# MS2VAN - Um Sistema para Gerência de Serviços em Redes Ativas Virtuais

*Fábio Luciano Verdi*

Dissertação de Mestrado

Instituto de Computação
Universidade Estadual de Campinas

# MS2VAN - Um Sistema para Gerência de Serviços em Redes Ativas Virtuais

Fábio Luciano Verdi

Março de 2002

**Banca Examinadora:**

- Prof. Dr. Edmundo Roberto Mauro Madeira
  Instituto de Computação - Unicamp (Orientador)
- Prof. Dr. Michael Anthony Stanton
  Instituto de Computação - UFF
- Prof. Dr. Nélson Luis Saldanha da Fonseca
  Instituto de Computação - Unicamp
- Prof. Dr. Paulo Lício de Geus
  Instituto de Computação - Unicamp

## FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Verdi, Fábio Luciano

V492m MS2VAN – Um sistema para gerência de serviços em redes ativas
V584m virtuais / Fábio Luciano Verdi -- Campinas, [S.P. :s.n.], 2002.

Orientador : Edmundo Roberto Mauro Madeira

Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computação – Gerência. 2. Telecomunicação. 3. Agentes inteligentes (Software). I. Madeira, Edmundo Roberto Mauro. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

# MS2VAN - Um Sistema para Gerência de Serviços em Redes Ativas Virtuais

Este exemplar corresponde à redação final da Dissertação devidamente corrigida e defendida por Fábio Luciano Verdi e aprovada pela Banca Examinadora.

Campinas, 18 de Março de 2002.

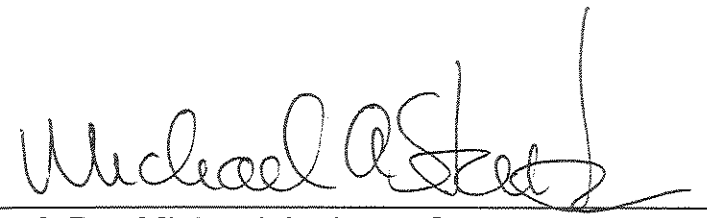
Edmundo R M Madeira

Prof. Dr. Edmundo Roberto Mauro Madeira
Instituto de Computação - Unicamp
(Orientador)

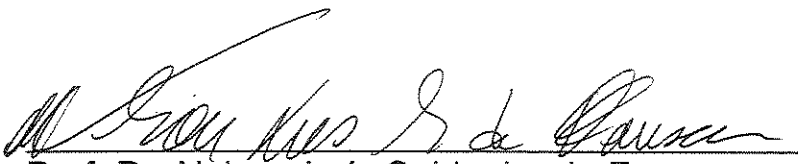
Dissertação apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

# TERMO DE APROVAÇÃO

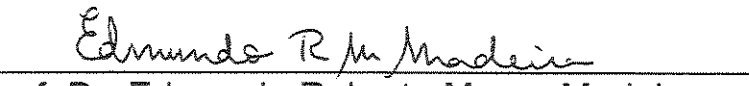
Tese defendida e aprovada em 18 de março de 2002, pela Banca Examinadora composta pelos Professores Doutores:



Prof. Dr. Michael Anthony Stanton
IC - UFF



Prof. Dr. Nelson Luís Saldanha da Fonseca
IC - UNICAMP



Prof. Dr. Edmundo Roberto Mauro Madeira
IC - UNICAMP

# Resumo

As Redes Ativas (*Active Networks*), diferentemente das redes tradicionais, permitem que pacotes carreguem códigos de programas e sejam instanciados em nós intermediários. Estas redes são ativas pois os nós podem realizar computações e modificar o conteúdo desses pacotes, ao contrário das redes tradicionais onde o processamento dos pacotes limita-se às operações realizadas nos cabeçalhos, principalmente com objetivo de roteamento sem modificar o dado do usuário, o qual é transferido de forma opaca pela rede. Com isso, a tecnologia das redes ativas adapta-se à necessidade de ambientes de Telecomunicação devido à facilidade de envio e instalação de serviços em locais específicos na rede. Além disso, no ambiente das redes ativas surge o conceito de Redes Ativas Virtuais. Uma Rede Ativa Virtual pode ser descrita como um grafo de nós ativos virtuais conectados por enlaces virtuais. Neste trabalho apresentamos um sistema para gerenciar serviços em Redes Ativas Virtuais e estamos particularmente interessados em gerenciar ambientes de Telecomunicação. O sistema de gerência possui alguns componentes desenvolvidos usando a tecnologia de Agentes Móveis e considera a existência de serviços móveis que podem migrar na mesma Rede Ativa Virtual ou de uma Rede Ativa Virtual para outra. O modelo proposto inclui três áreas funcionais de gerenciamento: contabilidade, desempenho e configuração, e suporta também a criação de Redes Ativas Virtuais e a instalação de serviços. A gerência de dois serviços de Telecomunicação foi implementada a fim de testar o sistema desenvolvido: o serviço de VPN (*Virtual Private Network*) e o serviço de *Call Forwarding*.

# Abstract

Active Networking (AN) is a new way to provide services in Telecom networks. Users can program the network by injecting their programs into it. The AN nodes can receive, send and run programs using local computational resource. In a Telecom environment, for example, services can be purchased according to the customer's requirements. Furthermore, a lot of different services is offered by service providers. In this work we outline a system for service management in Virtual Active Networks (VANs). A VAN can be described as a graph of virtual active nodes connected by virtual links. We are particularly interested to manage Telecom environments. Some management system components are developed using a mobile agent platform and the system considers the mobile service migration in the same VAN or from a VAN to another. We focus on three specific management functions: accounting, performance monitoring and configuration. The model also supports the VAN creation and the mobile service installation in Telecom environments. The system is tested for the management of two Internet-based Telecom services: Virtual Private Network (VPN) and Call Forwarding.

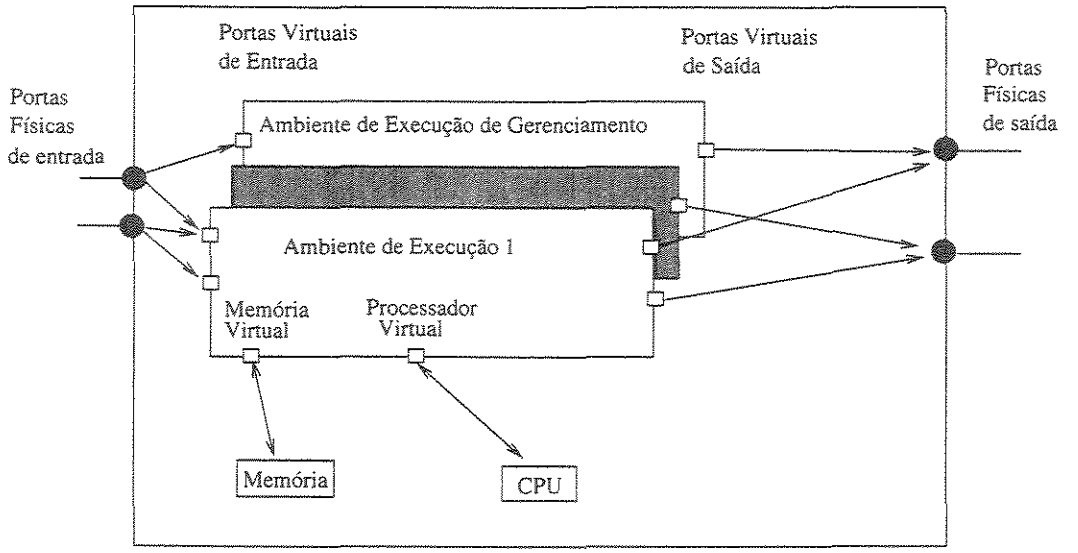



Figura 2.3: Nó de uma Rede Ativa Virtual com vários ambientes de execução.

a partir de blocos independentes e a lógica do serviço pode ser subseqüentemente distribuída para componentes que serão colocados em nós de controle da rede. Ao contrário das redes ativas, este tipo de arquitetura não considera que o cliente carregue e execute funcionalidades na rede [14].

Devido à facilidade de envio e disponibilização de serviços que as redes ativas oferecem, percebeu-se uma forte tendência no uso desta tecnologia em ambientes Telecom. Isso ocorre porque nesse tipo de cenário é necessário um mecanismo flexível de oferecimento de serviços para clientes. Duas figuras principais surgem nesse ambiente: o fonecedor de serviços, e o cliente ou usuário que utiliza os serviços. De um modo geral, o fornecedor de serviços é responsável pela disponibilização, envio, instalação e monitoração dos serviços. O cliente solicita serviços ao fornecedor os quais serão instalados e depois utilizados pelo cliente solicitante. Na maioria dos casos, onde a infra-estrutura de rede é compartilhada por vários clientes, é necessário que um determinado cliente solicite a instalação (configuração) de uma VAN antes da solicitação de serviços. A VAN será a entidade pela qual os recursos físicos serão compartilhados pelos vários usuários separando um cliente do outro. Pode haver uma negociação entre cliente e fornecedor em relação aos recursos a serem usados e a qualidade de cada serviço solicitado. O fornecedor é responsável por monitorar o uso dos recursos na rede e garantir que o acordo com cada cliente seja cumprido. O gerenciamento dos serviços pode ser feito tanto pelo fornecedor quanto pelo cliente-usuário do serviço.

O cliente, ao desejar instalar uma VAN, deverá avisar o fornecedor sobre suas ne-

cessidades e intenções. A partir disso, o fornecedor e o cliente entram em uma fase de negociação dos recursos a serem disponibilizados para o cliente. Os serviços a serem oferecidos também possuem algumas políticas de uso as quais devem ser respeitadas na medida em que cada serviço é instalado e usado. O projeto e implementação de serviços não necessariamente são feitos pelo fornecedor. Em muitos casos surge uma terceira parte responsável unicamente pelo desenvolvimento de serviços. Após estar implementado e testado, o serviço será registrado nos fornecedores de serviços. O cliente também poderá desenvolver seus próprios serviços, instalá-los em sua VAN, usá-los e gerenciá-los. Nesse caso o cliente desempenha o papel de cliente e fornecedor e poderá também disponibilizar seus serviços a outros clientes. Dessa forma, a oferta de serviços na rede pode crescer rápida e dinamicamente e tudo isso sem a interação com o fornecedor. Os clientes também podem fazer realocação de serviços com o objetivo de distribuir a carga em sua VAN sem prejudicar ou interferir nos outros clientes. A Figura 2.4 apresenta de forma simplificada a interação típica entre um cliente e um fornecedor para a instalação de uma VAN e seus serviços.

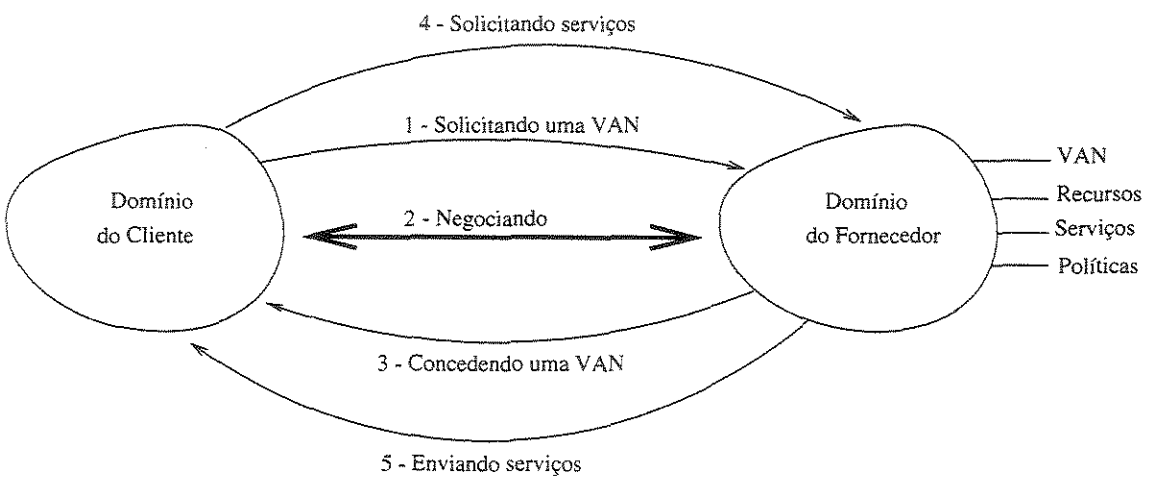



Figura 2.4: Interação entre um cliente e um fornecedor para a instalação de uma VAN e seus serviços.

## 2.3 Agentes Móveis

A tecnologia de agentes atualmente está inserida em muitos contextos tais como inteligência artificial, sistemas distribuídos, comunicação de computadores e engenharia de software [38]. Ao longo dos anos, essa tecnologia tem sido alvo de intensa pesquisa a

fim de definir melhor quais as reais e principais características encontradas nos agentes. Segundo [33] um agente é um programa de computador que age com autonomia em nome de uma pessoa ou organização, enquanto que uma *agência* ou *sistema de agentes* é uma plataforma que permite criar, interpretar, executar, transferir e terminar a execução de agentes. Mais precisamente, um agente deveria possuir algumas das seguintes características:

- Autonomia - Um agente possui controle sobre suas próprias ações e pode decidir o que fazer quando a tarefa é bem ou mal sucedida. Isso deve-se ao fato do agente conter seu código e os dados envolvidos;
- Comportamento Social - O agente deve ter capacidade de interação com o ambiente do local de execução, bem como com outros agentes e usuários;
- Reatividade - Agentes devem ser capazes de perceber o ambiente externo e responder a mudanças ocorridas de forma satisfatória;
- Eqüidade - Não existe uma hierarquia entre os agentes, eles comportam-se como pares;
- Delegação - Pode-se delegar tarefas aos agentes de forma a agirem em favor de seus usuários.

Além das características acima, os agentes podem ser incrementados atribuindo-se a eles mobilidade. Um agente simples passa então a se chamar agente móvel. Os agentes móveis são componentes de software capazes de migrar de uma localização física da rede para outra. Esse movimento para locais onde as informacões estão localizadas permitindo que os agentes realizem comunicação local ao invés de comunicação remota, é uma das grandes vantagens do uso de agentes móveis. A tecnologia de agentes móveis (*Mobile Agent Technology - MAT*) é atualmente usada em muitos contextos voltados para aplicações em redes, tais como recuperação de informação, gerência de redes e sistemas distribuídos, comércio eletrônico, computação móvel e ambientes de apoio à cooperação [38]. A facilidade com que objetos são enviados para locais específicos na rede é um dos motivos principais de seu uso. Um agente móvel é um objeto com autonomia e é isto que o diferencia de um simples objeto. Um agente móvel, portanto, possui um certo nível de inteligência, com algumas regras definidas permitindo que ele tome certas decisões baseadas nos dados coletados.

Apesar da tecnologia de agentes móveis estar crescendo muito, o número de aplicações que usam RPC (*Remote Procedure Call*), ou seja, o modelo cliente/servidor, ainda é grande. Em RPC, o cliente invoca métodos remotos em um servidor de forma síncrona,

ficando bloqueado até o retorno do método. Os agentes móveis são contruídos usando a abordagem da programação remota obtendo um ganho de flexibilidade em relação ao modelo cliente/servidor tradicional. Além do mais, evita-se principalmente o sucessivo número de requisições e respostas através da rede, situação presente no modelo RPC.

A plataforma de agentes móveis escolhida para a implementação do protótipo foi a plataforma Grasshopper 2.1 [28]. Na próxima seção apresentamos as principais características dessa plataforma, e as definições de agente, lugar (*place*), agência e região segundo a perspectiva da equipe do Grasshopper. Também apresentamos alguns breves conceitos sobre os modos de comunicação suportados pela plataforma em questão.

### 2.3.1 Grasshopper

Grasshopper é uma plataforma de agentes móveis desenvolvida para suportar aplicações em ambientes de sistemas distribuídos. O padrão MASIF (*Mobile Agent System Interoperability Facility*) [33] foi incorporado à plataforma a fim de obter a interoperabilidade entre diferentes plataformas de agentes móveis. Dois tipos de agentes são encontrados no contexto do Grasshopper, agentes móveis e agentes estacionários:

- **Agentes Móveis** - São os agentes capazes de se mover de uma localização para outra na rede. Aqui é importante destacar a diferença existente entre código móvel e agente móvel. No primeiro caso, também conhecido como execução remota, o programa é enviado para um local remoto antes de sua ativação, ficando no local durante todo o seu tempo de vida. No caso de agente móvel, o programa (agente móvel) é capaz de migrar de um local para outro durante seu tempo de vida.

- **Agentes Estacionários** - Esses agentes não possuem a capacidade de migração, ficando sempre no mesmo local.

A Figura 2.5 mostra a estrutura hierárquica dos componentes que pertencem à plataforma Grasshopper, os quais serão explicados abaixo.

#### Agência

A agência é um ambiente de execução para os agentes móveis e estacionários. Pelo menos uma agência deverá estar executando em cada host a fim de suportar a execução desses agentes. Uma agência no Grasshopper é formada de duas partes, uma agência central (*core agency*) e um ou mais lugares.

- **Agência Central** - A agência central fornece funcionalidades básicas necessárias para uma plataforma de agentes móveis. Algumas funcionalidades incluem: serviço

de comunicação, serviço de registro, serviço para tarefas de gerenciamento tais como criar e remover agentes, serviço de transporte, serviço de persistência, entre outras [26]. O serviço de registro de cada agência está conectado com o serviço de registro de regiões (ver abaixo) que mantém informações sobre agentes, lugares (ver abaixo) e agências.

- **Lugares** - Um lugar fornece um agrupamento lógico de funcionalidades dentro de uma agência. Pode existir, por exemplo, um lugar com um grau de segurança maior ou um lugar com mecanismos de comunicação mais eficientes. No Grasshopper toda agência possui um lugar padrão chamado *"InformationDesk"*. Todo agente que não possui um lugar informado será enviado para esse lugar padrão.

### Regiões

As regiões facilitam na tarefa de gerenciamento dos componentes distribuídos no ambiente Grasshopper, ou seja, agentes, lugares e agências. As agências e seus lugares podem ser associados com uma região específica sendo registrados usando um registro de regiões. Esse registro de regiões possui informações de todos os componentes que estão associados com uma região específica. Quando um novo componente (agente, lugar ou agência) é criado, ele automaticamente é registrado no registro da região correspondente. Quando um agente migra de uma agência para outra, o registro é também automaticamente atualizado. O registro de regiões funciona como se fosse um serviço de nomes pois mantém uma referência para a localização de cada agente. Quando um agente quer comunicar-se com outro, ele solicita ao registro de regiões a referência do agente desejado. Com essa referência a comunicação entre os dois agentes pode ser estabelecida. Consultando o registro de regiões, qualquer entidade, incluindo agentes e usuários humanos, pode localizar agentes, lugares e agências residindo em uma região.

### Métodos de Comunicação

O serviço de comunicação do Grasshopper suporta o protocolo IIOP (*Internet Inter-ORB Protocol*), RMI (*Remote Method Invocation*) e *socket*. Além disso, para conseguir um mecanismo para comunicação segura, RMI e *socket* podem ser protegidos usando SSL (*Secure Socket Layer*). A comunicação entre agentes pode ser feita de várias maneiras através do uso de comunicação síncrona, assíncrona, dinâmica e *multicast* [26].

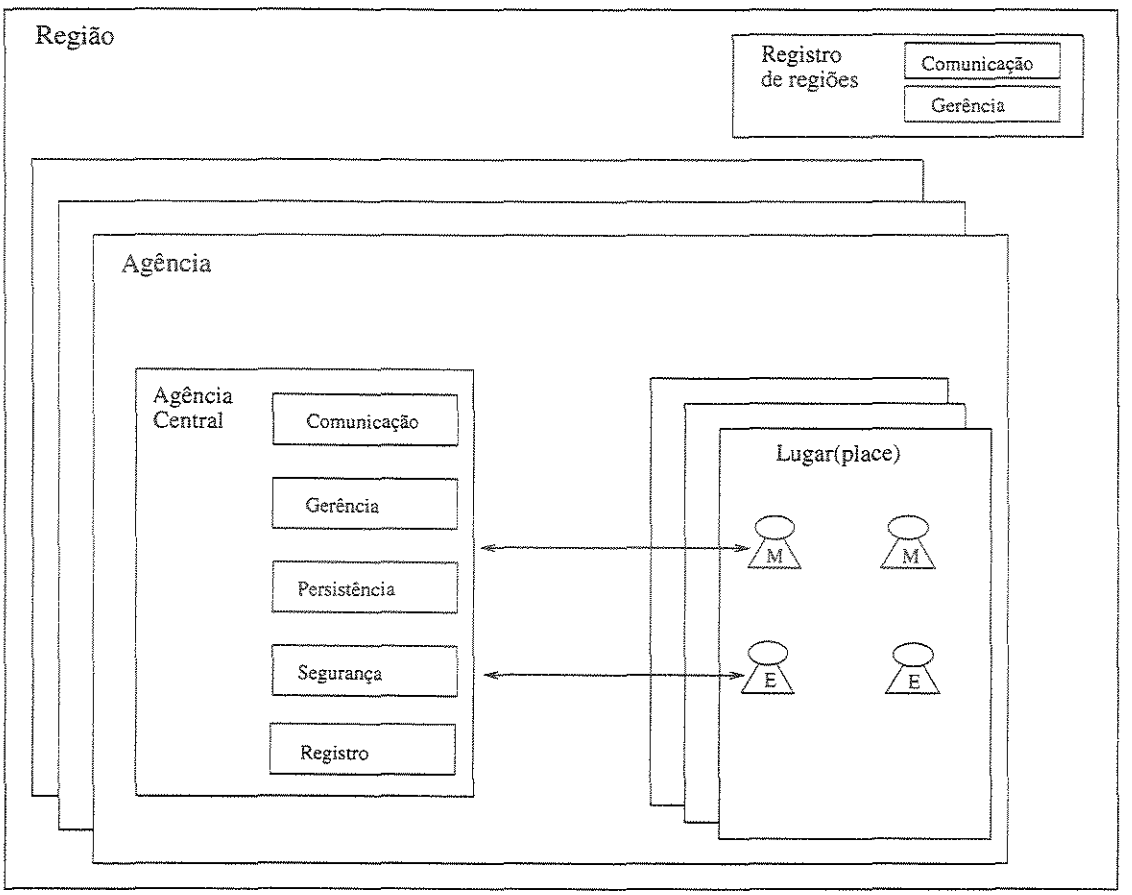



Figura 2.5: Composição hierárquica dos componentes Grasshopper.

## 2.4 Gerência de Redes e Sistemas Distribuídos

A gerência de redes basicamente preocupa-se em gerenciar dispositivos físicos. A gerência de sistemas distribuídos avança no sentido de gerenciar não somente dispositivos físicos e seus protocolos mas também serviços em nível de usuário, abrangendo os recursos para o processamento de serviços de maneira distribuída. Dessa forma a gerência de sistemas distribuídos preocupa-se em gerenciar elementos tais como processos, aplicações, uso de memória e CPU.

Visando padronizar o gerenciamento, a ISO (*International Organization for Standardization*) definiu a arquitetura de gerenciamento ISO/OSI que é baseado em três conceitos:

- Modelo agente/gerente
- Protocolo de gerenciamento

- Objeto gerenciado

Um Objeto Gerenciado (*Managed Object - MO*) representa um recurso, que pode ser um sistema hospedeiro (estação de trabalho, servidor de terminais, etc.), um protocolo, um *gateway* ou um equipamento de transmissão (modem, ponte, roteador, concentrador). Cada objeto gerenciado é visto como uma coleção de variáveis cujos valores podem ser lidos ou alterados.

Um agente é uma peça específica de software (ou *firmware*) que contém informações sobre um dispositivo específico e/ou seu ambiente. De fato, um agente coleta informações relevantes para o gerenciamento da rede junto aos objetos gerenciados. Um agente executa operações sobre os objetos, podendo notificar o gerente, armazenando as informações pertinentes na MIB (*Management Information Base*). A MIB é organizada na forma de árvore e armazena logicamente os dados que representam os objetos gerenciados de um sistema aberto. O papel do gerente é processar as informações recolhidas pelos agentes, com o objetivo de detectar a presença de falhas no funcionamento dos componentes da rede (hosts, *gateways*, processos executando os protocolos de comunicação), para que possam ser tomadas providências no sentido de contornar os problemas que ocorrem como consequências das falhas. O gerente envia comandos aos agentes, solicitando uma leitura ou modificação do valor das variáveis dos objetos gerenciados.

O modelo OSI/CMIP (*Common Management Information Protocol*) desenvolvido pela ISO em 1989, segue o paradigma de orientação a objetos e as seguintes operações podem ser feitas sobre os atributos:

- *Get*: usada na leitura de um atributo ou de uma lista de atributos;
- *Replace*: permite alterar valores de atributos por valores fornecidos;
- *Set*: substitui o valor de atributos por valores especificados quando da definição dos objetos;
- *Add*: é utilizada no caso de atributos cujos valores são definidos como conjunto, permitindo o acréscimo de novos elementos a este;
- *Remove*: usada na remoção de um elemento num conjunto de valores de um atributo;

As operações que podem ser feitas sobre os objetos gerenciados como um todo são:

- *Create*: permite criar um objeto e inicializar os valores dos atributos do mesmo;
- *Delete*: elimina um objeto em particular;

- *Action*: usada para determinar ao objeto que execute uma ação específica e apresente o seu resultado;

O protocolo SNMP surgiu em 1988, padronizado pelo IETF (*Internet Engineering Task Force*), utiliza os mesmos conceitos de agentes e gerentes do modelo OSI/CMIP e compõe a base do sistema de gerência de redes da arquitetura Internet TCP/IP (*Transmission Control Protocol/Internet Protocol*). O SNMP não é orientado a objetos pois não possui os conceitos relacionados a esta abordagem. O protocolo SNMP é centralizado e possui algumas limitações que dificultam a tarefa de gerência de sistemas distribuídos. Neste modelo, a tarefa de gerência da rede é feita pela estação (*host*) de gerência que atua como gerente. Os agentes SNMP recebem comandos do gerente e encaminham informações ou alteram os valores das variáveis que representam os objetos gerenciados. Embora existem outras PDUs (*Protocol Data Units*), abaixo mostramos as cinco PDUs definidas para a primeira versão do SNMP.

- *GET-REQUEST*: usada para recuperar uma informação de gerência específica;
- *GET-NEXT-REQUEST*: retorna o próximo valor armazenado;
- *SET-REQUEST*: altera os valores das variáveis dos objetos;
- *GET-RESPONSE*: responde a uma operação de *GET-REQUEST, GET-NEXT-REQUEST* ou *SET-REQUEST*, contendo os valores dos objetos ou uma mensagem de erro;
- *TRAP*: relata a ocorrência de algum evento.

A Figura 2.6 mostra a interação entre gerente, agente e objeto gerenciado, válida tanto para o modelo OSI/CMIP quanto para o modelo Internet/SNMP.

Existe uma forte tendência para a descentralização do controle de gerência. Um exemplo disso é a gerência por delegação (MbD - *Management by Delegation*) [20], onde o gerente delega funções de gerência ao agente, que poderá "agir" baseado em informações que ele "conhece". Outra alternativa, apresentada na seção 2.4.2, é usar agentes móveis para a gerência de serviços em sistemas distribuídos.

### 2.4.1 Gerência de Sistemas Distribuídos

Atualmente os sistemas distribuídos apresentam novos conceitos e requerem novas formas de gerenciar os serviços oferecidos na rede. Muitos desses serviços são representados por objetos desenvolvidos em linguagens de alto nível. É comum encontrarmos serviços sendo representados por objetos Java ou agentes móveis. É nesse contexto de gerência de

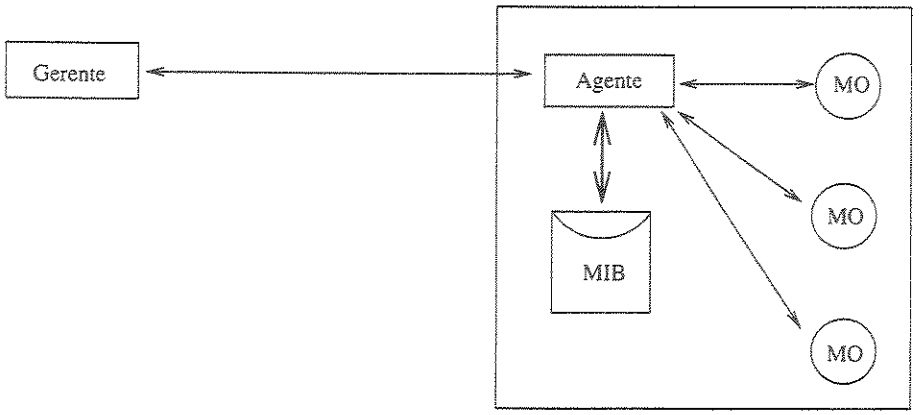



Figura 2.6: Arquitetura de Gerenciamento ISO/OSI e Internet.

sistemas distribuídos que a gerência de serviços está inserida. Para esse trabalho, estamos preocupados em gerenciar objetos Java que representam um serviço ativo a ser oferecido para um ou mais usuários em uma rede ativa. O modelo de gerenciamento OSI classificou as atividades de gerência em 5 áreas funcionais, sendo elas:

- Gerenciamento de Falhas: atua sob as falhas ocorridas nas entidades gerenciadas;
- Gerenciamento de Configuração: responsável pelas modificações a serem feitas na configuração das entidades gerenciadas;
- Gerenciamento de Contabilidade: mantém uma contagem ou contadores do uso de recursos de acordo com diferentes critérios;
- Gerenciamento de Desempenho: provê medidas de desempenho do sistema principalmente para otimização de recursos; e
- Gerenciamento de Segurança: atua sob o gerenciamento da funcionalidade de segurança suportada pelo sistema.

A gerência de sistemas distribuídos aborda as 5 áreas acima e está preocupada em responder questões do tipo:

- Qual recurso está sendo mais acessado?
- Qual o tempo de resposta de determinado servidor?
- Qual serviço está mais lento? Por que?

- Quantas requisições determinado servidor recebeu?
- Qual o serviço mais e menos acessado? Qual o *throughput*?

Assim, o objeto gerenciado, como sendo uma visão externa de um recurso (serviço) gerenciado, deve incluir funcionalidades adicionais que não seriam necessárias para a operação normal do serviço. Atributos que representam as propriedades dos componentes tais como contadores para eventos específicos e operações de gerência (métodos/ações) para acesso às variáveis específicas para gerenciamento devem ser acrescentados ao objeto e a sua interface. Estes métodos e atributos estão relacionados às cinco áreas funcionais, e permitem a coleta dos dados para posterior análise pelo gerente (humano) do sistema. A Figura 2.7 mostra, de uma forma geral, como um serviço (objeto) deve ser estendido a fim de que ele possa ser gerenciado. Essa idéia teve origem no projeto MAScOTTE (*MAnagement Services for Object oriented disTributed sysTEms*) [31] que apresentou as extensões para gerenciamento de objetos CORBA (*Common Object Request Broker Architecture*) .

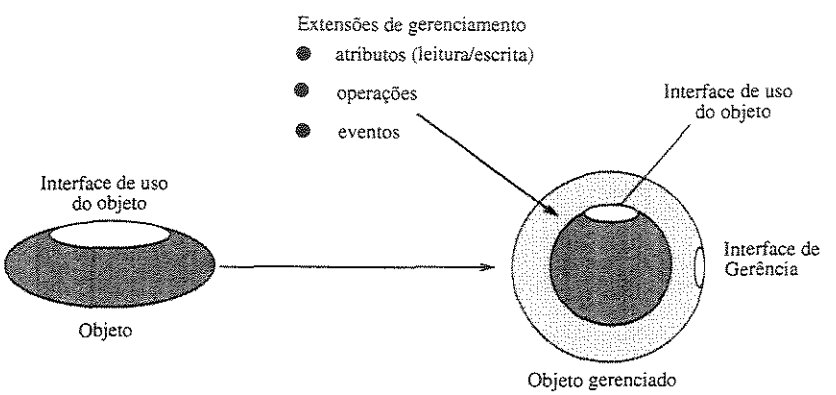



Figura 2.7: Um serviço (objeto) com extensões para gerenciamento.

### 2.4.2 Gerência de Sistemas Distribuídos usando Agentes Móveis

Na área de gerência de redes, principalmente gerência de serviços, os agentes móveis se apresentam como uma boa solução já que tais serviços estão distribuídos pela rede e a característica "saltar-executar-saltar" se ajusta naturalmente em tarefas de gerenciamento e recuperação de informações. Um agente migra para uma máquina, executa uma tarefa, migra para outra e executa outra tarefa dependente da saída precedente [11]. Além disso, a capacidade de realizar a compactação da semântica da informação [10] também é uma forte motivação para usar agentes móveis na gerência de sistemas distribuídos. No modelo

tradicional centralizado, se o gerente deseja pesquisar uma MIB segundo algum critério, uma função de pesquisa pode ser executada na estação de gerência em cada elemento recebido do agente remoto. No pior caso, todos os elementos precisam ser transferidos até a estação de gerência. Se a função de pesquisa fosse colocada no agente, somente o resultado seria enviado para a estação de gerência, reduzindo o custo de comunicação entre agente e estação.

Um agente móvel pode migrar autonomamente ou conforme o gerente ordenar, a fim de coletar dados de gerência, avaliá-los e agir quando detectar algum problema. Um agente poderá informar o gerente sobre algum evento inesperado que, por sua vez, enviaria outro agente móvel especificamente para solucionar o problema ocorrido. A interação autônoma assíncrona permite aos agentes móveis realizarem determinadas tarefas mesmo se a entidade que o delegou não está ativa. Este tópico é considerado em [20]. A facilidade para se definir interfaces e inserir novas funções de gerência nos agentes móveis também contribui para seu uso na área de gerenciamento de sistemas distribuídos.

Com relação aos desafios, o principal problema desta abordagem ainda é a segurança. É necessário que os ambientes de execução ofereçam um certo grau de segurança e limitação de uso dos recursos, a fim de proteger estes ambientes contra ações nocivas executadas por agentes neles recebidos [38].

### 2.4.3 Gerenciamento basedo em Políticas

Atualmente o desenvolvimento de programas que representam serviços tem considerado a aplicação de políticas em seu uso. As políticas definem as regras de uso de um determinado serviço ou de um grupo de serviços e servem para limitar e garantir um certo grau de eficiência e confiabilidade aos usuários do serviço.

Em uma abordagem orientada a objetos, o comportamento externo de um objeto define como ele interage com outros objetos em seu ambiente. Segundo [39], política é a informação que influencia nas interações entre um sujeito e um alvo, especificando o relacionamento entre este sujeito e este alvo. Uma política pode ser de autorização ou de obrigação. Uma política de autorização define quais atividades um sujeito pode realizar sobre um determinado alvo. Políticas de obrigação definem quais atividades um sujeito deve ou não realizar sobre um determinado alvo.

Em ambientes Telecom, a definição de políticas é muito importante já que um serviço pode ser usado por vários clientes. Nesse tipo de ambiente é muito comum um serviço migrar entre regiões para atender a demanda dos usuários.

A existência de políticas para um serviço é de extrema importância para a área de gerência de redes. Nessa dissertação a definição de políticas para os serviços nos permitiu simular de forma muito aproximada ambientes de Telecomunicações. Além do mais,

deve existir um eficiente mecanismo de comunicação entre a parte que gerencia e a parte gerenciada a fim de aplicar, para cada evento ocorrido, suas políticas definidas a fim de controlar e até mesmo evitar situações indesejadas no ambiente gerenciado.

## 2.5 ANTS

O ANTS (*Active Node Transfer System*) [43] é um *toolkit* desenvolvido para suportar aplicações ativas. Uma rede baseada em ANTS consiste de um grupo de nós conectados sendo que cada elemento do grupo executa o ambiente ANTS [7]. O ANTS segue o modelo da abordagem integrada (seção 2.1) carregando juntamente com os dados o código a ser executado nos ambientes de execução. Assim, no ANTS os pacotes das redes tradicionais são substituídos por cápsulas. As cápsulas representam o processamento a ser realizado em cada nó sobre os dados específicos. Dados que são carregados pelas próprias cápsulas ou coletados (gerados) no próprio nó ativo local. Seguindo a terminologia padrão das redes ativas, no ANTS os roteadores e nós finais são substituídos por nós ativos que, portanto, executam o código das cápsulas e mantêm estado. No ANTS é possível que os nós possuam sua própria *cache* onde os dados ou programas podem ser armazenados para posterior uso. O estado de um nó ANTS pode ser representado pelos programas que estão no nó e pelos dados que cada programa possui. O envio e recebimento de cápsulas é feito através de um mecanismo de distribuição de código que transfere rotinas automática e dinamicamente para os nós ativos onde elas são necessárias.

O crescente esforço a fim de flexibilizar o desenvolvimento de novos protocolos motivou a implementação do ANTS. Para a equipe do ANTS, os nós deveriam suportar uma grande variedade de novos protocolos, todos sendo usados simultaneamente na rede. Esses protocolos poderiam ser desenvolvidos e usados simplesmente através de um acordo entre as partes interessadas no protocolo, sem a necessidade de um registro centralizado ou de um acordo entre partes que não estão interessadas. Alcançar o consenso entre organizações ou grupos é algo extremamente complexo já que atender aos interesses, muitas vezes econômicos, de todos é quase impossível. Por último, desenvolver novos protocolos dinamicamente sem a necessidade de "parar" a rede, foi também uma motivação para a especificação do ANTS. A seguir, apresentamos o modelo de programação do ANTS e como suas classes são usadas para desenvolver aplicações ativas.

### 2.5.1 O Modelo de Programação do ANTS

O ANTS pode ser descrito de forma geral como um mecanismo para envio e recebimento de pacotes. Os pacotes incluem uma referência para o rotina que os envia (*forward*) entre os nós da rede. Essa rotina é avaliada quando o pacote chega em um nó. É através dela

que outras funções da cápsula serão disparadas, realizando tarefas específicas para um usuário ou aplicação. Isso permite que os usuários da rede controlem o fluxo dos pacotes que eles enviam, mas não os pacotes que eles recebem. A rotina de *forward* irá decidir para qual nó a cápsula deverá ser enviada. A decisão é baseada na definição do usuário e nas informações do nó atual.

Em relação à segurança, o ANTS desenvolveu um modelo simples mas que garante um certo nível de confiança. É necessário considerar que as rotinas de *forwarding* podem ser definidas por usuários não confiáveis e, portanto, devem ter suas capacidades limitadas. No ANTS espera-se que essas rotinas sejam executadas em um curto período de tempo e sem a possibilidade de escalonamento, ou seja, se outras cápsulas estiverem chegando no nó, seu processamento será adiado e colocado em uma fila por ordem de chegada. Não é possível que uma cápsula comece a ser avaliada, seja interrompida para que o ambiente de execução avalie outra cápsula, e retorne a ser avaliada posteriormente. Essas características apresentadas garantem algum nível de proteção, alocação e desempenho. A proteção é baseada na capacidade de realizar processamento somente nos pacotes que pertencem as suas aplicações, ou seja, é impossível o processamento de pacotes de outros usuários ou outras aplicações. A alocação é alcançada pois os recursos que cada pacote poderá usar em cada nó são limitados. Finalmente, aspectos relacionados ao desempenho são obtidos devido à simplicidade do modelo de processamento.

A transferência de código no ANTS ocorre usando a tecnologia de código móvel, serializando e desserializando objetos. A transferência, tanto de código como de estado, é feita através de *sockets* UDP (*User Datagram Protocol*).

O ANTS foi desenvolvido em Java e suas aplicações ativas, até o momento, também devem ser escritas em Java. Uma rede em ANTS é personalizada pelos seus usuários em termos de protocolo. Novos protocolos e as aplicações que fazem uso desses protocolos podem ser desenvolvidos muito facilmente apenas usando (herdando) as classes virtuais oferecidas pelo ambiente. Assim, um novo protocolo é desenvolvido herdando a classe virtual Protocolo (*Protocol*). É nesta classe que serão identificados os diferentes tipos de pacotes que formarão o protocolo. Cada pacote é especificado herdando a classe Cápsula (*Capsule*). Uma nova aplicação é criada herdando a classe Aplicação (*Application*) e uma instância da classe Nó (*Node*) representa o ambiente (*runtime*) ANTS local daquele nó. Existe ainda, entre as principais classes do ANTS, a classe Extensão (*Extension*) que permite aumentar a oferta de serviços disponíveis no nó. A seguir, uma visão mais detalhada de cada classe será apresentada.

### Cápsula

Uma cápsula é composta por um pacote e a sua rotina de *forwarding* que é executada em cada nó ativo que a cápsula visita enquanto estiver na rede. Todas as cápsulas herdam

os seguintes métodos para manipulação do cabeçalho:

- **getSource()** - Retorna o endereço origem da cápsula como um inteiro de 32 bits. O endereço origem é definido implicitamente pelo nó que inseriu a cápsula na rede.
- **getDst()** - Retorna o endereço destino da cápsula como um inteiro de 32 bits.
- **setDst()** - Muda o endereço destino da cápsula.
- **evaluate()** - Este é o principal método da cápsula, ou seja, é a rotina de *forwarding*. É o primeiro método a ser processado e é através dele que outras funções serão invocadas.
- **serialize()** - Na versão do ANTS que usamos (1.3), este método passou a se chamar *encode* mas com a mesma tarefa do anterior, ou seja, criar uma representação linear do objeto cápsula para transmissão na rede.
- **deserialize()** - O nome do método também mudou na versão que usamos passando a se chamar *decode*. Esse método cria um objeto cápsula recebido através de um buffer da rede.
- **length()** - Calcula o tamanho da cápsula serializada.

Os métodos de serialização são usados pelos nós para enviar e receber cápsulas. Entre os nós da rede a comunicação exige que os objetos sejam convertidos para uma representação especial, ou seja, uma seqüência linear de bytes. Esta conversão usa uma biblioteca conhecida como XDR (*eXternal Data Representation*) tornando a sequência de bytes universal e possível de ser reconhecida em outros nós. Nesses nós destinos o processo inverso é feito também usando a biblioteca XDR.

### Nó

A classe Nó (*Node*) implementa um único nó ativo. Ela exporta seus serviços para cápsulas, aplicações e extensões. As cápsulas precisam dos serviços do nó para processarem suas rotinas de *forwarding* e implicitamente transferirem seu código. As aplicações usam os serviços do nó para registrar os protocolos e enviar e receber cápsulas. Por fim, as extensões (ver abaixo) são usadas para aumentar as funcionalidades e os serviços locais do nó.

As cápsulas acessam os serviços do nó através de seu método *evaluate*. Um parâmetro é passado como referência para o nó local que então oferece os seguintes serviços:

- **getAddress()** - Obtém o endereço do nó local.

- **routeForNode()** - Envia uma cápsula para um determinado nó, usando a tabela de roteamento. Este método também é muito importante pois permite que uma cápsula seja transferida pelos nós intermediários somente utilizando as funções oferecidas pela classe nó.
- **deliverToApp()** - Entrega uma cápsula para uma determinada aplicação naquele nó. É através desse método que a cápsula transfere seus dados para uma aplicação.
- **softstate.put()** - Coloca um objeto na *cache* do nó local para posterior uso. É uma maneira de expandir as funcionalidades do nó já que pode-se colocar qualquer objeto que represente qualquer serviço a ser oferecido para os usuários.
- **softstate.get()** - Obtém um objeto que foi colocado na cache.

Um nó ativo do ANTS é identificado através de um único endereço. As funcionalidades oferecidas pelo nó podem ser usadas de várias formas conforme a necessidade dos usuários. Colocar objetos que representam serviços na *cache* será de extrema importância para nossa aplicação voltada para ambientes Telecom. Os serviços serão instalados pelas cápsulas em nós específicos. Após isso, serão enviadas as cápsulas que representam os clientes ou aplicações e que usarão os serviços anteriormente instalados.

### Extensão

As extensões de um nó aumentam a oferta de serviços daquele nó. Essas extensões podem ser componentes que serão colocados pelas cápsulas e que serão usados pelas aplicações específicas de cada usuário. Primeiramente, uma cápsula deverá localizar uma extensão no nó para depois acessar seus serviços diretamente. Novas extensões podem ser criadas herdando a classe Extensão que serão instaladas no nó como parte da configuração local.

### Protocolo

Os protocolos servem para agrupar tipos de cápsulas relacionados. Os protocolos são formados por grupos sendo que cada grupo é formado por um conjunto de cápsulas, como mostra a Figura 2.8. A cápsula é a única peça dessa hierarquia que transporta código pela rede.

Novos protocolos são desenvolvidos herdando a classe Protocolo. Quando o protocolo é instanciado, ele deve ser registrado no nó antes de uma aplicação enviar ou receber cápsulas que pertencem àquele protocolo. Abaixo criamos, a título de exemplificação, um protocolo com duas cápsulas definidas em um grupo. Esse protocolo é usado para instalar e remover serviços em nós da rede. A primeira cápsula instala um serviço em algum host na rede e a segunda cápsula remove um determinado serviço do nó.

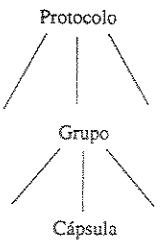



Figura 2.8: Composição hierárquica de uma cápsula.

```
//definições iniciais da classe.
startProtocolDefn();
startGroupDefn();
    addCapsule(''CapsulePutService'');
    addCapsule(''CapsuleRemoveService'');
endGroupDefn();
endProtocolDefn();
```

Por fim, os métodos *register* e *unregister* são usados para registrar e desregistrar, respectivamente, um protocolo do nó.

### Aplicação

Uma aplicação representa a entidade que usa a rede e serve para enviar e receber cápsulas e executar atividades independentes relacionadas a um determinado usuário. Novas aplicações são criadas herdando a classe Aplicação. Uma aplicação é a interface entre o usuário da rede ativa e a rede ativa propriamente dita. Cada usuário, que deseja usar a rede para enviar ou receber pacotes ativos, deverá possuir um objeto que instancie a classe Aplicação. Os principais métodos oferecidos pela classe são:

- **send()** - Insere uma cápsula na rede através do nó local.
- **receive()** - Recebe (*Upcall*) uma cápsula da rede através do nó local. Quando o nó local invocar o método *deliverToApp()* explicado anteriormente, a cápsula sendo processada pelo nó naquele instante será transferida para a aplicação que a receberá usando o método *receive*.

- **port()** - Retorna a porta na qual a aplicação está conectada ao nó local. O nó oferece o método *attachApplication* que conecta uma aplicação a uma determinada porta do nó.

Existe ainda a classe Canal (*Channel*) que fornece a interface para a camada de enlace, conectando os nós de maneira ponto-a-ponto ou por canais compartilhados. Até o momento, as cápsulas podem ser transferidas diretamente via protocolo Ethernet ou via UDP. Em redes menores é possível executar um nó por máquina e conectar os nós com canais Ethernet. Redes maiores são emuladas executando muitos nós por máquina conectando-os via UDP.

### 2.5.2 Configuração

Antes de poder realmente usar o ambiente ANTS é necessário definir a topologia da rede, ou seja, quais nós formarão a nova rede ativa. Essas definições são colocadas em um arquivo texto especificando toda a configuração da rede: endereços dos nós, endereços das aplicações, endereços e portas dos canais e os parâmetros de inicialização necessários para ativar cada nó da rede ativa. Após isso, utiliza-se a ferramenta *makeroutes* que irá gerar um arquivo com as rotas entre os nós ativos da rede. Abaixo mostramos um arquivo exemplo que define uma pequena rede ativa formada por quatro nós. Nesta demonstração, cada nó final possui uma aplicação ativa que é usada pelo usuário para o envio e recebimento de cápsulas. O parâmetro "destino" informa para onde as cápsulas criadas pelo nó local serão enviadas. É uma maneira de definir a comunicação entre aplicações ativas. O parâmetro *"routes"* especificado para cada nó serve para definir o nome do arquivo que será gerado através da ferramenta *makeroutes*.

```
node 1.1.1.3 -routes testeSendVN2.routes
channel 1.1.1.3 iguacu.dcc.unicamp.br:9001 -log 255
application 1.1.1.3 mgmtsw.AplicacaoSend -destino 1.1.1.1

node 1.1.1.5 -routes testeSendVN2.routes
channel 1.1.1.5 xingu:9001

node 1.1.1.2 -routes testeSendVN2.routes
channel 1.1.1.2 pinheiros:9001

node 1.1.1.1 -routes testeSendVN2.routes
channel 1.1.1.1 araguaia:9001
application 1.1.1.1 mgmtsw.AplicacaoReceive -destino 1.1.1.3
```

```
connect 1.1.1.3 1.1.1.1
connect 1.1.1.3 1.1.1.5
connect 1.1.1.3 1.1.1.2
```

Este arquivo especifica uma rede ativa formada pelos hosts iguaçu, xingu, pinheiros e araguaia, que estarão usando a porta 9001 para comunicação. As aplicações ativas estão localizadas nos hosts iguaçu e araguaia, e os outros nós possuem apenas uma instância da classe nó. A diretiva *connect* irá especificar as conexões entre os hosts definindo as rotas a serem geradas pelo *makeroutes* posteriormente. Essas rotas estarão em um formato específico permitindo que os nós inicializem suas tabelas de roteamento.

Após as configurações acima estarem prontas, uma instância da classe *ConfigurationManager* poderá ser criada. Este objeto irá ler os arquivos de configuração e iniciar um de seus nós localmente. Isto inclui criar a instância da classe nó (*node runtime*), as aplicações (se houver para aquele nó), e extensões (se houver para aquele nó) conectando uma com as outras e iniciando suas operações. O usuário, através da aplicação ativa, poderá então usar a rede ativa personalizando-a através do envio de serviços conforme desejar.

## 2.6 Trabalhos Relacionados

Primeiramente é importante destacar que existem alguns trabalhos sobre o uso das redes ativas para gerenciar as redes tradicionais (redes IP), seus dispositivos físicos e protocolos, como pode ser visto em [19]. Neste caso, a tecnologia ativa é usada como uma ferramenta para desenvolver aplicações de gerência. Pode-se criar cápsulas responsáveis pela coleta de dados de gerência e enviá-las para os dispositivos desejados. Em [15] são desenvolvidos alguns programas de gerenciamento que podem ser agrupados em um sistema de gerência e utilizados pelos gerentes. São aplicações típicas para descobrir informações específicas em nós, tráfego de pacotes IP (pacotes por segundo) em uma porta específica em um determinado intervalo, entre outras.

Nesta dissertação a tecnologia das ANs é estendida para fornecer serviços para ambientes de Telecomunicação. A flexibilidade e a padronização de interfaces ao invés de protocolos faz com que as ANs desempenhem um papel importante em áreas que exigem o desenvolvimento e a instalação de serviços customizados de maneira rápida e eficiente. Baseando-se nestas características, Brunner [13, 12, 1, 14] propôs um *framework* para instalar e gerenciar serviços em ambientes de Telecomunicação que usam a tecnologia ativa. O modelo propõe que o fornecimento de um determinado serviço seja dividido em duas partes. Na primeira parte o cliente solicita a instalação de uma VAN em seu domínio. Na segunda parte instala o serviço desejado. Pode haver uma negociação entre

cliente e fornecedor para definir a qualidade dos serviços e a disponibilização dos recursos, quando estes são propriedades do fornecedor. A gerência dos serviços pode ser feita pelo cliente, através de um sistema de gerência próprio, ou pelo fornecedor usando o seu sistema de gerência. O modelo proposto por Brunner não considera serviços móveis nem o gerenciamento baseado em políticas.

Alguns trabalhos consideram serviços móveis, não somente em ambientes baseados na tecnologia das redes ativas mas também baseados na tecnologia de agentes móveis. Em [16] são descritos os conceitos de um sistema de gerência para serviços baseados em agentes móveis. Neste caso, considera-se que os serviços são agentes móveis e, portanto, podem mover-se de um local para outro autonomamente. A idéia dos autores é criar um agente que "segue" o serviço na medida em que ele migra, ou seja, para cada serviço móvel haverá um agente móvel de gerência que armazenará as informações coletadas e notificará um gerente central sobre eventuais problemas. Este agente móvel de gerência possui autonomia para agir independentemente quando detectar situações indesejadas no ambiente. Este modelo também não considera o gerenciamento baseado em políticas e apresenta-se inviável para ambientes onde há um grande número de serviços já que é necessário um agente de gerência para cada serviço móvel.

Em [4], os autores consideram o uso de políticas de gerenciamento para os programas dos usuários e também definem um serviço de notificação responsável pela transmissão dos eventos para outras entidades. Além disso, é descrito como os Algoritmos Genéticos podem ser integrados com o gerenciamento baseado em políticas. Este modelo também não considera serviços móveis.

O modelo proposto nesta dissertação considera serviços móveis e o gerenciamento baseado em políticas. Não consideramos a etapa de negociação, definição da qualidade dos serviços e disponibilização dos recursos, sendo assunto para trabalhos futuros. Definimos um modelo hierárquico que minimiza o número de agentes de gerência. Com isso, o modelo apresenta-se viável para ambientes onde há um grande número de serviços, tipicamente em ambientes de Telecomunicação. Além disso, não desenvolvemos somente um sistema de gerência mas sim um *framework* para o fornecimento e instalação de serviços. As políticas definidas exigiram o desenvolvimento de componentes para controlar as migrações e a aplicação destas políticas sobre o ambiente gerenciado.

Tem-se discutido que a gerência de serviços ativos deveria ser delegada aos usuários da rede como um conjunto de pequenos sistemas de gerenciamento, diminuindo desta forma o custo de gerência [4]. O modelo proposto nesta dissertação atende a esta tendência podendo ser usado por um cliente para gerenciar seu domínio ou por um fornecedor de serviços (SP) para gerenciar vários clientes. Em ambos os casos pode-se realizar a gerência entre domínios (seção 5.2).

Atualmente existe um grande projeto de pesquisa e desenvolvimento formado por

instituições européias e uma universidade americana, cujo objetivo é desenvolver uma arquitetura de rede aberta, flexível, programável (segura, confiável e gerenciável) baseada nos conceitos das redes ativas [24]. O projeto propõe uma nova arquitetura para redes ativas integrando objetos distribuídos e agentes móveis e pretende realizar exaustivas comparações e testes entre as redes tradicionais e as rede ativas. Baseando-se nesta avaliação, o projeto pretende criar recomendações sobre o uso das redes ativas em grande escala.

# Capítulo 3

# O Modelo de Gerência

Nesta seção apresentamos os componentes da infra-estrutura desenvolvidos para essa dissertação. Dentre os componentes, temos os agentes de gerência responsáveis pela coleta e tratamento das informações para as áreas de contabilidade, desempenho e configuração. Outros componentes foram desenvolvidos para suportar funcionalidades exigidas pelos ambientes Telecom, tais como o Fornecedor de Serviços (SP) e o Gerente de Migrações (*Migration Manager - MM*).

Primeiramente, apresentamos as informações que estamos interessados em obter, informações estas relacionadas às áreas de contabilidade e desempenho. Mostramos como estas informações são coletadas nos diferentes níveis de gerência. Em cada nível de gerência é armazenado um conjunto de informações coletadas. A esse conjunto de informações chamamos de MIB, seguindo a terminologia comumente usada na literatura para representar as bases de dados de gerência. Em cada nível temos uma MIB criada pelo agente de gerência daquele nível.

## 3.1 Informações de Gerência

As perguntas que estamos interessados em responder estão diretamente relacionadas à necessidade de detectar problemas ou situações indesejadas no ambiente gerenciado. Por isso, as perguntas incluem a área de contabilidade, onde queremos saber, por exemplo, qual o serviço mais requisitado em uma VAN, ou em um host; e também incluem perguntas relacionadas à área de desempenho onde queremos saber, por exemplo, qual o host mais sobrecarregado em relação a média geral. Com base nessas respostas, o sistema de gerência permite que o gerente (humano) avance no sentido de agir em relação a uma possível (re)configuração do ambiente gerenciado, por exemplo, migrando um serviço de um host para outro para balanceamento de carga. Assim, as perguntas que estamos interessados em responder podem ser divididas em quatro grupos:

- por serviço;
- por host;
- por VAN; e
- em toda rede (domínio).

Esses quatro níveis para coleta de dados definidos de forma hierárquica garantem um grau maior de organização para o gerenciamento de serviços que se movem entre as redes ativas virtuais, evitando assim um excessivo custo de comunicação entre os agentes de gerência localizados em diferentes VANs. Além disso, fica fácil atribuir para cada gerente a responsabilidade de coletar dados somente nos níveis a ele especificados.

A distribuição dos agentes de gerência através dos níveis hierárquicos definidos ocorre da seguinte forma:

- **Um agente por host por rede virtual** - Uma rede virtual é composta por hosts. Em cada host haverá um agente de gerência responsável por aquela rede virtual naquele host. Um mesmo host poderá pertencer a mais de uma rede ativa virtual, então o número de agentes de gerência em um determinado host irá depender da quantidade de VANs que possuem o host em seu domínio. Nesse primeiro nível, o agente é responsável pela coleta de dados por serviço e por host.
- **Um agente por rede virtual** - Este agente será o gerente da rede ativa virtual.
- **Um agente central** - Este gerente está localizado no nível mais alto sendo responsável pela organização das informações de todas as VANs que pertencem ao domínio sendo gerenciado.

As perguntas de contabilidade selecionadas para o modelo são algumas das muitas que poderão ser acrescentadas conforme a necessidade mais específica de cada ambiente. Dessa forma, apresentamos abaixo algumas dessas perguntas para a área de contabilidade usadas nesse trabalho.

1. Qual é o serviço mais e menos requisitado;
2. Qual é a quantidade de invocações recebidas em um determinado serviço;
3. Qual é o host mais e menos requisitado;
4. Qual é a quantidade de invocações em um determinado host;
5. Qual é o tempo de residência dos serviços;

6. Qual é o *throughput* em um determinado período de tempo;

Essas perguntas representativas podem ser distribuídas pelos quatro níveis de gerência como mostra a Tabela 3.1.

Tabela 3.1: Distribuição das perguntas de contabilidade pelos quatro níveis de gerência.

| por serviço | por host | por VAN | toda rede |
|---|---|---|---|
| | 1 | 1 | 1 |
| 2 | | | |
| | | 3 | 3 |
| | 4 | | |
| 5 | | | |
| | 6 | | |

Da mesma forma que as perguntas de contabilidade, algumas das perguntas representativas relacionadas à área funcional de desempenho foram escolhidas. São elas:

1. Qual é o uso de memória e CPU;

2. Quais hosts estão com o uso de memória e/ou CPU acima dos limites especificados;

Essas perguntas podem ser distribuídas em três níveis de gerência, já que a área de desempenho não atinge o nível de gerência de serviços especificamente. Na Tabela 3.2 podemos analisar essa distribuição. A hierarquia de níveis modelada é unicamente usada para a coleta de dados de contabilidade já que para a coleta de dados de desempenho, os agentes de gerência serão enviados para os hosts específicos onde coletarão informações e retornarão com os dados coletados. As etapas para a coleta de dados tanto para a área funcional de contabilidade como para a área funcional de desempenho serão explicadas ainda nesta seção.

Tabela 3.2: Distribuição das perguntas de desempenho por três níveis de gerência.

| por host | por VAN | toda rede |
|---|---|---|
| 1 | | |
| | 2 | 2 |

Com os dados acima, coletados e tabulados, um conjunto de informações é fornecido para o gerente. A análise dessas informações permite a detecção de possíveis problemas no

ambiente. Dessa forma, definimos três ações de configuração que poderão ser executadas pelo gerente a fim de evitar ou amenizar situações indesejadas. Essas ações de configuração foram definidas com base nas políticas (seção 3.1.1) criadas para os serviços. As três ações de configuração são:

1. Mover um serviço de um host para outro dentro da mesma VAN com propósito de balancemanto de carga;
2. Criar uma nova instância de um serviço;
3. Excluir um serviço do ambiente.

A primeira ação de configuração pode ser usada quando o gerente detectar que um determinado host está muito sobrecarregado. Ele poderá então migrar algum dos serviços que estão localizados no host para outro local interno à VAN, de preferência para o host menos sobrecarregado. Migrar um serviço entre VANs somente é possível quando o cliente de uma VAN solicitar um serviço localizado em outra VAN (este cenário será detalhado na seção 3.5). Permitir que o gerente migre um serviço entre VANs causaria problemas relacionados à segurança, aspecto não considerado nesse trabalho. A segunda ação de configuração cria uma nova cópia do serviço enviando-a para algum host específico na rede. Por fim, a terceira ação de configuração assume que muitas cópias de um determinado serviço podem ser criadas devido a uma demanda crescente em um determinado período e, portanto, em algum momento no futuro elas deveriam ser eliminadas. A seguir, apresentamos as políticas definidas e a relação que as mesmas têm com as ações de configuração.

### 3.1.1 As políticas definidas e as relações com as ações de configuração

Nessa dissertação, além de definirmos as políticas para os serviços, definimos também uma política para o ambiente gerenciado (política 4 abaixo). Assim, temos as seguintes políticas:

1. Um serviço pode ser definido como interno ou externo à VAN;
2. Um serviço possui um limite máximo de migrações em um certo período de tempo;
3. Um serviço possui um número máximo de instâncias criadas na rede;
4. Existe um serviço menos usado recentemente na rede (*Least Recently Used Service - LRUS*);

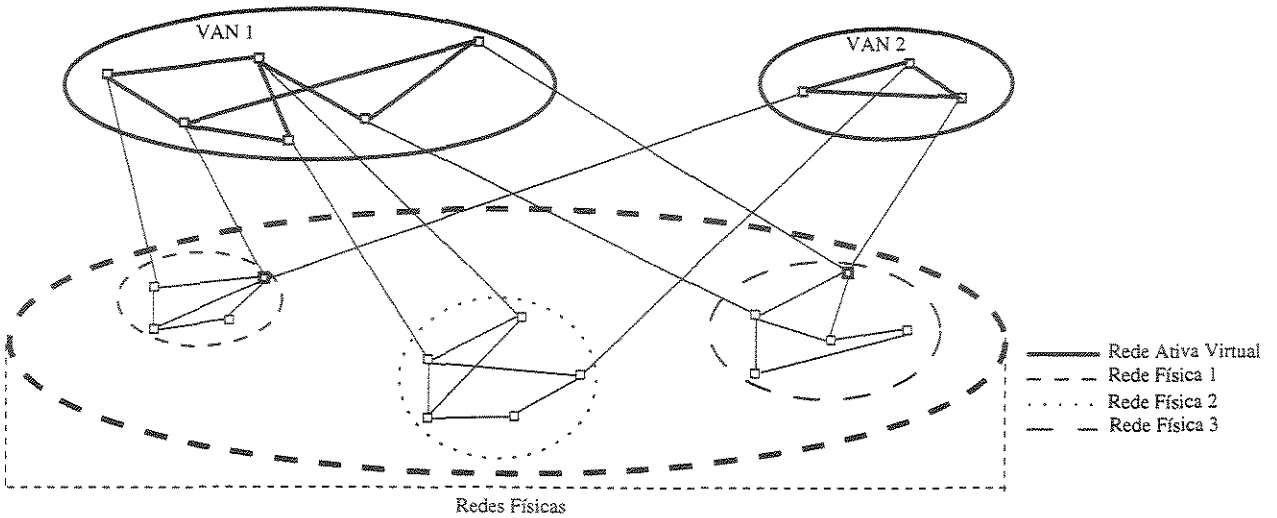



Figura 2.2: Uma Rede Ativa Virtual construída a partir de redes físicas.

lizados naquele nó. Nesses modelos, a gerência é feita usando a própria tecnologia ativa ao contrário do nosso modelo que usa tecnologia de agentes móveis. Na Figura 2.3 podemos observar que as portas físicas de entrada e de saída são compartilhadas pelos vários ambientes de execução do nó. Da mesma forma, a CPU e a memória também são compartilhados por todos os ambientes de execução. Cada ambiente de execução gerencia o acesso das suas aplicações aos recursos oferecidos pelo nó. E, no nível mais alto, o sistema operacional do nó gerencia o acesso aos recursos de cada ambiente de execução.

Uma VAN captura a funcionalidade e os recursos que um fornecedor oferece para um cliente. Da mesma forma que uma rede ativa pode ser entendida como uma generalização de uma rede tradicional, uma VAN pode ser vista como uma generalização de uma VPN tradicional. Como em uma VPN, uma VAN pode ser usada por um cliente para executar serviços usando a estrutura física de um fornecedor. Ao contrário de uma VPN, uma VAN oferece ao cliente um grau muito maior de flexibilidade e controle, permitindo não somente que serviços sejam criados mas também que sejam gerenciados pelos próprios clientes.

### 2.2.1 Redes Ativas Virtuais em ambientes Telecom

Arquiteturas de Telecomunicação tradicionais geralmente permitem o processamento de informações dentro da rede e suportam a criação de novas classes de serviços. No caso da rede telefônica, a arquitetura de Redes Inteligentes (*Intelligent Network - IN*) inclui um ambiente para criação de serviços. Nesses ambientes os serviços podem ser criados

relação à segurança. É necessário um eficiente mecanismo para garantir a confiabilidade de serviços disponibilizados na rede. Em [37] são citados e comentados alguns procedimentos que deveriam ser considerados na medida que o uso efetivo das redes ativas aconteça na prática. Aspectos relacionados à autenticação de usuários e de código, assim como autenticação de fornecedores e servidores (hosts) poderiam ser usados. Neste ambiente distribuído e aberto deve-se ter certeza de que a origem de cada cápsula que é transferida pelos nós ativos na rede seja totalmente confiável.

Uma questão importante que deve ser considerada em redes ativas refere-se ao grau de flexibilização de acesso aos objetos estáticos, ou seja, como uma cápsula sendo processada em um ambiente de execução poderá acessar outras informações que estão fora desse ambiente transiente. Muitas cápsulas necessitam acessar informações específicas do nó ou de serviços, tais como as tabelas de roteamento e o estado dos links de transmissão do nó. Uma das abordagens considera a implementação de métodos externos ao ambiente transiente que fornecem acesso controlado aos recursos do nó, como se fosse uma API (*Application Programming Interface*) do ambiente de execução para as aplicações. Tais aplicações, quando desejarem acessar componentes externos, deverão invocar métodos universais disponibilizados pela API.

## 2.2 Redes Ativas Virtuais

Do ponto de vista do fornecedor, uma VAN é a entidade pela qual os recursos de uma rede são compartilhados, assim como os clientes que usam a infra-estrutura do fornecedor são isolados um do outro. Do ponto de vista do cliente, uma VAN permite a instalação e o gerenciamento de serviços de redes ativas sem a interação com o fornecedor [14].

Uma Rede Ativa Virtual (*VAN*) pode ser descrita como um grafo de nós ativos virtuais conectados por links virtuais. Os nós ativos virtuais são também chamados de Ambientes de Execução, seguindo a terminologia citada em [29]. Uma VAN pode representar o interesse particular de um cliente conforme será explicado na próxima seção. Graficamente uma VAN pode ser visualizada conforme a Figura 2.2.

Um nó ativo virtual possui recursos para realizar processamento e armazenamento de informações sendo que um único nó ativo (físico) pode executar vários nós ativos virtuais pertencentes a diferentes VANs [14], ou seja, um nó ativo pode executar vários ambientes de execução, cada um representando uma rede ativa virtual específica. Note na figura acima, que um nó da rede física 1 e um nó da rede física 3 são compartilhados pelas VANs 1 e 2, caracterizando a sobreposição de VANs.

Da mesma forma, um nó de uma rede ativa virtual pode ser visto na Figura 2.3. Podemos perceber nesta figura a presença de um ambiente de execução para gerenciamento. Esse ambiente é usado em alguns modelos de redes ativas para gerenciar os serviços loca-

### 2.1.2 Vantagens e Aplicações

A principal motivação para uso de redes ativas é a facilidade de criação e disponibilização de serviços na rede. Algumas aplicações atualmente poderiam fazer uso da tecnolgia das redes ativas devido à necessidade de processamento interno à rede. A padronização de protocolos, que normalmente é um processo longo, pode ser evitada e substituída pela especificação de interfaces de programação de serviços. Da mesma forma, as cápsulas podem executar tarefas nos nós ativos e, portanto, as informações coletadas localmente servirão para a correção imediata de possíveis problemas evitando um excessivo tráfego na rede. Além disso, a possibilidade de *Caching* usada para armazenar informações nos nós intemediários da rede, juntamente com a filtragem de pacotes evitando que cápsulas menos importantes atravessem a rede sem necessidade, permitem também uma considerável redução de tráfego [1]. Esta filtragem de pacotes pode ser aplicada de forma mais específica em *Firewalls* para determinar quais pacotes poderiam ser transferidos de forma transparente de um ponto a outro na rede e quais deveriam ser bloqueados. Outra aplicação ocorre para nós que suportam mobilidade [6]. Nesse tipo de aplicação os nós estão localizados em pontos estratégicos na rede. Esses nós, além de outras tarefas, podem agir como uma ponte entre redes com largura de banda e características de confiabilidade diferentes, podendo, por exemplo, aplicar algoritmos de criptografia em níveis diferentes. Isto é comum em junções entre uma rede com fio (*wired*) e uma rede sem fio (*wireless*). Por último, as facilidades de transferência de código na rede possibilitam que as redes ativas sejam usadas para gerência de dispositivos de rede tradicionais tais como roteadores e comutadores, neste caso substituindo o protocolo SNMP (*Simple Network Management Protocol*) [19].

Recentemente, as redes ativas foram apontadas como uma tecnologia adequada para ser aplicada em ambientes de Telecomunicação. Nesse tipo de ambiente existe a necessidade de fornecer serviços de forma rápida e uniforme sem comprometer a qualidade do produto. Em [18], os autores apresentam algumas aplicações de ambientes Telecom que podem ser implementadas usando tecnologia de agentes móveis. Essas aplicações agem como serviços sendo oferecidos para usuários por um fornecedor ou uma companhia telefônica. O serviço de *Call Forwarding* (CF), apresentado naquele trabalho e comentado na seção 5 desta dissertação, foi usado para validar o nosso modelo de gerência. O serviço de CF realiza o roteamento de chamadas que chegam até o seu usuário para outros dispositivos, dependendo da data ou hora do dia, ou ainda, dependendo da ocorrência de eventos específicos.

### 2.1.3 Segurança

É claro que alguns desafios deverão ser enfrentados pelas redes ativas principalmente em

(CPU, memória, etc.) podendo mudar o estado não transiente do nó. O programa de uma determinada cápsula poderá acessar componentes estáticos localizados nos nós. Esses componentes estacionários podem obter dados sobre o nó, por exemplo, para tarefas de gerenciamento. Os dados podem ser posteriormente coletados pelas cápsulas enviadas por uma aplicação de gerência.

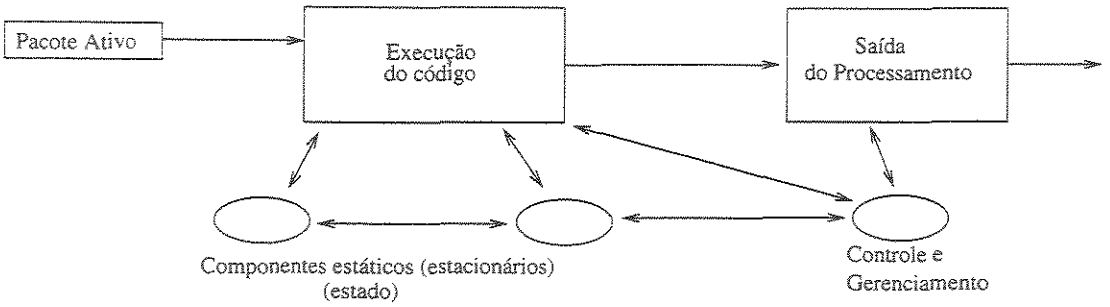



Figura 2.1: Processamento de pacotes ativos em um nó de uma rede ativa.

Como podemos ver, as redes ativas transportam *Pacotes Ativos* [29] também chamados de *Cápsulas*, e é neles que o código representando programas é transportado para diferentes localizações na rede. Com essa idéia, o paradigma das redes ativas permite que programas sejam instalados e executados na rede como se estivessem sendo instalados e executados em um computador. Através desta analogia, a rede desempenha o papel do computador que é simultaneamente compartilhado por várias partes (clientes) [14]. Além disso, as conexões e fluxos em redes ativas podem ser mais poderosos do que as conexões e fluxos dos sistemas atuais pois o estado deixado pelas cápsulas nos nós pode estar em forma de programas [9].

As cápsulas podem transportar códigos de programas que quando instanciados representam serviços específicos para um usuário ou aplicação. Estando estes serviços instanciados, os usuários podem usá-los enviando cápsulas para os nós determinados. Esse tipo de situação é muito comum em ambientes de Telecomunicação onde um serviço é colocado em algum nó da rede para atender a uma certa demanda de um usuário ou uma classe de usuários. As tarefas de criação e gerenciamento de serviços em redes ativas são diferentes das tarefas de criação e gerenciamento de serviços em outras tecnologias de rede. Na Internet de hoje, a criação de serviços não inclui a instalação dos serviços na rede, mas somente a instalação em terminais e servidores nas extremidades da rede.

### 2.1.1 Abordagens

Ao longo dos anos definiu-se duas abordagens que consideram a maneira pela qual uma rede ativa pode ser desenvolvida. São elas a Abordagem Discreta e a Abordagem Integrada [9].

**Abordagem Discreta ou comutação programável:** Esta abordagem mantém o formato atual de células / pacotes e fornece um mecanismo discreto para transferência de programas. Separar a injeção de programas do processamento das mensagens pode ser particularmente interessante quando a seleção desses programas é feita pelos administradores da rede e não por usuários finais. Além disso, esta separação também é útil quando se deseja um certo grau de confiabilidade da origem do código a ser executado ou quando o tamanho dos programas é relativamente grande. Aqui os usuários deveriam primeiramente colocar seus programas personalizados nos nós desejados. Após isso, poderiam enviar seus pacotes através desses nós "programáveis" da mesma forma que é feito atualmente. Quando um pacote chegar em um nó, seu cabeçalho será analisado e o programa apropriado é disparado para agir sobre o seu conteúdo que poderá ou não ser alterado. A quantidade de processamento de um nó depende da quantidade de programas que estão processando os conteúdos dos pacotes. Assim, vários pacotes pertencentes a diferentes aplicações ou usuários podem ser processados concorrentemente.

**Abordagem Integrada ou por cápsula:** Esta idéia vai além da abordagem anterior. Aqui os pacotes passivos da rede atual são substituídos por miniaturas de programas ativos que são encapsulados em quadros de transmissão e podem ser executados em cada nó de seu caminho na rede, ou seja, cada mensagem é um programa ou um fragmento dele. Assim, os dados dos usuários podem ser colocados dentro das cápsulas e transferidos juntamente com o código. Os nós da rede passam a se chamar *Nós Ativos* já que esses nós executam o código que chega através das cápsulas podendo mudar o estado do nó e possivelmente gerar novos pacotes para serem enviados a outros nós ativos. Os nós ativos possuem *Ambientes de Execução (Execution Environments - EE)* onde os pacotes ativos são executados. Estes ambientes são executados sobre um Sistema Operacional que age como uma camada que intermedia o acesso aos recursos físicos (*hardware*) do nó. Dessa forma, os recursos encontrados nesses nós ativos incluem memória e CPU para processamento dos pacotes ativos. Um nó ativo típico realizando o processamento de um pacote ativo é mostrado na Figura 2.1. Quando uma cápsula chega em um nó ativo, seu conteúdo é avaliado e enviado para um ambiente de execução transiente (temporário) onde seu conteúdo poderá ser seguramente analisado. A execução do conteúdo de uma cápsula pode resultar em acessos aos recursos externos ao ambiente transiente

ativas pois os nós podem realizar computações e modificar o conteúdo desses pacotes. O processamento pode ser realizado de forma personalizada por usuário ou por aplicação, ao contrário das redes tradicionais onde o processamento dos pacotes limita-se às operações realizadas nos cabeçalhos, principalmente com objetivo de roteamento.

O conceito de redes ativas surgiu a partir de discussões na DARPA (*Defense Advanced Research Projects Agency*) em 1994 e 1995, onde os pesquisadores da época questionavam sobre o futuro dos sistemas de rede. Vários problemas foram identificados nas redes atuais: a dificuldade de integrar novas tecnologias e padrões dentro de uma infra-estrutura de rede compartilhada, desempenho abaixo do esperado devido à redundância de operações nas camadas dos protocolos, e principalmente a dificuldade de acomodar novos serviços no modelo de arquitetura existente [9].

Nas redes tradicionais os pacotes transportam bits de um sistema final para outro sem modificar o dado do usuário, o qual é transferido de forma opaca pela rede. Nas redes ativas há uma computação personalizada sobre os dados que trafegam nos pacotes. Assim, as redes são consideradas Ativas devido a dois aspectos principais [6]:

- Os nós intermediários, tais como roteadores e comutadores, podem realizar computações sobre os dados dos usuários conforme os pacotes passam por eles;
- Usuários podem "injetar" programas na rede de forma a construir ou definir o processamento dos nós para que seja específico a uma determinada aplicação ou usuário. Assim, pode-se "moldar" o processamento em cada nó para atender certas necessidades específicas bastando para isso, instalar programas e definir o fluxo dos pacotes pelos nós programáveis.

Existem também alguns fortes motivos para investir na adaptação das redes tradicionais em redes ativas. Além dos problemas identificados pelos pesquisadores da DARPA, em [6] algumas vantagens da mudança na arquitetura atual das redes a fim de investir na troca de pacotes passivos por programas ativos são apresentadas. A troca de código na rede fornece uma base para interações mais ricas do que a troca de dados com formatos fixos. Colocar "inteligência" em nós intermediários possibilita o desenvolvimento de soluções mais adeqüadas para certos problemas, já que é possível implementar funções específicas em pontos estratégicos na rede. A abstração que as linguagens de alto nível fornecem permite que o usuário "personalize" sua infra-estrutura. Além disso, o desenvolvimento de novos serviços pode ser feito com uma considerável redução de tempo pois o processo de padronização não é mais conduzido pelo vendedor do serviço.

# Capítulo 2

# Conceitos Básicos e Trabalhos Relacionados

Nesta seção, definimos os conceitos de Redes Ativas, Redes Ativas Virtuais, Agentes Móveis, Gerência de Redes e Sistemas Distribuídos, enfatizando a Gerência de Serviços, e a Gerência baseada em Políticas. Apresentamos também o ANTS, ambiente no qual foi desenvolvida a infra-estrutura para suportar a aplicação ativa desejada a fim de validarmos o nosso modelo de gerência. Para concluir a seção, comentamos alguns trabalhos relacionados.

Com o objetivo de tornar as redes programáveis [21, 17], surgiram duas escolas de pensamento. A primeira delas, denominada *Opensig* [35], defende a utilização de um conjunto de interfaces abertas de programação de rede que modelam o hardware de comunicação, possibilitando o acesso completo a roteadores e comutadores. O projeto 1520 [36, 22], realizado pelo IEEE na área de interfaces de programação para redes, está seguindo a abordagem da comunidade *Opensig* e procura padronizar tais interfaces para roteadores IP (*Internet Protocol*), comutadores ATM (*Asynchronous Transfer Mode*) e redes móveis de telecomunicação. A segunda escola de pensamento é objeto de pesquisa desta dissertação. É a comunidade das redes ativas que defende a implantação dinâmica de novos serviços. Este dinamismo vai muito além daquele proposto pela comunidade *Opensig* e tal linha de raciocínio somente foi possível devido ao avanço tecnológico nos equipamentos de rede que podem suportar esta carga de processamento a mais sem degradar a rede como um todo [2].

## 2.1 Redes Ativas

As redes ativas, diferentemente das redes tradicionais, permitem que códigos de programas sejam carregados pelos pacotes e instanciados em nós intermediários. Estas redes são

funcionais de gerenciamento: contabilidade, desempenho e configuração; não abordando as áreas de segurança e falhas que completam o modelo de gerência OSI (*Open System Interconnection*) [42]. O modelo coleta dados sobre contabilidade e desempenho e permite que o gerente analise estas informações para detectar possíveis problemas atuando em direção a uma possível (re)configuração do ambiente gerenciado. Além disso, estamos considerando o uso de políticas e, portanto, a gerência de cada serviço deve considerar as políticas pré-definidas pelo desenvolvedor daquele serviço. O modelo, além do sistema de gerência propriamente dito, suporta toda a infra-estrutura para que os clientes criem suas redes ativas virtuais com os serviços desejados e acrescentem novos serviços à sua rede virtual conforme a necessidade.

O modelo é validado através da implementação de um protótipo dando origem ao Sistema de Gerência de Serviços Móveis para Redes Ativas Virtuais (*Mobile Service Management System for Virtual Active Networks - MS2VAN*). Alguns dos componentes do sistema foram desenvolvidos usando a tecnologia de Agentes Móveis. Toda a infra-estrutura para redes ativas foi criada usando o *toolkit ANTS (Active Node Transfer System)* [7]. No ANTS os pacotes são substituídos por cápsulas que transportam além de dados, código de programas Java. Os roteadores e os nós (tipicamente hosts), são substituídos por nós ativos que executam as cápsulas e mantém estado. A implementação tanto da infra-estrutura para a criação de redes ativas como do Sistema de Gerência foi feita usando a linguagem Java. A plataforma de agentes móveis *Grasshopper 2.1* foi utilizada para suportar os agentes de gerência, suas migrações e instanciações.

Esta dissertação está organizada em capítulos da seguinte forma: o próximo capítulo apresenta os conceitos básicos e alguns trabalhos relacionados. O modelo detalhado e alguns cenários de gerência são apresentados no Capítulo 3. A implementação do modelo é comentada no Capítulo 4. No Capítulo 5 o sistema é utilizado na gerência de dois serviços típicos de Telecomunicação. Finalmente, o Capítulo 6 apresenta as conclusões e considera possíveis trabalhos futuros.

são divididos seguindo critérios que facilitam na administração do ambiente. Isso introduz um conceito muito importante em redes ativas que são as Redes Ativas Virtuais (*Virtual Active Networks - VANs*). Com as VANs, é possível abstrair dispositivos intermediários e criar um ambiente com nós específicos atendendo às necessidades de cada cliente ou grupo de clientes. Em domínios empresariais, por exemplo, é interessante que uma empresa crie um ambiente de rede particular atribuindo a esse ambiente características de segurança mais concretas. As características conceituais das VANs foram particularmente herdadas das Redes Privadas Virtuais (*Virtual Private Networks - VPNs*). A vantagem das VANs sobre as VPNs é a possibilidade de envio de código de programas.

A tendência atual aponta para o uso das redes ativas em ambientes de Telecomunicação onde existe a necessidade de transferência de serviços de forma rápida, fácil e dinâmica para lugares específicos na rede [18]. Isso permite que as atualizações, remoções, substituições e outras tarefas relacionadas à vida útil de um serviço sejam feitas usando linguagens de programação de alto nível. Nesses ambientes de Telecom, é comum a presença de um Fornecedor de Serviços (*Service Provider - SP*) que centraliza os serviços disponíveis para os clientes. Esses clientes poderão definir uma rede ativa virtual e solicitar ao SP o envio de alguns serviços para a sua rede. É com base nessas características e na forte tendência do uso das ANs em ambientes Telecom que essa dissertação propõe um modelo para gerenciar serviços em redes ativas virtuais onde os objetos que representam os serviços podem migrar dentro da própria rede ativa virtual ou entre redes ativas virtuais.

A maioria dos trabalhos que abordam a gerência de serviços em redes ativas usam a própria AN para gerenciar os serviços ativos e não consideram serviços móveis. Uma solução particularmente interessante para serviços estáticos define um *framework* a fim de que os serviços sejam instalados por um fornecedor mas a gerência seja feita pelo usuário ou pelo próprio fornecedor do serviço [1]. Nestas soluções onde a gerência é feita pela própria AN, existe uma grande troca de cápsulas entre os nós ativos para envio e recebimento de informações de gerenciamento. Modelos mais genéricos que consideram serviços móveis, e não somente serviços estáticos, usam agentes móveis para gerenciar tais serviços [16]. Na maioria desses modelos, um agente móvel é responsável pela gerência de um serviço, migrando na medida em que o serviço migra. Entretanto, se o número de serviços aumenta muito, tipicamente o que ocorre em ambientes Telecom, esta solução apresenta-se inviável devido ao grande número de agentes móveis que seria necessário para gerenciar todos os serviços.

Ao contrário da solução que usa ANs para gerenciar o próprio ambiente e considera somente serviços estáticos, e da solução que considera serviços móveis mas é inviável para ambientes Telecom, desenvolvemos um modelo hierárquico para um domínio, onde o gerenciamento é dividido em quatro níveis: por serviço, por host, por VAN e no nível mais alto, em todo o conjunto de VANs (um domínio). São abordadas três das cinco áreas

# Capítulo 1

# Introdução

Atualmente, a maioria das tecnologias de comunicação (protocolos de comunicação) e os componentes físicos (repetidores, pontes, roteadores, etc.) têm obtido grande atenção da comunidade de padronização e das empresas. Conseqüentemente, o gerenciamento de redes deve fornecer mecanismos para monitoração, controle e coordenação da utilização de recursos pertencentes a estas redes [25]. Em paralelo a isto, recursos de *hardware* e *software* de sistemas finais (espaço no disco, tempo de CPU de aplicações) são incluídos nesta visão de gerência expandindo o conceito de gerência de redes para gerência de sistemas distribuídos. Portanto, a gerência de sistemas distribuídos envolve não somente a gerência de dispositivos que atuam como roteadores na rede mas também a gerência de serviços da infra-estrutura de comunicação e processamento de aplicações.

A oferta e a demanda de serviços têm aumentado muito atualmente, exigindo uma infra-estrutura de rede que permita um mecanismo rápido e eficiente para a disponibilização desses serviços na rede. As Redes Ativas (*Active Networks - ANs*) permitem que aplicações sejam transferidas para localizações estratégicas atendendo a uma certa demanda do usuário. Esse tipo de rede também permite e facilita o desenvolvimento e a disponibilização de novos serviços na rede de forma que qualquer desenvolvedor possa colocar suas aplicações à disposição de clientes. No ambiente das ANs, os pacotes são tipicamente substituídos por cápsulas que transportam dados e códigos. Uma cápsula é a combinação de um pacote e sua rotina de *forwarding* que é avaliada em cada nó a fim de executar tarefas pré-definidas [41].

Nesse ambiente das redes ativas é muito comum a existência de clientes (usuários) que solicitam serviços a algum fornecedor para serem executados em seus domínios. Esses domínios podem ser amplamente caracterizados como sendo qualquer ambiente específico que será criado para atender a uma demanda ou necessidade específica. Normalmente, um domínio pode ser tecnológico ou administrativo. Domínios tecnológicos são aqueles que seguem algum agrupamento definido pela tecnologia em uso. Domínios administrativos

# Lista de Tabelas

# Lista de Figuras

# Sumário

TCP - Transmission Control Protocol.
VAN - Virtual Active Network.
VPAN - Virtual Private Active Network.
VPN - Virtual Private Network.

# Acrônimos

AA - Active Application.
AN - Active Network.
ANTS - Active Network Transfer System.
CMIP - Common Management Information Protocol.
CORBA - Common Object Request Broker Architecture.
GMA - Global Manager Agent.
GNS - Global Naming Service.
IETF - Internet Engineering Task Force.
IIOP - Internet Inter-ORB Protocol.
IP - Internet Protocol.
ISO - International Organization for Standardization.
LM - Local Manager.
LRUS - Least Recently Used Service.
MAEA - Managed Active Element Agent.
MASIF - Mobile Agent System Interoperability Facility.
MC - Management Center.
MIB - Management Information Base.
MM - Migration Manager.
MO - Managed Object.
MRA - Management Remote Application.
MS - Managed Service.
MS2VAN - Mobile Service Management System for Virtual Active Networks.
ORB - Object Request Broker.
OSI - Open System Interconnection.
PM - Policy Manager.
RMI - Remote Method Invocation.
RPC - Remote Procedure Call.
SNMP - Simple Network Management Protocol.
SP - Service Provider.

# Agradecimentos

Agradeço...

Primeiramente a Deus, que em nenhum momento abandonou-me. Diante das dificuldades foi meu fiel servidor, dando-me saúde, fé, disposição e respostas em momentos de dúvidas.

Ao Prof. Edmundo pela orientação, dedicação e companheirismo. Não foi apenas orientador, mas amigo e conselheiro em todos os momentos desta caminhada.

À minha noiva Adriane pelo infinito apoio e compreensão. Sempre ao meu lado dando-me força e coragem, incentivando-me nos momentos de fraqueza. Obrigado por você existir.

À minha família, meu pai, minha mãe e meu irmão que de longe me diziam palavras de conforto e esperança.

Ao Maurício pelos incentivos e pelas muitas vezes que me ajudou.

Ao Leonardo e ao Hudo pelos momentos de conversa, conselhos e piadas.

Ao Rafael que apesar do pouco tempo de convívio, foi para mim um grande amigo de uma bondade infinita.

Aos meus grandes e inesquecíveis amigos da pensão e companheiros nas partidas de truco: Paulo, Chico, Rubens, Cláudio, etc.

À CAPES e ao projeto PRONEX/SAI pelo apoio financeiro.

A todos que de perto ou de longe me apoiaram com palavras ou gestos, pois foi através deles que consegui chegar ao fim de mais esta caminhada.

Obrigado!

5. O número mínimo de migrações de um serviço pode levá-lo a ser um candidato à remoção.

Essas políticas estão diretamente relacionadas com as ações de configuração. A primeira política refere-se a capacidade que um determinado serviço possui para migrar ou não para outras VANs. Serviços internos podem migrar somente dentro de sua VAN. Serviços externos podem migrar entre diferentes VANs. A segunda política evita que um determinado serviço migre excessivamente em um curto intervalo de tempo. Isso ocorre quando os clientes estão necessitando de um mesmo serviço ao mesmo tempo, fazendo com que tal serviço fique migrando de uma rede virtual para outra muito freqüentemente. Assim, quando um serviço atingir seu limite máximo de migrações no intervalo de tempo definido, a ação de configuração 2 poderá ser empregada para criar uma nova cópia do serviço na VAN do cliente solicitante. Mas para criar uma nova cópia do serviço, a política 3 deve ser respeitada, ou seja, isso somente será possível se o limite de instâncias do serviço na rede não foi atingido. Caso contrário, somente o gerente (humano) poderá decidir se o serviço poderá migrar, mesmo não obedecendo o seu limite de migrações, ou se uma nova cópia do serviço poderá ser criada, apesar da política de quantidade de cópias não permitir.

A política LRUS é definida para o ambiente gerenciado já que, a menos que não existam cópias de um determinado serviço na rede, sempre haverá o serviço menos usado recentemente. Quando um cliente solicita um serviço, o SP procura em todas as redes virtuais que possuam uma cópia desse serviço a fim de encontrar o menos usado. Esta política exige que seja feita uma coleta periódica e automática das requisições recebidas em cada serviço. O gerente responsável pelo serviço deverá coletar a quantidade de requisições recebidas a cada intervalo de tempo definido e armazenar em sua base de dados. Isso é necessário pois queremos descobrir o serviço menos usado recentemente e, portanto, esta informação deve estar disponível em algum gerente do modelo. Esta base de dados sempre será consultada quando se deseja obter o LRUS. A confiabilidade desta informação depende da freqüência da coleta e do momento em que a base de dados é consultada, conforme mostra a Figura 3.1.

Após a primeira coleta, é realizada uma busca pelo LRUS (a). Pode-se considerar que o tempo entre os dois eventos não é grande e muito provavelmente a informação da base de dados reflete o estado atual dos serviços. A segunda busca pelo LRUS (b) é feita após um grande período de tempo em relação a primeira coleta e, por isso, a informação da base de dados pode não refletir o estado atual dos serviços. À medida que a busca pelo LRUS se distancia temporalmente da coleta, a confiabilidade das informações da base de dados diminui. Para aumentar a confiabilidade das informações armazenadas, poderíamos aumentar a freqüência da coleta, já que a única métrica a ser coletada é a quantidade de invocações recebidas. Uma outra alternativa para garantir a confiabilidade seria realizar

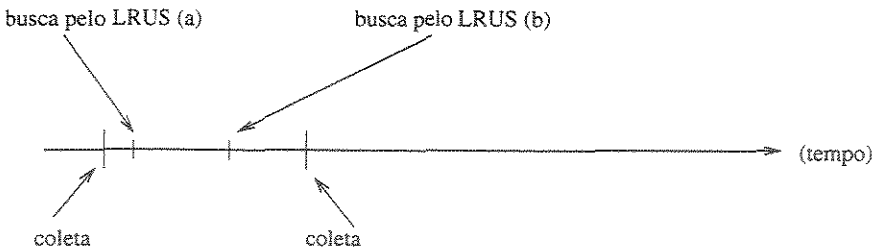



Figura 3.1: Confiabilidade da informação coletada.

uma coleta no momento da busca pelo LRUS, obtendo a informação atual sobre o serviço. Entretanto esta solução apresenta-se inviável já que o cliente teria que esperar pela coleta, causando um atraso na instalação do serviço.

A última política (política 5) refere-se à quantidade mínima de migrações de um serviço em um determinado período. Se o serviço não migrar pelo menos seu limite mínimo de vezes, ele será um candidato à remoção. A palavra candidato aqui possui um significado muito importante. Quando o sistema de gerência detecta que um determinado serviço pode ser removido, tal serviço é colocado em uma lista de serviços candidatos à remoção. Remover esses serviços imediatamente não é uma boa idéia pois pode ser que se um serviço não migrou seu número mínimo de vezes é porque ele está sendo muito usado no host onde ele se encontra no momento. E se um serviço é muito usado, provavelmente ele não atende a política 4, explicada acima. Portanto, antes da remoção de serviços, o gerente poderá coletar os dados de contabilidade para cada serviço, obtendo, por exemplo, a quantidade de invocações recebidas em um intervalo de tempo. Assim, é possível descobrir se um serviço está sendo muito usado ou não atualmente, e então decidir sobre sua remoção. Esse será o critério para a aplicação da ação de configuração 3, ou seja, os serviços candidatos a serem excluídos da rede serão aqueles em que a política 5 não foi atendida. A ação de configuração 3 aplica-se somente aos serviços solicitados após a criação da VAN, não se aplicando aos serviços solicitados durante a instalação da VAN. Isso é necessário pois os serviços enviados pelo SP, quando o cliente solicita a VAN, são considerados serviços básicos para o cliente e somente poderão ser excluídos pelo próprio cliente.

A política 2 e a política 3 minimizam o número de migrações de um serviço e ao mesmo tempo evitam que a instanciação de novos serviços aconteça ilimitadamente. Em ambientes onde pode haver uma grande demanda por serviços ao mesmo tempo, deve existir certas políticas a fim de evitar migrações e instanciações em excesso. Estas duas políticas definidas são uma tentativa para evitar este tipo de situação no ambiente proposto.

## 3.2 Os Componentes da Infra-Estrutura

Esta seção define os componentes desenvolvidos e as suas funcionalidades dentro do modelo proposto. Antes de explicarmos cada componente separadamente, mostramos através da Figura 3.2, a distribuição hierárquica desses componentes e seus relacionamentos. A figura também mostra duas redes ativas virtuais que compartilham um mesmo nó (nó X), caracterizando a sobreposição de VANs em diferentes hosts da rede.

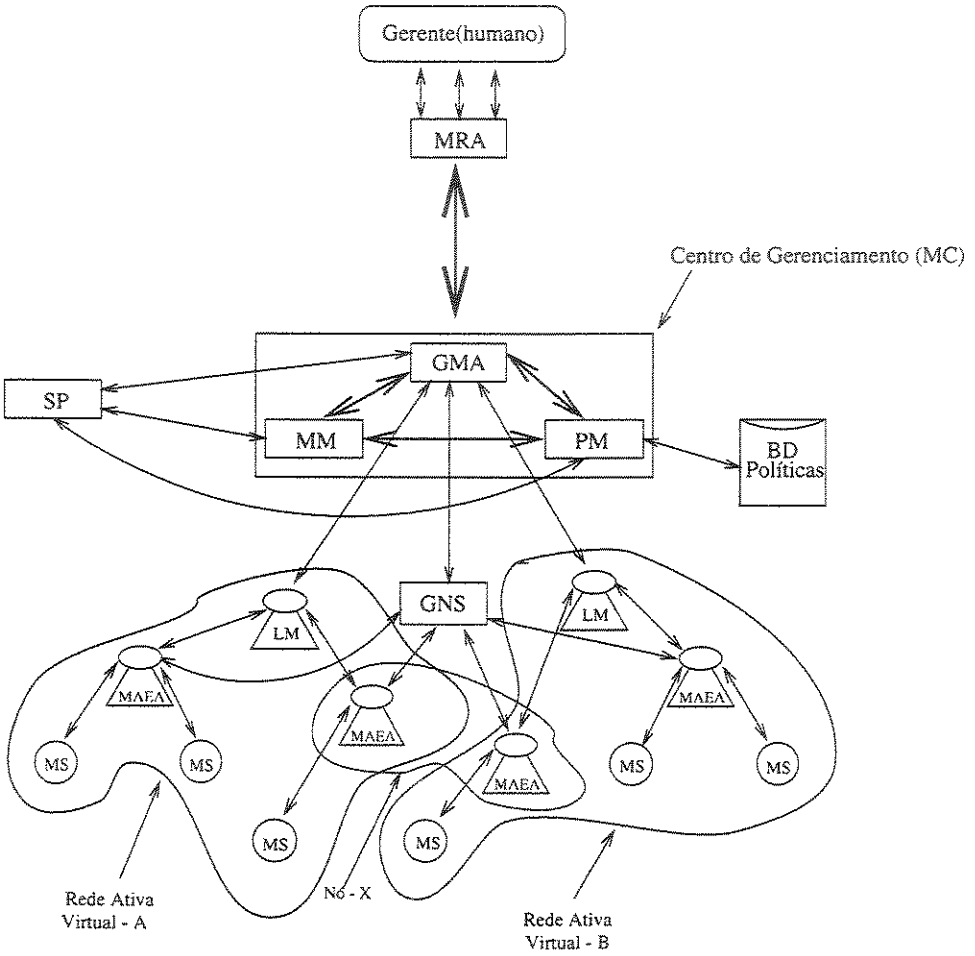



Figura 3.2: Os componentes da infra-estrutura e seus relacionamentos.

Na Figura 3.2 podemos ver os componentes desenvolvidos, faltando apenas o PMA (*Performance Manager Agent*), responsável pela coleta de dados de desempenho. O GMA (*Global Manager Agent - GMA*) é o gerente do domínio sendo responsável pela centralização e análise das informações de gerência. O LM (*Local Manager - LM*) é o

gerente responsável pela VAN e, portanto, coleta e gerencia informações sobre sua rede ativa virtual. O MAEA (*Managed Active Element Agent - MAEA*) coleta informações do Serviço Gerenciado (*Managed Service - MS*). O MS representa um serviço a ser oferecido na rede para os usuários. Um cliente solicita serviços quando ele deseja instalar uma VAN, ou posteriormente quando ele necessita de novos serviços em sua rede ativa virtual. O serviço passa a se chamar Serviço Gerenciado pois ele possui uma interface e atributos específicos para a gerência. O PM (*Policy Manager - PM*) gerencia as políticas dos serviços e o MM (*Migration Manager - MM*) é reponsável pela aplicação destas políticas para cada serviço. O SP oferece serviços e disponibiliza VANs para os clientes, e o GNS (*Global Naming Service - GNS*) é o serviço de nomes do domínio. Logicamente definimos um Centro de Gerenciamento (*Management Center - MC*) formado pelo GMA, PM e MM.

No modelo proposto, a maioria das características citadas na Figura 2.4 (seção 2.2.1) foram consideradas: solicitação pelo cliente para criação (disponibilização) de uma VAN, solicitação e instalação de serviços, aplicação de políticas de uso de cada serviço, migração de serviços internamente à VAN e entre VANs, e principalmente o gerenciamento desses serviços realizado através de agentes móveis. Uma das tarefas do fornecedor de serviços no nosso modelo inclui avisar o sistema de gerência que uma nova VAN está sendo criada. Assim, o sistema de gerência poderá enviar os agentes de gerência responsáveis pela monitoração de cada serviço nas diferentes VANs. Uma VAN no nosso modelo é definida através da especificação de um conjunto de hosts sendo que cada host poderá estar rodando um ou mais ambientes de execução pertencentes a diferentes VANs.

### 3.2.1 Agentes de Gerência

Os quatro agentes de gerência desenvolvidos que compõem o MS2VAN são: o MAEA, o LM, o PMA e o GMA. Os agentes de gerência, seguindo a proposta do modelo, podem ser móveis ou estáticos. Nesse trabalho definimos três agentes móveis e um estático. Os agentes móveis são o MAEA, o LM e o PMA, e o agente estático é o GMA. A mobilidade dos agentes MAEA e LM ocorre unicamente no momento em que uma rede ativa virtual é criada, já que eles são criados pelo MM (ver abaixo) e enviados para os hosts determinados. Após isso, esses agentes podem ser considerados estáticos pois ficarão no host até o cliente extingüir a rede virtual. O PMA é móvel pois será enviado para os hosts para coletar dados de desempenho e após o período de tempo definido, retornará com as informações coletadas.

A escolha por uma plataforma de agentes móveis está principalmente relacionada ao envio de agentes de gerência para hosts onde os serviços estão localizados com propósitos de coleta de dados para a área de contabilidade e desempenho. Além do mais, os serviços

são móveis em sua maioria podendo migrar internamente à VAN ou entre VANs. Assim, os mecanismos de comunicação entre agentes serão utilizados para a troca de informações entre os agentes de gerenciamento, já que no modelo proposto e implementado, os agentes móveis não seguirão os serviços que se movem de um host para outro. Devido a uma estrutura hierárquica de gerência, o movimento de agentes que "perseguem" os serviços não será necessário. Assim, em seguida a especificação de cada agente de gerência é apresentada.

### Gerente Global - GMA (*Global Manager Agent*)

Age como uma interface entre a MRA (*Management Remote Application*) e o sistema de gerência propriamente dito. A MRA é a aplicação de gerência utilizada pelo gerente para interagir com o sistema. A MRA invoca métodos no GMA que por sua vez dispara a coleta de dados de contabilidade ou dispara o envio de agentes para a coleta de dados de desempenho. Em ambos os casos, o GMA fica aguardando a entrega dos dados, agindo por um lado como servidor e por outro lado como cliente dos componentes da infra-estrutura. Quando as informações, tanto de contabilidade como de desempenho, são entregues para o GMA, ele as analisa conforme a requisição que está sendo feita. Por exemplo, se o método de gerência sendo invocado está apenas relacionado a uma rede virtual ou a um serviço, o GMA repassa essa informação diretamente para o gerente que está aguardando na MRA. Se for um método de gerência relacionado a todas as VANs, por exemplo, o método que resolve qual o serviço mais requisitado em todo o domínio (conjunto de todas as VANs), então o GMA deve analisar os dados enviados por cada LM e encontrar a informação desejada. Isso é possível devido ao modelo hierárquico criado e, portanto, a quantidade de informações a ser coletada, processada e transferida entre os agentes dos diferentes níveis depende unicamente do tipo de informação a ser obtida.

É no GMA que encontramos as informações sobre as redes ativas existentes e seus serviços. Através dele podemos saber quais os hosts que formam uma rede virtual, quais os serviços que uma determinada VAN possui e quais os serviços localizados em um determinado host, entre outras informações. Isso permite que o gerente usando a MRA consiga obter esse tipo de informação básica sem precisar descer a níveis inferiores de gerência, ou seja, sem precisar consultar o LM e o MAEA. Além do mais, as informações sobre as migrações dos serviços encontram-se no mesmo nível de gerência do GMA. Assim, para saber sobre as tabelas de migrações de cada serviço, o GMA precisa simplesmente consultar o MM e então enviar essas informações para o gerente remoto.

### Gerente da Rede Ativa Virtual - LM (*Local Manager*)

Coleta e gerencia informações da rede virtual da qual ele é responsável. O LM pode aplicar filtros a fim de diminuir a quantidade de informações a ser enviada para o GMA. Nesse nível de gerência sempre haverá a necessidade de consultar o MAEA para coletar os dados nos serviços gerenciados (MS).

### Gerente do Elemento Ativo Gerenciado - MAEA (*Managed Active Element Agent*)

Esse agente é responsável pela gerência de um ou mais serviços em um host pertencente a uma VAN. Assim como o GMA é uma interface entre o sistema de gerência e a MRA, o MAEA é uma interface entre o MS e o sistema de gerência. O MAEA acessa a interface de gerência do MS para coletar os dados sobre as métricas definidas e/ou ativar contadores de tempo. Nesse nível também são aplicados filtros sobre as informações a fim de enviar somente o que é solicitado pelo nível acima. Exceto quando o gerente desejar saber a quantidade de invocações recebidas no serviço a fim de ter certeza sobre sua remoção, o único componente que acessa a interface de gerência do MS é o MAEA. No caso do gerente desejar saber qual a quantidade de requisições recebidas por um serviço, o GMA irá acessar diretamente o MS sem passar pelo LM e pelo MAEA. Isso é mais eficiente pois, já que o gerente (humano) deverá informar o nome completo do serviço, fica fácil para o GMA consultar no GNS e obter a referência direta para o serviço sem a necessidade de seguir a hierarquia definida. Também é tarefa do MAEA avisar sobre a migração de um serviço para outro host. Nesse contexto, existe uma comunicação entre o MAEA de onde o serviço está saindo e o MAEA localizado no host para onde o serviço está migrando. Podemos então listar, resumidamente, as tarefas principais do MAEA:

- Ativar e desativar os filtros para coleta de métricas para a área de contabilidade por um período de tempo;
- Coletar as informações (métricas) nos serviços usando a interface de gerência;
- Enviar informações para outro MAEA sobre a migração de um serviço;
- Receber informações de outro MAEA sobre a chegada de um novo serviço no host;
- Processar eventos definidos previamente ou notificar o LM sobre eventos desconhecidos.

O último ítem relacionado a eventos não foi considerado nessa dissertação, sendo alvo de pesquisa de futuros trabalhos.

Observando novamente a Figura 3.2, nota-se que não mostramos a interação entre agentes MAEA localizados tanto na mesma VAN como em VANs diferentes. Esta interação será apresentada na seção sobre os cenários desenvolvidos para um possível ambiente Telecom. A comunicação entre agentes MAEA ocorre unicamente quando um serviço migra de um host para outro, interna ou externamente à VAN. A migração entre VANs ocorre somente quando um cliente de alguma VAN está solicitando um serviço de outra VAN. A migração interna pode ocorrer também quando o cliente deseja migrar serviços de um host para outro a fim de realizar balanceamento de carga.

### Gerente de Desempenho - PMA (*Performance Manager Agent*)

Esse agente é responsável pela coleta de dados para a área funcional de desempenho. Será enviado um agente desse tipo para cada host onde deseja-se coletar dados, diferentemente dos modelos encontrados na literatura atual, onde um único agente coleta os dados "pulando" de host em host armazenando informações sobre cada nó. Enviando-se um agente para cada host, garantimos que a coleta dos dados será em paralelo, obtendo-se assim, a situação de cada host no mesmo período de tempo. Este agente ficará coletando dados no host durante o intervalo de tempo especificado pelo gerente (humano). Após isso, o PMA retornará para o Centro de Gerenciamento (MC) onde irá entregar os dados coletados para o GMA.

A Figura 3.3 mostra os agentes de gerência distribuídos pela rede hierarquicamente, desde o GMA até o MS. Na figura não aparece o PMA já que ele não possui nenhuma relação hierárquica com os demais agentes de gerência. Ele apenas é criado quando o gerente deseja coletar dados de desempenho, e depois é eliminado do ambiente.

O acesso às informações de gerência nos serviços deveria ser limitado, coletando apenas informações indispensáveis para detectar possíveis problemas. Deveria-se, portanto, atribuir um certo grau de autonomia às VANs de forma que o sistema de gerência não precise ficar sabendo de todos os eventos ocorridos no ambiente. A questão a ser respondida é quais as informações o sistema de gerência precisa saber e quais ele não precisa. A autonomia está diretamente relacionada com a quantidade de informações coletadas. Quanto maior esta quantidade, menor será a autonomia do ambiente. Por exemplo, no modelo proposto, o GMA deverá saber, pelo menos, a localização de cada serviço, sendo informado sempre que ocorrer uma migração. Isto é necessário para realizar balanceamento de carga quando o sistema de gerência detectar hosts sobrecarregados na rede. Portanto, o controle desejado do ambiente limita a autonomia das VANs.

Outro aspecto a ser considerado está relacionado à quantidade de políticas existentes no ambiente gerenciado. No nosso caso, definimos quatro políticas, e a cada nova solicitação do cliente para instalação de um novo serviço em sua VAN, estas políticas devem ser aplicadas a fim de testar a possibilidade ou não desta instalação. Se não existissem essas

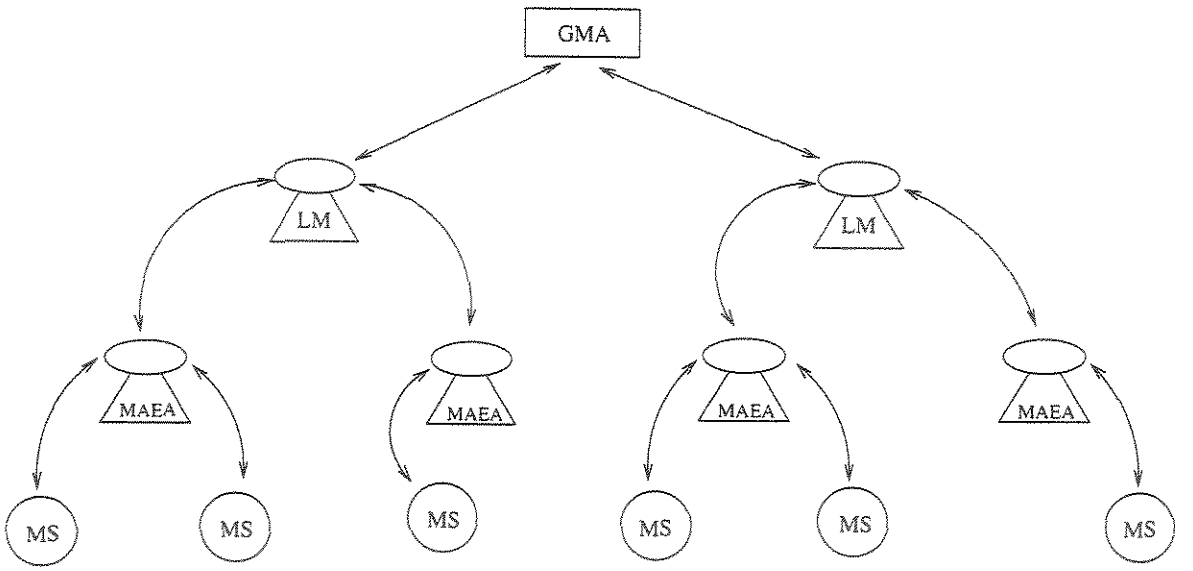



Figura 3.3: Distribuição hierárquica dos agentes de gerência.

políticas ou se elas fossem menos rígidas, esta verificação talvez não fosse necessária. Portanto, a quantidade de políticas e o grau com que elas interferem no ambiente influenciam diretamente no nível de autonomia que pode ser estabelecido.

### 3.2.2 Outros Componentes da Infra-Estrutura

Os componentes apresentados abaixo foram desenvolvidos para suportar as atividades relacionadas ao fornecimento de serviços em um ambiente Telecom: o Fornecedor de Serviços (SP), o Gerente de Políticas (PM), o Gerente de Migrações (MM), o Serviço de Nomes Global (GNS) e a Aplicação Remota de Gerenciamento (MRA). É importante comentar que estes componentes não são encontrados unicamente em ambientes de Telecomunicação. Eles são necessários em ambientes que necessitam de políticas e possuem serviços móveis instalados.

#### Gerente de Migrações - MM (*Migration Manager*)

O MM é responsável pelo envio de agentes de gerência da área de contabilidade e desempenho para os hosts especificados. O MM desempenha um papel muito importante no momento de disponibilização de uma VAN, enviando os agentes para a coleta de dados de contabilidade, ou seja, o MAEA e o LM, para os hosts que formarão a nova VAN. O MM recebe informações do SP sobre os serviços que cada MAEA deverá gerenciar e para qual

host cada gerente deverá ser enviado. O MM cria os agentes MAEA, insere as informações necessárias em cada MAEA criado e define o agente que será o LM da rede ativa virtual sendo criada. Finalmente, envia os agentes de gerência para os hosts especificados.

O MM é responsável pela aplicação das políticas de migração para os serviços, armazenando as informações de cada migração em uma tabela de migrações. Essa tabela possui informações de quantas vezes um determinado serviço já migrou, a hora da migração, host de origem e de destino e o número de invocações recebidas em cada host.

No envio de agentes para a área de desempenho, o MM apenas recebe a informação do período de tempo em que a coleta de dados deverá ser realizada. Sua tarefa é somente criar esses agentes e enviá-los para as agências especificadas, registradas no registro de agências. Os PMAs retornarão com os dados de desempenho coletados que serão entregues para o GMA.

### Gerente de Políticas - PM (*Policy Manager*)

O PM é responsável pelo controle, inserção, atualização e remoção de políticas na base de dados de políticas. Uma política é inserida na base de dados quando um serviço é registrado no SP (ver abaixo), e removida quando o serviço é desregistrado dele. O PM desempenha uma tarefa simples no sistema: armazenar e gerenciar as políticas dos serviços disponibilizando-as para os componentes que desejam obtê-las. Basicamente, o SP e o MM consultam o PM, respectivamente, para saber qual é o número máximo de cópias e qual é o número máximo de migrações permitido para um determinado serviço (ver seção 3.1.1 sobre políticas).

### Fornecedor de Serviços - SP (*Service Provider*)

O SP é o componente central para o fornecimento de serviços e disponibilização de VANs para os clientes. É através dele que o cliente poderá instalar uma VAN e seus serviços, assim como adicionar serviços isoladamente em sua VAN. Quando um determinado cliente deseja instalar uma VAN e seus respectivos serviços, ele deverá enviar uma cápsula para o SP informando quais hosts formarão a VAN e quais serviços pertencerão a cada host. O SP, ao receber esta cápsula, disponibiliza uma VAN enviando os serviços solicitados para o cliente. O SP deverá informar o MM e o GMA sobre a criação de uma nova VAN. Quando um cliente solicita um novo serviço, esse serviço poderá migrar ou não, com base nas políticas definidas e na quantidade de migrações já feitas por ele. Assim, se a migração não foi possível, e uma nova cópia do serviço foi criada, o SP deverá notificar o GMA e o MM. O primeiro precisa saber que um novo serviço existe na rede e necessita ser gerenciado. O segundo deverá criar uma entrada para esse serviço em sua tabela de migrações. Se a migração foi possível, o SP deverá avisar o GMA sobre isso. Note que

quando a migração é possível, há somente a necessidade de notificar o GMA pois o MM já está sabendo da possibilidade ou não da migração já que foi ele que fez essa verificação. As etapas para a instalação de uma VAN e seus serviços no domínio do cliente e para a solicitação de novos serviços serão explicadas com detalhes na seção sobre os possíveis cenários de um ambiente Telecom.

#### Serviço de Nomes Global - GNS (*Global Naming Service*)

O GNS é um serviço de nomes global e centralizado por meio do qual qualquer componente da infra-estrutura poderá obter uma referência para um serviço (objeto remoto). Esse componente recebe a localização do serviço e seu identificador, criando uma referência única que indica onde o serviço está localizado na rede. Os agentes de gerência consultam esse componente para obter as referências para os serviços.

#### Aplicação Remota de Gerenciamento - MRA (*Management Remote Application*)

A MRA é a aplicação de gerência usada pelo gerente (humano). Ela disponibiliza uma interface visual com os métodos disponíveis para a realização das tarefas de gerenciamento, inserção de políticas e registro de serviços. Os resultados são apresentados de forma gráfica ou em forma de tabelas, conforme a operação sendo realizada. O gerente poderá gerenciar o ambiente de uma forma geral ou refinando os parâmetros a fim de definir um escopo mais específico para a coleta de dados.

## 3.3 Gerência de contabilidade, desempenho e configuração

A coleta dos dados para a área de contabilidade e de desempenho difere completamente. A primeira usa o modelo hierárquico definido e invocação de métodos remotos. A segunda usa agentes móveis que migram para os hosts para coletar os dados de desempenho. Dados para a área funcional de contabilidade são basicamente relacionados ao número e tempo de resposta das invocações recebidas pelos serviços. A área de desempenho preocupa-se em coletar dados relativos ao uso de memória e CPU em cada host. A união das duas áreas permite ao gerente detectar problemas e atuar em direção a uma possível solução.

### 3.3.1 Gerência de Contabilidade

Para coletar os dados de contabilidade, os agentes distribuídos na rede comunicam-se a fim de trocarem informações, como mostra a Figura 3.4. A seqüência de passos para

realizar essa coleta é a seguinte:

1. O gerente (humano) invoca métodos em sua aplicação de gerência (MRA);
2. A aplicação invoca remotamente métodos no agente GMA, localizado no MC;
3. O GMA notifica cada LM para coletar dados durante o período de tempo especificado. Nesse intervalo de tempo definido, cada LM realizará a coleta em sua rede ativa virtual. Nesse ponto, o GMA espera todos os LMs enviarem os resultados;
4. Cada LM invoca métodos relacionados à coleta de dados de contabilidade nos MAEAs daquela VAN. O LM espera até que todos os MAEAs enviem seus dados. Após o LM obter os resultados de todos os MAEAs, os dados serão enviados para o GMA;
5. O MAEA acessa finalmente o serviço gerenciado (MS) para obter as métricas coletadas durante o período de tempo definido. Esses dados são retornados para o LM da VAN.

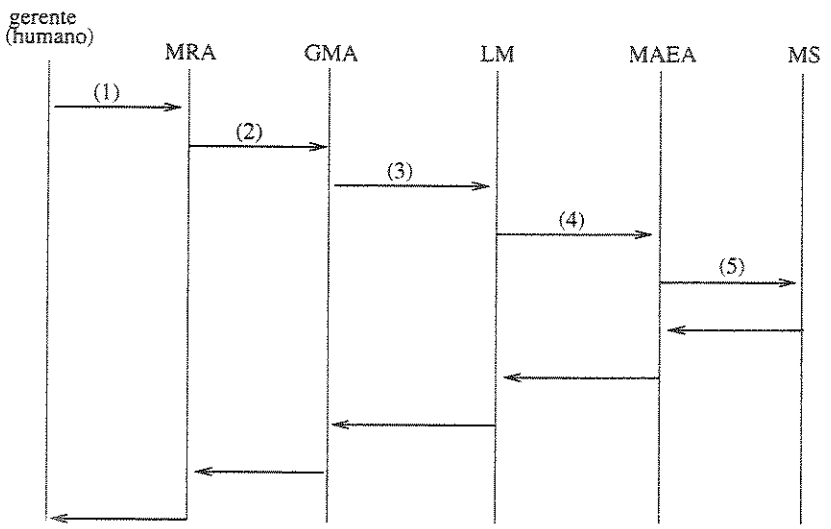



Figura 3.4: Diagrama de Seqüência para a coleta de dados de contabilidade.

A coleta de dados de contabilidade pode ser realizada em todas as VANs, mas também especificamente em uma VAN, host ou serviço. O grau de granularidade e o tempo para a coleta dos dados são definidos pelo gerente. Normalmente, coleta-se dados em todas as VANs a fim de detectar possíveis problemas de uma forma genérica. Após isso, futuras ações de gerência poderão estar unicamente relacionadas à coleta de dados para detalhar mais os problemas encontrados, refinando-se cada vez mais o escopo da gerência.

### 3.3.2 Gerência de Desempenho

Ao contrário da coleta de dados para a área de contabilidade, a coleta de dados para a área de desempenho é realizada através do envio de um agente móvel para cada host onde a gerência deve ser feita, como mostra a Figura 3.5. A seqüência de passos para realizar essa coleta é a seguinte:

1. O gerente (humano) invoca métodos em sua aplicação de gerência (MRA);
2. A aplicação invoca remotamente métodos no agente GMA, localizado no MC;
3. O GMA notifica o MM para que sejam enviados os agentes para coleta de dados de desempenho (PMA);
4. O MM cria os agentes e os envia para os hosts especificados;
5. Os agentes permanecem nos hosts durante o tempo determinado;
6. Os agentes de gerência retornam para o GMA onde entregarão os dados coletados para esse gerente;
7. O GMA analisa e retorna os dados coletados para a MRA que estava bloqueada aguardando a coleta ser realizada;
8. A MRA, após obter os dados, apresenta-os para o gerente de forma gráfica ou em tabelas.

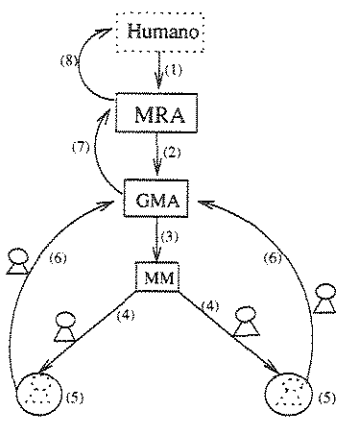



Figura 3.5: Seqüência realizada para a coleta de dados de desempenho.

Quando o gerente (humano) deseja coletar dados para a área de desempenho, automaticamente coleta-se dados para a área de contabilidade. Assim, no final do período de tempo especificado, pode-se apresentar para o gerente um conjunto completo de informações sobre o ambiente gerenciado, tais como: uso de memória e CPU, *throughput* dos hosts, host mais requisitado, host menos requisitado, entre outras. Além disso, o sistema de gerência fornece para o gerente uma tabela de quais hosts apresentam problemas, tais como uso de memória e CPU acima dos limites especificados. Dessa forma, o gerente poderá coletar informações específicas somente para esses hosts, e agir então em direção a uma possível (re)configuração do ambiente gerenciado.

### 3.3.3 Gerência de Configuração

A gerência de configuração usa as informações obtidas pela gerência de contabilidade e desempenho a fim de detectar problemas de sobrecarga de hosts. Além disso, a gerência de configuração é responsável pela instanciação e remoção de serviços do ambiente gerenciado. As três ações de configuração, definidas na seção 3.1, são as seguintes:

1. Mover um serviço de um host para outro dentro da mesma VAN com propósito de balancemanto de carga;
2. Criar uma nova instância de um serviço;
3. Excluir um serviço do ambiente.

Com as informações de desempenho coletadas, o gerente poderá saber se há algum host sobrecarregado. Neste caso, a ação de configuração a ser feita é migrar serviços de hosts sobrecarregados para hosts com menos carga. Pode-se também obter uma lista de serviços candidatos à remoção e então removê-los dos hosts sobrecarregados, usando a ação de configuração 3. A ação de configuração 2 é usada quando o número máximo de migrações de um determinado serviço é alcançado e, portanto, pode-se criar uma nova instância do serviço e enviá-la para o cliente solicitante.

## 3.4 Bases de Informações de Gerência - MIBs

O modelo de gerência proposto mantém algumas informações de gerenciamento nos diferentes níveis: MAEA, LM, e MC. Nesse trabalho, as bases usadas para armazenar esse tipo de informação são chamadas de MIBs.

Nesta seção apresentamos qual o tipo de informação está localizada em cada MIB e quais perguntas de gerência são respondidas em cada nível hierárquico. Para podermos coletar os dados de gerência, definimos algumas métricas simples e suficientes para esse trabalho. As métricas definidas foram:

a. invocações recebidas;

b. tempo de residência;

c. uso de memória; e

d. uso de CPU.

O serviço gerenciado possui atributos que representam as métricas a serem coletadas. A contagem das métricas é ativada em períodos de tempo definidos pelo gerente. No momento de coletar os dados para contabilidade, o MAEA ativa um contador (atributo) que é incrementado toda vez que o serviço é invocado. Após isso, o MAEA aguarda o período de tempo definido pelo gerente para a realização da coleta e desativa o atributo contador. Dessa forma, todas as invocações recebidas nesse período de tempo serão contadas. Para a obtenção de dados de desempenho, o PMA coleta dados relacionados ao uso de memória e CPU durante o período especificado, usando alguns aplicativos do próprio sistema operacional. Assim, a quantidade de métricas sendo coletadas em um determinado momento depende do método de gerência sendo invocado.

Cada MIB é capaz de responder algumas perguntas de gerência baseando-se nos dados coletados em seu nível. As Tabelas 3.3, 3.4 e 3.5 mostram quais questões são respondidas em cada nível de gerência e quais as métricas usadas para responder cada questão. Rotulamos as perguntas de contabilidade como sendo "C.*número-da-pergunta*" e as perguntas de desempenho "D.*número-da-pergunta*", por exemplo: C.1 significa a questão número 1 da área de contabilidade. As perguntas de gerência são aquelas definidas na seção 3.1, organizadas por serviço, por host, por VAN e em todo domínio, e serão repetidas aqui para facilitar a explicação.

As perguntas de contabilidade são:

1. Qual é o serviço mais e menos requisitado;
2. Qual é a quantidade de invocações recebidas em um determinado serviço;
3. Qual é o host mais e menos requisitado;
4. Qual é a quantidade de invocações em um determinado host;
5. Qual é o tempo de residência dos serviços;
6. Qual é o *throughput* em um determinado período de tempo;

As perguntas de desempenho são:

1. Qual é o uso de memória e CPU;
2. Quais hosts estão com o uso de memória e/ou CPU acima dos limites especificados;

Assim, temos os seguintes níveis com suas respectivas MIBs:

1. **MAEA**: Nesse nível o MAEA acessa os atributos de cada serviço. Esses atributos representam as métricas que contabilizam o número de requisições recebidas pelo serviço. As métricas nesse nível são: (a) e (b), e as questões que podem ser respondidas com essas métricas são:

Tabela 3.3: MIB do nível MAEA

| por serviço | por host | métrica |
|---|---|---|
| | C.1,C.4 | a |
| C.2 | | a |
| C.5 | | b |
| | C.6 | a,b |
| C.6 | | a,b |

2. **LM**: Esse é o nível intermediário de gerência responsável pelas perguntas relacionadas unicamente a uma determinada VAN. A única métrica aqui é a métrica (a), e as questões que podem ser respondidas com essa métrica são:

Tabela 3.4: MIB do nível LM

| por VAN | métrica |
|---|---|
| C.1, C.3 | a |

3. **MC**: Esse é o último e único nível de gerência que possui uma visão geral de todas as VANs. As métricas nesse nível são: (a), (c) e (d), e as questões que podem ser respondidas com essas métricas são:

Tabela 3.5: MIB do nível MC

| por host | por VAN | toda a rede | métrica |
|---|---|---|---|
| | | C.1,C.3 | a |
| D.1 | | | c,d |
| | D.2 | | c,d |
| | | D.2 | c,d |

Essa MIB pode ser considerada a mais importante de todas. É nela que as principais informações sobre as migrações de cada serviço ficam armazenadas. Isso ocorre devido à composição lógica do MC: GMA, MM e PM. No MM podemos saber quais foram as migrações de cada serviço e, para cada migração, saber qual foi o host origem e qual foi o host destino. Também é possível saber quantas requisições foram recebidas pelo serviço e qual o tempo de residência em cada host. É com base nessas informações que o MM analisa a possibilidade ou não de migração de um determinado serviço, ou seja, consultando a tabela de migrações e verificando o histórico do serviço, é possível saber se os limites especificados foram alcançados ou não. Nessa MIB, muitas informações básicas podem ser encontradas. É possível saber, por exemplo, os hosts e serviços que pertencem a cada VAN, os serviços que pertencem a cada host, as políticas de cada serviço e quais os serviços que atualmente estão registrados no SP. Além disso, é nesse nível que as informações coletadas em cada VAN serão processadas antes de serem enviadas para a MRA.

No modelo de gerência proposto, as métricas podem estar sempre sendo coletadas, coletadas em algum período de tempo ou podem não estar sendo coletadas. Coletar dados de gerência a todo momento não é conveniente devido a uma grande quantidade de informações que precisará ser armazenada.

A gerência normalmente coleta dados por um certo período de tempo, analisa esses dados e então possivelmente realiza futuras coletas, desta vez especificamente para os problemas encontrados. A idéia inicial é obter uma visão geral e global do ambiente gerenciado. Geral no sentido de que, em um primeiro momento, o gerente não está muito preocupado com detalhes sobre os serviços ou sobre as VANs, mas sim, detectar hosts possivelmente sobrecarregados. E global no sentido de que a gerência será realizada em todas as VANs, obtendo-se assim as informações mais recentes de todos os hosts em uso atualmente. Um exemplo típico seria realizar uma coleta sobre o desempenho e o *throughput* de todos os hosts que pertencem a alguma VAN. Com esta informação, pode-se identificar hosts sobrecarregados e então analisá-los de forma mais detalhada. Neste caso, poderíamos querer saber quais os serviços que estão em cada host e, então, gerenciar cada serviço obtendo a quantidade de requisições e seu *throughput* em um intervalo de tempo específico. Por fim, poderíamos migrar serviços de hosts sobrecarregados para hosts ociosos ou menos sobrecarregados. Essa é uma das perspectivas de gerência que pode ser adotada. A partir dela, pode-se criar variações restringindo a área de abrangência para a coleta de dados.

## 3.5 Cenários de Gerência

As possibilidades de uso de uma rede ativa foram desenvolvidas pensando nos possíveis cenários encontrados em ambientes Telecom. O cliente usa o ambiente ativo disponível através de uma Aplicação Ativa (*Active Application - AA*) desenvolvida para ser uma interface entre o usuário e a rede ativa virtual. Com a AA, o cliente poderá instalar uma VAN e seus serviços, solicitar novos serviços para sua VAN, remover serviços, migrar serviços internamente a sua VAN e extingüir uma VAN criada por ele. Com isso, tentamos fornecer para o cliente algumas funcionalidades básicas encontradas em ambientes Telecom e, portanto, criar cenários que pudéssemos gerenciar a fim de validar o protótipo desenvolvido.

A Aplicação Ativa criada comunica-se com o SP através do envio de cápsulas. Nessas cápsulas são colocadas todas as informações relacionadas ao pedido sendo feito. Os serviços são usados pela AA também através do envio de cápsulas para os hosts onde os serviços estão instalados. No modelo proposto, várias cápsulas foram definidas a fim de atender as necessidades da aplicação ativa implementada, tais como: cápsulas que instalam e removem serviços, cápsulas que avisam serviços para migrar, cápsulas que solicitam novos serviços, entre outras. A seguir, apresentamos de forma detalhada o comportamento dos componentes nos dois principais cenários desenvolvidos. O primeiro cenário representa a solicitação de um cliente para instalar uma VAN e seus serviços em seu domínio. O segundo cenário representa a solicitação de um cliente para instalar um novo serviço em sua VAN.

1. **Cliente solicita a instalação de uma VAN e seus serviços.**

   Essa funcionalidade permite que o cliente instale uma VAN e alguns serviços em seu domínio. Para isso, o cliente deverá informar os hosts que formarão a nova VAN, os serviços que pertencerão à VAN e o host destino para cada serviço. Após essas informações serem colocadas na cápsula, ela será enviada pela AA para o SP. Neste cenário, a seguinte seqüência de etapas é necessária, como mostra a Figura 3.6:

   (a) Cliente envia uma cápsula para o SP informando quais hosts formarão a VAN e quais serviços cada host terá;

   (b) O SP, ao receber a cápsula, analisa as informações contidas nela e então envia os serviços para os hosts especificados;

   (c) O SP notifica o MM enviando-lhe as informações sobre a nova VAN. O SP também notifica o GMA sobre a criação de mais uma rede ativa virtual;

   (d) O MM envia os agentes de gerência de contabilidade (MAEA e LM) para os hosts informados;

(e) Cada serviço se registra no GNS.

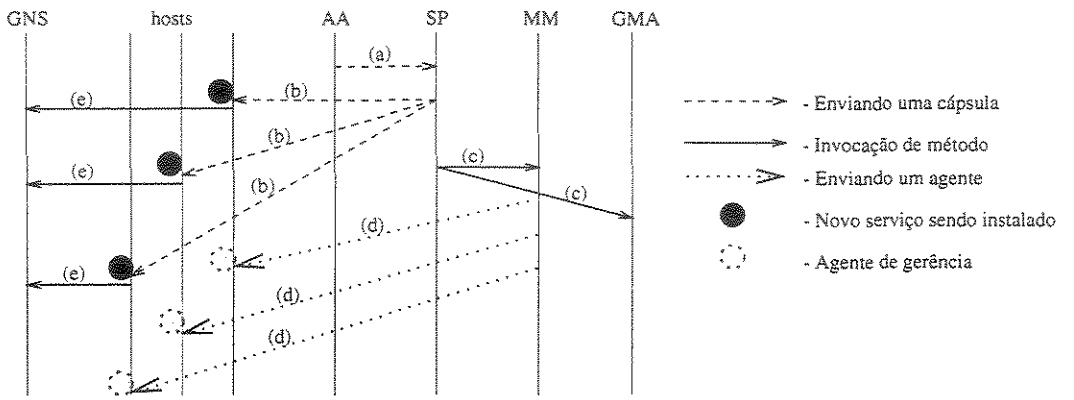



Figura 3.6: Cliente instalando uma VAN em seu domínio.

Note que neste cenário, o PM não é notificado. Isso ocorre porque nenhuma política é verificada quando o cliente deseja instalar uma VAN, ou seja, as políticas não serão aplicadas aos serviços solicitados juntamente com a instalação da VAN. As políticas são aplicadas somente aos serviços solicitados após a instalação de uma VAN. Isso garante que o cliente sempre poderá criar uma VAN, e os serviços solicitados por ele serão prontamente enviados para o seu domínio, a menos que o serviço não esteja registrado no SP.

2. **Cliente adiciona novos serviços à VAN.**

Esse cenário é considerado o mais importante já que envolve a interação entre quase todas as partes que pertencem ao ambiente gerenciado. Além disso, instalar novos serviços em uma VAN é um cenário bastante comum em ambientes Telecom.

A instalação de um novo serviço pode gerar a migração desse serviço de uma VAN para outra. Antes desta migração ocorrer, será necessário encontrar o serviço menos usado na rede (LRUS, seguindo as políticas definidas na seção 3.1.1) e verificar as políticas do serviço a fim de permitir ou não tal migração. O caso mais simples de instalação de um serviço é quando não existe nenhuma cópia do serviço solicitado na rede e, portanto, o SP simplesmente envia uma cópia desse serviço para o cliente. Nessa dissertação, para a demonstração desse cenário, consideramos o caso típico e mais complexo, ou seja, a situação onde existem várias instâncias do mesmo serviço (classe) na rede. Além disso, consideramos que o cliente deseja instalar um serviço externo à sua VAN ocorrendo, portanto, uma migração entre VANs. A Figura

3.7 ilustra graficamente a interação entre as várias partes do ambiente. Abaixo apresentamos a seqüência realizada para representar este cenário.

(a) O cliente envia uma cápsula para o SP informando qual serviço está sendo solicitado e para qual host ele deverá ser enviado. A partir desse momento o cliente deverá aguardar até todas as etapas de instalação do novo serviço serem completadas. O cliente será informado do término da instalação através do recebimento da cápsula que instalou o serviço no host especificado.

(b) O SP, ao receber a cápsula, procura em toda rede (domínio) pelo LRUS;

(c) O SP interage com o MM para verificar se a migração do LRUS encontrado é possível;

(d) O MM obtém as políticas do serviço a fim de aplicá-las sobre o mesmo;

(e) O MM responde para o SP sobre a possibilidade ou não da migração;

(f) Se a migração é possível, o SP envia uma cápsula para a AA da rede ativa virtual onde o LRUS está atualmente instalado. Essa cápsula será responsável pela remoção do serviço de uma VAN e a instalação do mesmo em outra. O envio da cápsula primeiramente para a AA, e não diretamente ao host onde o LRUS está localizado, deve-se a necessidade de avisar a AA para parar de usar o LRUS em sua VAN. Por outro lado, se a migração não é possível (aplicando a política 2 sobre quantidade máxima de migrações, ver seção 3.1.1), e a política 3 é satisfeita (número máximo de cópias do serviço não foi alcançado), o SP envia uma cápsula para o host destino especificado pelo cliente. Essa cápsula criará uma nova instância do serviço naquele host (f'). As etapas *l*, *m* e *n* são as mesmas tanto para uma nova instância quanto para a migração do serviço. Se a política 3 não for satisfeita, ou seja, se o número máximo de cópias daquele serviço foi alcançado, o gerente (humano) deverá decidir sobre esta situação. O gerente poderá forçar a migração ou enviar uma nova instância do serviço para o host;

(g) Se uma nova instância do serviço foi criada, o SP deverá notificar o GMA e o MM sobre o fato. Se a migração ocorreu, o SP notifica somente o GMA (g'). Note que no último caso, não é necessário avisar o MM sobre a migração do serviço. No passo (d), o MM realiza essa verificação e portanto ele já estará sabendo da possibilidade ou não da migração do serviço;

(h) A cápsula que realizará a migração do serviço é então enviada para o host onde o serviço está atualmente localizado;

(i) O serviço se desregistra do GNS e notifica o MAEA responsável pelo seu gerenciamento que ele (o serviço) está migrando;

(j) A cápsula remove o serviço do nó ativo local e migra para o novo host onde o serviço será instalado;

(k) O MAEA local (host de origem) avisa o MAEA do host destino que um novo serviço está chegando;

(l) A cápsula chega ao nó ativo destino, instancia o serviço solicitado e registra-o no GNS;

(m) O serviço notifica o MAEA local sobre sua chegada;

(n) A cápsula é finalmente enviada para a AA do cliente que solicitou o serviço. Com isso, o cliente ficará sabendo que o novo serviço foi instalado e que poderá ser usado.

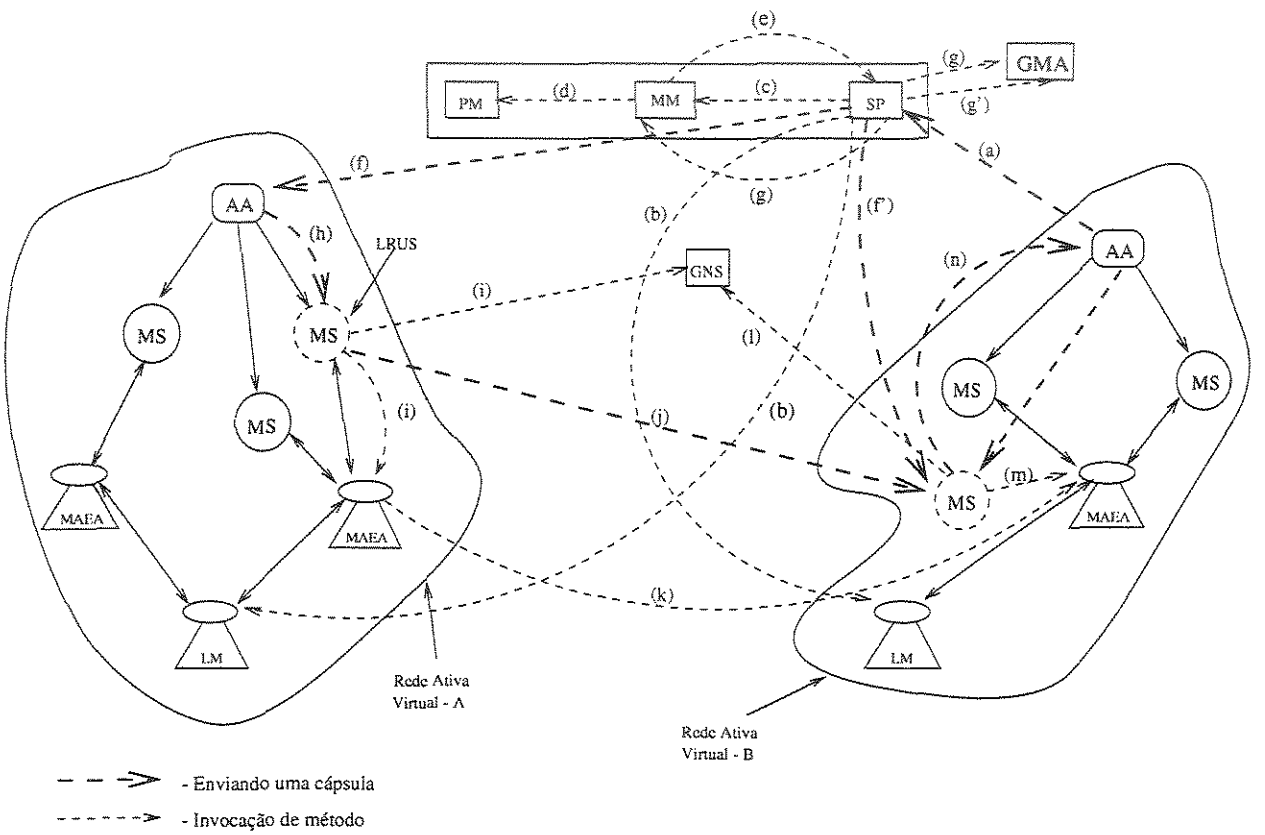



Figura 3.7: Cliente adicionando um serviço em sua VAN.

Cada LM é responsável por encontrar o LRUS em sua VAN. Note a comunicação entre MAEAs de redes virtuais diferentes ocorrendo unicamente devido à migração

de um serviço. A comunicação entre MAEAs pertencentes à mesma VAN ocorre da mesma forma que a comunicação entre MAEAs de redes diferentes.

Além dos cenários explicados acima, existem outros cenários comuns em ambientes Telecom. O primeiro deles é quando um serviço migra internamente à VAN. Normalmente a migração de serviços é realizada entre diferentes VANs, pois o cliente, na maioria dos casos, irá solicitar serviços que ainda não pertencem a sua VAN. Portanto, a migração interna de serviços ocorre principalmente quando o cliente deseja mover um serviço de um host para outro, ou quando o gerente (humano) detecta um host sobrecarregado e decide migrar serviços. No primeiro caso, a migração pode ser feita com propósitos de balanceamento de carga ou com propósitos de levar um serviço para locais onde ele está sendo mais freqüentemente usado. No segundo caso, ou seja, quando o gerente migra um serviço, tipicamente será com propósitos de balanceamento de carga. Se a migração ocorrer dentro da mesma VAN, os passos apresentados acima são os mesmos. Para o sistema de gerência, não importa o fato da migração ser externa ou interna à VAN.

Outro cenário é quando um cliente deseja remover um serviço de sua VAN. Neste caso, o cliente deverá enviar uma cápsula para o SP informando o serviço que ele deseja remover e sua localização. O SP enviará uma cápsula para o host informado a fim de remover o serviço da *cache* daquele nó ativo. O SP deverá informar o sistema de gerência sobre a remoção do serviço. O último cenário é quando o cliente deseja remover uma VAN previamente instalada (fornecida). Neste caso, o cliente deverá enviar uma cápsula para o SP informando o nome da VAN a ser removida. O SP enviará cápsulas para cada nó ativo que pertencia àquela VAN. Em cada nó ativo a cápsula removerá todos os serviços que pertenciam à VAN naquele host. Depois disso, o SP deverá notificar o sistema de gerência sobre a remoção da VAN.

# Capítulo 4

# Implementação

Neste capítulo explicamos alguns aspectos relacionados com a implementação da infra-estrutura para validarmos o MS2VAN. Abordamos o uso do ANTS para criar as VANs, as classes utilizadas, os arquivos de configuração definidos para a troca de cápsulas entre os nós ativos de uma mesma VAN e entre VANs. Mostramos também algumas das principais telas da MRA com resultados de operações de gerência realizadas pelo gerente.

## 4.1 Criando VANs com o ANTS

Enquanto [18] implementou os serviços usando agentes móveis, nessa dissertação, como sendo o principal foco do trabalho, implementamos os serviços usando a tecnologia de redes ativas. Desta forma, o cliente antes de solicitar a instalação de uma VAN, deverá criar um arquivo de configuração com os hosts que farão parte da VAN, informando também qual destes hosts terá uma aplicação ativa (AA). Este arquivo de configuração será usado pelo ambiente ANTS para envio e recebimento de cápsulas. Para as nossas simulações, criamos dois arquivos representando duas VANs. Abaixo mostramos os dois arquivos usados neste trabalho.

Arquivo de configuração para a VAN1:

```
node 1.1.1.2 -routes VAN1.routes
channel 1.1.1.2 pinheiros.dcc.unicamp.br:8000 -log 255
application 1.1.1.2 mgmtsw.ClientApplication -destino 1.1.1.1

node 1.1.1.3 -routes VAN1.routes
channel 1.1.1.3 iguacu:8000
```

```
node 1.1.1.1 -routes VAN1.routes
channel 1.1.1.1 araguaia:8000

connect 1.1.1.2 1.1.1.1
connect 1.1.1.2 1.1.1.3
connect 1.1.1.2 1.1.1.2
```

Arquivo de configuração para a VAN2:

```
node 1.1.1.3 -routes VAN2.routes
channel 1.1.1.3 iguacu.dcc.unicamp.br:9001 -log 255
application 1.1.1.3 mgmtsw.ClientApplication -destino 1.1.1.1

node 1.1.1.5 -routes VAN2.routes
channel 1.1.1.5 xingu:9001

node 1.1.1.2 -routes VAN2.routes
channel 1.1.1.2 pinheiros:9001

node 1.1.1.1 -routes VAN2.routes
channel 1.1.1.1 araguaia:8000

connect 1.1.1.3 1.1.1.1
connect 1.1.1.3 1.1.1.5
connect 1.1.1.3 1.1.1.2
connect 1.1.1.3 1.1.1.3
```

Nestes arquivos de configuração são definidos os endereços no formato do ANTS para cada nó ativo que fará parte da rede ativa (diretiva *node*), um canal de comunicação para cada nó alocado a uma porta (diretiva *channel*) e uma aplicação (diretiva *application*), caso o nó possua uma. A diretiva *connect* é usada para definir as rotas entre os hosts. Note que nem todos os hosts possuem conexão direta com outros hosts (ligação ponto-a-ponto), mas todos conseguem chegar a qualquer nó usando hosts intermediários (roteadores).

A Aplicação Ativa (AA) é denominada *ClientApplication* e estará executando no host pinheiros para a VAN1 e no host iguacu para a VAN2. A VAN1 é formada por dois hosts, pinheiros e iguacu. A VAN2 é formada por três hosts, iguaçu, xingu e pinheiros. Note que o argumento *destino* usado pela *ClientApplication* refere-se ao host 1.1.1.1 (araguaia). É

neste host que encontra-se o fornecedor de serviços (SP) do domínio e é para ele que serão enviadas as cápsulas quando o cliente desejar instalar uma nova VAN ou adicionar novos serviços a ela. O SP também possui um arquivo de configuração, o qual é apresentado abaixo.

```
Arquivo de configuração definido para o SP:

    node 1.1.1.1
    channel 1.1.1.1 araguaia.dcc.unicamp.br:8000 -log 255
    application 1.1.1.1 mgmtsw.ServiceProvider
```

Este arquivo de configuração é muito simples pois o fornecedor não necessita de rotas definidas, já que ele não sabe quais hosts formarão as futuras VANs. Desta forma, quando o cliente deseja instalar uma VAN, ele deverá informar o nome do arquivo de configuração criado para a VAN. Por exemplo, se um cliente deseja criar uma VAN usando o arquivo VAN2, definido acima, ele deverá colocar esta informação na cápsula que será enviada para o SP. No arquivo de configuração do SP definimos apenas a aplicação que irá executar no nó ativo 1.1.1.1, neste caso a classe que representa o SP será instanciada.

O SP, não possuindo rotas definidas em seu arquivo de configuração, deverá saber de alguma forma como enviar cápsulas para instalar os serviços nos hosts informados pelos clientes. Por isso, quando o SP receber uma cápsula solicitando a instalação de uma nova VAN, a primeira operação a ser realizada é ler o arquivo de configuração informado pelo cliente. Abaixo mostramos o trecho de código que realiza esta tarefa.

```
synchronized public void receive (Capsule cap) {
    super.receive(cap);
    if (cap instanceof RequireNewVANCapsule) {
        RequireNewVANCapsule capAux = (RequireNewVANCapsule)cap;
        try {
            thisNode().getRoutes().read(capAux.getFileName());
        }catch(Exception e) {
            e.printStackTrace();
        }
    }
}
```

A função *getFileName()* obtém o nome do arquivo de configuração da VAN sendo solicitada. A função *read* da classe *Node* irá ler este arquivo e criar rotas para os hosts

definidos no arquivo em tempo de execução. Essas rotas serão válidas até o nó ativo local ler outro arquivo de configuração. O código completo da cápsula *RequireNewVANCapsule* está no Apêndice A.1.

As principais classes usadas do *toolkit* ANTS foram:

- *DataCapsule*: estende a classe básica *Capsule*. Usada para a criação das cápsulas para a comunicação entre cliente e fornecedor, e para a instalação e remoção de serviços dos nós;
- *Protocol*: para a definição do protocolo para o ambiente. Um protocolo está associado com as cápsulas definidas para ele. Isto garante que somente estas cápsulas poderão ser enviadas e recebidas pelos nós que usam este protocolo;
- *Node*: representa uma instância de um nó ativo. É o ambiente onde as cápsulas são avaliadas e os serviços são instalados. Um único nó físico pode executar vários nós ativos de forma independente;
- *Application*: o cliente estende esta classe para enviar e receber cápsulas.

Para suportar a infra-estrutura das ANs no ambiente proposto, definimos as seguintes cápsulas que formam o protocolo usado neste trabalho:

- Uma cápsula para o cliente solicitar a instalação de uma VAN e seus serviços: *RequireNewVANCapsule*;
- Uma cápsula para instalar serviços em nós ativos: *InstallServiceCapsule*;
- Uma cápsula para realizar a remoção e instalação de um serviço na rede: *ServiceMigrationCapsule*. Na prática esta cápsula faz a migração de um serviço de um nó ativo para outro. Por fim, ela também notifica a aplicação do cliente que o serviço solicitado está instalado;
- Uma cápsula para o cliente instalar um novo serviço em sua VAN: *GetNewServiceCapsule*;
- Uma cápsula para remover um serviço de um nó ativo: *RemoveServiceCapsule*;
- Uma cápsula utilizada unicamente pelo cliente para usar os serviços instalados em sua VAN: *ServiceUse*;
- Uma cápsula de mensagens usada pelo SP para avisar o cliente que o serviço solicitado não pode ser instalado: *MessageCapsule*;

- Uma cápsula para o cliente remover um serviço de sua VAN: *RemoveServiceFromVANCapsule*;
- Uma cápsula para o cliente remover uma VAN previamente instalada: *RemoveVANCapsule*;
- Uma cápsula para remover todos os serviços de uma VAN em um nó ativo: *RemoveAllServicesCapsule*.

A cápsula de mensagens pode ser usada pelos componentes das ANs para a troca de avisos e para notificar sobre possíveis erros ocorridos. Neste trabalho ela é usada simplesmente pelo fornecedor para avisar o cliente que um determinado serviço não pode ser instalado. Poderia-se estender a utilidade desta cápsula criando códigos de erros a fim de especificar os motivos das falhas, por exemplo, quando o serviço solicitado não está registrado no SP.

Todas as cápsulas foram implementadas faltando apenas as três últimas que estão relacionadas a cenários simples não necessários para validar o modelo. Por isso, estas cápsulas não aparecem na classe que define o protocolo utilizado para os testes, classe esta apresentada no Apêndice C.

## 4.2 Interfaces da MRA

Para o gerente invocar operações de gerência assim como visualizar os resultados destas operações, desenvolvemos a Aplicação Remota de Gerenciamento (MRA). Esta aplicação comunica-se com o GMA através da invocação de métodos remotos (RMI) e usa o pacote *swing*, principalmente o componente *JTable*, para apresentação dos resultados.

As operações da MRA estão divididas em operações de contabilidade, desempenho e configuração. Além disso há opções para registrar serviços no SP e inserir políticas no banco de dados de políticas. Há também operações para saber os hosts de cada VAN e os serviços de cada host. Por último podemos mostrar graficamente a composição das VANs dando ao gerente uma visão rápida de como uma VAN está formada.

A coleta de dados de contabilidade é dividida por host, por VAN e em todo o domínio, conforme mostra a Figura 4.1.

Ainda podemos saber as invocações recebidas em um determinado host ou em um serviço específico. A Figura 4.2, mostra a MRA após o gerente invocar a operação para descobrir o serviço mais usado na VAN2. Da mesma forma, a Figura 4.3, apresenta o resultado após o gerente invocar a operação para saber o número de invocações recebidas em um determinado serviço. Neste caso, o gerente desejava descobrir o número de invo-

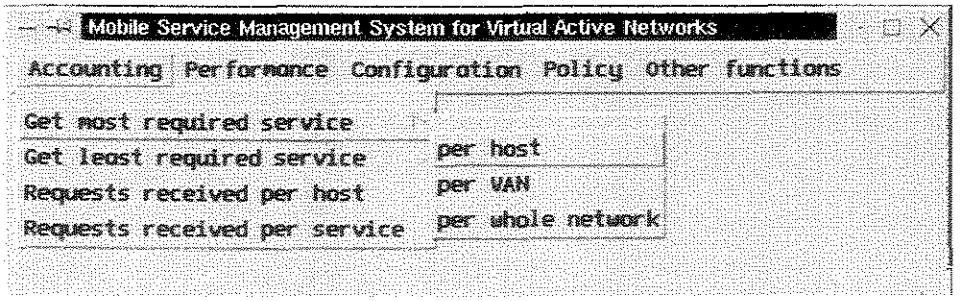



Figura 4.1: Opções para gerência de contabilidade.

cações recebidas no serviço cujo nome é **VAN11.1.1.2a**. Outras telas relacionadas com a área de contabilidade encontram-se no Apêndice D.

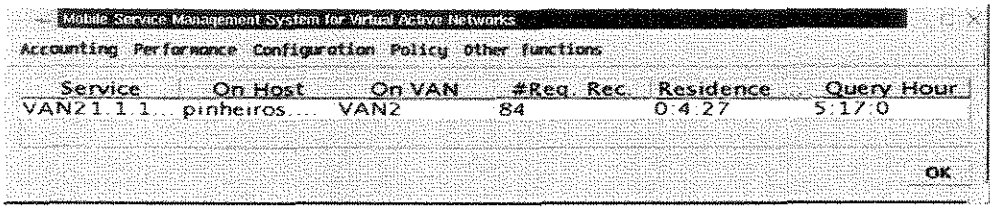



Figura 4.2: MRA após a gerência para descobrir o serviço mais usado na VAN2.

A coleta de dados de desempenho envolve unicamente a coleta de dados sobre o uso de memória e CPU. Além disso, quando o gerente invoca a operação para a coleta destes dados, o sistema de gerência dispara automaticamente a coleta de dados de contabilidade. Assim, ao final do tempo determinado pelo gerente, é mostrado na tela um conjunto de informações tanto de desempenho como de contabilidade. Com estas informações, podemos gerar uma tabela de decisão de hosts com possíveis problemas. A Figura 4.4 mostra como fica a MRA após este tipo de operação de gerência. Podemos observar nesta figura que o host iguaçu está com o uso de memória e CPU acima da média, e por isso necessita de uma gerência mais específica. O gerente poderia obter os serviços que estão instalados neste host e descobrir se algum deles está causando a sobrecarga. Uma possível solução neste caso seria realizar a migração de serviços para outros hosts a fim de distribuir a carga de processamento.

As operações para a área de configuração envolvem a remoção de serviços pouco usados conforme a política definida para eles, e a migração de um serviço de um host para

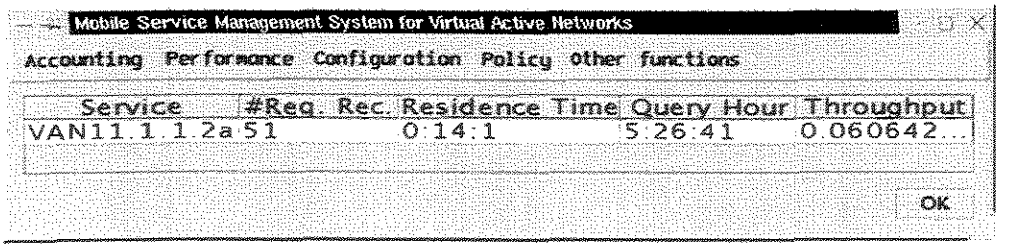



Figura 4.3: MRA após a gerência sobre um determinado serviço.

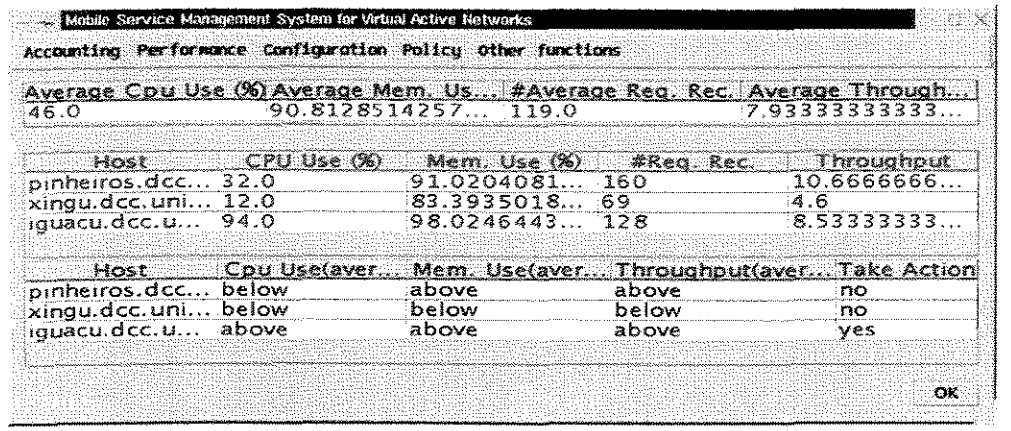



Figura 4.4: MRA após a gerência de desempenho e contabilidade.

outro dentro da mesma VAN. A primeira operação é dividida em duas etapas conforme explicamos no capítulo anterior. A primeira etapa é descobrir os serviços pouco usados e listá-los para o gerente. A segunda etapa é quando o gerente realmente remove o(s) serviço(s), baseando-se nas requisições recebidas pelo serviço ou baseando-se em uma decisão própria.

Dentre as operações restantes, implementamos uma operação que apresenta as migrações de um determinado serviço. Esta operação permite ao gerente saber quais os hosts em que um serviço passou, qual o tempo de residência e quantas requisições o serviço recebeu em cada host. A Figura 4.5 mostra a MRA após a invocação desta operação realizada para saber as migrações do serviço **VAN21.1.1.2c**

Outra operação permite ao gerente visualizar uma VAN na qual os hosts são representados por pontos e um ciclo conectando estes pontos representa uma VAN específica. Com isto, o gerente poderá saber quais são as VANs existentes em seu domínio, qual a composição de cada VAN e quais hosts pertencem a mais de uma VAN. Além disso, esta

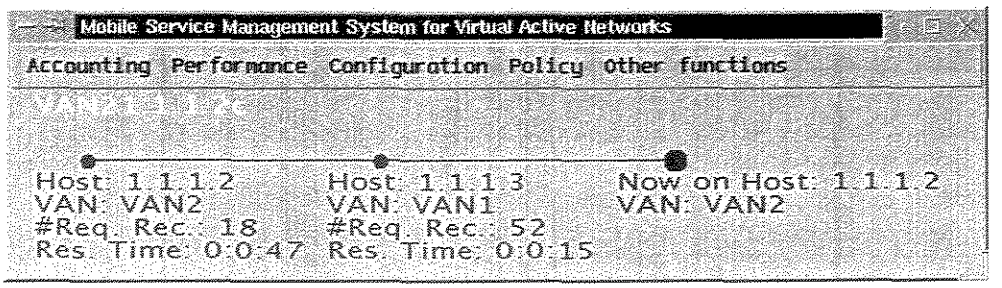



Figura 4.5: MRA depois de gerenciar as migrações de um serviço.

mesma operação permite ao gerente saber se alguma destas VANs pertence a uma Rede Ativa Privada Virtual (*Virtual Private Active Network-VPAN*). Uma VPAN é uma rede privada formada por hosts de diferentes domínios para permitir a troca de informações entre os domínios participantes. Explicaremos mais sobre as VPANs e também sobre sua gerência no capítulo 5. A Figura 4.6 apresenta a MRA após o gerente invocar a operação para visualizar as VANs instaladas em seu domínio. Podemos observar que há duas VANs instaladas neste domínio, a VAN1 com dois hosts e a VAN2 com três hosts. Note que os hosts 1.1.1.2 e 1.1.1.3 pertencem às duas VANs.

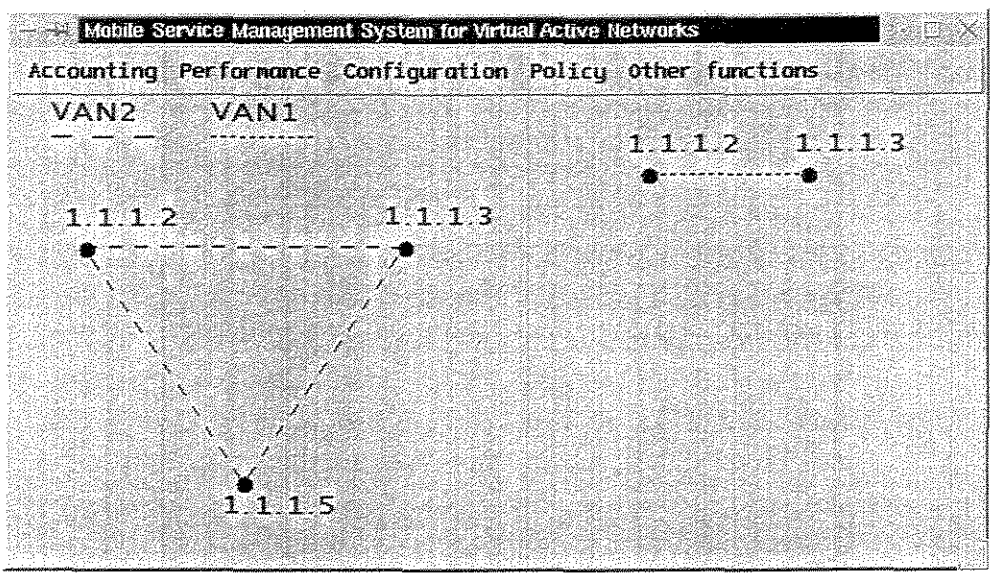



Figura 4.6: VANs instaladas em um domínio específico.

## 4.3 O Serviço de Nomes Global - GNS

O Serviço de Nomes Global (GNS) foi desenvolvido para facilitar a busca pelos serviços no ambiente distribuído. Todos os serviços estão registrados no GNS e qualquer componente que desejar invocar um serviço deverá obter a referência para o mesmo usando o GNS. O GNS foi implementado devido à falta de transparência de localização do RMI. Com o RMI, o cliente que deseja invocar um serviço informa, além do nome do serviço, o host onde ele está localizado. Se o cliente não sabe o host do serviço, ele deve procurar em todos os possíveis hosts em que o serviço pode estar localizado.

O GNS foi desenvolvido usando o pacote JNDI (*Java Naming and Directory Interface*). O JNDI é uma API que oferece funcionalidades para serviços de nomes e diretórios para aplicações Java [40]. Uma das grandes vantagens do JNDI é a sua independência em relação a outros serviços de diretórios específicos. Assim, uma grande variedade de serviços de diretórios tais como, DNS (*Domain Name System*), LDAP (*Lightweight Directory Access Protocol*), RMI e CORBA podem ser acessados.

Para este trabalho usamos o JNDI com o RMI. Assim, rodamos o *rmiregistry* nos hosts que pertencem ao domínio que estamos interessados em gerenciar. Isso é necessário pois o RMI exige que um *rmiregistry* esteja rodando na mesma máquina do servidor. Mas com o GNS, evitamos que a procura por um serviço seja feita em cada host até encontrar o objeto servidor desejado. O GNS mantém uma referência para o *rmiregistry* de cada host e, portanto, quando um cliente invocar o GNS para obter a referência para algum serviço, o GNS redirecionará a requisição para o *rmiregistry* do host onde o serviço está localizado.

O GNS possui dois métodos, *bind()* e *unbind()*. O método *bind()* recebe o nome do serviço e uma URL correspondente à localização do serviço. O GNS cria uma referência usando estas duas informações e a registra no contexto criado. O método *unbind()* apenas remove a referência para um serviço do contexto. Os dois métodos são mostrados abaixo e o código completo do GNS está no Apêndice B.

```
public void bind (String url, String nameService) throws RemoteException {
    Reference ref = new Reference(nameService,new StringRefAddr("URL", url));
    try {
        ctx.rebind(nameService,ref);
    }catch (NamingException e) {
        e.printStackTrace();
    }
}
```

```
public void unbind (String nameService) throws RemoteException {
    try {
        ctx.unbind(nameService);
    }catch (NamingException e) {
        e.printStackTrace();
    }
}
```

A URL está no formato *"rmi://+IP+:porta/+nomeDoServico"* indicando a localização exata do serviço. A porta fornecida na URL é a porta na qual o *rmiregistry* está registrado, já que não estamos usando a porta padrão (1099). Ela não está relacionada ao serviço sendo registrado, mas sim ao *rmiregistry* "escutando" nesta porta. O cliente obtém do GNS a referência para o *rmiregistry* do host onde o serviço está localizado. Através do *rmiregistry* daquele host, obtém-se a referência direta para o serviço.

O nome de um serviço é o resultado da concatenação entre o nome da VAN em que o serviço está instalado, o nó ativo onde o serviço está localizado e o nome do serviço dado pelo fornecedor. Um exemplo é mostrado abaixo.

Nome da VAN: **VAN2**;
Nome do host: **1.1.1.5**;
Nome do serviço dado pelo fornecedor: **serviceA**;

O nome final do serviço será: **VAN21.1.1.5serviceA**.

Este nome será único em toda rede. A cada migração, o nome do serviço é atualizado para refletir a nova localização.

## 4.4 Outros Aspectos

Para coletar os dados relacionados ao uso de memória e CPU, utilizamos funções típicas dos sistemas operacionais Unix. A coleta de dados sobre o uso de memória foi feita usando a função *top*. Para a coleta de dados sobre o uso da CPU foi usada a função *sar*.

A classe Extensão do ANTS não foi usada para oferecer serviços aos clientes. A oferta de serviços não deve estar vinculada a nenhum tipo de arquitetura, modelo ou linguagem. A rede ativa deve ser usada apenas como um meio para facilitar a disponibilização de serviços na rede. Para a implementação do protótipo atual, não faz diferença usarmos ou não a classe Extensão oferecida pelo ambiente ANTS, já que estamos trabalhando somente com objetos Java. Mas para flexibilizar possíveis trabalhos futuros, onde os serviços poderão ser objetos CORBA, optamos em não amarrar a oferta de serviços herdando

classes do próprio ambiente de redes ativas. Assim, na implementação do modelo, os serviços oferecidos são simples objetos Java-RMI que são instanciados e colocados nas *caches* em nós definidos pelo usuário.

A plataforma de agentes móveis Grasshopper foi utilizada simplesmente para enviar os agentes de gerenciamento para os hosts. A comunicação entre os agentes foi feita através de *sockets* usando invocações síncronas e assíncronas. As invocações assíncronas foram usadas pelo LM a fim de disparar a coleta de dados de contabilidade em cada MAEA de sua rede ativa. Para os testes, criamos uma única região onde todas as agências foram registradas. O lugar padrão, *"InformationDesk"*, foi usado pelos agentes de gerência.

O protótipo foi desenvolvido em ambiente Solaris usando JDK-1.2. A Figura 4.7 mostra a distribuição dos componentes desenvolvidos e os hosts utilizados para as simulações.

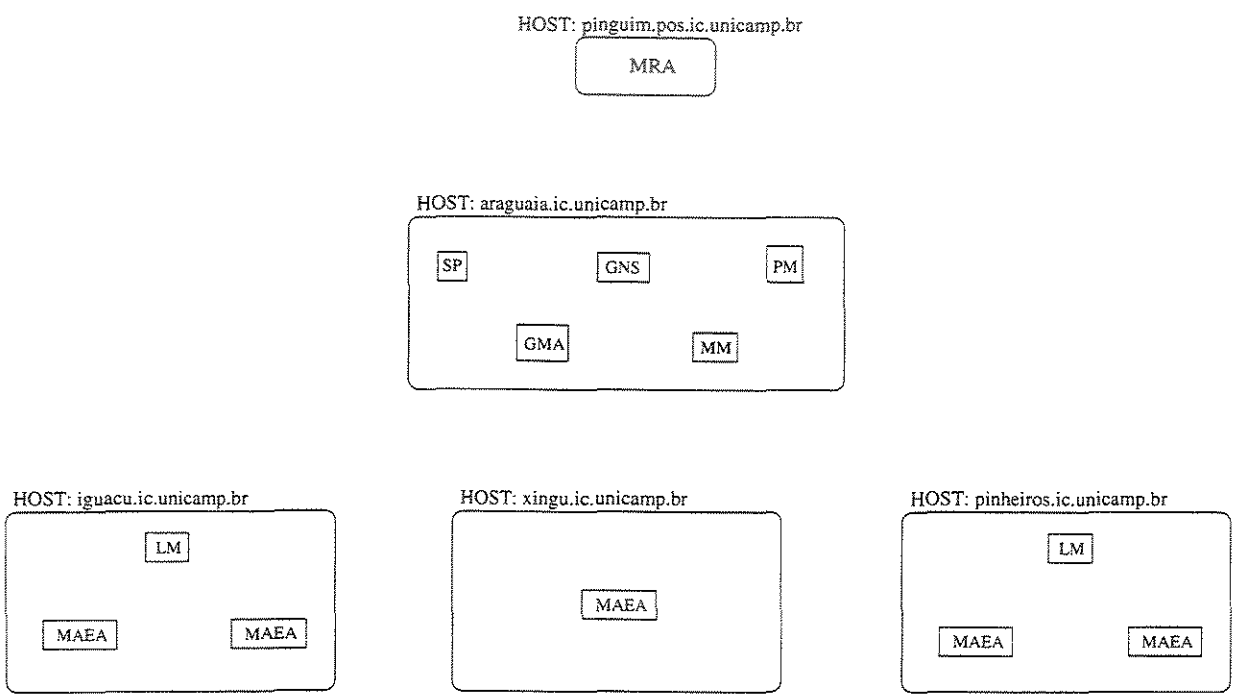



Figura 4.7: Distribuição dos componentes pela rede do Instituto de Computação.

A figura reflete exatamente a configuração usada nos testes. Os hosts pinheiros e iguaçu pertencem a duas VANs instaladas, por isso há dois MAEAs em cada um deles. Além disso, há um LM no host iguaçu responsável pela gerência de uma das VANs, e outro LM no host pinheiros responsável pela gerência da outra VAN. O host xingu pertence somente a uma VAN, portanto, um MAEA está localizado neste host. Os componentes GNS, PM, SP, MM e GMA estão todos no host araguaia. Por fim, a MRA está execu-

tando no host pinguim (PC com Linux), que desempenhará as tarefas de uma estação de gerência.

# Capítulo 5

# Aplicações

Nesta seção apresentamos como o nosso sistema gerencia dois serviços típicos de Telecomunicação. O primeiro serviço a ser gerenciado chama-se *CallForwarding* e serve para realizar o roteamento de chamadas telefônicas para outros dispositivos especificados pelo cliente. A possibilidade de criar Redes Privadas Virtuais (*Virtual Private Network - VPN*) caracteriza o segundo serviço gerenciado. Ambos os serviços podem ser encontrados em [18]. A seguir, explicaremos cada um separadamente.

## 5.1 Serviço de CallForwarding

A tarefa a ser realizada por esse serviço é muito simples. Ele possui um *script*, definido pelo seu usuário, que será analisado toda vez que uma chamada telefônica é recebida. O *script* possui informações necessárias para rotear uma chamada dependendo da hora e/ou de um evento. Cada usuário desse serviço poderá definir seu próprio *script* para atender as suas necessidades específicas. Abaixo mostramos um exemplo de *script* definido para este serviço:

```
Serviço de CallForwarding:

Roteamento baseado no tempo:
    09:00-12:00:  --> 282
    12:00-15:00:  --> 293

Roteamento baseado em eventos:
    if (origemDaChamada == 239)
        --> 200
```

Definimos um serviço de CallForwarding específico e um serviço de CallForwarding geral. O primeiro pode atender somente um setor ou um departamento de uma empresa. O serviço geral poderá atender vários setores da empresa ou até mesmo toda a empresa, e por isso deverá migrar entre setores. Cada setor é representado por uma VAN e por isso aplicaremos a política do serviço interno e externo definida no capítulo 3.1.1. O serviço específico poderá migrar somente dentro da mesma VAN, e o serviço geral poderá migrar entre VANs para atender aos vários setores. Assim, um cliente de uma VAN (setor) poderá solicitar a instalação de um serviço específico em sua VAN ou poderá solicitar a migração de um serviço geral localizado em outra VAN para atender a sua necessidade. Para a simulação deste cenário, nós criamos duas VANs: VAN1 e VAN2. Na primeira o cliente solicitou a instalação de um serviço de CallForwarding específico. Na segunda VAN o cliente solicitou a instalação de um serviço geral. Depois de algum tempo, usamos o sistema de gerência para coletar informações sobre as migrações dos serviços de CallForwarding instalados na rede. As Figuras 5.1 e 5.2 mostram, além das migrações dos dois serviços, algumas informações de contabilidade e tempo de residência de cada serviço em cada host.

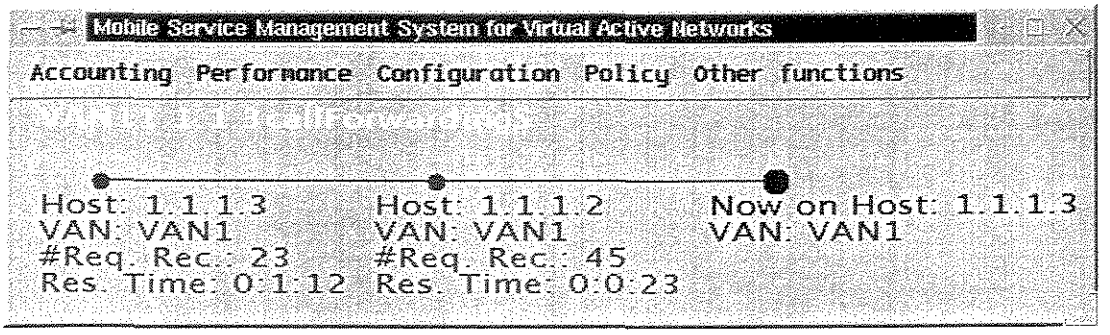



Figura 5.1: Migração do serviço de CallForwarding específico.

O serviço de CallForwarding específico (Figura 5.1), para esta operação de gerência, é denominado *VAN11.1.1.3CallForwardingS*, e da mesma forma o serviço de CallForwarding geral (Figura 5.2), é denominado *VAN11.1.1.2CallForwardingG*. Os nomes são criados conforme definido no capítulo 4.

Podemos observar que enquanto o serviço específico migrou somente dentro da VAN1, o serviço geral migrou entre a VAN1 e a VAN2. Inicialmente o serviço geral está no host 1.1.1.5 (xingu) que pertence à VAN2. Depois de algum tempo, ele está no host 1.1.1.2 (pinheiros) que pertence à VAN2. Este exemplo prático, típico de ambientes Telecom, serviu para demonstrarmos como os serviços móveis que migram somente dentro de uma VAN ou externamente entre várias VANs podem ser gerenciados usando o sistema

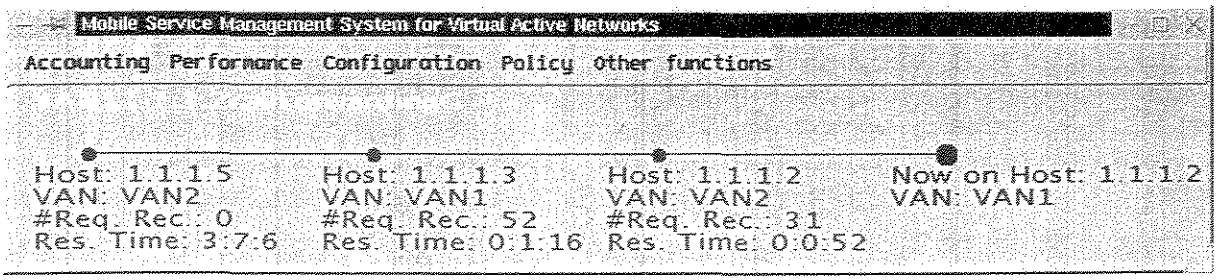



Figura 5.2: Migração do serviço de CallForwarding geral.

proposto.

## 5.2 Serviço de Rede Privada Virtual

O serviço de VPN permite que diferentes organizações ou companhias definam redes privadas entre elas usando os recursos de redes públicas. No nosso modelo, uma rede privada pode ser criada a partir de diferentes VANs de diferentes empresas formando uma Rede Ativa Privada Virtual (*Virtual Private Active Network - VPAN*). O objetivo deste serviço é mostrar como o modelo que propomos pode ser usado para gerenciar uma VAN que é formada por hosts de diferentes domínios. Até aqui nós gerenciamos VANs que pertencem ao mesmo domínio, e portanto um GMA é responsável por todas as VANs instaladas. Agora, com vários domínios a questão a ser respondida é: como um gerente de um domínio poderá coletar informações de hosts e serviços que pertencem a outros domínios? Para esse caso, foram acrescentadas algumas extensões unicamente no GMA para permitir a comunicação entre GMAs de diferentes domínios. Com isso, os GMAs podem interagir para instalar e gerenciar tanto as VPANs como seus serviços. O GMA de um domínio terá acesso somente às informações de gerência dos serviços que pertencem a sua VPAN. Um GMA de um determinado domínio não terá acesso às informações de gerência de serviços de outros domínios que não estejam incluídos na VPAN.

Neste novo cenário, um cliente que deseja instalar uma VPAN poderá escolher a topologia informando qual o host de seu domínio irá participar da VPAN e quais os outros domínios que farão parte desta VPAN. A instalação de uma VPAN é diferente da instalação de uma VAN. Primeiramente o cliente notifica seu GMA local para iniciar a instalação da VPAN. O GMA local irá interagir com os GMAs dos domínios informados pelo cliente a fim de negociar a instalação da VPAN. Se eles chegam a um acordo, a VPAN será instalada seguindo as regras que foram negociadas, e a troca de serviços entre domínios diferentes pode ser feita. As etapas necessárias para gerenciar uma VPAN

também são diferentes das etapas necessárias para gerenciar VANs instaladas no mesmo domínio. Para um gerente coletar dados de uma VPAN, será necessário interagir com os GMAs dos outros domínios a fim de que eles enviem as informações sobre os hosts e serviços daqueles domínios que pertencem à VPAN sendo gerenciada. Qualquer gerente de qualquer domínio pertencente a uma VPAN poderá gerenciá-la após ela ter sido instalada. A Figura 5.3 ilustra as etapas desse tipo de gerência. Nesse exemplo, o gerente do domínio 2 deseja obter informações sobre a VPAN a qual ele faz parte.

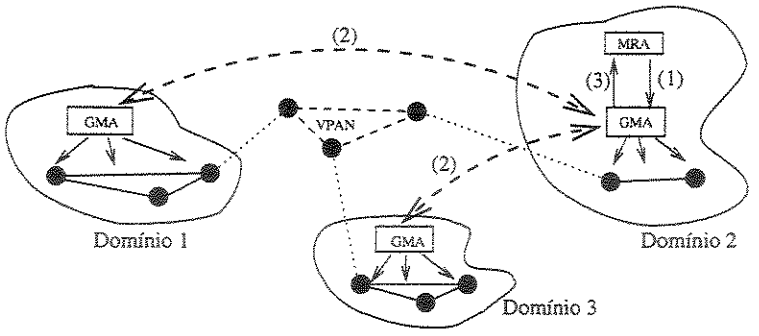



Figura 5.3: Etapas para gerenciar uma VPAN.

A figura mostra uma VPAN criada usando hosts dos domínios 1, 2 e 3. A operação de gerência é disparada pelo gerente do domínio 2 (1). O GMA do domínio 2 interage com os GMAs dos domínios 1 e 3 (2). Após o GMA do domínio 2 obter as informações desejadas, elas são enviadas para a MRA do gerente local (3). Com esse mecanismo, qualquer informação para a área de contabilidade ou desempenho pode ser obtida. A Figura 5.4 mostra a MRA após o gerente do domínio 2 ter invocado uma operação para saber quantas VANs estão instaladas naquele domínio. Essa operação também mostra as VPANs em que o domínio 2 está inserido. Nesse caso, o domínio 2 possui duas VANs instaladas e o host 1.1.1.2 é parte da VPAN1 formada também pelos domínios 1 e 3. A operação de gerência para coletar dados de contabilidade e desempenho em VPANs não foi implementada nesse trabalho.

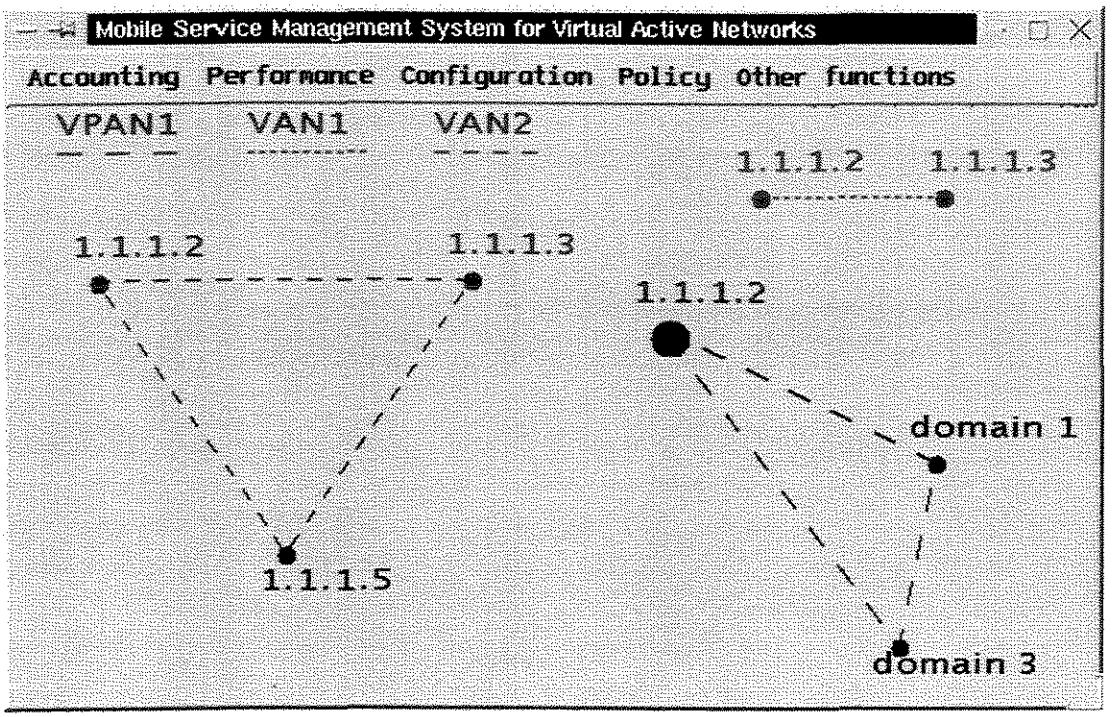



Figura 5.4: MRA mostrando as VANs e VPANs do domínio 2.

# Capítulo 6

# Conclusão

A tecnologia das redes ativas oferece facilidade e flexibilidade para instalar serviços na rede. Os ambientes de Telecomunicação normalmente exigem estas características e, portanto, podem fazer uso desta tecnologia para disponibilizar serviços para seus clientes. Estes serviços representam a necessidade de um certo usuário ou grupo de usuários que tipicamente os solicitam de um fornecedor para instalá-los em seu domínio. Muitas vezes estes serviços são móveis e podem migrar de um local para outro a fim de atender as solicitações dos clientes. A necessidade de gerenciar o uso destes serviços, assim como os recursos utilizados pelos mesmos, constitui um tópico de pesquisa importante.

Nesta dissertação, desenvolvemos um modelo de gerência para serviços em redes ativas virtuais (VANs), abordando três das cinco áreas funcionais de gerenciamento: contabilidade, desempenho e configuração. O modelo foi validado através da implementação de um protótipo dando origem ao MS2VAN, acrônimo atribuído ao sistema de gerência implementado. Alguns dos componentes do sistema foram desenvolvidos usando a tecnologia de Agentes Móveis e considera a existência de serviços móveis, onde um serviço pode migrar de uma VAN para outra.

O modelo proposto neste trabalho considera que os ambientes de Telecomunicação usam uma infra-estrutura baseada na tecnologia das redes ativas para o envio e instalação de serviços na rede. Por outro lado, o modelo pode ser aplicado em outros ambientes, até mesmo mais complexos, com serviços móveis desenvolvidos usando outra tecnologia para transferência dos objetos pela rede. Tais ambientes poderiam usar, por exemplo, a tecnologia de Agentes Móveis para suportar as migrações.

O modelo hierárquico definido facilita na atribuição de tarefas para a coleta de dados de contabilidade, assim como diminui o número de agentes de gerência necessário em ambientes onde há uma grande quantidade de serviços, tipicamente em ambientes de Telecomunicação. A aplicação de filtros em cada nível de gerência diminui a quantidade de dados que devem ser trocados pelos agentes.

A área de gerência de contabilidade preocupa-se em coletar métricas sobre a quantidade de requisições recebidas por um determinado serviço. Este conjunto de dados coletados, quando analisado isoladamente, não oferece informação suficiente ao gerente para detectar possíveis problemas no ambiente gerenciado. Por outro lado, a área de gerência de desempenho oferece métricas que permitem ao gerente detectar hosts cujo uso de memória e CPU estão acima da média. Entretanto, se estas métricas são também analisadas isoladamente, as informações não serão suficientes para o gerente tentar uma possível ação. É necessário, portanto, que as informações, tanto de contabilidade quanto de desempenho, sejam analisadas pelo gerente de forma conjunta a fim de que ele possa realizar a ação de configuração mais apropriada para tentar resolver o problema. Por isso, neste trabalho, os dados de contabilidade e desempenho podem ser coletados e analisados separadamente, mas ao mesmo tempo eles podem ser apresentados para o gerente simultaneamente em uma única tela, facilitando a tomada de decisão.

A área de gerência de configuração foi considerada e, por isso, algumas ações de configuração respresentativas foram definidas. O sistema permite ao gerente migrar um serviço de um host para outro a fim de balancear a carga entre hosts. Também é possível que o gerente exclua serviços que estão sendo pouco utilizados na rede. Com isto, tentamos oferecer ao gerente algumas funcionalidades para facilitar a gerência e a solução de problemas.

Atualmente, não somente em ambientes de Telecomunicação, há uma crescente necessidade de definição de políticas tanto para os serviços como para o ambiente. O modelo considera a existência de políticas e, portanto, desenvolvemos alguns componentes para o controle e a aplicação das políticas definidas. Percebemos que as ações de configuração estão relacionadas às políticas existentes e, portanto, na medida em que acrescentamos novas políticas, novas ações de configuração são necessárias.

O modelo não suporta somente a gerência mas também oferece a possibilidade para que os clientes criem suas VANs e instalem novos serviços. Com isso, consideramos que um *framework* com funcionalidades para o fornecimento e instalação de serviços e para a gerência de um típico ambiente de Telecomunicação foi desenvolvido.

A implementação usando Agentes Móveis auxiliou no envio e recebimento dos agentes de gerência para os locais onde a gerência deve ser realizada. O ANTS, apesar de suportar somente serviços desenvolvidos na linguagem Java, é muito flexível e permite criar redes ativas de uma forma muito simples. Além disso, existe uma separação bem definida entre a infra-estrutura para fornecer e instalar VANs e serviços em redes ativas e os componentes responsáveis pela gerência deste ambiente. Com isso, incrementar o modelo e inserir novas funcionalidades, tanto nos componentes de gerência como nos componentes para o fornecimento de serviços, torna-se uma tarefa simples.

Para testar o modelo, a gerência de dois serviços de Telecom foi realizada: o serviço

de VPN e o serviço de *Call Forwarding*. Em ambos, a instalação de VANs e de serviços, a gerência do ambiente e a existência de serviços móveis foram consideradas. Com isto, o modelo apresenta-se viável para ambientes de Telecomunicação onde existem políticas e os serviços podem ser móveis ou estáticos.

Alguns dos ítens que vislumbramos como trabalhos futuros são:

- Considerar não somente serviços implementados na linguagem Java mas também em outras linguagens. Uma solução é definir serviços CORBA e então utilizar a interface MASIF [32] padronizada pelo OMG (*Object Management Group*) [34] para gerenciar estes objetos;

- Acrescentar uma etapa de negociação entre cliente e fornecedor. Nesta etapa pode-se negociar a qualidade do serviço que está sendo fornecido e a disponibilização dos recursos que serão utilizados pelo cliente, caso estes sejam propriedades do fornecedor. Pode-se definir um contrato entre as partes e criar um mecanismo de gerência para garantir que o contrato seja cumprido;

- Realizar uma gerência antes da instalação dos serviços. Este tópico é comentado em [3] e considera que os serviços possuem políticas e necessidades para a instalação. Desta forma, é necessária a realização de uma gerência na rede a fim de analisar a disponibilização dos recursos para, após isto, instalar cada serviço no local mais adequado. No nosso modelo, o cliente não precisaria informar a localização de cada serviço, isto seria definido pelo fornecedor usando o sistema de gerência para definir o melhor local para a instalação de cada serviço, seguindo as políticas definidas para cada um deles;

- Considerar a existência de mais de um fornecedor oferecendo o mesmo serviço. Este aspecto sugere uma nova abordagem a ser considerada no momento em que um determinado cliente deseja instalar um serviço. Neste caso, o cliente deverá ficar sabendo, de alguma forma, que existem várias implementações do mesmo serviço sendo oferecidas em mais de um fornecedor. Assim, deve haver um mecanismo (*facility*) que faça a negociação entre um cliente e vários fornecedores, objetivando minimizar o uso dos recursos e garantir a qualidade do serviço;

- Permitir que os serviços mantenham seu estado após as migrações. No modelo atual o estado dos serviços é perdido quando os mesmos migram de um local para outro. Para transferir este estado, é necessário o desenvolvimento de cápsulas responsáveis por esta transferência. O desenvolvedor de cada serviço deve criar as cápsulas correspondentes a cada classe de serviço a fim de transferir o estado dos mesmos. Para o sistema de gerência isto seria transparente já que o envio de estados de um local para outro seria tarefa da infra-estrutura das redes ativas.

# Referências Bibliográficas


[1] M. Brunner. Active Networks and its Management. *Journal Annals of Telecommunications,* pp. 144-151, 2001.

[2] C. A. C da Rocha, A. P. Braga e J. N. de Souza. Uma Abordagem para Implantação de Nós Programáveis em Redes de Comunicação. *19º Simpósio Brasileiro de Redes de Computadores (SBRC'01),* Florianópolis-SC, Brasil, pp. 696-710, Maio de 2001.

[3] R. Haas, P. Droz e B. Stiller. Distributed Service Deployment over Programmable Networks. *IEEE 12th International Workshop on Distributed Systems: Operations & Management - DSOM'01,* Nancy, França, pp. 113-127, Outubro de 2001.

[4] I. W. Marshall, H. Gharib, J. Hardwicke e C. Roadknight. A Novel Architecture for Active Service Management. *Seventh IFIP/IEEE International Symposium on Integrated Network Management,* Seattle, Washington, USA, Maio de 2001.

[5] R. Orfalli e D. Harkey. *Client/Server Programming with JAVA and CORBA*. Editora John Wiley & Sons, Inc., second edition, 1998.

[6] D. L. Tennenhouse e D. J. Wetherall. Towards an Active Network Architecture. *Computer Communication Review*, 26(2), Abril de 1996.

[7] D. J. Wetherall, J. V. Guttag, e D. L. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *IEEE OPENARCH'98*, San Francisco, CA, Abril de 1998.

[8] F. L. Verdi e E. R. M. Madeira. A Mobile Agent-based Model for Service Management in Virtual Active Networks. *IEEE 12th International Workshop on Distributed Systems: Operations & Management - DSOM'01,* Nancy, França, pp. 101-112, Outubro de 2001.

[9] D. L. Tennenhouse, M. Smith, W. D. Sincoskie, D. J. Wetherall e G. J. Minden. A Survey of Active Network Research. *IEEE Communications Magazine,* pp. 80-86, Janeiro de 1997.

[10] M. Baldi e P. Picco. Evaluating the Tradeoffs of Mobile Code Design Paradigms in Network Management Applications. *20th International Conference on Software Engineering (ICSE'98)*, Kyoto, Japão, Abril de 1998.

[11] B. Schulze, E. R. M. Madeira e P. Ropelatto. MomentA: Service Management using Agents in a CORBA Environment. *Journal of Network and Systems Management (JNSM)*, Estados Unidos, vol. 9, No. 2, pp. 203-222, Junho de 1999.

[12] M. Brunner e R. Stadler. The Impact of Active Networking Technology on Service Management in a Telecom Environment. In *Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, Boston, USA, 1999.

[13] M. Brunner e R. Stadler. Service Management in Multiparty Active Networks. *IEEE Communications Magazine*, pp. 144-151, Março de 2000.

[14] M. Brunner, B. Plattner e R. Stadler. Service Creation and Management in Active Telecom Networks. *Communications of the ACM*, pp. 55-61, Março de 2001.

[15] R. Kawamura e R. Stadler. Active Distributed Management for IP Networks. *IEEE Communications Magazine*, pp. 114-120, Abril de 2000.

[16] M. Breugst e S. Choy. Management of Mobile Agent Based Services. *Intelligence in Services and Networks (IS&N'99)*, Barcelona, Spain, Springer, pp. 143-154, Abril de 1999.

[17] Y. Yemini e S. da Silva. Towards Programmable Networks. *IFIP/IEEE International Workshop on Distributed Systems: Operations & Management - DSOM'96*, L'Aquila, Itália, Outubro de 1996.

[18] M. Breugst e T. Magedanz. Mobile Agents - Enabling Technology for Active Intelligent Network Implementation. *IEEE Network*, pp. 53-60, Maio/Junho de 1998.

[19] D. Raz e Y. Shavitt. Active Networks for Efficient Distributed Network Management. *IEEE Communications Magazine*, pp. 138-143, Março de 2000.

[20] G. Goldszmidt e Y. Yemini. Distributed Management by Delegation. *15th International Conference on Distributed Computing Systems*, Junho de 1995.

[21] A. T. Campbell et al. A Survey of Programmable Networks. *ACM SIGCOMM Computer Communication Review*, Vol. 29, No. 2, pp. 7-24, Abril de 1999.

[22] J. Biswas et al. The IEEE P1520 Standards Initiative for Programmable Network Interfaces. *IEEE Communications Magazine*, Vol. 35. No. 10, pp. 64-70, Outubro de 1998.

[23] M. F. Arnett et al. *Desvendando o TCP/IP*. Editora Campus, 1997.

[24] Future Active IP Networks. http://www.ist-fain.org.

[25] International Organization for Standardization. Information Processing Systems - Open Systems Interconnection - Basic Reference Model. *International Standard 7498*, 1984.

[26] Grasshopper Basics and Concepts. IKV++ GmbH.

[27] Grasshopper Programer's Guide. IKV++ GmbH.

[28] http://www.grasshopper.de.

[29] AN Architecture Working Group. Architectural Framework for Active Networks. *Calvert, K. (editor)*, Julho de 1999.

[30] Institut Informations- und Datenverarbeitung. *Monitoring of CORBA-based Applications*, Junho de 1997.

[31] The MAScOTTE Project: MAnagement Services for Object oriented disTributed sysTEms. Esprit project 20804. http://tes.iitb.fhg.de/corba-assistant.

[32] http://www.omg.org/cgi-bin/doc?formal/2000-01-02.

[33] Object Management Group, OMG Document: orbos/97-10-05. *Mobile Agent System Interoperability Facilities Specification*, Novembro de 1997.

[34] http://www.omg.org.

[35] Open Signaling Working Group. http://comet.columbia.edu/opensig.

[36] IEEE P1520. http://www.ieee-pin.org.

[37] K. Psounis. Active Networks: Applications, Security, Safety, and Architectures. *IEEE Communications Surveys*, pp. 2-16, janeiro-março de 1999.

[38] P. Ropelatto. Gerência de Monitorização de Sistemas Distribuídos em um Ambiente CORBA usando Agentes Móveis. Tese de Mestrado, Instituto de Computação - Universidade Estadual de Campinas - UNICAMP, Campinas - SP, junho de 1997.

[39] M. Sloman. Policy Driven Management for Distributed Systems. *Journal of Network and Systems Management*,Vol. 2, No. 4, pp. 333-360, 1994.

[40] http://java.sun.com.

[41] D. Wetherall. Developing Networks Protocols with the ANTS Toolkit. *Design Review*, Agosto de 1997.

[42] Y. Yemini. The OSI Network Management Model. *IEEE Communications Magazine*, pp. 20-29, Maio de 1993.

[43] http://www.cs.washington.edu/research/networking/ants/.

# Apêndice A

# Algumas Cápsulas

## A.1 Cápsula para solicitação de uma nova VAN e seus serviços

```
package mgmtsw;
import ants.*;

public class RequireNewVANCapsule extends DataCapsule {

    final private static byte[] MID = findMID("mgmtsw.RequireNewVANCapsule");

    protected byte[] mid() {
        return(MID);
    }

    final private static byte[] PID = findPID("mgmtsw.RequireNewVANCapsule");

    protected byte[] pid() {
        return(PID);
    }

    String[][] hostsServices;
    int tam,col,l=-1,c=0,k;
    protected String vnName;
    protected String fileName;
```

```
/*Calcula o tamanho da cápsula*/
public int length() {
    int size=0;
    for(int i=0;i<hostsServices.length;i++)
        for (k=0;k<hostsServices[i].length;k++)
            size = size + Xdr.STRING(hostsServices[i][k]);
    return(super.length() + (Xdr.INT * hostsServices.length ) +
        (Xdr.INT * k) + size + Xdr.STRING(vnName) + Xdr.STRING(fileName)
}

/*Usa a biblioteca XDR para serialização dos dados em um formato padrão*
public Xdr encode() {
    Xdr xdr = super.encode();
    xdr.PUT(vnName);
    xdr.PUT(hostsServices.length);
    xdr.PUT(fileName);
    for (int i=0;i<hostsServices.length;i++){
        xdr.PUT(hostsServices[i].length);
        for (int j=0;j<hostsServices[i].length;j++)
            xdr.PUT(hostsServices[i][j]);
    }
    return (xdr);
}

/*Desserializa os dados */
public Xdr decode() {
    Xdr xdr = super.decode();
    vnName = xdr.STRING();
    tam = xdr.INT();
    hostsServices = new String[tam][];
    fileName = xdr.STRING();
    for (int i=0;i<tam;i++){
        col = xdr.INT();
        hostsServices[i] = new String[col];
        for (int j=0;j<col;j++)
            hostsServices[i][j]=xdr.STRING();
    }
```

```
    return(xdr);
}

/*Retorna o nome da VAN dado pelo cliente*/
public String getName(){
    return vnName;
}

public void setVirtualNetName(String name){
    vnName = name;
}

/*Nome do arquivo de rotas para esta VAN*/
public void setFileName(String filename) {
    this.fileName = filename;
}

public String getFileName() {
    return fileName;
}

/*Adiciona os hosts que formarão a VAN*/
public void addHost(String host,int tam) {
    l++;
    c=0;
    hostsServices[l]=new String[tam];
    hostsServices[l][c] = host;
}

/*Adiciona os respectivos serviços para cada host*/
public void addService(String s) {
    c++;
    hostsServices[l][c] = s;
}

public String[][] getHostsServices() {
    return hostsServices;
}
```

```
    /*Método de avaliação da cápsula.  É invocado em cada nó ativo*/
    public boolean evaluate(Node n) {
        if (n.getAddress()==getDst()) {
            return(n.deliverToApp(this,dpt));/*Entrega a cápsula
                                              para a aplicação*/
        }else {
            return(n.routeForNode(this,getDst()));/*Nó roteador*/
        }
    }

    public RequireNewVANCapsule() { }

    public RequireNewVANCapsule(short sa, short da, int na,
                                ByteArray d, int lin) {
        super(sa,da,na,d);
        hostsServices = new String[lin][];
    }
}
```

## A.2 Cápsula para instalação de um novo serviço na VAN

```
package mgmtsw;

import ants.*;
import java.util.Hashtable;
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.net.*;
import java.rmi.server.*;

public class InstallServiceCapsule extends DataCapsule {

    protected String serviceName,agentName,className,
                     hostDestFormatoAnts,shortServiceName;
```

```
InetAddress hostAddress;
public boolean newService=false;/*Atributo que informa se a cápsula está
                                  adicionando um novo serviço à VAN ou está
                                  instalando um serviço durante o
                                  fornecimento da VAN*/
int aplicationAddr;

final private static byte[] MID = findMID("mgmtsw.InstallServiceCapsule")

protected byte[] mid() {
    return(MID);
}

final private static byte[] PID = findPID("mgmtsw.InstallServiceCapsule")

protected byte[] pid() {
    return(PID);
}

public int length() {
    return(super.length()+Xdr.STRING(serviceName) +
        Xdr.STRING(agentName) +
        Xdr.STRING(className)+Xdr.BOOLEAN+Xdr.INT+
        Xdr.STRING(shortServiceName)+Xdr.STRING(hostDestFormatoAnts));
}

public Xdr encode() {
    Xdr xdr = super.encode();
    xdr.PUT(serviceName);
    xdr.PUT(agentName);
    xdr.PUT(className);
    xdr.PUT(newService);
    xdr.PUT(aplicationAddr);
    xdr.PUT(shortServiceName);
    xdr.PUT(hostDestFormatoAnts);
    return (xdr);
}
```

```
public Xdr decode() {
    Xdr xdr = super.decode();
    serviceName = xdr.STRING();
    agentName = xdr.STRING();
    className = xdr.STRING();
    newService = xdr.BOOLEAN();
    aplicationAddr = xdr.INT();
    shortServiceName = xdr.STRING();
    hostDestFormatoAnts = xdr.STRING();
    return(xdr);
}

/*Atribui o nome do serviço a ser instalado*/
public void setNameService(String s){
    serviceName = s;
}

/*Atribui o nome da classe do serviço a ser instanciada no host*/
public void setClassName(String s) {
    className = s;
}

/*Atribui o nome do MAEA pelo qual o serviço será gerenciado*/
public void setAgentName(String agentName) {
    this.agentName = agentName;
}

/*Endereço da aplicação destino que está solicitando o serviço.*/
/*É usado somente no caso de ser um novo serviço.*/
public void setAddressOfAplicationDest(int address) {
    aplicationAddr = address;
}

public void setDestinoFormatoAnts(String dest) {
    hostDestFormatoAnts = dest;
}

public String getDestinoFormatoAnts() {
```

```
    return hostDestFormatoAnts;
}

/*Nome dado pelo fornecedor do serviço*/
public void setShortServiceName(String shortName) {
    shortServiceName = shortName;
}

public String getShortServiceName() {
    return shortServiceName;
}

public boolean evaluate(Node n) {
    if (n.getAddress()==getDst()) {
        try {
            hostAddress = InetAddress.getLocalHost();
        }catch (UnknownHostException e) {
            e.printStackTrace();
        }
        String hostIP = hostAddress.getHostAddress();
        String rmiurl = "rmi://"+hostIP+":2000/"+serviceName;
        try {
            Class serviceClass = Class.forName(className);
            /*Cria Serviço*/
            ManagementInterface s =
                    (ManagementInterface)serviceClass.newInstance();

            s.setAgentName(agentName);/*Seta o MAEA para este serviço*/

            /*Registra serviço no serviço de nomes local*/
            Naming.rebind(rmiurl,s);

            /*Coloca o serviço na cache do nó ativo*/
            n.getCache().put(serviceName,s,60000);
            BindInterface s1 =
            (BindInterface) Naming.lookup("rmi://araguaia:2000/register")
            s1.bind(rmiurl,serviceName);/*Registra serviço no GNS*/
            System.out.println("Servico Registrado!");
```

```
                if (newService) {/*Se for um serviço novo*/
                    /*O serviço deverá avisar o MAEA sobre sua chegada*/
                    s.avisarMAEA(serviceName);

                    /*Uma cápsula é criada para avisar a Aplicação Ativa do*/
                    /*cliente que o serviço solicitado chegou*/
                    /* e pode ser usado*/
                    ServiceMigrationCapsule c = new ServiceMigrationCapsule()
                    c.origem=2;
                    c.setShortServiceName(shortServiceName);
                    c.setDestinoFormatoAnts(hostDestFormatoAnts);
                    c.setDst(aplicationAddr);
                    c.prime(this);
                    return n.routeForNode(c,c.getDst());/*Envia cápsula p/ AA
                }
            }catch (Exception e) {
                e.printStackTrace();
            }
            return(true);
        }else
            return(n.routeForNode(this,getDst()));
    }

    public InstallServiceCapsule() { }

    public InstallServiceCapsule(short sa, short ds, int na, ByteArray da) {
        super (sa,ds,na,da);
    }
}
```

# Apêndice B

# Serviço de Nomes Global - GNS

```
package mgmtsw;

import java.util.Hashtable;
import javax.naming.directory.*;
import javax.naming.*;
import java.rmi.RemoteException;
import java.rmi.Naming;

public class GlobalNamingService extends java.rmi.server.UnicastRemoteObject
                implements BindInterface {

    private Hashtable env;
    private Context ctx;
    private ContextFactory ictx;

    /*Cria um contexto inicial e registra no serviço de*/
    /*nomes do RMI (rmiregistry)*/
    public GlobalNamingService( ) throws RemoteException {
        super();
        env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY,
            "com.sun.jndi.rmi.registry.RegistryContextFactory");
        env.put(Context.PROVIDER_URL, "rmi://localhost:2000");
        try {
            ctx = new InitialContext(env);
```

```
            try {
                ctx.rebind("rmi://localhost:2000/register", this);
                System.out.println("Registry OK!");
            } catch (Exception e) {
                System.out.println("Registry falhou:  "+ e.getMessage());
                e.printStackTrace();
            }
        }catch (NamingException e) {
            e.printStackTrace();
        }
    }

    /*Remove um serviço do contexto*/
    public void unbind (String nameService) throws RemoteException{

        try {
            ctx.unbind(nameService);
        }catch (NamingException e) {
            e.printStackTrace();
        }
        System.out.println("Servico removido do naming service");
    }

    /*Adiciona um serviço no contexto*/
    public void bind (String url, String nameService) throws RemoteException{

        Reference ref =
                    new Reference(nameService,new StringRefAddr("URL", url));
        try {
            ctx.rebind(nameService,ref);
        }catch (NamingException e) {
            e.printStackTrace();
        }
     System.out.println("Servico registrado no naming service");
    }
}
```

# Apêndice C

# Classe que define o protocolo utilizado

```
package mgmtsw;

import ants.*;

public class TelecomProtocol extends Protocol {

    public TelecomProtocol() throws Exception {

        startProtocolDefn();

        startGroupDefn();
        addCapsule(``mgmtsw.RequireNewVANCapsule'');
        addCapsule(``mgmtsw.InstallServiceCapsule'');
        addCapsule(``mgmtsw.ServiceUse'');
        addCapsule(``mgmtsw.ServiceMigrationCapsule'');
        addCapsule(``mgmtsw.GetNewServiceCapsule'');
        addCapsule(``mgmtsw.RemoveServiceCapsule'');
        addCapsule(``mgmtsw.MessageCapsule'');
        endGroupDefn();

        endProtocolDefn();
    }
}
```

# Apêndice D

# Outras Interfaces da MRA

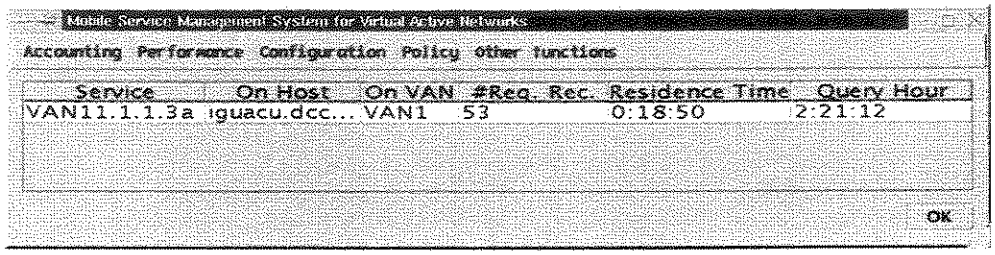



Figura D.1: MRA após a busca pelo serviço mais requisitado em toda a rede.

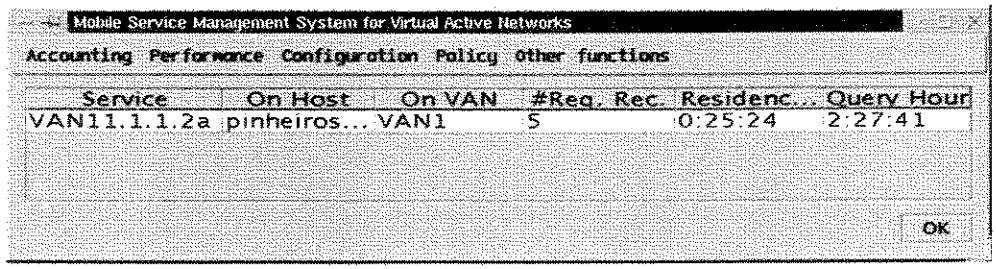



Figura D.2: MRA após a busca pelo serviço menos requisitado em um determinado host.

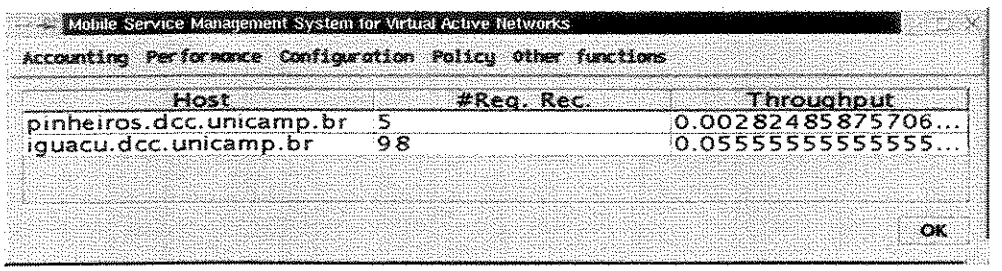



Figura D.3: MRA após a gerência para saber a quantidade de requisições recebidas em cada host.