Estação Gerente** - parte fundamental de uma ferramenta integrada de gerenciamento este elemento atua como agente centralizador das informações de gerenciamento, ficando responsável por prover todas as funções de controle e monitoração ao administrador.
- **Interface do Administrador** - é o elemento responsável por permitir a interação entre a ferramenta NetTracker e o administrador da rede. É importante notar que esta cumpre não apenas o papel de agente difusor de informação, mas também o papel de elemento descentralizador de gerenciamento. Em princípio, determinadas partes da interface serão codificadas como códigos transportáveis (*Applets Java*) que poderão ser executado por qualquer *Browser* com capacidades Java.

Sob a ótica de gerenciamento, a Figura 3.2 apresenta uma visão de alto nível da arquitetura da ferramenta NetTracker, que é composta por cinco camadas interdependentes: *Módulos de Programas de Aplicação (APM), Interface de Programas de Aplicação (API), Módulos de Extensão (EM), Biblioteca de Objetos Java e Máquina Virtual Java.* A seguir são descritos os papeis funcionais de cada uma destas camadas.

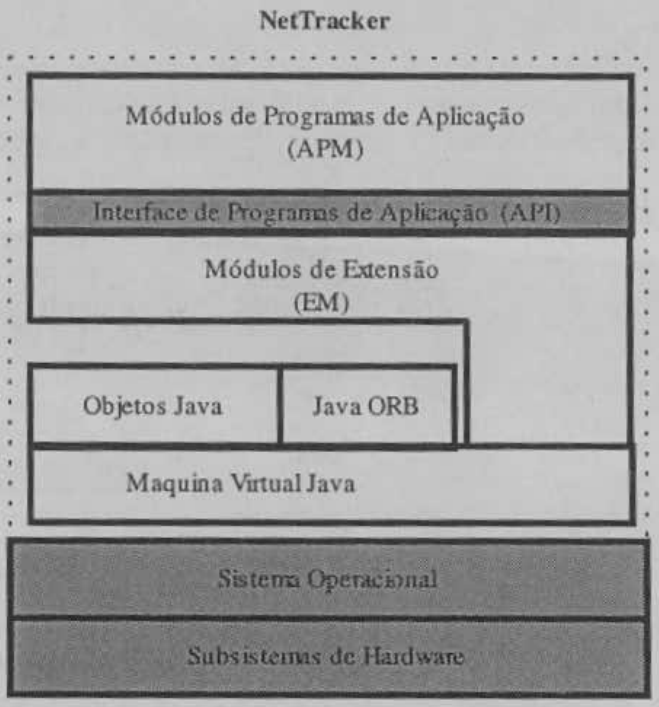

Figura 3.2 - Arquitetura Operacional NetTracker

3.1 Camada de Módulos de Programas de Aplicação

A camada APM trata dos aspectos funcionais agregados ao sistema NetTracker, incorporando as aplicações responsáveis por proverem as funções de gerenciamento e de comunicação com o administrador. Como exemplos de elementos componentes desta camada, podemos citar a aplicação de Filtragem de Eventos SNMP, juntamente com as aplicações de gerenciamento de ações (*Trigger*) e de busca e reconhecimento automático da topologia da rede (*Discovery*).

3.2 Camada de Interface de Programas de Aplicações

Nesta camada, encontram-se disponíveis as interfaces de programação que fornecem acesso às funções implementadas pelos Módulos de Extensão. No contexto da tecnologia de objetos, que será amplamente empregado neste projeto, esta camada é composta por todas as interfaces dos objetos codificados na camada EM.

3.3 Camada de Módulos de Extensão

Todos os objetos responsáveis por proverem funções básicas de apoio às funções de gerenciamento serão integrados à esta camada. Tais funções desempenharão o papel de alicerce para os programas de aplicações que se encontram na camada superior. Basicamente, esta é composta pelos seguintes sub-módulos:

- **HTTPServer** - Permite que a Ferramenta de gerenciamento disponibilize serviços de configuração remota através de requisições HTTP. Este módulo possibilita que as interfaces de usuário possam ser transparentemente enviadas ao administrador, independentemente de sua localidade no ambiente.
- **Database** - Este módulo é responsável por prover, de forma transparente, acesso à mais variada gama de SGBDs disponíveis no mercado. Para tal prevê-se a utilização do protocolo JDBC (*Java Database Connection*) ou ainda de *drivers* nativos oferecidos pelos fornecedores dos produtos. Em princípio, no protótipo, é utilizado a implementação do mSQL para a plataforma Unix®.
- **SNMP** - todas os recursos de gerenciamento presentes no protocolo SNMP serão agrupados neste módulo. Em uma primeira instância, serão agregadas as funcionalidades de uma implementação de domínio público. Prevê-se, contudo, a mobilização de esforços no sentido de implementá-lo no futuro.
- **SNMPv2** - com propósito semelhante ao anterior, este módulo disponibilizará as funções de gerenciamento da nova versão do SNMP (SNMPv2). Existe um trabalho paralelo, de implementação deste módulo, sendo efetuado como parte do programa de mestrado do ICMSC.

- **Security** - este módulo tem o objetivo de disponibilizar todas as funções que tratem dos aspectos de segurança da ferramenta NetTracker (i. é, SSL, MD5, DES, RSA, S/Key, entre outros). Além do mais, existem perspectivas em se incorporar funcionalidades extras de segurança ao NetTracker, mediante a integração de um módulo adaptativo de detecção de intrusão baseado em redes neurais.

3.4 Biblioteca de objetos Java

Nesta camada encontram-se todos os objetos que compõem as classes da linguagem Java, e que são responsáveis por permitirem a implementação de grande parte das camadas superiores.

3.5 Maquina Virtual Java

Sem dúvida, é o coração da nova arquitetura do NetTracker. Sobre ela estão assentadas todas as principais características inovadoras da ferramenta: carga e descarga dinâmica de módulos, independência de plataforma, linkedição dinâmica e em tempo de execução de classes, acesso remoto a objetos, códigos binários transportáveis, etc.

4. A estrutura e o relacionamento dos objetos no sistema.

A ferramenta NetTracker é composta por duas categorias principais de objetos: Objetos de Sistema e Objetos de Aplicação. Os primeiros definem os objetos que integram o ambiente básico de execução do sistema, efetuado a carga dinâmica e posterior ativação dos Objetos de Aplicação. Estes últimos, por sua vez, definem de fato as aplicações do sistema NetTracker. A Figura 4.1 nos dá uma visão mais detalhada desta estrutura organizacional.

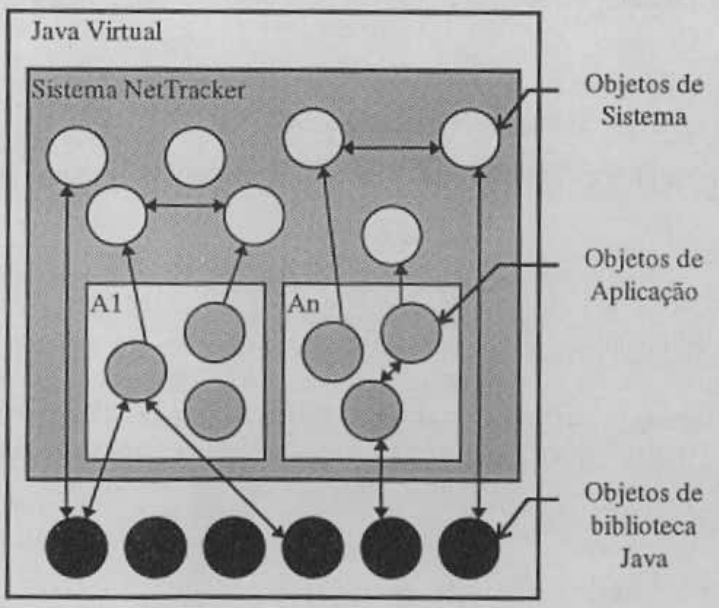

Figura 4.1 - Estruturação dos Objetos NetTracker

Desta forma, o processo de ativação e execução do sistema NetTracker pode ser descrito pelos seguintes passos:

1. Após o processo de ativação da Máquina Virtual Java, a classe principal do sistema (NetTracker) é carregada, checadas suas dependências e posteriormente ativada. Este procedimento é comumente aplicado às aplicações Java;
2. Após ser instanciado pela etapa anterior, e de posse do controle executivo, o método construtor configura o sistema, criando as estruturas de suporte e controle necessárias. Neste processo são instanciados e ativados os objetos ditos essenciais, tais como MVHttpServer, MVManager, MVLoader, entre outros. O primeiro funciona como um servidor HTTP e fica responsável por permitir o interfaceamento do sistema NetTracker com o

administrador da rede, através de um *Browser* Web qualquer. Já o segundo é responsável por controlar a criação e o gerenciamento dos objetos NetTracker (MVApplication) que são instanciados a partir de classes carregadas dinamicamente pelo ambiente. O objeto MVLoader é tratado logo a seguir. Ao término desta etapa, todo os objetos de sistema estão estabelecidos e em processo de execução;

3. Ativado como uma *thread* Java, o objeto MVLoader inicia o procedimento de carga e registro de todos os objetos de aplicação. Para tal é examinado o arquivo de cadastro de aplicações (MVAppReg.cfg), buscando e efetuando a carga dinâmica de cada um dos módulos de aplicações cadastradas. A princípio estão cadastradas as aplicações de suporte ao sistema (*ManagerSNMP, Logger, Trigger, Database e Trapper*).
4. Após a carga, e durante o processo de ativação dos módulos de aplicação mencionados anteriormente, cada qual efetua o registro de seus atributos em um objeto de registro de atributos (MVApplicationAttributes). Isto é necessário, pois, através deste registro é possível o controle e monitoração de cada atributo pelo gerenciador do sistema (MVManager).

Ao término deste processo de ativação o sistema está plenamente operacional e apto a interagir com o administrador da rede. Segue-se uma descrição das aplicações de suporte ao sistema.

4.1 *ManagerSNMP*

A aplicação *ManagerSNMP* é responsável por gerenciar as informações de gerenciamento SNMP contidas em MIBs (*Management Information Base*) (MCC91) controladas e mantidas por agentes SNMP. O processo de obtenção de informações é realizado de duas formas: *polling*, a aplicação gerente faz requisições periódicas aos clientes; e o *trapping* , que são mensagens enviadas ao gerente sobre algum evento anormal ou pré-determinado (Arquivo de configuração *ManagerSNMP.conf*), sem que estas tenham sido requisitadas pelo gerente.

Seguindo a estrutura das aplicações do NetTracker, tem-se que o objeto *ManagerSNMP* é composto das seguintes classes: *ManagerSNMP*, *MibControl* e *MibTrapper* que compõem o Módulo de Extensão.

- *ManagerSNMP*: Implementa as funções que monitoram os limites aceitos para as variáveis do ambiente gerenciado, para que no momento que estas alcancem os seus limiares o gerente SNMP dispare um evento que irá tratar o problema.
- MibControl: As funções de monitoramento e controle dos agentes SNMP serão implementadas nesta classe, no qual a interface as aplicações será realizada através de páginas HTML.
- MibTrapper: Responsável por receber as notificações (Traps) enviadas pelos agentes SNMP da rede.

4.2 *Logger*

A função do *Logger* é gerenciar os arquivos de *log* das aplicações do sistema NetTracker, que devem referenciar o objeto *Logger* para poder enviar as mensagens. Estas mensagens possuem três campos: *type* (determina o tipo da mensagem), *priority* (define a prioridade de tratamento da mensagem) e *message* (mensagem que será armazenada no log). A determinação em qual arquivo de *log* será armazenada a mensagem é de responsabilidade do administrador, que através de uma interface HTTP (via método *perfom()*) determina qual arquivo está relacionado ao par (*type, priority*).

4.3 *Trigger*

O *Trigger* é responsável por executar funções ou tarefas que sejam dependentes de algum evento que possa ocorrer no ambiente. Estas funções são configuradas seguindo um relacionamento com algum evento determinado, por exemplo: o recebimento de uma notificação da perda de comunicação entre um par de roteadores, pelo módulo de filtragem de *traps* poderia disparar uma tarefa que estabeleceria uma rota alternativa, através da inserção de uma rota extra na tabela de roteamento de um roteador disponível.

4.4 *Database*

A *Database* gerencia as consultas à base de dados realizadas pelas aplicações, executando os comandos SQL passados e retornando o resultado da consulta. Para se ter acesso as funcionalidades do *Database* as aplicações referenciam o objeto *Database* que possui todos os métodos necessários para a criação e manutenção da base. Por exemplo: as informações recebidas e coletadas pelo *ManagerSNMP* são armazenadas em uma base de

dados para que estas estejam disponíveis às outras aplicações, que também fazem acesso às informações através do *Database*.

4.5 Trapper

Esta aplicação é responsável por receber notificações de ocorrências de eventos importantes, que possam vir a ocorrer em um agente SNMP ou SNMPv2, filtrá-las baseadas em padrões e critérios pré-definidos pelo administrador da rede, e apresentar tais informações, ou requisitar a execução de alguma tarefa pelo módulo *Trigger*. Desta forma pode-se estabelecer ações que possam ser disparadas automaticamente pelo sistema, em reação a um estímulo fornecido por um dos filtros.

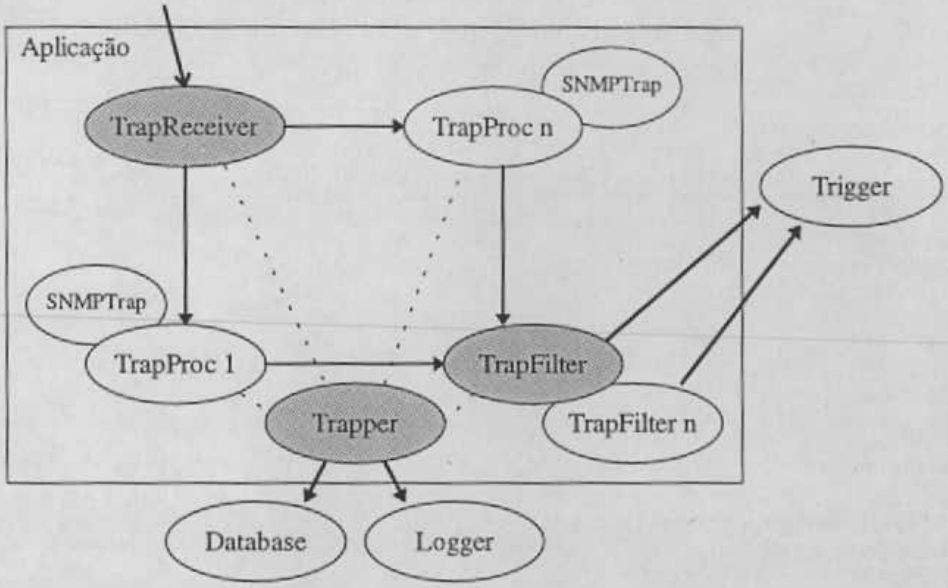

Figura 4.2 - Objetos da Aplicação de Filtro de Traps

Em princípio está sendo tratado apenas o protocolo SNMP via Advent API (ADV96), dado a falta de uma API SNMPv2 disponível. Existe, entretanto, um trabalho que está sendo desenvolvido que visa implementar uma API SNMPv2, a qual poderá ser incluída posteriormente no sistema NetTracker. A composição de objetos e o interrelacionamento destes, dentro da aplicação de filtragem de eventos, pode ser observado na

Figura 4.2. De fato, iniciada através da carga e ativação da classe principal (*Trapper*), são criados e ativados os objetos TrapReceiver, TrapFilter, cujas funções estão fora do escopo deste artigo. O Trapper, por outro lado, atua como o objeto essencial da aplicação, implementando as funções run e perform.

5. Definição e operação das aplicações

Cada aplicação ou módulo de aplicação a ser integrado ao sistema é composto por um conjunto de definições de objetos, os quais interagem entre si para proporcionar a funcionalidade necessária e esperada. Para tal, um destes objetos, dito objeto de ativação, deve obedecer a duas regras essenciais:

- A definição (Classe) de um objeto principal de aplicação deve necessariamente herdar os atributos e métodos da superclasse MVApplication, como apresentado na Figura 5.1. Esta última, por sua vez, define os métodos abstratos de interfaceamento dos objetos do sistema, apresentados anteriormente, com os objetos de aplicação;
- Todo objeto de ativação deve implementar obrigatoriamente os método **run** e **perform**, mostrados na Figura 5.2. O primeiro deve codificar as instruções e operações que estabelecem, de fato, a aplicação no ambiente operacional do sistema, e será invocado logo após sua carga pelo mesmo. O último deve conter código de manuseio de solicitações HTTP.

```
/* @ (#) MVManager.java r0.1 alpha        16/12/96 Rogério Borges Mouro
 *
 * NetTracker 1.0 alpha (LEGO)
 * Copyright (c) 1996, University of São Paulo at São Carlos.
 * Intermidia Research Lab. All Rights Reserved.
 */
public class MVTrapper extends MVApplication {

        . . .

} // end of MVManager
```

Figura 5.1 - Definição de uma plicação NetTracker

```
        . . .
        protected run() {
                while( alive ) {
                        thread.suspend();

                        . . .

                }
                end();
        } // end of run method

        protected Reply perform( Request request ) {
                Reply reply = null;
                String URI = request.getURI();

                . . .

                return reply;
        } // end of perform method
```

Figura 5.2 - Funções essenciais

Além da obrigatoriedade do cumprimento destas regras, e caso haja necessidade de se definir atributos de controle para a aplicação, cada um destes atributos devem ser definidos utilizando-se os objetos preestabelecidos para este propósito (MVInteger, MVFloat, MVString, MVBoolean e MVByte). É importante ainda ressaltar que cada atributo deverá registrar-se explicitamente, através do método **register**, no objeto de atributos (MVApplicationAttributes) da aplicação do qual fazem parte.

Seguindo estas regras na definição de uma aplicação, seu relacionamento com outros objetos no sistema se dá de forma transparente, mesmo quando as classes alvo do relacionamento ainda não estão carregadas e disponíveis. Neste caso o próprio ambiente Java se incumbe de resolver tais dependências, carregando-as e ativando-as quando necessário.

6. Conclusões.

A ferramenta NetTracker apresenta-se amplamente flexível no sentido de permitir a inclusão de aplicações e, consequentemente, de novas tecnologias, sem a necessidade de recodificação das partes já existentes. É evidente que esta capacidade somente é verificada caso não haja dependência das aplicações anteriores com a aplicação que esteja sendo incorporada (i.e. uma aplicação já funcional não pode ter acesso aos métodos de uma aplicação desenvolvida posteriormente a ela).

Em termos do desempenho, é evidente que o ônus operacional introduzido pela maquina virtual Java traz como conseqüência um degradação geral no processamento da ferramenta. Entretanto é factível a utilização de uma maquina virtual que incorpore um módulo JIT (*Just-in-time compile*) em sua base, agindo de forma perceptível no incremento da velocidade de execução da ferramenta e, conseqüentemente, das aplicações por ele controladas.

Atualmente o protótipo implementado, possui as capacidades de carga e descarga dinâmica de módulos, controle parcial da ferramenta e dos módulos carregados (ativação e desativação, visualização e edição dos atributos dos módulos de aplicações), tudo via protocolo HTTP. O módulo *Trapper* e *Trigger* encontram-se em fase final de codificação. Os demais módulos estão sendo desenvolvidos pelo grupo de Redes do Laboratório Intermídia.

De fato, a única preocupação de quem venha a desenvolver aplicações de gerenciamento que tenham como meta agregar funcionalidades extras ao NetTracker, é a de codificar as funções *run* e *perform* no corpo de suas aplicações, como discutidas anteriormente, além de utilizar adequadamente os recursos de classes e aplicações já implementadas.

7. Referências

[ADV96] Advent Network Management, Inc., *Uniform Resource Locators* (URL), http://www.advent.com/index.html, 1996.

[ANS87] ANSI X3T5/87-305: Tutorial Material for Development of OSI Management Model Section V - Information Model, Novembro de 1987.

[BEN95] Bem-Natan, R., A guide to Common Object Request Broker Architecture, McGraw Hill, 1995

[BOO91] Booch, G.: Object Oriented Design with Applications. The Benjamin/Cummings Publishing Co., Inc., 1991

[CAS90] CASE, J. et. al. A Simple Network Management Protocol (SNMP), Network Working Group, *Request for Comments* 1157, 1990

[CAS93a] Case, J. et. al. Introduction to version 2 of the Internet-standard Network Management Framework. SNMP Research, Inc., Hughes LAN System, Dover Beach Consulting, Inc., Carnegie Mello University, *Request for Comments* 1441, 1993.

[CAS93b] CASE, J. et. al. Manager-to-Manager Management Information Base. SNMP Research, Inc., Hughes LAN System, Dover Beach Consulting, Inc., Carnegie Mello University, *Request for Comments* 1451, 1993.

[DAV97] Dave, R., *Uniform Resource Locator* (URL), http://www.w3.org/pub/WWW/TR/REC-html32.html, 1997.

[DER96] Deri, L, Surfin' Network Resources across the Web, *Uniform Resource Locator* (URL), http://misa.zurich.ibm.com/Webbin/~Ide/Surfin.ps.Z, 1996.

[FIE97] Fielding, R, et al, Hypertext Transfer Protocol - HTTP/1.1, Request for Comment 2068, 1997.

[FRE96] Freir, A., O.; Karltan, P.; Kacher, P., C., The SSL Protocol version 3.0, Transport Layer Security Working Group, *Uniforn Resource Locator* (URL), ftp://ds.internic.net/internet-drafts/draft-ietf-tls-ssl-version3-00.txt, 1996.

[GOS96] Gosling, J.; Joy, B.; Steele, G., The Java Language Specification, Addison-Wesley, 1996.

[KLE88] Klerer, S. M.: The OSI Management Architecture: An Overview. IEEE Network, Vol. 2, No. 2, pg. 20-29, Março de 1988.

[LIN97] Lindholm, T., et al., The Java Virtual Machine Specification, Addison-Wesley, 1997.

[MCC91] McCloghrie, K.; ROSE, M. T., Management Information Base for Network Management of TCP/IP-based internets: MIB-II. (DDS Network Information Center, SRI International), Request for Comment 1213, 1991.

[RIV78] Rivest, R., L.; Shamir, A.; Adleman, L. M., A method for obtain digital signature and public-key cryptosystems, Communications of the ACM, 21(2), p.120-126, 1978.

[ROS90] Rose, M. T., Structure and Identification of Management Information for TCP/IP-based internets. (DDS Network Information Center, SRI International), Request for Comment 1155, 1990

[ROS91] Rose, M. T., The Simple Book - *An Introduction to Management of TCP/IP-based Internets*. Englewood Cliffs, Prentice-Hall, 1991.

[STE90] Stevens, R., Unix Network Programming, Prentice Hall, 1990.

[STA93] Stallings, W., SNMP, SNMPv2, and CMIP - *The Pratical Guide to Network-Management Statandards*. Addison-Welley, 1993.

[TAN89] Tanenbaum, A., S., Computer Network 2.ed, Englewood Cliffs, Prentice Hall, 1989.