Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa

Edmundo Lopes Cecilio

Dissertação de Mestrado

Luci Pirmez, D.Sc., COPPE/UFRJ, Brasil, 1996
Luiz Fernando Rust da Costa Carmo, Dr. UPS, França, 1995

Instituto de Matemática / Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Dezembro de 2002

Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa

Edmundo Lopes Cecilio

Dissertação submetida ao corpo docente do Núcleo de Computação e Eletrônica / Instituto de Matemática da Universidade Federal do Rio de Janeiro – UFRJ, como parte dos requisitos necessários à obtenção do grau de Mestre em Ciências em Informática.

Aprovada por:

_____

Profa. Luci Pirmez - Orientadora
D. Sc., COPPE/UFRJ

_____

Prof. Luiz Fernando Rust da Costa Carmo - Orientador
Dr., UPS, France

_____

Membro da banca

_____

Membro da banca

Rio de Janeiro – RJ

Dezembro de 2002

FICHA CATALOGRÁFICA

CECILIO, EDMUNDO LOPES.

Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa, [Rio de Janeiro], 2002.

xii, 93 p., 29,7 cm (IM/NCE/UFRJ, MSc., Informática, 2002)

Dissertação (Mestrado) - Universidade Federal do Rio de Janeiro, IM/NCE

1. Gerenciamento pró-ativo 2. Gerência de Redes 3. Gerenciamento Distribuído 4. Tecnologia Ativa

I. IM/NCE/UFRJ II. Título (série)

A João Alberto Neves e Ana Maria Moura, que motivaram e viabilizaram o meu retorno ao Mestrado

Aos meus amigos da PUC, Sérgio, Luiz Fernando, Guido, Rogério, Débora, Rodrigo, entre outros, por terem me iniciado neste caminho e por terem me dado a oportunidade de trilhar nele nos últimos cinco anos

Aos professores do Departamento de Engenharia de Sistemas do Instituto Militar de Engenharia que demonstraram ter infinita paciência com as ausências do seu coordenador de graduação

A todos os meus alunos que, por vezes, deixaram de ter o "algo mais" que sempre me caracterizou e que eles merecem

Aos meu amigos Ana Paula, Reinaldo, Renata, Flávia, Werner, Coelho, Alexandre, Noel, Roberta, Leonardo, que sempre estiveram à disposição em qualquer momento

Aos meus familiares, que toleraram pacificamente e com compreensão os meus afastamentos das funções de pai, marido e filho

Aos meus orientadores, Luci e Rust, que, sempre que foi preciso, tiveram paciência, mostraram o caminho para a realização desta dissertação e disponibilizaram os meios necessários

Resumo

CECILIO, Edmundo Lopes. **Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa**. Orientadores: Luci Pirmez e Luiz Fernando Rust da Costa Carmo. Rio de Janeiro. UFRJ/IM/NCE, 2002. Diss.

A garantia de níveis de qualidade de serviço específicos depende fundamentalmente da presença de um mecanismo de gerenciamento preciso, eficiente e rápido, não só dos recursos da rede como também dos serviços de comunicação (fim a fim), das estações e das aplicações. Esse mecanismo deve ser de fácil atualização e flexível o suficiente para a implantação de novas funcionalidades com a devida rapidez além de, preferencialmente, ser independente de plataforma. Outra característica desejável é que seja pró-ativo, de forma a permitir que sejam executadas ações com o objetivo de se tentar reverter quedas nos níveis de QoS ou, caso isso seja impossível, minimizar os prejuízos para os usuários. O gerenciamento de desempenho, em particular, é de vital importância, pois é quem deve monitorar a situação da rede no que diz respeito ao desempenho dos serviços que estão sendo prestados e indicar quando e porque os níveis desejados de QoS não estão sendo alcançados. A tecnologia ativa – redes ativas e agentes móveis – vem sendo considerada em recentes pesquisas para o provimento da flexibilidade e distribuição necessárias a próxima geração de futuros sistemas de gerenciamento. Esta dissertação apresenta uma proposta de arquitetura de gerenciamento de desempenho que, além de usar tecnologia ativa, é pró-ativa. O trabalho proposto é baseado na arquitetura de gerenciamento distribuída ativa AGAD, que foi desenvolvida no NCE/UFRJ. O protótipo que foi implementado tinha como objetivo investigar a viabilidade de se implementar um sistema de gerenciamento de desempenho pró-ativo em tempo real usando um paradigma de mobilidade de código baseado em Java, o μCode. Foram realizados três tipos de testes com o protótipo que disponibilizaram dados referentes ao desempenho do seu funcionamento. Esses dados permitiram que fossem feitas análises das vantagens e desvantagens da arquitetura que é proposta nesta dissertação.

Abstract

CECILIO, Edmundo Lopes. **A Pro-Active and Distribbuted Performance Management Architecture Using Active Technology**. Advisors: Luci Pirmez and Luiz Fernando Rust da Costa Carmo. Rio de Janeiro. UFRJ/IM/NCE, 2002. Diss.

Only a precise, efficient and rapid management mechanism will be able to guarantee the specific quality of service levels overseeing a network's resources, workstations and applications as well as its (end-to-end) communications services. This mechanism must have the capacity to be easily updated and yet be flexible enough to promptly assimilate the introduction of new functionalities, besides being, yet, platform independent. Another desirable characteristic is to be pro-active, to allow actions to be executed to try to reverse QoS falling or, if this turn to be impossible, to minimize user perception of the QoS failure. Performance Management is of vital importance as it is responsible for both monitoring network status as regards services being offered and indicating when and why desired QoS levels have not been achieved. Recent research has shown that Active Technology – active networks and mobile agents – offers the flexibility and distribution assets required for management systems of the future. This work proposes an architecture of Performance Management that, in addition to utilizing active technology, is also pro-active. This proposal is based on an Active, Distributed Management Architecture (AGAD) developed at the Núcleo de Computação Eletrônica (NCE) of the Federal University of Rio de Janeiro (UFRJ). The goal of the prototype was to investigate the viability of implementing a real time pro-active performance system running on a Java-based mobile agent paradigm μCode. Three kinds of tests were executed and data were collected about the performance of the prototype. These data allowed the advantages and disadvantages of the proposed architecture to be analyzed.

Lista de Siglas e Abreviaturas

AGAD	Arquitetura de Gerenciamento Ativa e Distribuída
ANEP	*Active Network Protocol*
ANTS	*Active Node Transfer System*
ATM	*Asynchronous Transfer Mode*
BCD	Base de Códigos
BCN	Base de Conhecimento
BGC	Base Geral de Códigos
BICD	Base de Informações de Componentes de Domínio
BIG	Base de Informações Gerenciais
BIT	Base de Informações Temporárias
BTpG	Base de Topologia Geral
CANES	*Composable Active Networks Elements*
CMIP	*Common Management Information Protocol*
CORBA	*Common Object Request Broker Architecure*
DAN	*Distributes code caching for Active Networks*
DARPA	*Defense Advanced Research Projects Agency*
DCOM	*Distibuted Component Object Mode*
GD	Gerente de Domínio
GDD	Gerente de Desempenho de Domínio
GDPA	Gerenciamento de Desempenho Pró-ativo
GM	Gerente Mor
ID	Inspetor de Desempenho
MIB	*Management Information Base*
MPEG	*Motion Picture Expert Group*
MPLS	*Multiprotocol Label Switching*
NCE	Núcleo de COmputação e Eletrônica
NMF	*Network Management Fórum*
NMS	*Network Management System*
OSI	*Open Systems Interconnection*
OSPF	*Open Shortest Path First*
QoP	*Quality of Presentation*
RIP	*Routing Information Protocol*
RMI	*Remote Method Invocation*
RMON	*Remote Monitoring*
RPC	*Remote Procedure Call*
RSVP	*Reservation Protocol*
RTCP	*Real Time Control Protocol*
RTP	*Real Time Protocol*
SLA	*Service Layer Agreement*
SNMP	*Simple Network Management Protocol*

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1 - Introdução

A abrangência da Internet ou, de forma mais genérica, das redes baseadas na arquitetura TCP/IP, aumentou significativamente. Essas redes deverão, em breve, assumir o papel de principal infra-estrutura de comunicação, que ainda é da rede telefônica convencional. A tecnologia utilizada na implementação dessas redes - hardware e protocolos – terão, no entanto, que evoluir, uma vez que essas redes, no futuro, deverão oferecer suporte a complexos serviços com requisitos de qualidade de serviço (QoS – *Quality of Service*[1]) diferentes e específicos para cada tipo de aplicação.

O desempenho das redes baseadas em TCP/IP, em sua configuração atual, é, em geral, aquém do desejado, em função do serviço prestado ser apenas de melhor esforço. Não há priorizações no encaminhamento de pacotes e nem há um mecanismo de controle de admissão de fluxos, além de haver redundâncias em algumas camadas de protocolo ao longo da rede, entre outros fatores.

O desempenho é diretamente dependente da presença de um sistema de gerenciamento, que é o responsável pela monitoração do estado da rede e pela comunicação de anormalidades.

Ainda que existentes e variados, os mecanismos de gerenciamento de desempenho que estão em uso atualmente, caracterizam-se por uma excessiva centralização no processamento, com grande troca de dados via rede, além de serem meramente reativos, ou seja, emitem notificações somente após a ocorrência de falhas.

Além do gerenciamento de desempenho insatisfatório, o que se tem, não somente na Internet, mas nas redes de comunicação atuais em geral, é a dificuldade na instalação de novos padrões, de novas tecnologias e de novos serviços. O ciclo de vida para a produção e implementação de novas funcionalidades é muito longo, podendo se estender por mais de uma década. A implementação das funcionalidades necessárias às novas redes vai requerer uma infra-estrutura mais flexível com a finalidade de se reduzir o prazo desse longo ciclo de vida.

A seguir, será apresentada uma visão geral de um mecanismo de gerenciamento de desempenho que vem ao encontro dessas necessidades que foram citadas nos parágrafos anteriores. Em seguida, serão apresentados os objetivos e a organização dessa dissertação.

---

[1] O termo QoS denomina o efeito coletivo provocado pelo conjunto das características quantitativas e qualitativas de um serviço oferecido por um sistema que é necessário para alcançar a funcionalidade requisitada pelas aplicações [16].

1.1. Gerenciamento de desempenho pró-ativo e tecnologia ativa

A garantia de níveis de serviço específicos depende fundamentalmente da presença de um mecanismo de gerenciamento preciso, eficiente e rápido. O gerenciamento de desempenho, em particular, é de vital importância, pois é ele quem deve monitorar a situação da rede no que diz respeito ao desempenho dos serviços que estão sendo prestados e indicar quando e porque os níveis desejados de qualidade não estão sendo alcançados. É desejável ainda que o gerenciamento, particularmente de desempenho, seja pró-ativo, de forma que ações sejam desencadeadas antecipadamente, com o objetivo de se tentar evitar ou, pelo menos, minimizar as conseqüências das falhas ou degradações que venham a ocorrer nos serviços que estão sendo prestados aos usuários.

O gerenciamento não deve se limitar apenas à rede. As próprias estações e as aplicações que estão nas extremidades de um serviço de comunicações, devem ser gerenciadas. Um desempenho deficiente nessas estações pode ser responsável pela degradação da QoS. O mesmo pode ser dito em relação aos serviços (fim a fim) prestados pelas redes e às aplicações. O gerenciamento deve estender-se também a eles.

O cenário de uma aplicação multimídia distribuída pode ser utilizado como exemplo para a ilustração do que é esperado de um sistema de gerenciamento de desempenho pró-ativo que se estenda até além dos domínios da rede. A adaptabilidade das aplicações em relação à QoS disponível é uma funcionalidade que vem sendo pesquisada como uma forma de se manter a qualidade de percepção (QoP[2]) do usuário na apresentação de um documento multimídia [27][28], ainda que os níveis de QoS venham a ser diminuídos. A interação entre o sistema de gerenciamento de desempenho e este tipo de aplicação propicia a possibilidade de que as adaptações sejam adiantadas, minimizando o tempo de interrupção da apresentação do documento multimídia para o usuário, no caso da ocorrência de uma falha de QoS. Esse adiantamento da adaptação pode ser realizado tanto de forma reativa, ou seja, após a ocorrência de uma falha de QoS, quanto de forma pró-ativa. Nesse último caso, o objetivo da adaptação seria reduzir as exigências de QoS do serviço de comunicações quando da detecção – por um sistema de gerenciamento de desempenho pró-ativo – de uma tendência de que venha a ocorrer uma degradação da QoS. Essas adaptações de caráter pró-ativo podem até mesmo evitar que a falha de QoS venha, de fato, a ocorrer.

---

[2] A QoP corresponde a uma medida que engloba não apenas a satisfação do usuário com respeito a um serviço prestado por um sistema multimídia distribuído, mas também, à potencialidade do usuário perceber, sintetizar e analisar os conteúdos informacionais de uma apresentação [26].

O mecanismo de gerenciamento de desempenho, a exemplo do que ocorre com as próprias redes, deve ser também de fácil atualização e flexível o suficiente para a implementação de novas funcionalidades com a devida rapidez. De preferência, esse mecanismo deve ser também independente de plataforma. A tecnologia ativa, que compreende o uso das redes ativas e dos agentes móveis tem se mostrado como a tecnologia mais adequada ao provimento desta flexibilidade, uma vez que permitem o carregamento, remoto, de programas nos diversos elementos da rede e nas estações. Além da flexibilidade disponibilizada pelo uso dessa tecnologia, ela também viabiliza a presença de capacidades locais de processamento nos elementos gerenciados, fato que pode tornar mais dinâmicas as ações oriundas no gerenciamento e reduzir o tráfego na rede.

1.2. Objetivos e organização da dissertação

O objetivo dessa dissertação é apresentar uma proposta de arquitetura de gerenciamento de desempenho pró-ativa e distribuída que se utilize da tecnologia ativa[3]. O gerenciamento de desempenho proposto não é limitado apenas à rede. Ele se estende não somente aos serviços de comunicação que são prestados pela rede e mas também às aplicações e às estações.

A arquitetura proposta é baseada na Arquitetura de Gerenciamento Ativa Distribuída (AGAD) [19] que está em desenvolvimento no NCE da UFRJ.

Foi implementado um protótipo da arquitetura que teve como objetivo a investigação da viabilidade, das vantagens e das desvantagens do gerenciamento pró-ativo distribuído baseado em tecnologia ativa, usando, em particular, no caso do protótipo, os agentes móveis.

Os testes que foram realizados com o protótipo estão relacionados a uma aplicação multimídia distribuída que está em desenvolvimento no NCE, o ServiMídia [27][28].

1.3. Organização da dissertação

Além dessa introdução, a dissertação está organizada em mais cinco capítulos. O capítulo 2 descreve os conceitos básicos sobre os quais a proposta de arquitetura está baseada, como gerenciamento centralizado x distribuído, gerenciamento de desempenho, redes ativas e agentes móveis e o uso de agentes móveis e redes ativas no gerenciamento distribuído. O capítulo 3 apresenta a proposta de arquitetura de gerenciamento pró-ativo distribuída que é o foco deste trabalho. São detalhados os elementos da arquitetura e o funcionamento da mesma é explicado. São também descritos os possíveis cenários onde a utilização de um sistema de gerenciamento de desempenho pró-ativo seria vantajosa. Os trabalhos relacionados são discutidos ao final deste capítulo. O

[3] Tecnologia ativa engloba, nesse trabalho, conforme será explicado posteriormente, as tecnologias de redes ativas e agentes móveis.

capítulo seguinte descreve o ambiente de desenvolvimento que foi utilizado e apresenta, de forma detalhada, o protótipo que foi implementado. O capítulo 5 apresenta os testes que foram realizados com o protótipo, os seus resultados, e uma análise desses resultados, discorrendo sobre as vantagens e as desvantagens da arquitetura proposta que puderam ser deduzidos a partir desses resultados. O último capítulo apresenta as conclusões da dissertação e sugere trabalhos futuros decorrentes do que foi proposto.

Capítulo 2 - Conceitos básicos

Nesse capítulo serão apresentados os conceitos básicos que são necessários para o correto entendimento da arquitetura de gerenciamento de desempenho pró-ativo distribuído usando tecnologia ativa que está sendo proposta nesta dissertação.

A primeira seção compara as abordagens de gerenciamento centralizado e distribuído, apresentando as vantagens do segundo sobre o primeiro. As principais formas de implementação do gerenciamento distribuído são resumidas. A seção 2.2 descreve, já inserida no contexto do gerenciamento distribuído, a área funcional de gerenciamento de desempenho. A importância do gerenciamento de desempenho para o transporte de tráfegos multimídia é destacada. Os requisitos, a abrangência, os problemas e os parâmetros a serem monitorados e controlados por um sistema de gerenciamento de desempenho são também descritos. O paradigma de gerenciamento pró-ativo e as vantagens de sua utilização são apresentados na seção 2.3. A seção seguinte, Tecnologia ativa, apresenta as características dos paradigmas de redes ativas e de agentes móveis. As vantagens e desvantagens dessas abordagens são também discutidas. A seção 2.5 trata da integração de tecnologia ativa no gerenciamento distribuído de redes, em particular, no gerenciamento de desempenho. Por fim, na última seção, um resumo do capítulo é apresentado.

2.1. Gerenciamento de redes centralizado versus *distribuído*

A abordagem clássica de gerenciamento – centralizada – que é utilizada, por exemplo, nas implementações iniciais das arquiteturas SNMP [51] e CMIP [37][4] é baseada em um paradigma cliente-servidor, onde este último, instalado em um elemento da rede, na forma de um agente, sofre pollings freqüentes para que envie dados a uma estação de gerenciamento (a NMS – *Network Management Station*), que nesse caso faz o papel de cliente. A NMS constrói então uma visão do estado da rede e gera Alarmes quando problemas são detectados. Normalmente, a intervenção do operador ou administrador da rede é requerida. A NMS pode também enviar comandos aos agentes para que estes desencadeiem ações visando resolver esses problemas. A funcionalidade dessas ações, no entanto, é limitada ou demorada. Situações de exceção podem ser configuradas nos agentes para que estes enviem, por iniciativa própria, avisos à NMS como, por exemplo, as *traps*.

Essa abordagem gera um grande tráfego na rede, que, na maioria das vezes, após ser analisado, serve apenas para deduzir-se que a situação é normal. Esse grande tráfego não só pode prejudicar o desempenho da rede como também pode fazer com que o tempo decorrido entre a detecção de uma situação anormal e a intervenção apropriada para resolvê-la seja muito grande, podendo torná-la até mesmo inútil. À medida que o número de elementos gerenciados cresce, a comple-

xidade, os requisitos de poder de computação da estação de gerenciamento e de consumo de banda da rede gerenciada também aumentam. Ainda, em redes com grande abrangência geográfica, a NMS pode estar distante dos elementos gerenciados e, conseqüentemente, os retardos na comunicação para o envio de dados e comandos podem ser excessivamente grandes.

Abordagens alternativas – distribuídas – para o gerenciamento vêm sendo pesquisada nos últimos anos. Desde o trabalho de Yemini [17], sobre gerenciamento distribuído por delegação, até o uso de agentes móveis [10], diversas são as opções sendo pesquisadas ou já em uso comercial para a implementação da distribuição no gerenciamento.

As abordagens mais simplificadas dividem a rede em domínios hierárquicos de gerenciamento, cada qual com sua NMS. Atividades de monitoramento e certas capacitadas de filtragem são delegadas aos próprios elementos gerenciados, que enviam apenas Alarmes ou relatórios resumidos às NMS. Exemplos dessas abordagens são as funções de monitoramento de métricas [29] e sumarização [54] do modelo OSI de gerenciamento e RMON no modelo SNMP [51]. Essas abordagens, no entanto, apesar de reduzirem o tráfego na rede e o tempo entre a detecção de uma anomalia e a emissão do correspondente alarme ou ação, não oferecem flexibilidades para atualização ou troca de softwares de forma rápida.

Opções mais avançadas de gerenciamento distribuído utilizam-se de paradigmas de objetos distribuídos como CORBA [31]e Java-RMI [32] ou da chamada remota de procedimentos (RPC) [33]. O uso desse paradigma, embora vantajoso sob os pontos de vista de projeto e de reuso, abstrai-se dos detalhes de implementação. Exemplos como CORBA, DCOM e RMI são muito úteis no projeto e construção de sistemas distribuídos, mas escondem os verdadeiros custos de suas implementações. O tamanho e a complexidade da infra-estrutura que tem que estar instalada na plataforma normalmente são grandes. Como resultado, o uso dos recursos da rede pode se dar de forma ineficiente, principalmente no que diz respeito ao consumo de capacidade de transmissão dos enlaces, dada a (possível) grande quantidade de dados a ser trocada entre os objetos. Outra desvantagem dessas abordagens é que não oferecem o suporte a ambientes verdadeiramente distribuídos, onde agentes (softwares) podem se comunicar com vizinhos para se encarregarem, em conjunto, de executar tarefas de forma distribuída e eficiente.

As pesquisas mais avançadas para a distribuição do gerenciamento usam o que é chamado de tecnologia ativa [10]. Essa tecnologia integra as pesquisas atuais em redes ativas, redes programáveis e em agentes móveis, que serão descritas na próxima seção. De forma resumida, pode-se dizer que programas podem ser instalados em nós de comutação da rede que oferecem suporte a execução de programas ou em estações. Esses programas são dotados de certa capacidade de

processamento, sendo capazes de agir localmente com o objetivo de resolver problemas ou consolidar informações de gerenciamento. Dessa forma, além de reduzirem o tráfego na rede e a demora para o desencadeamento de ações de gerenciamento, a atualização desses programas pode ser realizada de forma flexível e rápida. Essa abordagem não deve ser confundida com aquelas que pregam a distribuição de objetos. Com o uso de tecnologia ativa, programas inteiros, autônomos, e não apenas procedimentos ou objetos são instalados nos elementos a serem gerenciados, além de poderem migrar.

2.2. Gerenciamento de desempenho

O conjunto de serviços disponíveis via Internet é e será cada vez mais variado. Durante muitos anos eles resumiam-se a aplicações do tipo Telnet, FTP, Usenet e e-mail. Esse conjunto vem sendo aumentado com aplicações como WWW, transações financeiras, educação à distância, transmissão de fluxos de áudio e vídeo via multicast isoladamente ou sincronizados, com interatividade em tempo real ou não, computação distribuída e acesso a bancos de dados distribuídos.

Estudos do desempenho[4] da Internet em sua configuração atual mostram [3], no entanto, que há muitos usuários insatisfeitos. O serviço de entrega de pacotes baseado apenas no melhor esforço faz com que ainda seja impossível garantir os níveis desejáveis de QoS [3]. A tecnologia da futura Internet terá que transportar tráfegos que poderão ser agrupados em diferentes classes de serviço[5]. O tráfego de cada classe terá suas características e seus requisitos de QoS específicos.

A garantia de QoS está diretamente relacionada ao gerenciamento de desempenho, que é uma das cinco áreas funcionais de gerenciamento segundo classificação do Fórum de Gerenciamento de redes (NMF – *Network Management Forum*) [2]. O correto gerenciamento do desempenho viabiliza ao administrador ou operador da rede gerenciá-la de forma a prover níveis diferenciados de QoS ou, pelo menos, descobrir que não está sendo possível atingi-los. O gerenciamento de desempenho também trás vantagem financeira, uma vez que, quanto maior for a capacidade de medir, analisar e garantir certos níveis de QoS; acordos de níveis de serviço (SLA – *Service Level Agreement*) mais estritos poderão ser negociados, gerando maior receita.

O gerenciamento de desempenho visa atingir dois objetivos conflitantes, a garantia de QoS para os usuários (aplicações) e a obtenção de um alto grau de multiplexação no uso da rede [1]. Ele deve prover funções para medir, monitorar, avaliar e gerar informações sobre os níveis de de-

---

[4] Segundo Ferreira [15], desempenho é o conjunto de características ou de possibilidades de atuação de um sistema.

[5] Classes de serviço implicam que serviços podem ser categorizados em classes separadas, que podem ser gerenciadas individualmente .

sempenho alcançados pela rede. Essas informações devem ser utilizadas principalmente com duas finalidades. A primeira é a geração de estatísticas periódicas que venham a ajudar no planejamento de capacidade da rede, enquanto que a segunda é indicar a ocorrência de desrespeitos aos níveis de QoS desejados. Nesse último caso, controles dos recursos da rede e/ou das aplicações devem ser acionados com o objetivo de melhorar a QoS que está sendo prestada aos usuários [2]. Exemplos desses controles podem ser alterações de roteamento, realocação de *buffers* e a adaptação dos tráfegos, entre outras.

O provimento de serviços com QoS consistente e confiável tem sido associado a mecanismos como escalonamento com reserva de certa capacidade de processamento, classificação de tráfego em classes de serviço e sinalização para a reserva tanto de banda quanto de *buffers*, entre outros. No entanto, mais do que isso é necessário. Mecanismos de gerenciamento de desempenho que ajudem na seleção de rotas através da escolha de enlaces de acordo com as suas capacidades e da obtenção de métricas precisas de roteamento podem fazer mais pela obtenção do desempenho esperado pelo usuário do que complexos mecanismos como Weighted Fair Queueing, RSVP e tarifação diferenciada. Mesmo que esses complexos mecanismos tornem-se disponíveis em caráter geral, ainda serão necessárias ferramentas para a administração e gerenciamento do desempenho da rede através do ajuste de parâmetros como agregação e pesos de cada classe de serviço [3]. Isso requer um conhecimento praticamente instantâneo da situação dos enlaces da rede.

O gerenciamento de desempenho não deve se limitar apenas à rede. Os sistemas de computação ligados a ela também devem ter seu desempenho gerenciado com a finalidade de se identificar degradações que sejam neles originadas. Um servidor com pouca capacidade de processamento para a tarefa que deve executar, por exemplo, pode impedir que certo nível de QoS seja atingido.

Os principais requisitos para a execução do gerenciamento de desempenho propriamente dito são, então, aumentar o mínimo possível o tráfego na rede e disponibilizar informações sobre o desrespeito aos níveis de serviço desejados o mais rapidamente possível. As aplicações de gerenciamento atuais concentram-se basicamente na divulgação do estado da rede em vez de localizar e desencadear ações para que problemas de desempenho sejam resolvidos. A próxima geração de aplicações de gerenciamento de desempenho deverá tratar de aspectos como descobrimento automático de topologia, monitoração (via coleta e análise de dados), simulações rápidas, planejamento de capacidade, configuração de roteadores e comutadores e a realização de diagnóstico de falhas [3].

Os desafios na implementação de gerenciamento de desempenho são integração, escalabilidade e flexibilidade. A integração possui três componentes: a necessidade de se coletar dados de diferentes origens, analisá-los e alimentar com os dados obtidos as aplicações de gerenciamento do proprietário da rede. É necessário trabalhar por toda a rede e em suas bordas em diferentes camadas de diferentes fornecedores e realizar a correlação das informações colhidas. Na escalabilidade há duas dimensões a serem consideradas, o número de usuários e o tamanho (físico) da rede. Quanto maiores forem, maior será o tráfego gerado pelo gerenciamento de desempenho e maiores serão os retardos. A flexibilidade significa que os elementos gerenciados e as aplicações que tratam do gerenciamento devem ser configuráveis e adaptáveis para que possam prestar diferentes serviços e atender a diferentes objetivos de desempenho.

Pode-se dizer que o trabalho do gerenciamento de desempenho resume-se, então, na monitoração e no controle, mediante ações, de parâmetros de QoS. Estes são expressos em unidades diferentes nas diferentes camadas de processamento. Aplicações têm parâmetros em função de seus objetivos, como por exemplo, um número de quadros de imagem por segundo, enquanto recursos ou elementos de rede definem parâmetros de acordo com a sua natureza, como, por exemplo, para um enlace de comunicações, uma taxa em bits por segundo. Chatterjee [12] propõe uma taxonomia para os parâmetros de QoS.

2.3. Gerenciamento pró-ativo

A quase totalidade das abordagens de gerenciamento é reativa, ou seja, trata da coleta de dados sobre o funcionamento da rede e dos serviços e, no máximo, da indicação da ocorrência de problemas. Tem sido pesquisada, ultimamente, a abordagem pró-ativa[6] de gerenciamento [23][24]. O gerenciamento de redes pró-ativo tem por objetivo principal identificar problemas na rede antes que alguma degradação ou falha de QoS nos serviços que estão sendo prestados venha a ocorrer. Algumas áreas funcionais, como configuração e contabilização, são inerentemente reativas. Já as áreas de gerenciamento de desempenho, de falhas e de segurança podem ser mais eficientes caso o gerenciamento seja realizado de forma pró-ativa. A Figura 1 apresenta uma visão esquemática do relacionamento entre as áreas funcionais, o estado da rede e o tipo de gerenciamento predominante para cada área.

---

[6] "Pró": a favor. "Ativo": que exerce ação, que age, funciona, trabalha, se move, está apto a agir, com rapidez, prontidão, é atuante, enérgico.

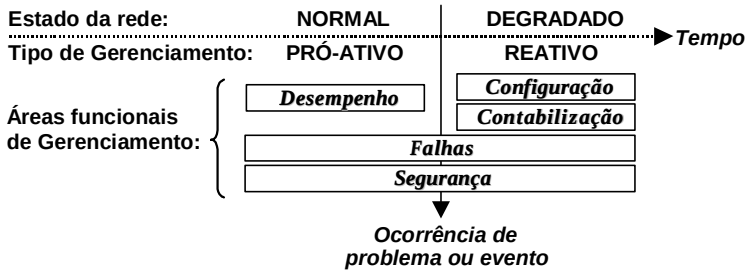


Figura 1 - Gerenciamento pró-ativo x gerenciamento reativo

No gerenciamento pró-ativo, o comportamento da rede deve ser constantemente monitorado para que estatísticas sejam coletadas. Análises dessas estatísticas permitem que sejam identificados sintomas indicadores de que um ou mais problemas podem vir a acontecer. Podem ser utilizadas, nessas análises, simulações rápidas, extrapolações ou técnicas de inteligência artificial, com o objetivo de se precisar ou estimar o momento quando determinado problema ou evento vai ocorrer [24]. As operações que têm que ser realizadas nessas análises podem vir a consumir grande processamento. Esse tem sido o fator limitante do emprego de técnicas pró-ativas no gerenciamento de redes. A distribuição do processamento de gerenciamento, juntamente com a flexibilidade oferecida pelo uso de tecnologia ativa, aliados ao aumento da capacidade de processamento disponível no hardware podem viabilizar o gerenciamento pró-ativo. Uma possível solução é a colocação de uma capacidade de processamento (destinada ao gerenciamento) limitada nos elementos a serem gerenciados e uma maior capacidade de processamento em estações destinadas a esse fim, próximas àqueles elementos.

2.4. Tecnologia ativa

A partir de 1995, após as primeiras publicações originadas em pesquisa financiada pela DARPA, a distribuição do controle e a programabilidade dos nós de uma rede passaram a ser investigadas com grande seriedade com o objetivo de se disponibilizar uma maior flexibilidade tanto no gerenciamento quanto na disponibilização rápida de novos serviços em redes de comunicação [18].

Três são os paradigmas sendo considerados para a implementação dessa distribuição: redes ativas, redes programáveis e agentes móveis. Os três paradigmas oferecem o suporte a modelos que utilizam recursos computacionais no interior e/ou nas bordas da rede para a execução de software “sob demanda”. Desta forma, novos tipos de aplicações de controle e de gerenciamento podem ser implementadas. Esses paradigmas têm sido postos à prova, na sua maioria, implementados via CORBA ou Java. Embora os conceitos desses três paradigmas tenham sido originados em diferentes comunidades de pesquisas visando resolver diferentes problemas, eles co-

meçam a se superpor em termos de foco e aplicabilidade, fato que está popularizando o termo *tecnologia ativa* para as referências a um ou mais desses paradigmas em conjunto [10].

2.4.1. Redes ativas

Uma rede ativa é uma rede onde os nós intermediários (roteadores, comutadores, gateways, etc), que são chamados de nós ativos, em vez de realizarem apenas a entrega de dados, podem realizar computações, normalmente, mas não apenas, em prol da entrega de dados. Os fluxos de dados podem então transportar programas (códigos) e dados. Nem todos os nós da rede precisam ser ativos. Quando esse for o caso, deve haver o tunelamento entre dois nós ativos adjacentes, a exemplo do que ocorre no MBone. Cada nó ativo é composto por um hardware, por um sistema operacional, por um ou mais ambientes de execução e por aplicações ativas, conforme pode ser observado na Figura 2.

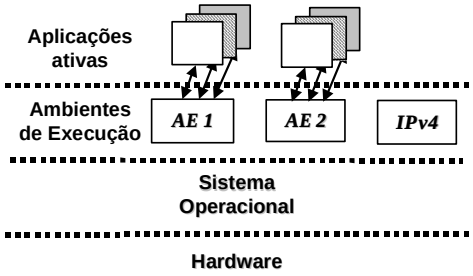


Figura 2 - Arquitetura de um nó de comutação ativo

O sistema operacional é o responsável pela alocação e escalonamento dos recursos do nó, como, por exemplo, ciclos de CPU e capacidades de memória e armazenamento. Os ambientes de execução implementam máquinas virtuais que executam as aplicações (ativas) que, por sua vez, tratam os fluxos de pacotes. A capacidade de processamento dos ambientes de execução pode variar entre completamente genérica, ou seja, capaz de executar qualquer programa, até especializada em uma única tarefa, como, por exemplo, encaminhar pacotes baseado em cabeçalhos do protocolo IPv4. Os usuários obtêm serviços fim a fim de uma rede ativa através de aplicações ativas que são executadas nas máquinas virtuais (uma em cada nó ativo da rota) providas pelos ambientes de execução.

Duas abordagens de implementação de redes ativas estão sendo pesquisadas: integrada e discreta. Na abordagem integrada, os códigos que implementam as aplicações ativas são encapsulados nos próprios pacotes do fluxo que devem ser por elas processados. Nesse caso, a aplicação ativa é instalada imediatamente, na hora de sua utilização. Os pacotes ativos referentes a essa abordagem são chamados de cápsulas. Exemplos de projetos que utilizam essa abordagem são IP Option, ANEP [35] e Smart Packets [13]. Na abordagem discreta, os programas não são integrados aos dados. Os pacotes ativos transportam apenas identificadores das aplicações ativas (ou rotinas

das mesmas) que devem neles ser executadas. Nesse caso, as aplicações ativas são instaladas por um esquema de transferência de código sob demanda a partir de servidores ou mesmo de outros nós ativos[7]. Essa abordagem, embora trate os nós da rede como "comutadores programáveis", não pode ser confundida com a tecnologia de "redes programáveis", onde a programação do nó é realizada via operações de gerenciamento, por um administrador, conforme será visto na próxima seção. Exemplos de projetos que se utilizam da abordagem discreta são ANTS, CANES e DAN [18].

Alguns projetos como SwitchWare e NetScript empregam uma combinação das duas abordagens [11].

O paradigma das redes ativas oferece grandes vantagens, como eficiência no uso da capacidade de transmissão da rede, rápida instalação de novos protocolos e a também rápida disponibilização de flexibilidades como, por exemplo, a atualização de um determinado processamento em um nó de comutação. Assim, essas redes podem acompanhar a rápida evolução das tecnologias de comunicação de dados mediante a introdução de novos serviços sem que os longos processos de padronização de protocolos e de formato de pacote e desenvolvimento de plataformas específicas (hardware e software) tenham que ser esperados.

A tecnologia de redes ativas possui, em contrapartida, algumas desvantagens. A primeira delas é a questão da segurança [11]. Esquemas de autenticação têm que ser implementados de forma a impedir que códigos não autorizados sejam colocados em funcionamento em nós ativos. A proteção de um nó ativo contra erros em tempo de execução também é um fator a ser considerado [11]. Dependendo das funcionalidades que forem disponibilizadas para as linguagens a serem utilizadas para a implementação dos códigos, o correto funcionamento de um nó ativo pode ser comprometido por problemas como entrada em laço infinito, bloqueio indevido de processo ou consumo excessivo de memória. Outra desvantagem significativa, que pode vir a ser minimizada com a constante evolução da capacidade de processamento do hardware computacional, é a queda no desempenho inerente ao uso de códigos interpretados ou mesmo parcialmente interpretados. A exigência de autenticações para a execução de códigos também contribui para a queda no desempenho das aplicações ativas de um nó ativo.

2.4.2. Redes programáveis

A tecnologia de redes programáveis prevê a disponibilização de interfaces abertas de programação nos elementos da rede. Com o uso dessas interfaces, a rede pode ser considerada um grande

[7] Normalmente, o código é obtido do nó ativo anterior da rota do fluxo em questão.

computador, que pode ser programado para que novos serviços, sinalizações e controles sejam rapidamente implementados [39].

A programação que é prevista nessa tecnologia não é dinâmica como aquela que está disponível na tecnologia de redes ativas. Aqui a programação é desencadeada através de operações de gerenciamento, realizadas por um administrador ou operador, e não pelos próprios pacotes de um fluxo.

Essa tecnologia, no que diz respeito ao gerenciamento, pode oferecer vantagens apenas para a atualização de NMS ou de agentes de gerenciamento fixos. O dinamismo do gerenciamento, no sentido de que são necessárias capacidades diversas de processamento em momentos e locais diversos, requer mais do que apenas interfaces abertas de programação.

2.4.3. Agentes móveis

Tendências atuais deixam claro que a complexidade do software continuará a crescer dramaticamente nas próximas décadas. A natureza cada vez mais dinâmica e distribuída das aplicações e de seus dados requerem que os sistemas de aplicação não somente respondam às requisições por informações mas que também, de forma inteligente, comuniquem-se, antecipem, adaptem e busquem formas de ajudar os usuários. Esses sistemas devem não só auxiliar a coordenação entre humanos, mas também devem cooperar entre si. Em resposta a esses requisitos, os esforços de pesquisadores de diversas áreas, como inteligência artificial, robótica e computação distribuída, entre outras, começam a convergir para o desenvolvimento de agentes de software [7].

A abordagem para comunicação em ambientes de software mais utilizada na prática ainda é a chamada remota de procedimentos (RPC – *Remote Procedure Call*), que foi concebida nos anos 70. Cada mensagem transferida pela rede inclui os argumentos para a execução do procedimento ou resultados da execução do mesmo. Nesse caso, os computadores que se comunicam combinam previamente quais são os procedimentos que podem ser chamados e quais são as mensagens que devem ser trocadas. Os procedimentos propriamente ditos são internos ao sistema onde executam e fixos. A interação via RPC requer comunicação constante entre cliente e servidor. A execução do procedimento é dita síncrona, pois só acontece por estímulo do software que o chamou remotamente. O tráfego gerado é grande e a atualização dos procedimentos é difícil, por estes serem fixos, requerendo novas instalações manuais e troca de componentes. Além de RPC, outras tecnologias para comunicação em ambientes de software, ainda que também baseadas no paradigma cliente-servidor, são CORBA e Java-RMI.

Uma alternativa para a chamada remota de procedimentos é a programação remota. Nesse caso, deve ser combinada previamente entre os sistemas de computação apenas a linguagem que será

utilizada. Um agente de software é então enviado ao sistema que deve servir como servidor em algum processamento. Essa abordagem dispensa a interação constante entre cliente e servidor.

Outra abordagem é o uso de agentes. Um agente é uma entidade computacional que age ou tem o poder e a autoridade de agir ou representar outra entidade. Em particular, espera-se que um agente de software execute uma tarefa específica para a qual tenha sido designado. Para isso, ele atua de forma contínua e autônoma, tanto pró-ativa quanto reativa, tem capacidade de aprender e cooperar e dispensa constante intervenção ou orientação, em um ambiente particular, possivelmente habitado por outros agentes e processos [5] [7]. A execução do agente é dita assíncrona, uma vez que não depende exclusivamente de estímulos de outro software, como no caso da RPC.

Uma grande vantagem quantitativa e tática dessa abordagem é o seu melhor desempenho, pois dados podem ser analisados e ações podem ser desencadeadas localmente. Quanto menor for a capacidade de transmissão de dados da rede e menor for a sua disponibilidade, maior será a vantagem. Isto é particularmente útil quando da presença de redes de acesso com baixas taxas e/ou sem conexões permanentes. Outra grande vantagem, qualitativa e estratégica, é a possibilidade de configuração e adaptação. A manutenção também é facilitada, pois o agente pode ser sempre o mais atualizado possível, uma vez que o mesmo é enviado somente quando necessário.

Muito da pesquisa a respeito dos aspectos de inteligência de agentes de software advêm da inteligência artificial (IA) distribuída. Esta é uma extensão de idéias derivadas da IA que se aplicam a sistemas multi-agentes. Em vez de uma única – muito grande e complexa – aplicação que codifica toda a inteligência de um sistema, um certo número de agentes são envolvidos em um esforço colaborativo para a resolução de problemas. Isto não implica que o sistema grande seja meramente dividido em partes menores. Agentes que tratam de problemas específicos, interligados por um sistema de comunicação, podem trocar pontos de vista e colaborarem para a produção de estratégias para a resolução de problemas. Esse paradigma é chamado de resolução distribuída de problemas. A abordagem baseada em agentes difere da abordagem cliente-servidor no sentido de que não há uma clara distinção em quem é o cliente e quem é o servidor. Todos os agentes participam da computação de acordo com os papéis para eles designados [5].

Agentes móveis são caracterizados pela sua habilidade de moverem-se entre diferentes localidades via redes locais (LAN) ou geograficamente distribuídas (WAN). A tecnologia de agentes móveis oferece também o suporte ao encapsulamento de protocolos o que é bom para a implementação de redes ativas.

Os agentes móveis viabilizam a transformação das redes atuais em plataformas programáveis remotamente. As principais vantagens no uso de agentes móveis em comparação com a abordagem cliente-servidor podem assim ser resumidas: [5] eficiência, economia de espaço, redução do tráfego, interação assíncrona e em tempo real, robustez, operação em ambientes heterogêneos, extensibilidade em tempo real e fácil atualização.

Para se fazer uso de agentes móveis, um sistema de comunicação, bem como os sistemas de computação, têm que incorporar um framework de mobilidade. Este vai prover as facilidades que vão oferecer o suporte aos modelos de ciclo de vida, computacional, de segurança, de comunicação e de navegação dos agentes [5].

Em contraste, a atualização de um agente local é um processo bem mais longo [5] do que o envio de dados coletados ou um resumo dos mesmos. Há também que se considerar o impacto a ser causado no desempenho da rede pela transferência dos próprios agentes móveis. O desempenho do gerenciamento de redes baseado em agentes móveis é, de maneira geral, superior àquele do SNMP, exceto em casos onde o número de elementos gerenciados é pequeno ou quando o tamanho do agente, que é relacionado a sua funcionalidade, for grande. Maiores detalhes sobre essa comparação podem ser obtidos em [34]. Adicionalmente, agentes não controlados podem inundar a rede e tomar parte significativa dos recursos. Sob o ponto de vista da segurança, nem todos, humanos ou aplicações, devem ser capazes de poder injetar agentes na rede.

2.4.4. Agentes móveis *versus* Redes ativas

A tecnologia de redes ativas é mais genérica do que a tecnologia de agentes móveis em termos de encapsulamento de protocolos, configuração e instalação de serviços e manutenção. A fundamental diferença entre as duas tecnologias é que redes ativas usam o conceito de processamento na camada de rede, ou seja, voltado para o encaminhamento de pacotes, enquanto agentes móveis executam como sistemas de aplicação. Sistemas baseados em agentes móveis são projetados para a construção de um ambiente de computação distribuído e interligado por um sistema de comunicação, enquanto o propósito das redes ativas é disponibilizar e tornar mais eficientes facilidades no sistema de comunicação.

No entanto, pode se considerar que os programas (códigos) transportados em cápsulas das redes ativas são agentes móveis. Da mesma forma, um elemento de rede que seja capaz de executar um agente móvel pode ser considerado um nó ativo [14].

Agentes móveis podem migrar baseados em decisões autônomas, além de poderem também gerar processos filhos ou *threads* para tratarem de problemas específicos. Essas funcionalidades

não estão previstas nas redes ativas. Por fim, agentes podem trocar mensagens entre si, algo também não previsto nas redes ativas [10].

No plano de gerenciamento, pode-se considerar que as funcionalidades a serem executadas estão, principalmente, no nível das aplicações, uma vez que o gerenciamento não tem como objetivo principal tratar diretamente da entrega de pacotes. Sendo assim, é mais aconselhável, no que diz respeito a frameworks[8] de gerenciamento, empregar-se o termo tecnologia ativa, pois tanto redes ativas quanto agentes móveis podem ser utilizados.

2.5. Tecnologia ativa no gerenciamento de redes

Nesta sub-seção serão integrados os conceitos básicos apresentados no capítulo com o objetivo de se formar a base conceitual sobre a qual a arquitetura proposta nesse trabalho é construída.

2.5.1. Gerenciamento distribuído com tecnologia ativa

A implementação do gerenciamento distribuído com o uso de tecnologia ativa vem sendo muito pesquisada nos últimos anos. Os elementos da rede que devem ser gerenciados, tanto nós de comutação quanto estações, podem receber softwares[9] para que tarefas específicas sejam executadas localmente. O simples polling via rede em busca apenas de dados pode ser praticamente abolido.

Os elementos da rede e as estações podem ser providos apenas de funcionalidades básicas no que diz respeito à obtenção de informações de gerenciamento. Capacidades genéricas e específicas podem ser providas via softwares enviados a esses elementos. O comportamento desses softwares e o seu nível de especialização podem ser modificados a qualquer momento. A flexibilidade das aplicações de gerenciamento é enorme nesse caso, uma vez que os elementos da rede podem ser configurados para atividades de gerenciamento de acordo com as necessidades do momento, sem as restrições do longo processo de padronização, desenvolvimento e implementação de novas funcionalidades que já foi citado. As atualizações de regras para a tomada de decisões, de limites para desencadeamento de ações ou de um protocolo de gerenciamento, por exemplo, poderiam ser feitas mediante simples atualizações dos softwares em questão. O mesmo se aplica a mudanças de políticas e inclusão de novos serviços de gerenciamento.

Além da facilidade de atualização de funcionalidades, esses softwares podem ser providos de capacidades avançadas no que diz respeito ao processamento das informações de gerenciamen-

---

[8] Frameworks são arquiteturas semi-prontas reutilizáveis. Não só os componentes de um sistema, mas também as decisões de projeto a ele associadas são reutilizados.

[9] Esses softwares podem ser agentes móveis ou aplicações ativas.

to. Os tempos de detecção de problemas e de restabelecimento do funcionamento normal do elemento gerenciado podem então ser bastante reduzidos, uma vez que o software de gerenciamento está sendo executado localmente. Pela mesma razão, a utilização da capacidade de transmissão dos enlaces da rede para fins de gerenciamento pode ser significativamente reduzida, pois o tráfego de gerenciamento é menor. Granularidades mais finas para monitoração e controle dos parâmetros de QoS podem também ser utilizadas. Apenas um mínimo de informações, como relatórios de ocorrências e estatísticas, precisam ser enviadas para as NMS. Além de agir diretamente no gerenciamento de elementos, softwares podem estar executando nos nós de comutação da rede também com a finalidade de filtrar e fundir as informações destinadas às NMS, reduzindo ainda mais o tráfego referente ao gerenciamento.

A possibilidade de se enviar programas para serem executados automaticamente nos elementos gerenciados trás, associada a si, os riscos de tal funcionalidade ser utilizada de forma mal intencionada. É imperativo, portanto, a presença de um mecanismo de segurança eficaz e eficiente. Os requisitos de um mecanismo desse tipo são discutidos por Reiser [38].

2.5.2. Gerenciamento de desempenho com tecnologia ativa

Certos aspectos na monitoração do desempenho de redes, serviços e aplicações, especialmente quando medidas de tempo estão envolvidas, são difíceis de serem considerados na abordagem centralizada de gerenciamento. O retardo gerado pela transmissão dos dados pela rede torna a precisão de medidas questionável [5].

O envio de um software com capacidade de processamento, como um agente móvel ou uma aplicação ativa, em vez do uso de polling via rede permite a análise local do elemento a ser gerenciado. Dessa forma, não há retardos da rede envolvidos pelo menos na fase de monitoramento do desempenho.

Softwares locais podem, por intermédio de cálculos sobre valores armazenados temporariamente, calcular a tendência de comportamento de certos parâmetros. Notificações são então enviadas à NMS quando limites dessa tendência forem ultrapassados. O processamento realizado localmente por esses agentes ou aplicações, no entanto, não pode ser excessivo em relação à capacidade de processamento disponível no elemento gerenciado em questão.

O paradigma de mobilidade pode ser utilizado, por exemplo, para a busca da causa de uma queda de desempenho ao longo de uma rota. Um agente se desloca a partir de uma extremidade de um serviço de comunicação até a outra, de nó de comutação em nó de comutação, colhendo informações sobre o desempenho nos enlaces que formam a rota. Ao final dessa busca, pode ser precisado onde e porque está ocorrendo uma queda no desempenho.

A capacidade de processamento local que pode ser instalada nos elementos gerenciados com o emprego das redes ativas e o paradigma de mobilidade viabilizam uma distribuição no gerenciamento que possibilita ao mesmo ser pró-ativo.

2.6. Resumo do capítulo

Este capítulo apresentou os conceitos básicos sobre os quais a arquitetura de gerenciamento de desempenho pró-ativo que é o assunto dessa dissertação é construída. Inicialmente foi discutida a mudança do paradigma de gerenciamento centralizado para distribuído. Foi então definido o gerenciamento de desempenho pró-ativo. Em seguida, foram apresentadas as características, vantagens e limitações das redes ativas, programáveis e dos agentes móveis. Ainda nesta subseção foram comparadas as redes ativas com os agentes móveis, que, juntas, compõem a tecnologia ativa. Finalmente, esses conceitos básicos foram integrados para que fossem apresentadas as vantagens do gerenciamento distribuído, em particular, do gerenciamento de desempenho, usando tecnologia ativa.

Capítulo 3 - Gerenciamento de desempenho pró-ativo

O gerenciamento de desempenho de redes, serviços, estações e aplicações pode ter seus objetivos reunidos em duas diretrizes, que são conflitantes entre si: (i) viabilizar o provimento dos níveis adequados de QoS que atendam às necessidades dos usuários e (ii) definir estratégias de alocação de recursos que maximizem a utilização desses elementos gerenciados, oferecendo benefícios, particularmente, aos provedores de redes e dos serviços. As atividades e os mecanismos que são necessários para a busca ao atendimento a esses objetivos podem ser organizados, de acordo com proposta de Pacifici [25], no modelo[10] apresentado na Figura 3.

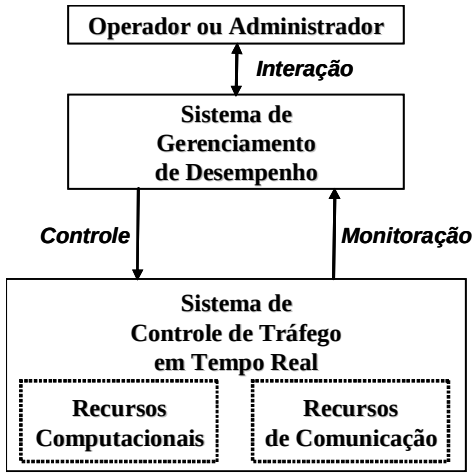


Figura 3 - O Sistema de gerenciamento de desempenho

O Sistema de controle de tráfego em tempo real regula a competição pelos recursos disponíveis nas estações e nos nós de comutação em tempo real. Um mecanismo de roteamento, por exemplo, faz parte desse Sistema de controle. O Sistema de gerenciamento de desempenho controla a operação do Sistema de controle de tráfego em tempo real, enquanto o Operador ou Administrador supervisiona toda as atividades, interferindo quando e onde for necessário. As interações entre os dois sistemas se dão tanto de forma assíncrona, em diferentes escalas de tempo, quanto de forma síncrona, quando assim for necessário. Já a interação entre o sistema de gerenciamento de desempenho e o operador se dá sempre de forma assíncrona.

O controle de tráfego em tempo real é realizado por uma coleção de mecanismos, cada um com uma tarefa específica, que atua nos diversos elementos gerenciados, que operam também de forma assíncrona entre si. Exemplos de controles realizados por esses mecanismos são: gerenciamento de armazenamento temporário em memória, de escalonamento de processos ou *threads* no processador de uma estação ou de um nó de comutação; de controle de fluxo, de roteamento

[10] Considera-se que as aplicações estão incluídas em “Recursos Computacionais”.

e/ou re-roteamento, de controle de admissão, entre outros. A operação desses mecanismos pode ser sintonizada através de *parâmetros de controle* associados a cada um dos mecanismos. O estado da rede, dos serviços, das aplicações e das estações é formado e pode ser obtido a partir da união, composição e interpretação desses parâmetros. O Sistema de gerenciamento de desempenho executa a sua tarefa monitorando os valores desses parâmetros e executando ações de controle, alterando o estado dos elementos gerenciados, também por intermédio desses parâmetros. O operador monitora esse estado através de abstrações dinâmicas visuais apresentadas em uma interface gráfica e exerce o controle através da manipulação de parâmetros de gerenciamento de desempenho.

A arquitetura proposta neste trabalho é baseada no modelo de sistema de gerenciamento de desempenho proposto por Pacifici [25], que foi apresentado na Figura 3. Adicionalmente, o paradigma de tecnologia ativa é usado, tendo como base a Arquitetura de Gerenciamento de Rede Ativa Distribuída (AGAD) [19], que foi desenvolvida no NCE da UFRJ.

O Sistema de gerenciamento de desempenho proposto é pró-ativo porque monitora os parâmetros desejados do Sistema de controle de tráfego em tempo real para detectar tendência de queda no desempenho dos mesmos que possam levar à ocorrência de *falhas de QoS*. Falhas de QoS causam impacto nos serviços que estão sendo prestados aos usuários como, por exemplo, a interrupção de um fluxo de vídeo que esteja sendo exibido. Um vez que sejam detectadas as tendências de queda no desempenho, ações são desencadeadas para que seja tentada a reversão dessa tendência ou, caso isso seja impossível, pelo menos a redução das conseqüências para o usuário. Essas ações se materializam na forma de ajustes nos parâmetros de controle do Sistema de controle de tráfego em tempo real. O comportamento do retardo em um serviço de comunicação pode ser utilizado como exemplo. Um mecanismo integrante do Sistema de controle de tráfego em tempo real, que está fora do escopo desse trabalho, disponibiliza informações sobre o parâmetro de controle retardo. O Sistema de gerenciamento de desempenho pró-ativo monitora a tendência de comportamento desse parâmetro através de sua variação e estima, mediante extrapolação, que a ultrapassagem do seu limite máximo, e a conseqüente falha de QoS, ocorrerá em determinado instante, caso aquela tendência se mantenha. Ações são então desencadeadas automaticamente, via agentes móveis, aplicações ativas ou outros meios, para que seja pelo menos tentada a reversão da tendência que foi detectada. Um exemplo dessas ações poderia ser a alteração de uma ou mais rotas em uma rede.

As ações desencadeadas pelo Sistema de gerenciamento de desempenho podem não ser suficientes para que seja evitada uma falha de QoS. Nesse caso, ações podem ser desencadeadas junto

aos usuários dos serviços de comunicações, ou seja, as aplicações, para que as adaptações cabíveis possam ser realizadas de forma a, pelo menos, se tentar evitar a queda na QoP percebida pelo usuário. Nesse caso, como as adaptações certamente reduzirão os níveis de QoS exigidos, pode ser possível até que a falha de QoS prevista não venha a acontecer e que o prejuízo para quem assiste a um vídeo, por exemplo, se limite apenas à mudança de resolução espacial do mesmo. A característica pró-ativa da arquitetura de gerenciamento de desempenho ora proposta é quem viabiliza que as adaptações possam vir a ser desencadeadas, se não totalmente, pelo menos parcialmente antes da ocorrência de falhas de QoS.

Este capítulo está organizado em cinco seções além de um resumo ao seu final. A próxima seção apresenta um resumo da AGAD, sobre a qual um sistema baseado na arquitetura sendo proposta deve ser executado. A seção 3.2 descreve a Arquitetura de gerenciamento de desempenho pró-ativo propriamente dita, apresentando a sua organização, seus elementos e o seu funcionamento. A seção seguinte descreve, em detalhes, os elementos da arquitetura e como os mesmos funcionam. A seção 3.4 descreve casos onde o emprego do gerenciamento de desempenho pró-ativo pode ser vantajoso e a seção 3.5 posiciona a arquitetura sendo proposta em relação aos trabalhos existentes na área, tecendo comparações.

3.1. Arquitetura AGAD

A Arquitetura de Gerenciamento Ativa Distribuída (AGAD) [19] tem por objetivo prover uma infra-estrutura baseada em tecnologia ativa que permita o gerenciamento distribuído das redes, serviços, estações e aplicações de um ou vários sistemas autônomos. A Figura 4 apresenta uma visão geral da AGAD, cujo funcionamento será resumido em seguida.

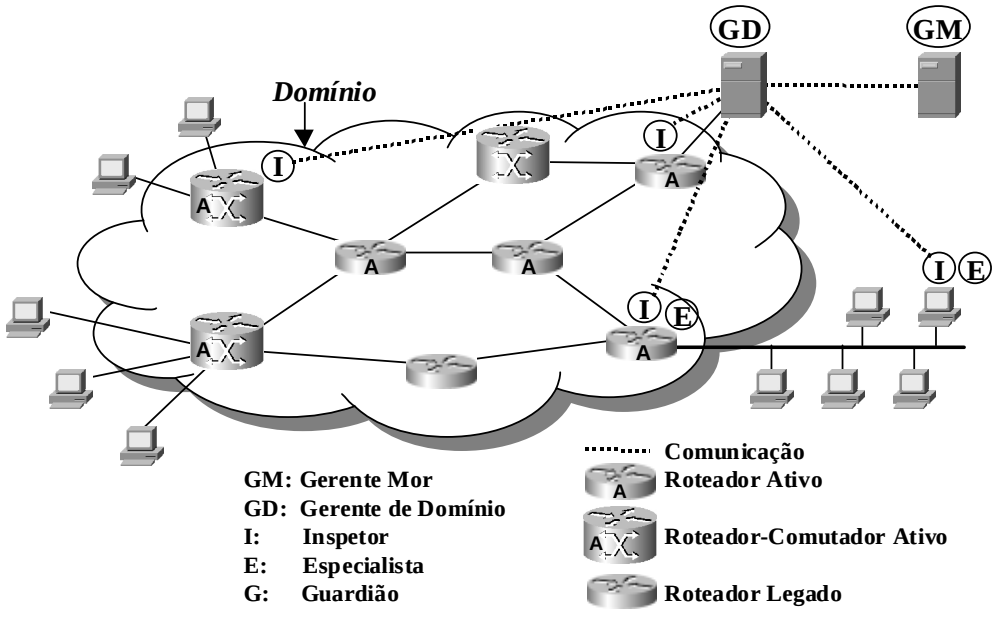


Figura 4 - Visão geral da AGAD

Os principais elementos da AGAD são: (i) Gerentes Mor (GM), (ii) Gerentes de Domínio (GD), (iii) Inspetores, (iv) Especialistas e (v) Guardiões. A comunicação entre esses elementos é realizada por um protocolo próprio. A seguir serão descritas as funcionalidades básicas de cada um desses elementos.

3.1.1. Gerente Mor

O GM é implementado de forma fixa, em uma estação de trabalho, e oferece uma interface gráfica para um administrador ou operador. O GM é o responsável pelo gerenciamento de um conjunto de domínios, via Gerentes de Domínio. Informações de gerenciamento consolidadas do conjunto de domínios são disponibilizadas pelo GM. Para tal, ele possui, em seu código, diferentes componentes de software que tratam das diferentes áreas funcionais de gerenciamento.

O GM utiliza-se de três bases de dados: a Base Geral de Códigos (BGC), a Base de Informações de Gerência (BIG) e a Base de Topologia Geral (BTpG). A BGC contém os códigos dos demais componentes da AGAD. A BIG armazena informações consolidadas sobre o gerenciamento dos domínios (recebidas de cada GD) e a BTpG contém informações sobre as topologias dos domínios gerenciados pelo GM.

O GM cria os GD e os envia para seus respectivos domínios. O funcionamento desses GD passa então a ser monitorado via mensagens do tipo *keep alive* e a integridade, não só dos GD, como dos demais elementos da AGAD, passa a ser monitorada pelos guardiões. Na inicialização do gerenciamento ou em caso de falha de um guardião já existente, o GM cria Guardiões e os envia aos seus GD para que cumpram tarefas relativas ao monitoramento da integridade dos mesmos.

Dependendo da quantidade de domínios gerenciados, pode haver mais de um GM.

3.1.2. Gerente de Domínio

Um domínio equivale a um conjunto de elementos a serem gerenciados sob uma mesma autoridade administrativa. Da mesma forma que o GM, o GD também disponibiliza uma interface gráfica para o administrador do sistema, com as mesmas funcionalidades da interface do GM. O GD é responsável pelo gerenciamento de seu domínio, que inclui a disponibilização de informações de gerenciamento de nível domínio e o desencadeamento de ações, quando necessárias, para a reversão de alguma situação desfavorável ou para, pelo menos, a minimização dos efeitos dessa situação. Para isso, são mantidas na GD quatro bases de informações: a Base de Topologia do Domínio (BTpD), a Base de Códigos de Domínio (BCD), a Base de Conhecimento, (BCN) e a Base de Informações dos Componentes do Domínio (BICD).

O GD cria e envia Inspetores e Especialistas aos elementos do domínio que devem ser gerenciados. As informações de gerenciamento (consolidadas) recebidas dos Inspetores são armazenadas

na BICD e, após a análise e classificação das mesmas, de acordo com o que estiver prescrito na BCN, Especialistas podem ser criados pelo GD para que sejam enviados para onde forem necessários para a realização de tarefas especializadas relativas ao gerenciamento. A análise e classificação podem se utilizar de técnicas de IA, daí a presença da BCN.

Além da interação com Inspetores e Especialistas, o GD também emite, após consolidações, relatórios com informações de gerenciamento sobre o seu Domínio. Quando necessário, informações relevantes são enviadas ao GM.

3.1.3. Inspetor

Os Inspetores deverão realizar, de fato, as tarefas relativas ao monitoramento no gerenciamento dos elementos do domínio. São eles os responsáveis pela obtenção, local, dos dados referentes às funcionalidades que são gerenciadas, ou seja, a monitoração dos parâmetros de controle do Sistema de controle de tráfego em tempo real, conforme apresentado na Figura 4. A obtenção desses dados é feita com o uso das facilidades disponíveis localmente, desde acesso direto às MIB até o uso de agentes *proxy* (explicados posteriormente). Os dados podem então sofrer um processamento local, simplificado o suficiente para não sobrecarregar o elemento gerenciado em questão, realizado pelo próprio Inspetor, para que apenas um mínimo de informações sejam enviadas, via rede, ao GD. O processamento local tem por objetivo realizar desde a filtragem de informações irrelevantes até a fusão de informações relevantes diferentes em uma mesma mensagem a ser enviada ao GD. Além dessa operação assíncrona em relação ao GD, o Inspetor também pode operar de forma síncrona, atendendo a pedidos específicos daquele Gerente.

O Inspetor é implementado segundo a organização que é apresentada, de forma esquemática, na Figura 5.

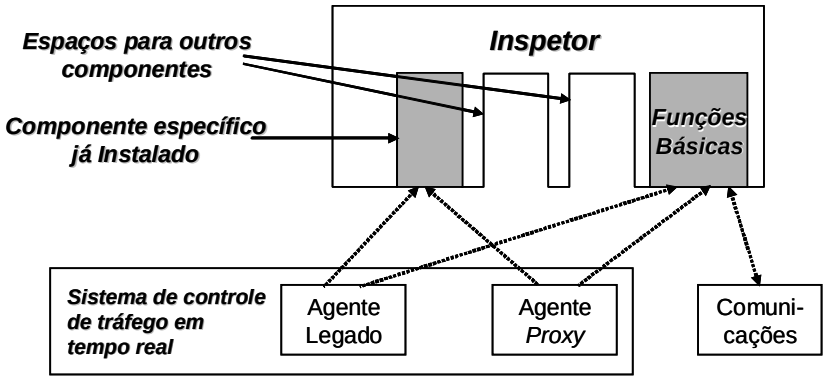


Figura 5 - Implementação do Inspetor

O bloco “comunicações” refere-se às facilidades de comunicação de dados do sistema operacional do elemento gerenciado que hospeda o Inspetor. Agentes proxies têm por finalidade disponibilizar uma interface padrão com o Inspetor e, ao mesmo tempo, interagir com uma tecnologia

proprietária. Agentes legados são aqueles diretamente acessáveis, via interfaces bem definidas, como por exemplo, uma MIB.

A implementação do Inspetor é configurável dinamicamente, de forma a permitir que as diferentes áreas funcionais de gerenciamento sejam tratadas independentemente, de acordo tanto com a natureza do elemento a ser gerenciado quanto com as intenções do administrador do domínio. É possível também a atualização do código dos componentes em tempo de execução. As funções básicas tratam de tarefas referentes ao funcionamento do Inspetor em si e de tarefas comuns às áreas funcionais de gerenciamento, como, por exemplo, a comunicação com o GD e a interação com um Guardião.

3.1.4. Especialista

Uma vez que os Inspetores detectem situações que exijam ações ou mesmo análises mais especializadas, agentes Especialistas devem ser empregados. Para estes não há uma organização configurável dinamicamente semelhante àquela do Inspetor. Eles são softwares autônomos, com funções bem definidas. Os Especialistas apropriados são enviados pelo GD ao elemento gerenciado em questão ou a outros elementos com um objetivo bem definido, para o qual foram programados. Exemplos de ações que podem ser realizadas por Especialistas são: a alteração de políticas de escalonamento, reconfiguração de utilização de memória, interagir com uma aplicação adaptativa, etc. Um Especialista pode até mesmo ser implementado na forma de uma mensagem para sinalizar a necessidade de que seja desencadeada uma adaptação em nível aplicação ou um re-roteamento.

3.1.5. Guardião

O Guardião tem como objetivo verificar se os demais componentes da AGAD estão funcionando corretamente e se os códigos dos mesmos estão íntegros, ou seja, se não foram alterados inadvertida ou intencionalmente. Cada um dos elementos da AGAD implementa uma interface para interação com o Guardião, onde há o suporte para funcionalidades como autenticação, autorização, controle de execução e verificação.

3.1.6. Comunicação entre os elementos da AGAD

A comunicação entre os elementos da AGAD se dá por intermédio de um protocolo próprio, de nível aplicação. Essa comunicação serve tanto para a troca de mensagens quanto para a mobilidade de código. Diversas são as opções para a mobilidade de código, característica que será apresentada com maiores detalhes no capítulo referente ao ambiente de desenvolvimento e implementação.

3.2. Arquitetura de gerenciamento de desempenho pró-ativa

A arquitetura de gerenciamento de desempenho pró-ativa proposta nesse trabalho tem por objetivo realizar o gerenciamento de desempenho relacionado aos elementos apresentados na Figura 6.

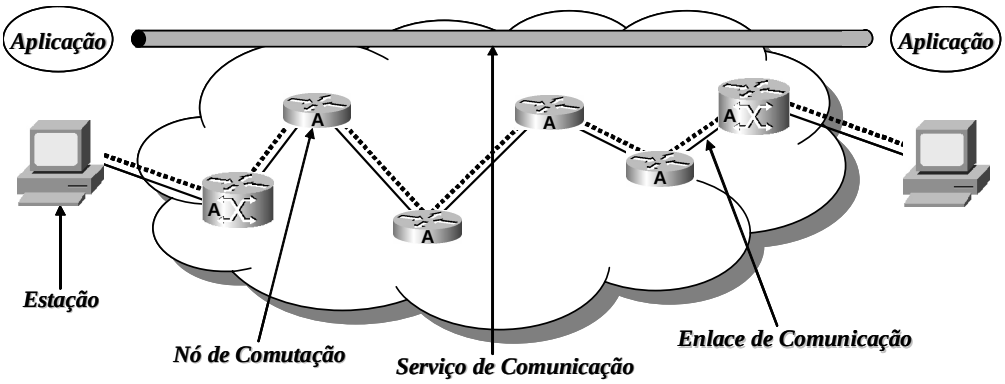


Figura 6 - Elementos cujo desempenho é gerenciado

Serão descritos, a seguir, os parâmetros[11] de controle relevantes para o gerenciamento de desempenho relativo a cada um desses elementos. O *tempo de resposta* é relevante para uma aplicação. Esse é o tempo decorrido entre o recebimento, pela aplicação, de uma solicitação e a disponibilização de uma resposta. A estação, que é composta por um hardware e um sistema operacional, deve ter a *capacidade de processamento*, a *quantidade de memória disponível*, e o *tamanho da fila de saída* gerenciados. No caso do serviço de comunicação, são relevantes o *retardo*, a *variação do retardo*, a *taxa de erros*, a *taxa de perdas* e a *vazão*. Os aspectos a terem seu desempenho gerenciados em um nó de comutação são os mesmos de uma estação, exceto pela presença de várias filas de saída. Em um enlace de comunicações são importantes a *utilização*, a *taxa de erros* e a *taxa de perdas*. Além desses parâmetros considerados relevantes, o Sistema de controle de tráfego em tempo real pode disponibilizar outros parâmetros para monitoração e controle, como, por exemplo, a *taxa de colisões* de um canal de comunicação compartilhado ou a *taxa de broadcasts* por unidade de tempo.

Conforme já foi mencionado, a associação destes parâmetros de controle com o elemento gerenciado é de responsabilidade dos mecanismos do Sistema de controle de tráfego em tempo real, que está fora do escopo desse trabalho. Ao Sistema de gerenciamento de desempenho só cabe a monitoração (obtenção do valores) e o controle (determinação de valores, quando possível e necessário) desses parâmetros.

---

[11] Esses parâmetros de controle são utilizados para que sejam definidos *níveis de QoS*.

O gerenciamento de desempenho[12] é, tradicionalmente, realizado através da monitoração dos valores momentâneos dos parâmetros de controle, citados no parágrafo anterior, em relação aos limites mínimos e/ou máximos da faixa de valores toleráveis para o elemento gerenciado em questão. Nas abordagens tradicionais de gerenciamento, o desrespeito a esses limites leva à geração de Alarmes (como as *Traps* do SNMP, por exemplo) que, por sua vez, deverão desencadear, normalmente, apenas sinalizações por parte de uma aplicação de gerenciamento ao administrador da rede. As desvantagens desse modelo centralizado, mencionadas na Seção 2.1, devem-se à centralização excessiva no processamento das informações sobre aqueles parâmetros e, adicionalmente, ao fato de que as ações são desencadeadas para tentar se reverter um quadro desfavorável em relação ao desempenho somente após o desrespeito aos limites, ou seja, após a ocorrência de uma falha de QoS.

A utilização de tecnologia ativa e a disponibilidade de mecanismos de adaptação, associadas à capacidade de processamento cada vez mais poderosa do hardware computacional presente tanto em estações quanto em nós de comutação, torna possível a exploração do monitoramento das tendências de comportamento dos parâmetros de controle de desempenho dos elementos apresentados na Figura 6.

A tecnologia ativa viabiliza não somente a presença de agentes[13] de gerenciamento dotados de inteligência e capacidade de processamento para monitorar as tendências localmente, bem como também viabiliza o envio de agentes (móveis) ou aplicações ativas para a realização de ações onde as mesmas forem necessárias, quando assim for desejado e viável. As aplicações adaptativas, por sua vez, após o recebimento de informações sobre as tendências de desempenho dos parâmetros que lhes são relevantes, como tempo de resposta, retardo, variação do retardo, taxas de perdas e de erros, podem começar a desencadear o processo de adaptação e, se for o caso, abrir novas sessões de um serviço de comunicações. Quando estas adaptações forem de fato necessárias, o tempo perdido com as mesmas que é percebido pelo usuário será menor, pois parte das tarefas já foram realizadas. Essas ações, normalmente não são desencadeadas de forma automática nas abordagens tradicionais de gerenciamento, onde não é utilizada tecnologia ativa.

Uma aplicação adaptativa que exibe um fluxo de vídeo pode ser utilizada como exemplo. Normalmente, a adaptação é realizada após a ocorrência de uma falha de QoS. Isso implica na per-

---

[12] Deste ponto em diante, nesta dissertação, o termo Gerenciamento de desempenho faz alusão, na realidade, ao *Sistema de gerenciamento de desempenho*, de acordo com o que foi mostrado na Figura 3.

[13] Nesse caso, o significado é de agente de gerenciamento, conforme previsto pelo IETF para o SNMP [51].

cepção, pelo usuário, da ocorrência da falha, na forma de uma paralização do fluxo de vídeo, até que uma nova mídia seja apresentada. Usando o gerenciamento pró-ativo, pode ser possível o adiantamento da adaptação, tendo como base as tendências de comportamento dos valores dos parâmetros de QoS, para que a queda de qualidade de percepção do usuário quando da ocorrência de uma falha de QoS seja minimizada.

A seguir serão descritas as quatro atividades que são realizadas pelo gerenciamento de desempenho pró-ativo que está sendo proposto, juntamente com o(s) momento(s) de seu desencadeamento.

1. ***Monitoração*** dos parâmetros de controle do elemento gerenciado: é realizada continuamente, com freqüência configurável para cada parâmetro, para detectar as tendências de variação dos mesmos.
2. ***Comunicação*** com o Gerente de Desempenho de Domínio e, quando for o caso, com o elemento gerenciado: é realizada quando for necessária, sendo causada pela detecção de uma tendência de que venha a ocorrer uma falha de QoS. As condições para que uma tendência seja caracterizada também são configuráveis.
3. ***Ação*** automática ou manual junto aos elementos gerenciados: é realizada quando ordenada.
4. ***Emissão de relatórios*** sobre o gerenciamento de desempenho: é realizada periodicamente, com freqüência configurável, ou quando ordenada.

3.2.1. Monitoração

A monitoração é realizada pelo componente (de software) de gerenciamento de desempenho pró-ativo instalado no Inspetor associado ao elemento a ser gerenciado. Esse componente de software de Inspetor será tratado, deste ponto em diante, como Inspetor de Desempenho (ID), embora, de fato, seja implementado na forma de um componente de software do verdadeiro Inspetor, de acordo com o que é previsto pela AGAD.

Durante a monitoração são coletados dados das fontes disponíveis, (como MIBs, informações obtidas do RTCP, de agentes *proxy* do sistema operacional da estação ou do nó de comutação, ou das próprias aplicações), sobre os parâmetros de controle que devem ser monitorados. A interface que existe entre o Inspetor de Desempenho e o elemento gerenciado, que é a interface entre o Sistema de gerenciamento de desempenho e o Sistema de controle de tráfego em tempo real do modelo apresentado na Figura 3, será detalhada na Seção 3.3.

A partir desses que foram dados coletados são calculadas as variações dos parâmetros que devem ser monitorados ao longo do tempo, para que sejam detectadas as tendências dessas variações. As variações são então utilizadas para que, mediante extrapolações, sejam calculados os instantes de tempo quando, caso as tendências se mantenham, limites mínimos ou máximos venham a ser desrespeitados. Em função do tempo disponível até que, de fato, as falhas de QoS ocorram, diferentes ações podem ser desencadeadas. Essas extrapolações são calculadas nos Gerentes de Domínio. O Inspetor de Desempenho trata apenas da obtenção dos dados, do cálculo das variações ocorridas nos mesmos e do envio dessas variações, quando necessárias e já consolidadas, ao Gerente de Desempenho de Domínio.

Visando um melhor entendimento do funcionamento do Inspetor de Desempenho, será usado como exemplo a monitoração de um parâmetro que expressa a *disponibilidade* de um recurso, que é expressa (e obtida) em valores percentuais. O Inspetor de Desempenho estará preocupado, então, com a diminuição desta disponibilidade. A Figura 7 apresenta uma escala representando: a faixa de valores na qual o parâmetro pode variar (0 a 100%), um valor que é o limite mínimo para o envio de Alarmes (80%) e três observações obtidas na monitoração (n, n+1 e n+k) pelo Inspetor de Desempenho. Só serão enviados Alarmes de Tendência caso a variação calculada a partir das observações assim exigir (será explicado a seguir), desde que o valor da observação que levou à necessidade de enviar o Alarme seja igual ou menor do que valor limite. Em caso contrário, todas as informações são calculadas e armazenadas, mas o Alarme de Tendência não é enviado.

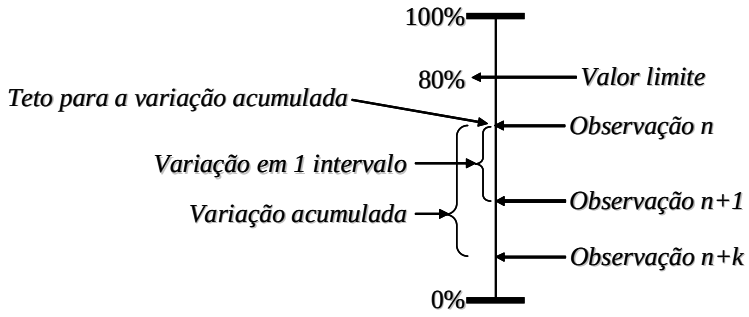


Figura 7 - Exemplo de parâmetro monitorado

Dois limites máximos devem ser estabelecidos para a variação: o valor máximo para uma variação em um único intervalo, ou seja, entre duas observações consecutivas (**n** e **n+1** na figura), e o valor máximo para a variação acumulada desde o último Alarme que foi enviado, (na figura, entre **n** e **n+k**), independentemente do número de observações (k, no caso do exemplo). O valor da observação **n** da figura é o valor de referência (chamado de teto) para cálculo da variação acumulada. Esse teto pode ser o maior de dois valores: da última observação que motivou o envio

de um Alarme de Tendência ou de uma observação posterior, cujo valor tenha sido maior do que aquela que motivou o envio do Alarme. Cabe destacar que no exemplo que está sendo analisado, "maior" significa "melhor", uma vez que o parâmetro monitorado representa disponibilidade. No caso do segundo valor para o teto (observação posterior já citada), como uma observação posterior veio a ser maior ("melhor", portanto) do que aquela que motivou o envio de um Alarme, ela passa a ser considerada como o teto para o cálculo da variação acumulada.

Visando ilustrar o que está sendo explicado, será utilizada uma amostra de teste com 50 observações. Os dois tipos de variações que levam ao desencadeamento de um Alarme de Tendência (ser maior do que a variação máxima tolerada em um intervalo ou ser maior do que a variação máxima acumulada tolerada) e o comportamento do teto (bola cheia) da variação acumulada estão ilustrados na Figura 8. O parâmetro utilizado como exemplo ainda é o mesmo da Figura 7, ou seja, a disponibilidade de um recurso. Nesse caso, o valor máximo tolerado para a variação em um intervalo entre duas observações consecutivas é de 10% e o valor máximo para a variação acumulada é de 12%. O valor limite (mínimo) para envio de Alarmes é aquele que já foi citado, 80%.

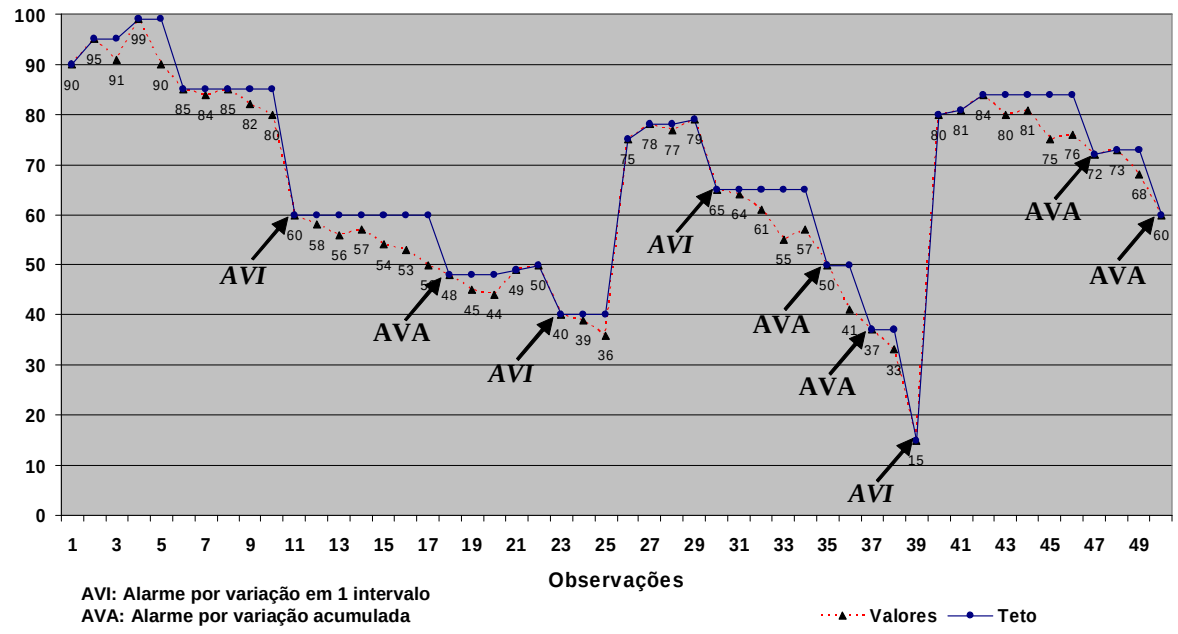


Figura 8 - 50 observações para teste

O teto (inicialmente coincidente com o valor da primeira observação) é aumentado de acordo com o aumento dos valores das observações de 1 até 4. Nas observações 3 e 5 o valor da observação decresce, mas o teto é mantido, pois ele é a referência para o cálculo da variação acumulada. Na observação 6 a variação acumulada chega a -14 (85% – 99%), excedendo o limite que foi configurado, que é de 12%. Um Alarme de Tendência deveria então ser enviado. Nesse caso, isso não ocorrerá, pois o valor da observação 6 (85%), que motivou o Alarme, ainda é superior

ao limite estabelecido para o envio de Alarmes, que é de 80%. O teto para cálculo de variação acumulada é então atualizado para o valor da observação 6. Na observação 11 (60 %, variando -20%, referentes a última observação, que foi de 80%) tanto o limite para a variação acumulada (12%) quanto o limite para a variação em um intervalo foram ultrapassados (10%), o que levará ao envio de um Alarme de Tendência, já que o limite mínimo para o envio de alarmes, que é 80%, também foi desrespeitado. Entre as observações 11 e 18 (valores 60% e 48%, respectivamente) o valor para a variação acumulada é igualado, o que levará ao envio de um Alarme. Deve-se notar que o valor da observação 14 (57%), embora tenha aumentado em relação à observação anterior (56%), não causa atualização do valor do teto, pois não é maior do que o valor do teto no momento, que é de 60%. Pode-se concluir que o valor do teto só é reduzido quando do envio de Alarme. Já o aumento do valor do teto ocorre sempre que uma observação apresentar valor superior ao valor atual do teto.

No caso do parâmetro de controle associar “melhor desempenho” a menores valores numéricos das observações e “pior desempenho” a maiores valores numéricos, tudo que foi explicado anteriormente continuará sendo válido, sendo necessária apenas a inversão do sentido do eixo de referência. Esse seria o caso se o parâmetro representasse o percentual de utilização de um recurso em vez do percentual de disponibilidade que foi usado como exemplo.

As informações que são necessárias para o cálculo de variações e para a tomada de decisão a respeito do envio ou não de um Alarme de Tendência devem ser mantidas em memória principal, visando uma minimização do tempo de acesso às mesmas. Memória secundária deve ser utilizada apenas para o registro perene dessas e outras informações que são julgadas necessárias, até que as mesmas sejam enviadas ao Gerente de Domínio. O acesso à memória secundária não deve inviabilizar o respeito ao intervalo entre obtenções consecutivas das observações dos valores do parâmetro monitorado. Essa freqüência de monitoração do parâmetro de controle em questão é limitada pelo tempo de execução do Inspetor de Desempenho, que deve realizar o que foi descrito nesse exemplo cada vez que for obtida uma nova observação do valor do parâmetro.

Além dos Alarmes de Tendência, o Inspetor de Desempenho armazena e envia, periodicamente, para o Gerente de Desempenho de Domínio, informações sobre o parâmetro de controle monitorado que venham a servir para o planejamento de capacidade. Os Alarmes de Tendência podem ser também enviados a outras aplicações além do Gerente de Desempenho de Domínio, desde que o Inspetor de Desempenho seja assim configurado.

3.2.2. Comunicação

A comunicação desencadeada pelo Inspetor de Desempenho visa atingir três objetivos: (i) transmitir para o Gerente de Desempenho de Domínio as informações que devem ser armazenadas para fins de emissão de relatórios e de planejamento de capacidade, (ii) enviar ao Gerente de Desempenho de Domínio Alarmes de Tendências, (iii) enviar às aplicações que assim o desejarem, os Alarmes de Tendência sobre os parâmetros monitorados.

Essa comunicação é realizada, no caso (i), através da interface existente entre o Inspetor de Desempenho e o Inspetor propriamente dito. No caso (ii), a comunicação é feita diretamente entre o Inspetor de Desempenho e o Gerente de Desempenho de Domínio. No caso (iii), a comunicação é realizada diretamente entre o Inspetor de Desempenho e as aplicações. A Figura 10 apresenta essas interfaces de forma esquemática.

Essa comunicação produz o menor tráfego possível na rede, sendo desencadeada apenas quando for necessária. Ela consiste apenas no envio de códigos e valores de parâmetros, além da identificação do Inspetor de Desempenho que as originou. O detalhamento dessa comunicação será feito também na Seção 3.3.

3.2.3. Ações

As ações referentes ao gerenciamento de desempenho são realizadas por Especialistas de Desempenho.

A seguir são citadas possíveis ações que podem ser realizadas por Especialistas nos elementos gerenciados.

- **Aplicações**: podem ser avisadas da tendência de queda no desempenho de serviços de comunicação e da própria estação. Nesse caso, a ação do Especialista a realização de uma comunicação. As ações propriamente ditas que são necessárias, como, por exemplo, adaptações, devem ser desencadeadas pelas próprias aplicações.
- **Estação**: redimensionamento de espaços para armazenamento temporário, alteração de prioridades e/ou de políticas de escalonamento e encerramento ou ativação de processos e/ou *threads*.
- **Serviço de comunicações**: renegociação de parâmetros de QoS e mudanças de rotas.
- **Nós de comutação**: as mesmas previstas para uma estação e a alteração da política de escalonamento de pacotes nas filas de saída.

- **Enlace de comunicações**: este elemento não pode ser objeto, diretamente, de nenhuma ação. No entanto, alterações nos algoritmos de detecção e correção de erros podem ser feitas, com o objetivo de se melhorar o desempenho de um enlace.

A descrição detalhada do Especialista de Desempenho, da sua composição e o seu funcionamento, bem como a interface entre os Especialistas e os elementos gerenciados que devem sofrer essas ações, será feita na Seção 3.3.4.

3.2.4. Relatórios

Dois tipos de relatórios são disponibilizados: *Relatórios de Estado* e *Relatórios para Planejamento de Capacidade*. O primeiro tipo, *Relatório de Estado*, consolida e apresenta as últimas informações disponíveis relativas ao desempenho dos elementos desejados, individualmente ou em conjunto. O segundo tipo, *Relatório para Planejamento de Capacidade*, consolida informações relativas ao desempenho ao longo de um período de tempo, que pode ser definido pelo operador ou administrador.

Uma interface gráfica presente nos Gerentes Mor e de Domínio disponibiliza as interações que são necessárias para a emissão desses relatórios.

O detalhamento e a implementação desses relatórios foram deixados fora do escopo deste trabalho.

3.3. Elementos integrantes da arquitetura

Nesta seção, serão apresentados de forma detalhada os elementos da Arquitetura de Gerenciamento de Desempenho Pró-ativo, incluindo o objetivo de cada um, os seus relacionamentos com os demais elementos da AGAD e os seus ciclos de vida.

A Figura 9 apresenta (sem usar uma notação específica) os relacionamentos dos componentes da Arquitetura de Desempenho Pró-ativo[14] do Gerente Mor e do Gerente de Domínio com os seus respectivos Gerentes. Esses componentes são denominados Gerente de Desempenho Mor e Gerente de Desempenho de Domínio. As interfaces entre esses componentes de software e os seus respectivos Gerentes da AGAD são também apresentadas.

[14] Deste ponto em diante será empregado apenas o termo *Desempenho*. O significado desse termo, no contexto desse trabalho é, no entanto, o de *Desempenho Pró-ativo*.

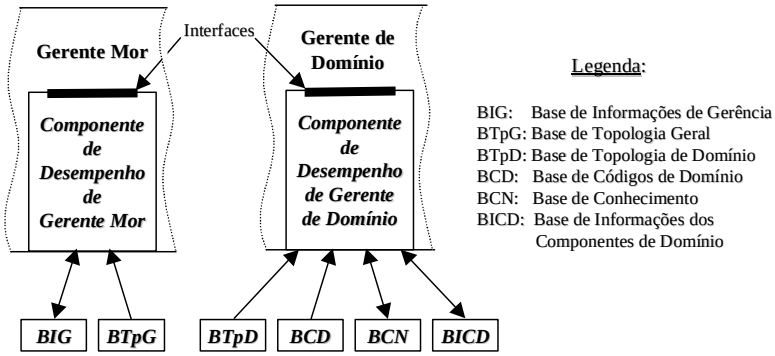


Figura 9 - Componentes de gerenciamento de desempenho dos Gerentes

A Figura 10 apresenta os relacionamentos do componente de software que trata do gerenciamento de desempenho (ou Inspetor de Desempenho) com o Inspetor (da AGAD) e com o elemento gerenciado. É também apresentado o relacionamento do Especialista de Desempenho com o elemento no qual serão desencadeadas ações. As interfaces também estão assinaladas.

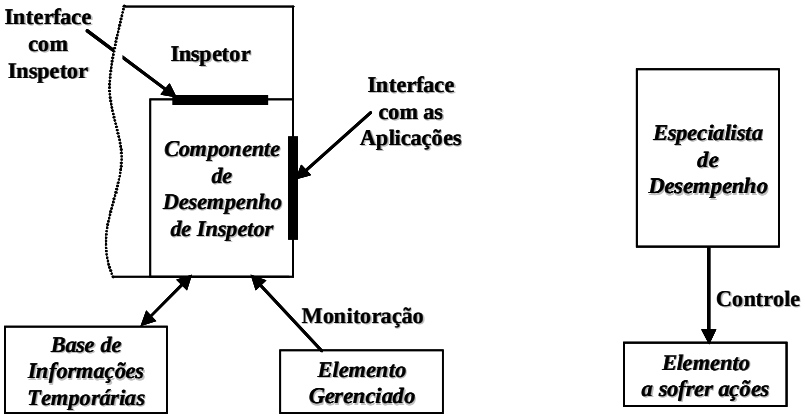


Figura 10 - Inspetor de Desempenho e Especialista de Desempenho

O termo “componente” que é empregado nesta e nas próximas sub-seções refere-se a componente de software, que é a forma pela qual são implementados os componentes da arquitetura de desempenho pró-ativo junto aos seus respectivos hospedeiros da AGAD.

As interfaces mostradas nessas duas figuras se dão em nível aplicação, à luz da orientação a objetos e do desenvolvimento de software orientado a componentes [22], que viabiliza a atualização de código em tempo de execução. A interação entre o Inspetor de Desempenho e elemento gerenciado, bem como entre Especialista de Desempenho e esse mesmo elemento, pode se dar de forma direta, através a obtenção e/ou a determinação de valores dos parâmetros de controle do Sistema de controle de tráfego em tempo real, ou de forma indireta, por intermédio de elementos integrantes deste último sistema. Nesse último caso, conforme já foi citado, a ação do Especialista pode se resumir apenas a uma notificação.

3.3.1. Gerente de Desempenho Mor

Esse componente de software do Gerente Mor trata do armazenamento de informações e da disponibilização de dados para a emissão de relatórios referentes ao gerenciamento de um conjunto de domínios. As informações são recebidas dos Gerentes de Desempenho de Domínio via facilidades de comunicação da AGAD e armazenadas na BIG.

O operador ou administrador do Gerente Mor pode então, via interface gráfica para interação com esse Gerente, ordenar a emissão de relatórios, que podem ser dos dois tipos já citados: relatórios sobre o estado do conjunto de domínios no que diz respeito ao desempenho ou relatórios para o planejamento de capacidade do conjunto de domínios.

O funcionamento do Gerente de Desempenho Mor pode ser entendido através do estudo do seu ciclo de vida, que está ilustrado na Figura 11:

1. O Gerente de Desempenho Mor deve ser iniciado pelo Gerente Mor, quando o operador ou administrador optar pela realização do gerenciamento de desempenho.
2. Quando o gerenciamento de desempenho de domínio for desejado e, conseqüentemente, ordenado, obter o código do Gerente de Desempenho de Domínio da BGCódigos.
3. Enviar o código do Gerente de Desempenho de Domínio para o Gerente de Domínio do elemento da rede onde o mesmo será executado.
4. Trocar informações com os Gerentes de Desempenho de Domínio através das facilidades de comunicações da AGAD
5. Armazenar as informações relevantes na BIG.
6. Verificar, periodicamente, se os Gerentes de Domínio estão ativos.
7. Quando ordenado, apresentar relatórios de estado ou para o planejamento de capacidade ou ainda qualquer outra informação solicitada.

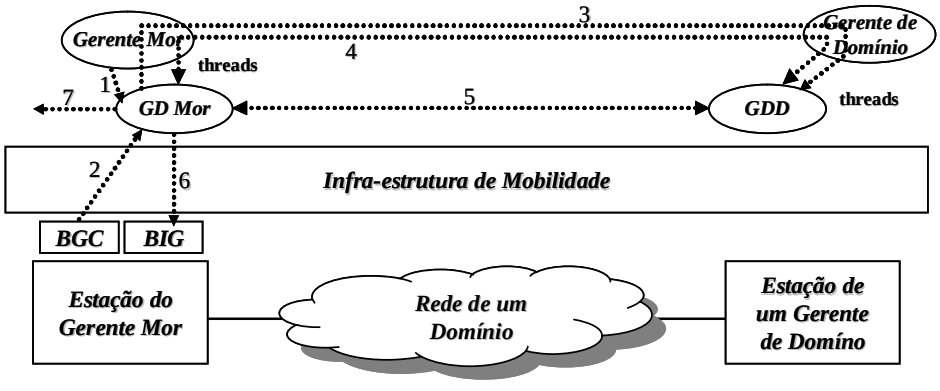


Figura 11 - Ciclo de vida do Gerente de Desempenho Mor

3.3.2. Gerente de Desempenho de Domínio

Esse componente de software do Gerente de Domínio é executado para que os oito objetivos citados a seguir sejam atingidos:

(i) Enviar, quando ordenado, os Inspetores de Desempenho para os elementos a serem gerenciados.

O Gerente de Desempenho de Domínio recupera da BCD o código dos Inspetores de Desempenho que devem ser enviados aos elementos a serem gerenciados, de acordo com o que foi determinado pelo operador e/ou administrador. Em seguida, esse código é enviado ao elemento em questão, via a infra-estrutura de mobilidade da AGAD.

(ii) Armazenar, após consolidar, as informações relativas ao gerenciamento de desempenho do domínio do qual faz parte, na BICD.

Essas são as informações que são recebidas dos Inspetores de Desempenho. A consolidação trata de minimizar as informações a serem, de fato, registradas. Informações decorrentes da execução de outras tarefas pelo próprio Gerente de Desempenho de Domínio e pelos Especialistas também são armazenadas na BICD.

(iii) Realizar análises e extrapolações, a partir de informações sobre tendências de comportamento do desempenho obtidas dos Inspetores de Desempenho.

Essas análises e extrapolações são realizadas a partir das informações monitoradas e enviadas pelos Inspetores de Desempenho sobre as tendências de comportamento dos parâmetros de controle do Sistema de controle de tráfego em tempo real.

O Inspetor de Desempenho, por poder estar instalado em equipamentos com pouca capacidade de processamento disponível, trata apenas da obtenção dos valores de momento dos parâmetros de controle e do cálculo da variação do valor desses parâmetros. As variações podem ser enviadas ao Gerente de Desempenho de Domínio de duas formas, de acordo com configuração determinada por esse mesmo Gerente: na forma de Alarmes de Tendência (diretamente do Inspetor de Esempenho ao Gerente de Desempenho de Domínio) ou individualmente (via AGAD). Esse último caso contempla a necessidade de que as próprias variações sejam armazenadas ou processadas pelo Gerente de Desempenho de Domínio. Nesse caso a exigência de capacidade de transmissão de dados da rede será maior.

O Gerente de Desempenho de Domínio, ao receber essas variações (diretamente ou em na forma de um Alarme) analisa-as e realiza um cálculo de extrapolação com o objetivo de determinar quando a variação em questão poderá causar uma falha de QoS, ou seja, quando um limite má-

ximo ou mínimo para aquele parâmetro em questão será ultrapassado. Um exemplo desse limite pode ser um valor de retardo que, se atingido, leve ao esvaziamento de um buffer de recepção, gerando uma interrupção no fluxo de vídeo que está sendo apresentado. Essa análise e a extrapolação podem se utilizar desde um simples ajuste de curva até avançadas técnicas de IA, como é exemplificado em [24]. O comportamento do desempenho do parâmetro em questão que já é conhecido também é levado em consideração na extrapolação, de forma que fatores como, por exemplo, a distribuição do tráfego na rede (*baseline)* ou do volume de transações de uma determinada aplicação ao longo das horas do dia, que já são conhecidas, sejam computados nas estimativas.

(iv) Decidir, de acordo com a análise e as extrapolações realizadas, se é necessário o envio de Especialistas e para quais elementos gerenciados os mesmos devem ser enviados.

O tempo disponível até a ocorrência de falhas de QoS, o parâmetro em questão e o tipo de elemento gerenciado são levados em conta para o Gerente de Desempenho de Domínio decidir se é o caso enviar um ou mais Especialistas de Desempenho e para quais elementos gerenciados os mesmos devem ser enviados.

A Base de Conhecimento (BCN) é consultada para que essas decisões sejam tomadas. No caso de uma adaptação [27], por exemplo, dois Especialistas podem ser enviados, um para o servidor e outro para o cliente, para que as respectivas partes da adaptação sejam desencadeadas. No caso de mudança de rotas, um único Especialista, na forma de agente móvel, pode percorrer a nova rota alterando tabelas. Técnicas de Inteligência Artificial podem ser utilizadas também na escolha dos Especialistas e de seus destinos.

(v) Enviar Especialistas de Desempenho.

Caso seja concluído, pela análise da extrapolação, que Especialistas devem ser enviados e para onde devem ser enviados, o Gerente de Desempenho de Domínio deve obter, junto à BCD, os códigos dos Especialistas de Desempenho em questão para que eles sejam enviados aos elementos onde vão ser executados. A infra-estrutura de mobilidade da AGAD é utilizada para esse envio.

(vi) Enviar informações consolidadas ao Gerente de Desempenho Mor.

Resumos de todas as informações que são armazenadas na BICD são enviados ao Gerente de Desempenho Mor, via AGAD. Esse envio pode ser configurado por aquele Gerente para ser realizado com uma determinada freqüência, condicionalmente ou a pedido.

(vii) Emitir os relatórios que forem solicitados.

Da mesma forma que o Gerente de Desempenho Mor, o Gerente de Desempenho de Domínio disponibiliza os dados necessários para a emissão dos Relatórios de Estado e para Planejamento de Capacidade que forem solicitados.

Para que esses sete objetivos descritos anteriormente sejam alcançados, o funcionamento do Gerente de Desempenho de Domínio é o que está descrito a seguir em seu ciclo de vida e está ilustrado na Figura 12:

1. Ser iniciado pelo Gerente de Domínio.
2. Obter, junto à BCD, os códigos dos Inspetores de Desempenho que forem necessários para o gerenciamento de desempenho dos elementos que forem determinados pelo operador e/ou administrador.
3. Enviar os Inspetores de Desempenho aos elementos selecionados pelo administrador da rede, via AGAD.
4. Receber dos Inspetores de Desempenho via AGAD informações relativas ao gerenciamento de desempenho do domínio e armazená-las, de forma consolidada, na BICD e enviar aos mesmos, também via AGAD, comandos e configurações, se necessário.
5. Receber Alarmes de Tendências de comportamento dos parâmetros de controle gerenciados dos Inspetores de Desempenho.
6. Analisar os Alarmes e realizar extrapolações, consultando a BCN, e decidir quais as ações que devem ser desencadeadas para que seja tentada a reversão das tendências.
7. Quando necessário, escolher Especialistas que devem ser enviados e decidir para onde enviá-los. Recuperar o código dos Especialistas da BCD e enviá-los via AGAD.
8. Quando ordenado, apresentar Relatórios de Estado ou para Planejamento de Capacidade, consultando as bases de dados que forem necessárias.

No caso ilustrado na Figura 12, o Especialista foi enviado para o mesmo elemento gerenciado onde está sendo executado o Inspetor de Desempenho, o que não é obrigatório.

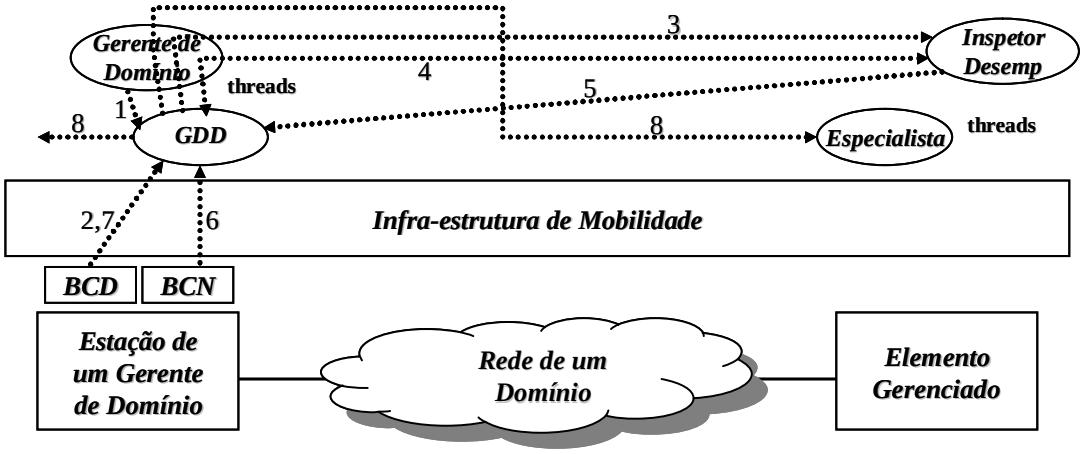


Figura 12 - Ciclo de vida do Gerente de Desempenho de Domínio

3.3.3. Inspetor de Desempenho

O Inspetor de Desempenho é executado em cada elemento da rede que tem algum parâmetro de controle cujo desempenho deva ser monitorado e o operador assim o desejar. Os objetivos do Inspetor de Desempenho podem ser assim resumidos:

(i) Monitorar os valores dos parâmetros de controle do Sistema de controle de tráfego em tempo real.

A monitoração consiste na obtenção periódica dos valores de momento dos parâmetros de controle que foram determinados para que o seu desempenho seja gerenciado. As obtenções desses valores a partir de parâmetros diferentes é realizada de forma assíncrona entre si, pois cada elemento e cada parâmetro possui a escala de tempo mais apropriada para o seu gerenciamento. Os parâmetros de controle do Sistema de controle de tráfego em tempo real que são monitorados podem ter seus valores obtidos de diferentes formas. MIBs podem ser acessadas diretamente pelo Inspetor de Desempenho, via SNMP, valores referentes aos parâmetros de um serviço de comunicações que se utilize dos protocolos RTP/RTCP para sua implementação podem ser obtidos a partir da aplicação que usa o serviço e, quando não houver a possibilidade de acesso direto a esses valores, agentes proxies podem ser utilizados. Esses agentes *proxies* comunicam-se com a facilidade disponível no elemento em questão e passam os valores desejados ao Inspetor de Desempenho, quando solicitado.

(ii) Calcular as tendências de variação dos valores dos parâmetros.

O Inspetor de Desempenho, por ser implementado na forma de uma aplicação ativa ou de um agente móvel, poderia até mesmo realizar complexas computações. No entanto, o mesmo não pode consumir uma quantidade significativa de capacidade de processamento do sistema

que o hospeda, especialmente quando este for um equipamento de rede, como um roteador, um comutador, uma ponte ou até mesmo um repetidor. Com a evolução da tecnologia utilizada nos enlaces de comunicações, o desempenho desses equipamentos de rede tende a ser tornar crítico para um bom desempenho do sistema de comunicação.

Dessa forma, o processamento realizado pelo Inspetor de Desempenho, conforme já foi detalhado na seção 3.2.1, trata apenas da realização do cálculo da variação dos valores dos parâmetros gerenciados a partir de valores consecutivos desses parâmetros, bem como da comparação do resultado desse cálculo com limites configuráveis.

O armazenamento das informações que são necessárias à execução do Inspetor de Desempenho (cálculo de variações e comparações), que não é grande, é feito em memória volátil, na Base de Informações Temporárias – BIT – visando ter-se uma maior rapidez no acesso às mesmas. Todas as operações e valores calculados pelo Inspetor de Desempenho são registrados, para serem usados na emissão de relatórios, em memória secundária, na Base de Informações Permanentes – BIP.

(iii) Comunicar-se com o Gerente de Desempenho de Domínio ou com aplicações para o envio de Alarmes de Tendência ou de variações.

A comunicação com o Gerente de Desempenho de Domínio é realizada por intermédio do Inspetor propriamente dito (da AGAD) ou diretamente ao Gerente de Desempenho de Domínio. Para a comunicação via AGAD, o Inspetor de Desempenho utiliza para isso a interface que foi mostrada na Figura 10. Essa comunicação consiste no envio, pelo Inspetor de Desempenho, das variações na freqüência que for ordenada e também do recebimento de comandos sobre quais os parâmetros a monitorar e quais os limites de variação a serem observados para esses parâmetros. Os fragmentos de código que atualizam os componentes do Inspetor de Desempenho são recebidos via AGAD. Já no caso do envio de Alarmes de Tendência, os mesmos são enviados diretamente ao Gerente de Desempenho, da forma que será explicada na seção referente à implementação.

Esses três objetivos são atingidos durante o ciclo de vida de um Inspetor de Desempenho, que é descrito a seguir:

1. Ser iniciado pelo Inspetor (da AGAD).
2. Aguardar o recebimento das mensagens do Inspetor (da AGAD) que especificam os elementos e desses, quais os parâmetros, a freqüência de observação, bem como os limites, que devem ter o desempenho monitorado.

3. Monitorar os parâmetros de controle dos elementos que foram selecionados para terem o seu desempenho gerenciado, obtendo valores, calculando variações, comparando-as com limites e, em seguida, armazenando-as na BIT.
4. Enviar para o Gerente de Desempenho de Domínio, via AGAD e na freqüência ordenada, informações sobre as variações dos valores dos parâmetros de controle.
5. Enviar Alarmes de Tendência referentes às tendências que ultrapassaram seus limites ao Gerente de Desempenho de Domínio.

3.3.4. Especialista de Desempenho

O Especialista é um software autônomo, que tem por objetivo realizar uma única ação e enviar um relatório ao Gerente de Desempenho de Domínio. O seu código deve ter o menor tamanho possível, para que o retardo relativo a sua transferência não seja significativo. O código do Especialista deve também ser desenvolvido especificamente para a tarefa para a qual foi projetado, podendo até ser dependente de plataforma, ou seja, enviado já na forma de código executável.

A ação a ser realizada pode exigir que o Especialista seja um agente móvel, com mobilidade fraca ou forte [40], ou ainda que seja simplesmente enviado e executado, deixando de existir após o envio de relatório sobre a ação, sem as características de um agente móvel.

Os objetivos do Especialista são descritos a seguir:

(i) Desencadear as ações comandadas pelo Gerente de Desempenho de Domínio.

O Especialista é enviado a um elemento gerenciado via infra-estrutura de mobilidade e deve ser imediatamente instanciado para executar a ação para a qual foi projetado.

(ii) Enviar relatório das ações realizadas ao Gerente de Desempenho de Domínio.

Toda ação realizada por um Especialista gera o envio de uma comunicação ao Gerente de Desempenho de Domínio relatando o resultado da mesma. No caso do Especialista ser um agente móvel, a comunicação pode se dar apenas ao final do caminho percorrido pelo mesmo, caso a ação assim o exija, ou, em caso contrário, ao final de cada ação realizada. Após envio do relatório, ao Gerente de Desempenho de Domínio, o Especialista pode migrar ou se encerrar.

3.3.5. Protocolo de comunicação para o gerenciamento de desempenho

A comunicação que ocorre entre os elementos relacionados ao gerenciamento de desempenho se limita a apenas duas situações: (i) entre o Gerente de Desempenho de Domínio e o Inspetor de Desempenho, e (ii) entre o Especialista e o Gerente de Desempenho de Domínio.

O formato das mensagens está apresentado na Figura 13. Esse formato de mensagem é uma extensão do formato previsto na AGAD [19]. Os tamanhos dos campos estão mostrados em caracteres (bytes).

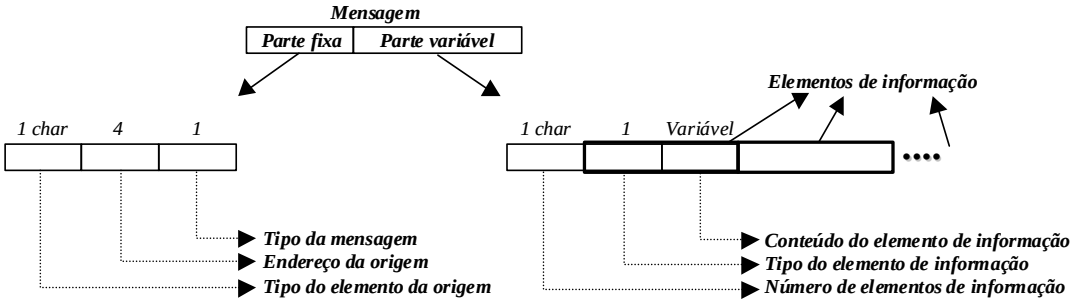


Figura 13 - Formato das mensagens do protocolo de comunicação da AGAD

O campo *Tipo do Elemento da origem* informa se a mensagem é oriunda de um Gerente de Desempenho de Domínio, Inspetor de Desempenho ou Especialista. Cada mensagem pode ser composta por quantos *Elementos de* Informação forem necessários. Cada elemento de informação transporta uma informação específica, sendo que a sua composição é variável. O primeiro campo indica qual é o elemento de informação e os demais transportam as informações propriamente ditas. Os elementos que não forem reconhecidos por uma versão de código são ignorados. Maiores detalhes sobre os campos do protocolo podem ser encontrados em [19].

3.3.6. O desempenho do gerenciamento de desempenho pró-ativo

É esperado que a utilização de linguagens como Java, bem como a presença de uma infra-estrutura de mobilidade baseada também nessa linguagem, venham a causar impactos no desempenho da arquitetura de gerenciamento de desempenho que é proposta nesse trabalho. No entanto, uma vez que a evolução da capacidade de processamento é constante e o desenvolvimento de processadores que executam Java ou linguagens similares está cada vez mais próximo de se tornar uma realidade prática [47], [48], a pesquisa de tal abordagem de gerenciamento de desempenho é justificada.

Um detalhamento das camadas de processamento que podem existir na implementação da arquitetura que é proposta nesse trabalho é apresentado na Figura 14. Nem sempre, no entanto, todas as camadas serão necessariamente utilizadas. Quando o código do Especialista, por exemplo, estiver disponível em linguagem compilada para a plataforma na qual o mesmo deverá agir, algumas camadas de processamento não serão utilizadas no momento da execução desse Especialista.

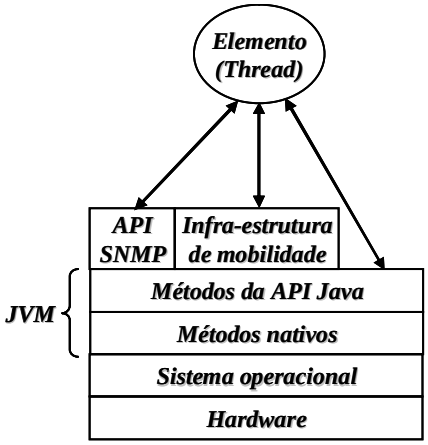


Figura 14 - Camadas de processamento existentes na arquitetura sendo proposta

O elemento da Figura 14 simboliza qualquer componente da arquitetura de gerenciamento de desempenho: Gerente, inspetor ou Especialista. Algumas medições sobre o desempenho do funcionamento da arquitetura usando essas camadas de processamento serão apresentadas no capítulo referente aos testes e análise dos resultados, quando então poder-se-á ter uma idéia do impacto causado pela presença das mesmas.

3.4. Aplicações do gerenciamento de desempenho pró-ativo

Nesta seção serão apresentadas algumas possíveis aplicações da arquitetura de gerenciamento de desempenho pró-ativo no gerenciamento dos diversos elementos citados na Figura 6, ou seja, aplicações, serviços de comunicação, estações de trabalho, nós de comutação e enlaces de comunicações. Serão também discutidas as vantagens e desvantagens de seu uso.

3.4.1. Adaptação em aplicações

Aplicações adaptativas vêm sendo pesquisadas [27], [28] com o intuito de que as mesmas venham a manter um certo nível de QoP de uma dada apresentação, ainda que com declínio da QoS disponível, principalmente na rede, mas também nos servidores, conforme será melhor detalhado na seção 3.4.3.

No caso de aplicações multimídia distribuídas, a adaptação envolve processamento tanto no cliente quanto no servidor de uma apresentação. No caso de uma mesma mídia estar disponível em diversas versões, com diferentes exigências de tráfego e de processamento, é suficiente que o servidor detecte uma falha de QoS para realizar a troca (adaptação) de versão da mídia em questão. No caso de não haver uma versão menos exigente de QoS da mídia para a qual ocorreu a falha de QoS, tanto o cliente quanto o servidor têm que realizar processamento na adaptação. O cliente tem que reestruturar a apresentação de forma a exigir outras mídias em substituição àquelas que falharam, mas mantendo a coerência da apresentação que estava sendo realizada.

Após essa reestruturação, o servidor ou os servidores das mídias da apresentação em questão têm que ser acionados para que passem a fornecê-las àquele cliente.

O gerenciamento de desempenho pró-ativo nesse caso pode monitorar as tendências de comportamento dos parâmetros de QoS dos fluxos das mídias da apresentação e enviar Alarmes de Tendência caso as mesmas ultrapassem certos limites. O Gerente de Desempenho de Domínio envia Especialistas tanto para o cliente quanto para o servidor, para que os respectivos processamentos necessários à adaptação possam ser adiantados. As conseqüências desse adiantamento podem ser, no pior caso, um ganho de tempo na própria execução da adaptação, minimizando o tempo de interrupção da mídia em questão, caso a falha da mesma venha de fato a ocorrer. No melhor caso, a adaptação pode ser realizada com um mínimo de prejuízo para a apresentação, sem que venha a ocorrer uma falha de QoS. Os usuários perceberiam apenas, por exemplo, a troca de versão da mídia, com a ocorrência de uma interrupção mínima.

Uma abordagem alternativa é o envio dos Alarmes de Tendência diretamente às aplicações, com o objetivo de se poupar o tempo referente ao processamento de análise e classificação no Gerente de Desempenho de Domínio.

3.4.2. Mudança de rotas em serviços de comunicação

As redes IP podem obter proveito das facilidades de comutação baseada em rótulos [49] disponível em tecnologias de nível de enlace, como ATM e Frame Relay, ou mesmo em tecnologias de acesso múltiplo e compartilhado como Ethernet, através do uso de um *shim header* [50]. Circuitos virtuais são criados em nível de enlace e os pacotes IP são então encaminhados apenas com base nas informações dos rótulos, sem que seja necessário o processamento em nível de rede em cada nó de comutação. MPLS [45] é um exemplo de tecnologia que se utiliza dessa técnica, que é ilustrada na Figura 15.

A atualização dos circuitos virtuais em nível de enlace no MPLS é feita, normalmente, com base nas informações de controle de nível IP, ou seja, nas informações oriundas dos protocolos de roteamento, como RIP, RIP-II, OSPF e BGP, entre outros [50]. A freqüência dessa atualização é a mesma da troca de informações desses protocolos, ou seja, na ordem de 30 segundos ou superior.

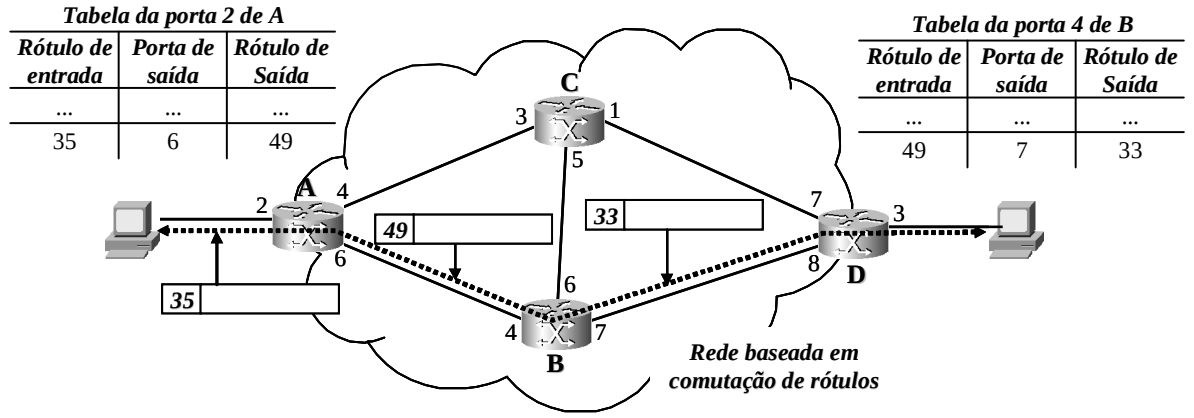


Figura 15 - Rede IP com comutação de rótulos em nível de enlace

Um sistema de gerenciamento de desempenho pró-ativo pode, ao detectar a tendência da ocorrência de uma falha de QoS em um determinado fluxo, desencadear, via um Especialista, a atualização das tabelas de rótulos dos roteadores-comutadores de forma que uma nova rota seja criada para aquele fluxo. Nesse caso, o Especialista poderia ser um agente móvel para percorrer a rota de um circuito virtual. Usando ainda a Figura 15 como referência, um agente poderia percorrer o caminho A-C-D e alterar as tabelas de rótulos desses roteadores-comutadores para que um novo circuito virtual fosse criado e passasse a ser a rota para o fluxo de dados existente entre as duas estações.

O Especialista tem que ser enviado para fazer este trabalho já de posse da nova rota para a qual o circuito virtual será criado. Isso exige que o Gerente de Desempenho de Domínio tenha conhecimento das rotas de seu domínio. Tal funcionalidade tem que ser obtida do Sistema de controle de tráfego em tempo real, de acordo com o que foi apresentado no início deste capítulo.

O tratamento individualizado de fluxos se caracteriza por ser uma solução pouco escalável. É requerida então uma política de gerenciamento que limite a quantidade desses tratamentos individualizados. Em nível domínio, no entanto, podem ser escolhidos apenas aqueles fluxos que são críticos e para os quais se justifique esse tipo de tratamento.

3.4.3. Utilização de recursos em estação de trabalho

Os sistemas de gerenciamento atuais concentram a maior parte de seus esforços na monitoração e controle das redes. As estações, no entanto, quer sejam clientes ou servidoras, podem ser responsáveis por falhas de QoS. Basta um servidor de vídeo atender a uma quantidade de solicitações de envio de fluxos que exceda sua capacidade de processamento e/ou de memória para que ocorram uma ou mais falhas de QoS, que certamente serão percebidas pelos clientes. É necessário então o gerenciamento do desempenho também das estações, principalmente daquelas que hospedam servidores.

O gerenciamento de desempenho pró-ativo pode realizar o monitoramento dos valores dos parâmetros capacidade de CPU e de memória utilizada (via MIB ou através de chamadas ao sistema) e quando estes valores excederem limite configurado pelo Gerente de Desempenho de Domínio, um Alarme de Tendência é enviado àquele Gerente. Este então pode enviar um Especialista que, de acordo com o tipo de estação, analise os processos e/ou *threads* em execução e encerre aqueles que não sejam essenciais. As aplicações críticas podem também ser avisadas para que passem a realizar um controle de admissão mais estrito, limitando a aceitação de novos pedidos de fornecimento de fluxos.

Esse trabalho exige Especialistas dedicados para cada tipo de estação, de forma que sejam conhecidos quais processos e/ou *threads* são essenciais ou críticos. No entanto, no âmbito de um único domínio, não se espera que exista uma quantidade grande de estações que executem aplicações críticas ao ponto de tornar o gerenciamento pró-ativo em um problema intratável.

3.4.4. Alteração de política de escalonamento de pacotes em nós de comutação

As mesmas monitorações e ações referentes às estações são aplicáveis também para os nós de comutação pois, em termos de hardware e software, são similares às estações, talvez apenas menos genéricos. No caso da utilização de tecnologia ativa em nós de comutação, por exemplo, a monitoração da capacidade de processamento e/ou de memória disponível pode vir a ser críticas, pois diversos agentes móveis – Inspetores de Desempenho, Especialistas ou outros tipos de agentes – podem estar sobrecarregando o nó até mesmo sem serem necessários, prejudicando a atividade principal em questão, que é de encaminhar ou rotear pacotes.

Adicionalmente em relação às estações, em um nó de comutação (roteador, comutador ou ambos), congestionamentos nos enlaces de saída podem previstos através do monitoramento das suas filas de saída. Nesse caso, a alteração da política de escalonamento de pacotes no atendimento das filas de saída é uma das ações possíveis de serem desencadeadas com o objetivo de se evitar a ocorrência de falhas de QoS nos fluxos afetados por essas filas. Outra possível ação é a implementação de diferentes políticas de descarte seletivo de pacotes, em função da variação da quantidade de pacotes nas filas e/ou da variação do tempo de espera dos pacotes nas filas. As políticas utilizadas com esses fins, tanto para o escalonamento quanto para o descarte, são, normalmente, definidas por configuração manual feita nos roteadores e/ou comutadores, sendo, portanto, estáticas.

3.4.5. Alteração de protocolos em enlaces de comunicações

Em nível enlace de comunicações, uma possível aplicação do gerenciamento de desempenho pró-ativo é a monitoração da taxa de erros no meio físico que é utilizado por aquele enlace. Em

função da variação das taxas de erros e de perdas, monitorada por um Inspetor de Desempenho, pode ser feita a modificação dos algoritmos de detecção e correção de erros que estão sendo utilizados. Um retardo adicional seria introduzido, porém, a forma tradicional de se lidar com erros acontecidos no nível físico, pelo menos em tecnologias de comutação rápida de pacotes, como ATM e Frame Relay, é quase sempre a detecção e/ou correção de erros fim a fim, o que, certamente, impõe um retardo maior no serviço de comunicações que está sendo prestado, caso ocorram erros, pois serão necessárias retransmissões.

Uma outra aplicação do gerenciamento de desempenho pró-ativo em nível enlace de comunicações é a alteração, de acordo com as variações detectadas por um Inspetor de Desempenho, do tipo de serviço que deve ser implementado por um protocolo de nível 2, ou mesmo a alteração do protocolo a ser utilizado. A troca de um serviço sem conexão e sem reconhecimento por um serviço sem conexão e com reconhecimento, por exemplo, pode ser feita, caso estejam ocorrendo colisões fora da banda em um meio físico compartilhado baseado em Ethernet.

3.5. Trabalhos relacionados

A distribuição do processamento, transferindo parte do processamentos do servidor de gerenciamento, como nas abordagens CMIP e SNMP, para próximo aos dados necessários ao gerenciamento, ou seja, para os próprios elementos a serem gerenciados, têm sido pesquisada desde o surgimento de abordagens padronizadas de gerenciamento. O trabalho de Yemini [17] destacou o conceito pela primeira vez. A primeira geração de sistemas de gerenciamento distribuído utilizava basicamente o paradigma cliente-servidor. A segunda geração usa RPC ou a distribuição estática de objetos, como CORBA e Java RMI para a distribuição das tarefas. As pesquisas atuais, configurando uma terceira geração, concentram-se no emprego de tecnologia ativa para a implementação da distribuição no gerenciamento [4].

A maioria dos trabalhos que foram publicados usam tecnologia ativa, porém, não implementam o gerenciamento pró-ativo. O trabalho consultado que emprega uma abordagem pró-ativa não se utiliza de tecnologia ativa.

Smart Packets [13] utiliza o paradigma de redes ativas. Um protocolo específico é utilizado, o Active Network Protocol (ANEP), para a transferência do programas que não podem exceder 1 KB. A proteção ao nó ativo é forte, tendo sido implementada uma linguagem de alto nível, S-procket, que não dispõe de ponteiros, não realiza o acesso a arquivos e nem faz gerenciamento de memória, que são ações potencialmente problemáticas. Um programa em Sprocket é compilado em outra linguagem, Spanner, com o objetivo de minimizar o tamanho do programa a ser transmitido na rede. Um ambiente de execução capaz de executar programas Spanner está pre-

sente em cada nó ativo. Um esquema de segurança realiza a autenticação dos programas a serem executados nos nós ativos. O paradigma de mobilidade não é utilizado.

A arquitetura Active Distributed Management (ADM) [10] é organizada em três camadas: gerenciamento de processos, básica de operação e ferramentas de gerenciamento. A primeira é o ambiente de execução, provendo as funcionalidades de redes ativas e de mobilidade, disponibilizando funções para criação, remoção, interrupção, continuação, duplicação, movimentação, comunicação, serviços de diretório e de segurança. A camada de básica de operação disponibiliza padrões de navegação para os fragmentos de código ativo (da terceira camada) que de fato realizam tarefas de gerenciamento. Na terceira camada são implementadas as funções relativas ao gerenciamento propriamente dito, na forma das aplicações ativas. A linguagem utilizada é Java.

O projeto MIAMI [1] implementa o gerenciamento de desempenho com o uso de três agentes, dois estáticos e um móvel. O primeiro, estático, realiza a interface entre o gerenciamento de desempenho e o sistema de gerenciamento. Este agente, quando ordenado, cria um agente móvel e o envia a um elemento de rede para que execute tarefas de monitoramento e de consolidação localmente. Um terceiro agente, estático, serve como agente *proxy*, realizando a interface entre o agente móvel e o elemento a ser gerenciado. O agente móvel envia informações consolidadas ao primeiro agente estático periodicamente ou assincronamente, quando limites configurados para parâmetros de desempenho forem ultrapassados. A linguagem utilizada também é Java.

Francheschi [24] investigou o gerenciamento pró-ativo utilizando técnicas de inteligência artificial para a geração de alarmes. Um módulo (equivalente a um agente de gerenciamento SNMP) pró-ativo é carregado no elemento gerenciado contendo um serviço de verificação que se utiliza de IA. Este serviço obtém valores dos parâmetros gerenciados via RMON, compara-os com dados do baseline do elemento gerenciado e, após verificá-los, se for o caso, emite alarmes de tendência

3.6. Resumo do capítulo

Esse capítulo diferenciou, inicialmente, o sistema de gerenciamento de desempenho, do sistema de controle de tráfego em tempo real. O primeiro sistema apenas monitora e determina ações sobre os parâmetros de controle do tráfego que são, de fato, controlados pelo segundo sistema. Na seção seguinte, a arquitetura AGAD, sobre a qual o sistema proposto é executado, é resumida. Em seguida, é apresentada a arquitetura de um sistema de gerenciamento de desempenho pró-ativo que engloba não só a rede, como também serviços, aplicações e estações de trabalho. Na seção seguinte, são detalhados os elementos integrantes do sistema e o seu funcionamento.

As possibilidades de aplicação da arquitetura proposta no gerenciamento de aplicações, serviços de comunicação, nós de comutação, estações e enlaces são também discutidas. Por fim, os trabalhos relacionados são comentados.

Capítulo 4 - Ambiente e implementação

Nesse capítulo será detalhado o ambiente de desenvolvimento que foi escolhido, bem como as razões que levaram a sua escolha e as vantagens e desvantagens decorrentes do seu uso. Também será detalhada a implementação de um protótipo da arquitetura de Gerenciamento de Desempenho Pró-Ativo (GDPA) composto de um Gerente de Desempenho de Domínio, um Inspetor de Desempenho e um Especialista de Desempenho que foram utilizados para a realização dos testes que serão descritos no próximo capítulo.

4.1. Ambiente de desenvolvimento

O ambiente de desenvolvimento que foi escolhido tem como plataformas computadores pessoais (PC) com sistema operacional aberto Linux da distribuição de versão 7.2 da Red Hat [46] com kernel 2.4.7-10 e como linguagem de programação, Java (JDK) [32], em sua versão 1.3.1 .

Essa infra-estrutura foi escolhida por utilizar softwares gratuitos e baseados em Java, que é independente de plataforma.

Por terem sido utilizadas plataformas com diferentes capacidades de processamento e configurações de hardware, as mesmas serão detalhadas no capítulo referente aos testes que foram realizados.

O sistema de gerenciamento de desempenho pró-ativo integra-se à AGAD, conforme já foi citado, e esta também se utiliza do mesmo ambiente de desenvolvimento.

A infra-estrutura de rede era baseada em Fast Ethernet compartilhada. Em alguns testes, conforme será explicado no próximo capítulo, foram usados segmentos Ethernet de 10 Mbps. A tecnologia de rede foi escolhida de forma que representasse a tecnologia mais disponível atualmente.

A seguir serão apresentadas a infra-estrutura de mobilidade utilizada, o μCode [41], e a interface usada para programação de aplicações em Java para a implementação de gerenciamento baseada no padrão SNMPv1 [43].

4.1.1. Infra-estrutura de mobilidade

A infra-estrutura utilizada é o μCode [41], que provê a mobilidade fraca, segundo a taxonomia de mobilidade de código de Fuggeta [40]. O ambiente da infra-estrutura é baseado na presença de servidores, denominados μServers, sendo executados sobre máquinas virtuais Java (JVM) em cada plataforma. Um μServer é um ambiente computacional para *threads* móveis, que quando migram mantêm seu estado referente aos dados, mas não seus estados de execução, daí a mobilidade implementada ser classificada como fraca. As operações básicas disponibilizadas pelos

μServers são a criação remota de *threads*, a cópia de *threads* e a relocação tanto de um conjunto de classes quanto do fechamento de uma classe, ou seja, a própria classe e todas que são por ela invocadas. A execução do código que é enviado pode se dar de forma síncrona ou assíncrona e com a execução imediata, na chegada do código, ou não (retardada).

Essa infra-estrutura de mobilidade é implementada sobre a JVM na forma de uma camada simples, que não requer processamento adicional significativo em relação à máquina virtual. A unidade de migração é um grupo, que é um conjunto de classes e objetos. Classes individuais ou o fechamento de uma classe podem ser adicionadas a um grupo. O uso do fechamento de uma classe faz com que não seja mais necessária nenhuma busca à plataforma de origem.

As *threads* geradas a partir de classes recebidas são mantidas em um espaço de classes privado, de forma a se evitar conflitos de nomes com outras classes locais. É possível, no entanto, publicar-se classes em um espaço de classe compartilhado associado ao μServer, de forma que as mesmas possam ser acessadas por outras classes do próprio μServer ou de μServers remotos.

A programação com o μCode é realizada utilizando-se três níveis de abstração: o núcleo, composto pelas classes Group e ClassSpace e pelas interfaces GroupHandler e Mobile; o nível intermediário, formado pela classe MuServer e o nível do usuário, que é formado pelas abstrações que foram implementadas pelo programador. A programação em nível de usuário é feita usando-se as primitivas disponíveis no nível intermediário, que poupa o programador de usar as primitivas de mais baixo nível de abstração que estão presentes no núcleo.

O μCode foi escolhido como infra-estrutura de mobilidade em função da baixa complexidade no seu uso, da sua simplicidade por exigir poucos processos e poucas *threads* em execução e, principalmente, da fina granularidade disponibilizada para a mobilidade de código, além de ser gratuito. É possível se enviar desde um simples objeto até o fechamento de uma classe.

4.1.2. Interface de programação de aplicações de Gerenciamento

Foi utilizada uma interface para programação de aplicações em Java, para implementação de gerenciamento baseada no padrão SNMPv1, da empresa AdventNet, em sua versão 3.3 [43]. O daemon SNMP que foi usado é aquele instalado pelo Red Hat 7.2, que é do pacote UCD-SNMP v4.2.1-7 [44].

Essa API foi escolhida por ser gratuita e oferecer um alto grau de encapsulamento, em Java, dos detalhes referentes à interação com agentes SNMP.

4.2. Implementação realizada

A apresentação da implementação que foi realizada será feita com o auxílio da *Unified Model Language* (UML) [42], para que sejam descritos os principais casos de uso, o modelo conceitual, os diagramas de seqüência e a hierarquia de classes do protótipo.

O enfoque da implementação, conforme será visto na próxima seção, referente aos testes realizados, foi sobre o Inspetor de Desempenho. A principal razão para a adoção desse enfoque foi o desejo de se investigar a viabilidade de se realizar tarefas de gerenciamento com fina granularidade de tempo utilizando-se uma linguagem de alto nível e parcialmente interpretada, como Java, juntamente com uma infra-estrutura de mobilidade, também baseada em Java.

O protótipo implementado é composto por três componentes de software, um Gerente de Desempenho de Domínio, um Inspetor de Desempenho e um Especialista de Desempenho. Conforme já foi citado, foi deixada fora do escopo dessa dissertação a exploração de técnicas de análise dos dados obtidos do Inspetor de Desempenho que é feita pelo Gerente de Desempenho de Domínio, bem como da análise e extrapolação de variações originadas nos Alarmes por este último elemento para a seleção de Especialistas. O Gerente de Desempenho de Domínio implementado trata somente da recepção de um alarme e do envio de um Especialista pré-determinado. Já o Inspetor de Desempenho que foi implementado é totalmente funcional, realizando tudo que foi previsto na Seção 3.2.1.

A utilização do protótipo (daquelas que foram citadas na seção 3.4) escolhida para a realização dos testes, descritos no próximo capítulo, é o gerenciamento de desempenho que é realizado em benefício de uma aplicação adaptativa, que é o ServiMídia [28]. Nesse caso, o parâmetro mais relevante a ter o seu desempenho gerenciado é o retardo entre a transmissão, pelo servidor, e a recepção, pelo cliente, dos pacotes que contêm o vídeo e o áudio de uma apresentação multimídia. Assim, o protótipo implementado trata do monitoramento deste parâmetro de controle, o retardo. Quanto ao desencadeamento de ações, duas opções são consideradas: a tradicional, onde o envio do Alarme de Tendência é feito para o Gerente de Desempenho de Domínio e uma alternativa, onde o Alarme é enviado diretamente à aplicação.

4.2.1. Casos de Uso do protótipo

Os requisitos exigidos do protótipo que foi implementado podem ser entendidos com o auxílio dos Casos de Uso [42] apresentados a seguir, que mostram os principais processos implementados. Cabe destacar que os atores dos casos de uso são sempre entidades externas ao protótipo, como, por exemplo, o Gerente Mor, o Gerente de Domínio, o Inspetor, todos da AGAD.

- **Caso de Uso**: Iniciar a monitoração de um parâmetro de controle.

- **Atores**: Gerente de Domínio (iniciador) e Inspetor.
- **Finalidade do Caso de Uso**: Capturar a inicialização de um Inspetor de Desempenho para o monitoramento de um parâmetro de controle.
- **Visão geral**: O Gerente de Domínio inicia o Gerente de Desempenho de Domínio e, de acordo com a configuração determinada pelo administrador da rede, envia um Inspetor de Desempenho a um elemento gerenciado para monitorar o retardo de um fluxo de vídeo recebido por um cliente ServiMídia. O elemento gerenciado já deve possuir um Inspetor (da AGAD) em execução.
- **Seqüência de eventos**:

Ação do ator	Resposta do protótipo
1. O gerenciador do sistema determina o parâmetro de controle a ser monitorado, identifica o elemento gerenciado onde esse parâmetro será monitorado e determina os limites das variações para o envio de Alarmes de Tendência que foram descritos na seção 3.2.1, a freqüência de monitoramento e para quem devem ser enviados os Alarmes.	2. O Gerente de Desempenho de Domínio é iniciado, recupera o código do Inspetor de Desempenho apropriado e inicializa (ou atualiza, caso já tenham sido inicializadas) as Bases de Informações que são necessárias ao gerenciamento de desempenho, BCD, BCN e BICD.
	3. O Inspetor de Desempenho, configurado com os valores já citados no evento 1, é enviado para o Inspetor do elemento gerenciado em questão, via infra-estrutura de mobilidade.
4. O Inspetor recebe o Inspetor de Desempenho e o instancia.	5. Inspetor de Desempenho inicia a sua execução, registrando seus métodos no Inspetor e apresentando-se à aplicação adaptativa para a qual monitorará o retardo.
	6. Inspetor de Desempenho inicia, se for necessário, o *proxy* do parâmetro de controle para poder obter valores do mesmo.
	7. Inspetor de Desempenho inicia o monitoramento do parâmetro retardo.

- **Caso de Uso**: Monitorar um parâmetro de controle.
  - **Atores**: Sistema operacional (iniciador) e parâmetro retardo do sistema de controle.
  - **Finalidade do Caso de Uso**: Capturar o monitoramento do comportamento do retardo.
  - **Visão geral**: Obter o valor do retardo com a freqüência configurada, calculando a sua variação e, de acordo com os limites pré-determinados, decidir pelo envio de um Alarme de Tendência.
  - **Seqüência de eventos**:

Ação do ator	Resposta do protótipo
1. Sistema operacional desbloqueia o Inspetor de Desempenho, após um intervalo de tempo equivalente à freqüência configurada para a monitoração do retardo.	2. Inspetor de Desempenho obtém valor atual do parâmetro retardo.
	3. A variação desde a última observação feita é calculada e armazenada na Base de Informações Temporárias.
	4. O valor da variação é comparado com o valor máximo da mesma que é permitida entre duas observações consecutivas. É tomada a decisão de se enviar ou não um Alarme de Tendência.
	5. A variação acumulada é calculada e comparada com o seu valor máximo que é permitido. É tomada a decisão de se enviar ou não um Alarme de Tendência.
	6. O valor do retardo é comparado com o limite mínimo a partir do qual devem ser desencadeados Alarmes. Se havia sido decidido o envio de um Alarme de Tendência, o mesmo é enviado.
	7. As informações necessárias são armazenadas na Base de Informações Permanentes.

- **Caso de Uso**: Desencadear ação com o uso de Especialista de Desempenho.
  - **Atores**: Retardo, Gerente de Domínio (iniciador), Inspetor.
  - **Finalidade do Caso de Uso**: Capturar o tratamento de um Alarme de Tendência com o uso de um Especialista de Desempenho.
  - **Visão geral**: O Gerente de Domínio, após análise do Alarme de Tendência recebido recupera e envia o Especialista apropriado ao elemento gerenciado.
  - **Seqüência de eventos**:

Ação do ator	Resposta do protótipo
1. Sistema operacional desbloqueia o Inspetor de Desempenho, após um intervalo de tempo equivalente à freqüência configurada para a monitoração do retardo.	2. Inspetor de Desempenho realiza o que está previsto nos eventos 2 a 7 no caso de uso anterior, enviando o Alarme de Tendência ao Gerente de Desempenho de Domínio.
	3. O Gerente de Desempenho de Domínio recebe um Alarme de Tendência e o analisa, escolhendo o Especialista apropriado e decidindo para qual elemento gerenciado enviá-lo.
4. Gerente de Desempenho de Domínio recupera o código do Especialista selecionado e, via infra-estrutura de mobilidade, o envia ao elemento gerenciado.	
5. O Inspetor (da AGAD) do elemento gerenciado recebe o código do Especialista e o instancia.	6. O Especialista inicia a sua execução.

- **Caso de Uso**: Enviar Alarme de Tendência diretamente à aplicação.

- **Atores**: Retardo, Aplicação Adaptativa.
- **Finalidade do Caso de Uso**: capturar a comunicação à Aplicação Adaptativa da ocorrência de um Alarme de Tendência de variação do retardo.
- **Visão geral**: O Inspetor de Desempenho comunica, diretamente à Aplicação, a ocorrência do Alarme de Tendência.
- **Seqüência de eventos**:

Ação do ator	Resposta do protótipo
1. Sistema operacional desbloqueia o Inspetor de Desempenho, após um intervalo de tempo equivalente à freqüência configurada para a monitoração do retardo.	2. Inspetor de Desempenho realiza o que está previsto nos eventos 2 a 7 no caso de "Monitorar Parâmetro de Controle", sem enviar o Alarme de Tendência ao Gerente de Desempenho de Domínio.
	3. Inspetor de Desempenho comunica à Aplicação, diretamente, a ocorrência do Alarme de Tendência, passando o valor do retardo bem como a variação eu foi excedida e causou o Alarme.

4.2.2. Modelo Conceitual do protótipo

Os principais objetos do protótipo, as suas associações e os seus principais atributos podem ser visualizados no Modelo conceitual [42] que é apresentado na Figura 16.

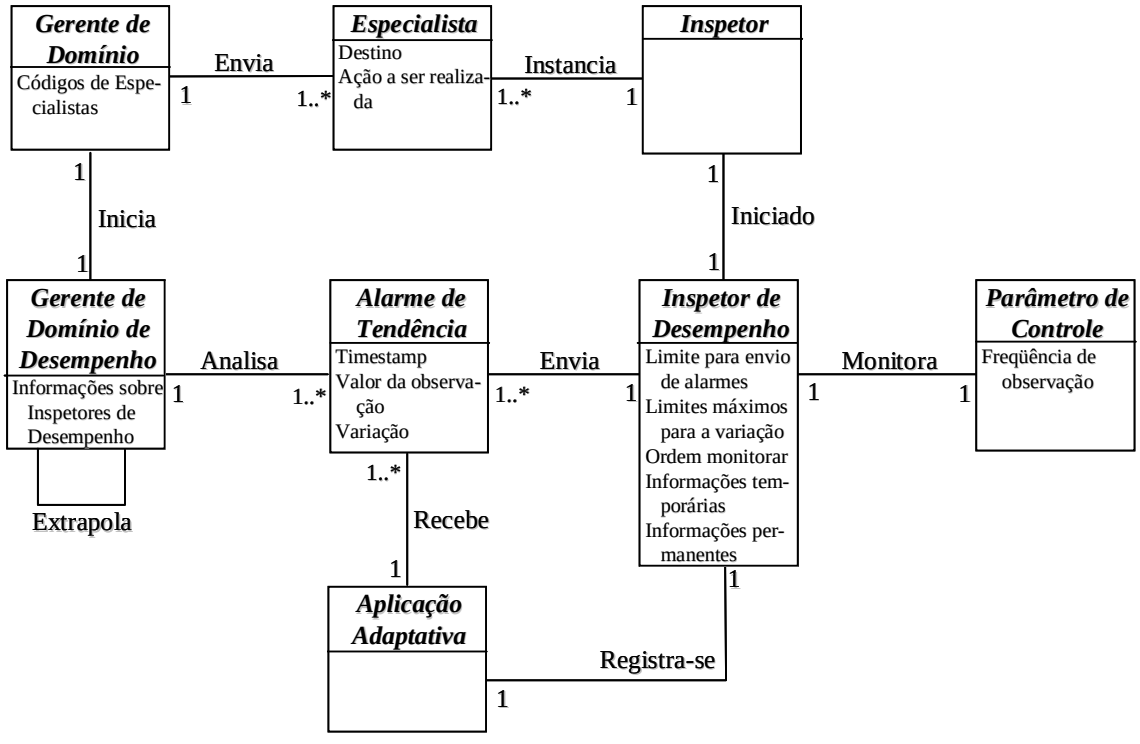


Figura 16 - Modelo conceitual do protótipo

4.2.3. Diagramas de Seqüência do protótipo

Os Diagramas de Seqüência apresentados a seguir descrevem as interações que ocorrem entre os objetos do protótipo. O primeiro diagrama (Figura 17) ilustra o início do gerenciamento de de-

sempenho. A seqüência de mensagens trocadas na monitoração do parâmetro de controle, juntamente com o uso de um Especialista para o desencadeamento de ações está apresentada no segundo diagrama (Figura 18). O envio do Alarme de Tendência diretamente para a aplicação adaptativa pode ser deduzido com facilidade a partir deste último diagrama e, portanto, não será apresentado.

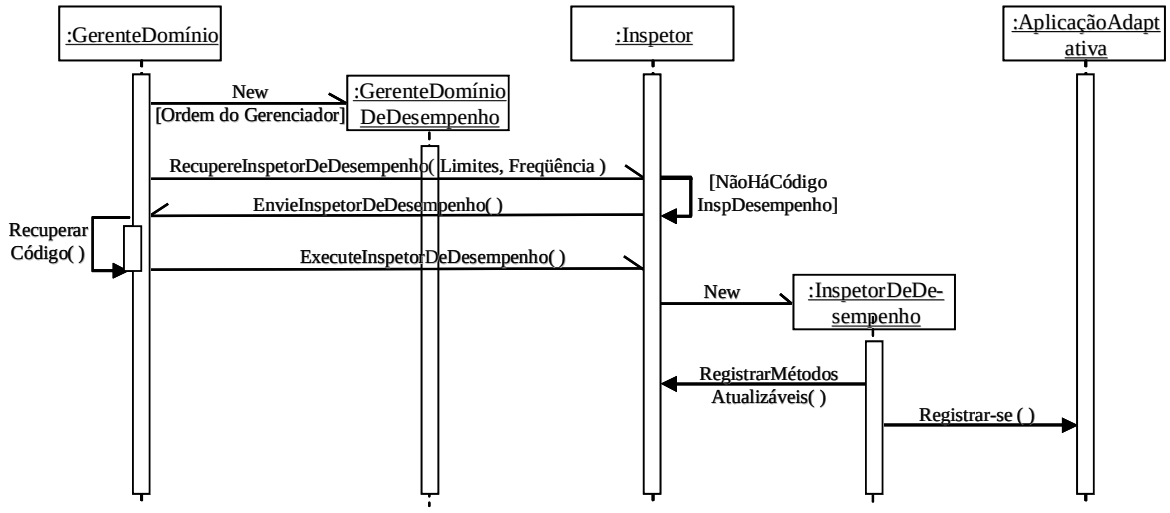


Figura 17 - Diagrama de Seqüência Iniciar monitoramento de parâmetro de controle

O objeto ProxyParâmetroControle do diagrama apresentado a seguir é quem realiza a interação com o sistema de controle em tempo real para a obtenção do valor do parâmetro no momento desejado. No protótipo implementado, esse objeto é instanciado a partir da API de gerenciamento SNMP que é utilizada para a obtenção de valores de variáveis de MIB.

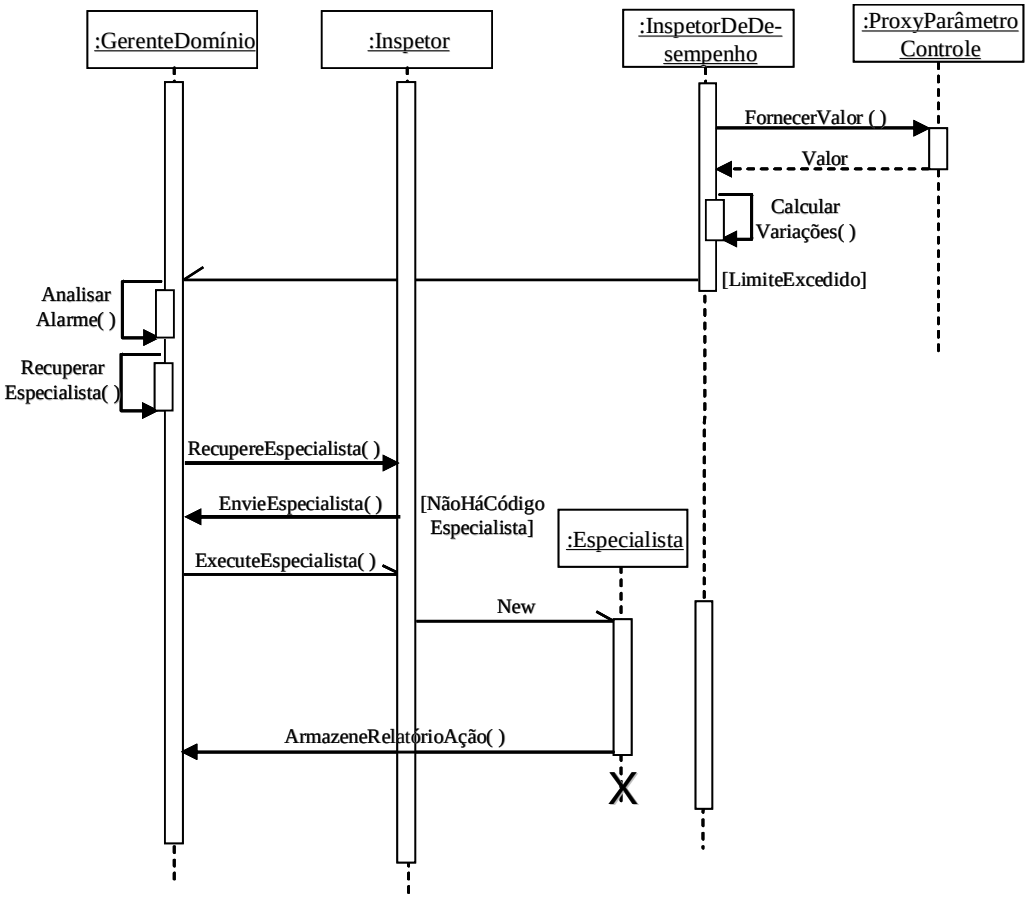


Figura 18 - Diagrama de Seqüência de Monitoração e de ação com o uso de Especialista

4.2.4. Hierarquia de Classes do protótipo

Serão apresentadas de forma mais detalhada, conforme pode ser observado na Figura 19, apenas as classes do protótipo que foi implementado. As classes da AGAD que são relevantes ao entendimento do protótipo são também mostradas, porém, sem maiores detalhes.

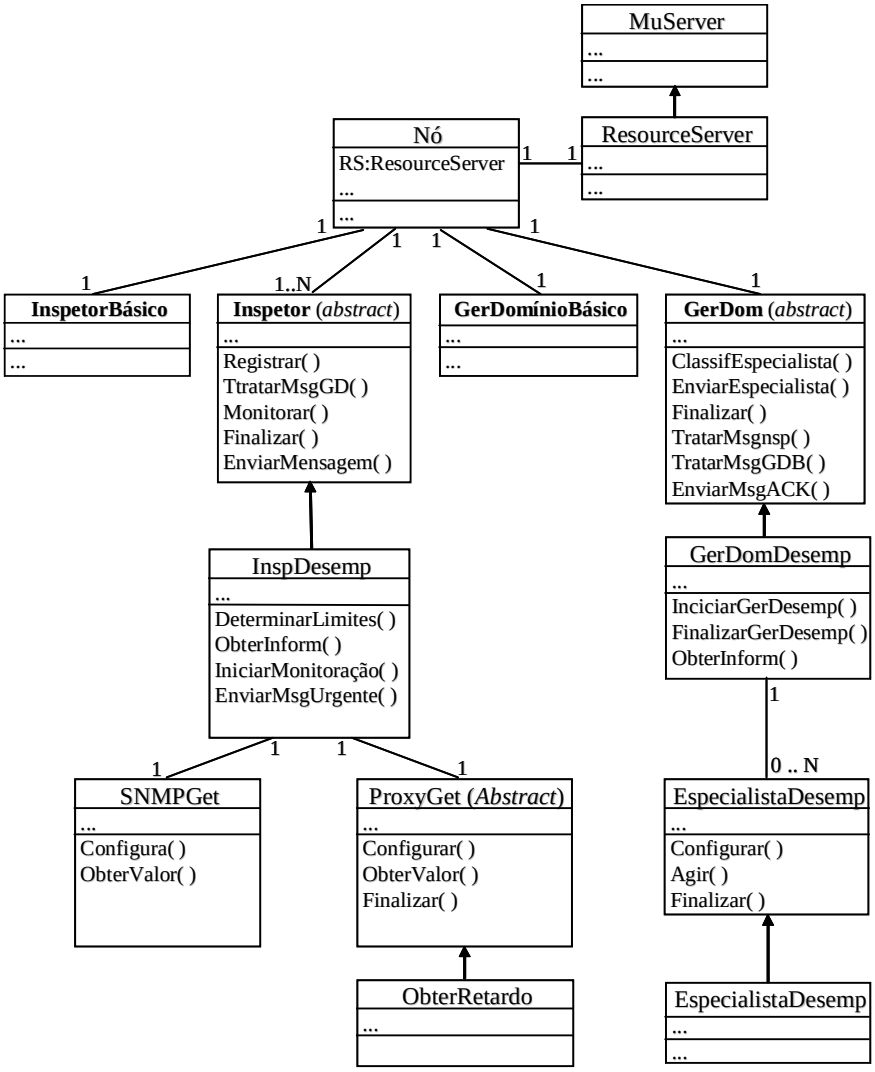


Figura 19 - Diagrama de Classes do protótipo

Apenas a classe do principal elemento do protótipo, o Inspetor de Desempenho, terá os seus métodos descritos com maiores detalhes.

- Classe **InspDesemp**:
    - Método **Registrar**: É herdado da AGAD. Registra a instância da classe junto ao Inspetor (da AGAD) para fins de atualização de código.
    - Método **TratarMsgGD**: É herdado da AGAD. Realiza o tratamento das mensagens que são recebidas do Gerente de Desempenho Mor via AGAD.
    - Método **Monitorar**: É herdado da AGAD. Implementa o laço de monitoração do Inspetor de Desempenho, cujo funcionamento foi explicado na seção 3.2.1. O pseudo-código deste método está apresentado na próxima seção (4.2.5). A monitoração, de fato, não é iniciada quando este método é chamado. Outro método é usado com essa finalidade.

- Método **Finalizar**: É herdado da AGAD. Permite a finalização da instância da classe, quer seja por encerramento das atividades do Inspetor de Desempenho ou para que o mesmo seja atualizado.
- Método **EnviarMensagem**: É herdado da AGAD. Permite o envio de mensagens não urgentes. Essas mensagens são enviadas com o uso das facilidades de comunicação da AGAD.
- Método **DeterminarLimites**: Permite a determinação dos limites que podem ser configurados no Inspetor de Desempenho. Os limites, que são os parâmetros do método, são aqueles explicados na seção 3.2.1. Esse método pode ser usado durante a execução do Inspetor para que os valores dos limites sejam alterados.
- Método **IniciarMonitoração**: Permite o desencadeamento e a interrupção do laço de monitoração.
- Método **ObterInform**: Permite que as informações referentes à monitoração, ou seja, os valores dos parâmetros ao longo do tempo, as variações dos mesmos e os alarmes que foram enviados e as estampas de tempo sejam recuperadas pelo Gerente de Desempenho de Domínio.
- Método **EnviarMsgUrgente**: Permite o envio de mensagens diretamente ao Gerente de Desempenho de Domínio. É usado para o envio de Alarmes de Tendência.

4.2.5. Implementação do Inspetor de Desempenho

As diferentes monitorações que devem ser realizadas por um Inspetor de Desempenho em um determinado elemento gerenciado devem ser implementadas em diferentes *threads*, de forma a simplificar os controles dos tempos de intervalo entre a obtenção de observações consecutivas dos valores dos parâmetros de controle. Isso significa que deve haver uma *thread* para o monitoramento de cada parâmetro de controle. Estas devem ser as *threads* de maior prioridade do Inspetor de Desempenho. Tarefas como armazenamento de dados em memória secundária e envio de mensagens de rotina podem ser realizadas por *threads* de menor prioridade.

As diferentes *threads* do Inspetor de Desempenho que realizam o monitoramento dos diferentes parâmetros de controle devem ter seu código desenvolvido especificamente para o comportamento do parâmetro em questão, visando minimizar o seu tempo de execução. O exemplo já apresentado na seção 3.2.1 continuará a ser usado nesta seção, para visualização da implementação do protótipo.

Os parâmetros passados para o construtor do Inspetor de Desempenho incluem aqueles que foram explicados na seção 3.2.1, (i) o limite mínimo para o envio de Alarmes (*ThresholdValue*), (ii) a variação máxima tolerada em um intervalo entre medições sucessivas (*MaxOneSleepVariationLimit*) e (iii) a variação acumulada máxima tolerada (*MaxAccumulatedVariationLimit*) e acrescenta outros valores relevantes: (iv) o intervalo a ser aguardado entre a obtenção de dois valores consecutivos do parâmetro monitorado (*SleepInterval*) e (v) uma ordem para que seja determinado se o Inspetor de Desempenho já deve iniciar sua execução realizando a monitoração (*BeginMonitoring*).

O diagrama da classe Inspetor de Desempenho é apresentada na Figura 20.

```
class PerformanceInspector extends Thread {
   public PerformanceInspector (    String Name,
                    int ThresholdValue,
                    int MaxOneSleepVariationLimit,
                    int MaxAccumulatedVariationLimit,
                    int SleepInterval,
                    boolean BeginMonitoring )
   public void BeginMonitor ( boolean ToMonitor )
   public void SetLImits ( int Value, int Value, int Value, int Value )
   public void Finalize ( void )
   public String[ ] GetInfo ( int InfoType )
   private void SendTendencyAlarm( String Alarm )
   private Register( )
   public void run ( )
}
```

Figura 20 - Diagrama da classe do Inspetor de Desempenho

Os diversos métodos permitem que a execução do Inspetor de Desempenho seja controlada e configurada pelo Gerente de Desempenho de Domínio. A parada ou a retomada da monitoração podem ser comandadas pelo método ***ToMonitor***. Os limites citados no parágrafo anterior podem ser alterados com o uso do método ***SetLimits***. A execução do Inspetor de Desempenho pode ser encerrada com o uso do método ***Finalize***. Essa parada é usada quando do encerramento das atividades do Inspetor de Desempenho em questão ou quando da substituição do seu código por uma nova versão. O Gerente de Desempenho de Domínio pode solicitar qualquer tipo de informação adicional, cuja obtenção foi previamente programada no Inspetor de Desempenho, com o uso do método ***GetInfo***.

A implementação do Inspetor de Desempenho do exemplo já citado, ou seja, o código presente no método ***run*** do diagrama de classe apresentado na Figura 20, é apresentada na forma de pseudo-código na Figura 21.

As variáveis iniciadas com "*Priv*" são atualizadas pelo método ***SetLimits***, e são elas quem armazenam, de forma privativa, os limites que eu podem ser configurados para a monitoração.

O tempo gasto na execução de um laço de monitoração é descontado do intervalo que deve ser aguardado para a realização do próximo laço. Esse tempo pode envolver o acesso a dispositivos

cujo acesso seja bem mais demorado do que o processamento realizado na CPU, como, por exemplo, o acesso a MIBs e/ou a dispositivos de memória secundária. Adicionalmente, todas as informações relevantes para posterior utilização em aplicações que venham a analisar o desempenho do parâmetro monitorado são armazenadas: o número da observação, o valor obtido, a variação e o tipo de alarme que foi desencadeado. Uma vez que o código do Inspetor de Desempenho é desenvolvido especificamente para o parâmetro que é por ele monitorado, o tempo de acesso ao dispositivo de armazenamento pode ser bem conhecido, de forma que o intervalo entre observações possa ser precisamente programado.

```
// controls first loop execution: no variation need to be calculated
Beginning = true
LoopCount = 1
// PrivToMonito controls starting and stopping of monitoring
while ( PrivToMonitor )
   InitialTime = get system time of the beginning of the loop
   AlarmSent = null
   Get UpdatedValue for monitored parameter
   if ( Beginning )
      // starts reference values because it´s the first time loop runs
      LastValue = UpdatedValue
      UpdatedVariation = 0
      CeilingValue = UpdatedValue
      Beginning = false
   else
      // calculates variation
      UpdatedVariation = LastValue – UpdatedValue
      // verifies if ceiling value has to be changed
      if ( UpdatedValue > CeilingValue )
         CeilingValue = UpdatedValue
      else
         // verifies if new value is worts than last value
         if ( UpdatedValue < LastValue )
            // verifies if variation is greater than limit for 1 interval
            if ( UpdatedVariation >= PrivMaxOneSleepVariatiLimit )
               CeilingValue = UpdatedValue
               // verifies if threshold to send alarm was reached
               if ( UpdatedValue < PrivThresholdValue )
                  AlarmSent = max one sleep limit exceeded
                  sends tendency alarm
            else
               // verifies if variation is greater than limit of acumm var
               if ( (CeilingValue - UpdatedValue) >= PrivMaxVariationLimit )
                  CeilingValue = UpdatedValue
                  // verifies if threshold to send alarm was reached
                  if ( UpdatedValue < PrivThresholdValue )
                     AlarmSent = max accumulated variation exceeded
                     Sends tendency alarm
   LastValue = UpdatedValue
   stores LoopCount, UpdatedValue, UpdatedVariation, AlarmSent
   FinalTime = get system time
   LoopTime = FinalTime - InitTime
   if ( LoopTime < PrivSleepInterv )
      Thread.sleep( PrivSleepInterv - LoopTime )
   if ( ContinuesRunning == null )
      PrivToMonitor = false
      LoopCount++
```

Figura 21 - Pseudo-código de um Inspetor de Desempenho

4.2.6. Considerações sobre a implementação

A implementação do protótipo contempla todas as facilidades previstas na AGAD [19], das quais, as principais são a utilização da programação *multithreading* e a possibilidade de atualização de componentes de software em tempo real.

O Inspetor de Desempenho é implementado na forma de *thread*, sendo que uma *thread* deve ser didicada à monitoração de cada parâmetro de controle. A comunicação do Inspetor de Desempenho com o Gerente de Desempenho de Domínio para o envio de Alarmes de Tendência se dá via socket, diretamente, e é realizada, no Inspetor de Desempenho, por intermédio de uma *thread* dedicada a essa função, que é chamada, de forma sincronizada, pela *thread* que realiza a monitoração. Tal solução se faz necessária porque podem existir mais de uma *thread* de monitoração, quando houver mais de um parâmetro a ser monitorado no mesmo elemento gerenciado.

A atualização de componentes em tempo real é implementada de forma que seja possível o envio, a partir do Gerente de Desempenho de Domínio, um novo código para um método ou para uma classe que estejam sendo executados (intanciados). Para isso, há um método denominado Registrador em cada elemento da AGAD que implementa (por ocasião da instanciação) a Interface para cada método presente no elemento. O novo código, ao ser recebido pelo μServer, tratará de comandar o encerramento oportuno do objeto a ser substituído e se registrará, assumindo então o lugar do objeto substituído.

4.3. Resumo do capítulo

Neste capítulo foram apresentados o ambiente de desenvolvimento utilizado e a implementação do protótipo que foi realizada. O ambiente é todo baseado tanto em softwares gratuitos quanto na linguagem Java. A infra-estrutura de mobilidade e a interface de programação de aplicações de gerenciamento que foram utilizadas também são baseadas na linguagem Java. O protótipo implementado consistiu de um Gerente de Desempenho de Domínio, um Inspetor de Desempenho e um Especialista. A implementação do protótipo foi realizada com o uso da análise e projeto orientados a objeto e os diagramas referentes ao desenvolvimento foram apresentados via UML.

Capítulo 5 - Testes e análise dos resultados obtidos

Os testes realizados tinham como cenário de referência um domínio (da AGAD) onde é executada uma aplicação multimídia distribuída como o ServiMídia [28]. Este cenário representa a utilização do gerenciamento de desempenho pró-ativo que foi descrita na seção 3.4.1, ou seja, no gerenciamento de aplicações.

O objetivo geral dos testes é determinar se o desempenho de um sistema de gerenciamento pró-ativo baseado em uma linguagem interpretada viabiliza tanto a monitoração de parâmetros de QoS em tempo real, quanto o desencadeamento de ações que venham a, prever e, até, se possível, evitar a ocorrência de falhas de QoS e/ou minimizar os seus efeitos para a QoP.

Inicialmente, na seção 5.1, será descrita a metodologia que foi utilizada na realização dos testes. Na seção 5.2 serão descritos os testes que foram realizados com o protótipo, os objetivos desses testes e os resultados obtidos. Esses resultados serão analisados na seção 5.3. Por fim, na seção 5.4 serão apresentadas as considerações finais do capítulo.

5.1. Metodologia de testes

Os testes consistiram, quando necessário, na execução de cinco rodadas do evento em questão, cada uma delas com pelo menos trinta experiências, para cada plataforma considerada. Foram calculados, para cada rodada, a média, o desvio padrão, o intervalo de confiança de 95% e grau de ajuste à distribuição normal, expresso percentualmente e calculado pelo teste Qui-quadrado [52]. Tal teste foi realizado com o auxílio do software Satistica [55] e teve por finalidade verificar se os valores coletados eram suficientes, tanto em quantidade quanto em dispersão, para que a média obtida fosse considerada válida. É também calculada a média das cinco médias (das rodadas) para cada plataforma em questão. Esse valor será utilizado na análise dos resultados como uma forma de se associar uma única medida, por plataforma, para cada teste.

5.2. Testes realizados e resultados obtidos

Foram realizados três tipos de testes com o protótipo de Gerenciamento de Desempenho Pró-Ativo que foi implementado: (i) execução isolada do Inspetor de Desempenho com o envio de Alarmes, (ii) desencadeamento de uma ação por um Especialista após o envio de um Alarme de Tendência pelo Inspetor de Desempenho e (iii) monitoramento do parâmetro de controle retardo ao longo de um intervalo de tempo em uma rede real.

5.2.1. Testes do Inspetor de Desempenho isolado

O objetivo específico do primeiro tipo de teste é a obtenção de dados referentes ao desempenho do funcionamento do próprio Inspetor de Desempenho em plataformas com diferentes capacida-

des de processamento. Com esses dados pode-se inferir qual é a freqüência máxima de monitoramento de parâmetros de controle que pode ser utilizada e qual é a capacidade de processamento necessária para essa monitoração.

Esse teste consistiu na determinação de quanto tempo leva a execução de um laço completo de monitoramento pelo Inspetor de Desempenho. As plataformas que foram utilizadas estão apresentadas na Tabela 1.

Tabela 1 - Plataformas de hardware utilizadas nos testes isolados

Plataforma	CPU	Clock (MHz)	Cache (KBytes)	RAM (MBytes)
1	AMD K6-II	475	64	128
2	AMD Athlon	1333	256	256

Em todas as experiências foi colocado em execução, em nível usuário, apenas o processo referente ao Inspetor de Desempenho. Uma vez que a utilização (daquelas sugeridas na Seção 3.4) escolhida para que o gerenciamento aqui proposto fosse testado é o trabalho em prol de uma aplicação multimídia distribuída, onde é bem provável a utilização de um ambiente gráfico para a interface com o usuário, o ambiente de área de trabalho (KDE) foi deixado em execução em todos os testes. A funcionalidade de *garbage collection* do Java não teve a sua configuração alterada.

A execução do laço de monitoramento do Inspetor de Desempenho deve ser feita da forma mais rápida possível, de forma que não só seja permitida uma maior freqüência de monitoramento, como também seja consumida pouca capacidade de processamento do elemento gerenciado onde o Inspetor de Desempenho é executado. Uma execução típica do laço de monitoramento envolve as seguintes fases:

- Obtenção de uma observação do valor do parâmetro monitorado. Essa observação pode ser obtida de três formas: (i) diretamente do sistema operacional, (ii) diretamente de uma aplicação, como é o caso quando se usa RTCP, ou, ainda, (iii) de daemons do sistema operacional cujo acesso se dá via pilha de protocolos, como no caso de um agente SNMP. A terceira opção foi a utilizada neste teste.
- Execução de um trecho de programa com cálculos e comparações simples, cujo pseudo-código foi apresentado na Figura 21.
- Envio de um Alarme de Tendência, caso seja necessário, via pilha de protocolos do sistema operacional.

- Armazenamento em memória secundária do valor da observação; da variação da mesma em relação ao valor da observação anterior; de uma estampa de tempo e do tipo de alarme que foi enviado.

O laço de monitoramento que foi testado foi programado para representar o pior caso em termos de tempo de execução, pois vai desde a obtenção do valor da observação do parâmetro monitorado a partir de um agente SNMP até o envio de um Alarme de Tendência e o conseqüente armazenamento de dados em memória secundária. A obtenção do valor da observação via SNMP é uma das formas mais demoradas de se fazê-lo, pois envolve o preparo de uma PDU de requisição e a transmissão da mesma via UDP, com a participação do sistema operacional. A resposta também é recebida via UDP. Uma chamada ao sistema operacional ou uma interação direta com uma aplicação certamente levaria menos tempo. Foram configurados valores para os limites (descritos na Seção 3.2.1) que levaram, neste teste, à ocorrência de Alarmes em todas as experiências de cada rodada.

Os resultados obtidos estão apresentados na Tabela 2. Cabe destacar que, em função dos valores observados na execução do teste na pior plataforma serem próximo a 10 ms, optou-se por medir o tempo, em cada experiência, para a realização de 200 laços e dividir o valor encontrado por 200 para se chegar ao valor apresentado na tabela.

Em uma primeira realização do teste de cinco rodadas com trinta experiências na plataforma Athlon de 1,3 MHz, ao se tentar realizar o ajuste dos resultados obtidos à curva normal pelo teste do Qui-quadrado, não se obteve sucesso, uma vez que o desvio padrão era demasiadamente grande. O teste foi então repetido com 41 experiências, quando então foi possível o ajuste, conforme pode ser observado pelos valores do coeficiente de ajuste à normal apresentados na Tabela 2. Já no caso da plataforma AMD K6 de 475 MHz, obteve-se o ajuste com apenas trinta experiências, por isso, o teste não foi repetido.

A primeira experiência de cada rodada de cada teste sempre apresenta um valor excessivamente alto em relação às experiências posteriores. Tal fato se deve às instanciações de objetos e do carregamento de variáveis em memória que são realizados pela máquina virtual Java quando do início da execução dos programas. Uma vez que o principal objetivo deste teste é a obtenção de dados referentes à execução do Inspetor de Desempenho, ou seja, já inicializado, os valores dessas primeiras experiências, embora apresentados, serão omitidos tanto dos cálculos da média, do desvio padrão e do intervalo de confiança de cada rodada.

Tabela 2 – Execução completa de um laço de monitoração com envio de alarme

	AMD K6 475 MHz					Athlon 1,3 MHz				
Rodadas	**1**	**2**	**3**	**4**	**5**	**1**	**2**	**3**	**4**	**5**
	13,20	14,74	12,94	16,34	16,04	3,63	2,17	2,83	6,23	4,28
1ª	10,02	10,07	9,97	10,09	9,97	0,13	0,07	0,92	0,27	0,12
2ª	10,02	9,97	10,02	9,97	9,98	0,08	0,07	1,12	0,42	0,42
3ª	9,97	10,02	10,02	9,97	10,07	0,18	0,07	1,92	0,77	0,17
4ª	10,01	9,97	10,01	10,02	10,02	0,30	0,27	1,17	0,07	0,52
5ª	10,02	10,02	9,97	10,02	10,02	0,23	0,32	2,02	0,07	0,12
6ª	10,02	10,02	10,02	10,02	9,97	0,58	0,92	2,57	0,62	0,07
7ª	9,97	10,01	10,02	9,97	10,02	0,08	0,07	1,98	0,22	0,07
8ª	10,02	10,02	9,97	10,02	10,02	0,88	0,07	0,82	0,52	0,07
9ª	10,02	10,03	10,02	10,02	10,12	0,18	0,07	1,87	0,12	1,07
10ª	10,02	10,02	10,02	10,02	10,02	4,08	0,17	2,93	1,32	0,22
11ª	10,02	9,97	9,97	10,02	10,02	1,13	0,07	1,97	0,87	0,07
12ª	10,01	10,02	9,97	9,97	10,02	0,18	0,62	1,17	1,82	0,92
13ª	10,02	9,97	10,02	10,02	10,02	1,33	0,27	0,92	0,42	0,07
14ª	10,02	10,02	9,97	9,97	10,02	0,23	0,42	0,82	1,57	0,07
15ª	10,01	10,02	10,02	10,02	9,97	0,83	0,62	1,17	0,27	0,22
16ª	9,97	9,97	10,02	10,07	10,02	1,93	0,07	1,27	0,72	0,07
17ª	9,97	10,02	9,97	10,02	9,97	0,33	0,27	0,77	0,82	0,07
18ª	10,77	10,77	10,77	10,98	10,77	0,43	0,27	1,47	0,17	0,07
19ª	10,02	10,02	10,02	10,02	9,97	0,93	0,17	1,12	0,97	0,07
20ª	9,97	10,01	9,97	10,02	10,02	0,28	0,07	1,07	0,47	0,07
21ª	9,97	10,02	9,97	10,02	10,02	0,03	0,07	0,97	0,82	0,07
22ª	10,02	9,97	10,02	10,37	10,02	0,18	0,12	1,52	1,82	0,07
23ª	10,02	10,02	10,02	10,02	9,97	0,18	0,07	2,67	3,63	0,07
24ª	9,97	9,97	10,02	10,02	10,17	0,03	0,22	2,17	0,24	0,07
25ª	10,02	10,02	9,97	10,02	10,22	0,03	0,07	1,77	9,43	0,17
26ª	10,02	9,97	10,02	10,02	10,02	0,78	0,07	2,93	12,39	0,07
27ª	10,02	10,02	9,97	10,02	9,97	1,43	0,07	1,17	8,68	0,17
28ª	9,97	9,97	10,02	9,97	10,02	0,28	3,18	2,38	9,23	0,32
29ª	9,97	10,02	10,02	10,02	9,97	0,24	3,32	1,67	10,23	0,07
30ª	10,02	9,97	10,02	10,02	9,97	0,69	3,13	2,02	13,99	0,27
31ª	- -	- -	- -	- -	- -	0,49	1,52	2,62	1,44	0,12
32ª	- -	- -	- -	- -	- -	1,04	0,22	3,17	2,49	0,22
33ª	- -	- -	- -	- -	- -	1,64	0,32	2,93	0,88	0,17
34ª	- -	- -	- -	- -	- -	0,59	0,12	4,83	2,99	0,07
35ª	- -	- -	- -	- -	- -	0,29	0,12	7,63	3,29	0,17
36ª	- -	- -	- -	- -	- -	0,45	0,07	7,03	2,93	0,07
37ª	- -	- -	- -	- -	- -	1,04	0,07	7,28	3,98	0,12
38ª	- -	- -	- -	- -	- -	0,19	0,07	5,98	1,58	0,07
39ª	- -	- -	- -	- -	- -	0,04	0,12	2,07	3,29	0,12
40ª	- -	- -	- -	- -	- -	0,04	0,07	0,82	4,24	0,17
41ª	- -	- -	- -	- -	- -	0,13	0,27	2,83	0,27	0,17
Média da rodada	10,03	10,03	10,03	10,06	10,04	0,67	0,49	2,33	2,83	0,28
Média da plataforma	10,04					1,32				
Grau ajuste à normal (%)	99,99	99,99	99,99	100,0	99,99	100,0	100,0	99,9	100,0	100,0
Desvio padrão	0,14	0,14	0,14	0,19	0,15	0,87	0,88	1,77	3,63	0,67
Int. de confiança de 95%	±0,05	±0,05	±0,05	±0,07	±0,05	±0,27	±0,27	±0,55	±1,12	±0,21

Obs: tempos em milissegundos

5.2.2. Testes de desencadeamento de ação via Especialista

O objetivo específico desse teste é a obtenção de dados referentes ao desempenho da interação do Inspetor de Desempenho com o Gerente de Desempenho de Domínio utilizando-se não só do ambiente Java como também de uma infra-estrutura de mobilidade, além da própria rede. Com esses dados pode se saber qual é a latência mínima que deve ser esperada do sistema de gerenciamento pró-ativo. Essa latência tem que ser considerada para a escolha das ações que devem ser desencadeadas pelos Especialistas.

Estes testes foram realizados a partir da execução do Inspetor de Desempenho e do Gerente de Desempenho sendo executados em estações diferentes, sendo ambas plataformas AMD K6-II de 500 Mhz, com 256 MB de memória RAM e 128 KB de memória cache, interligadas via rede. O Gerente Mor iniciava todo o processo, a partir de uma terceira estação, cuja plataforma era a mesma. Um esquema simplificado do teste, onde são mostradas as infra-estruturas que são utilizadas, é apresentado na Figura 22.

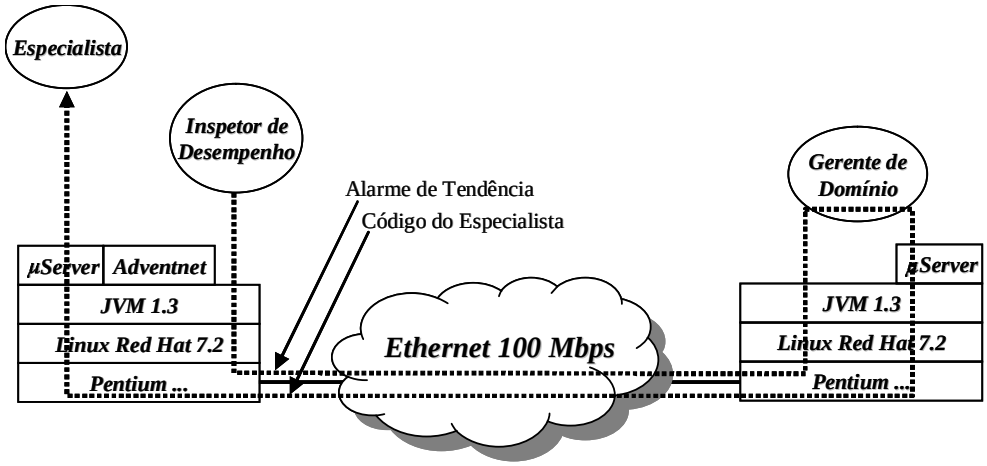


Figura 22 - Esquema simplificado da interação Insp. de Desempenho – Ger. de Desempenho

Dois foram os tempos medidos nos testes: (i) o tempo total, desde a decisão, pelo Inspetor de Desempenho, do envio de um Alarme de Tendência, até o fim da instanciação de um Especialista, na mesma estação do Inspetor de Desempenho, e (ii) o tempo entre o recebimento do Alarme pelo Gerente de Desempenho de Domínio e o envio do código do Especialista. O tempo desse teste ii está inserido no tempo do teste i e deve-se à análise do Alarme recebido, ao processamento da infra-estrutura de mobilidade e à recuperação e envio do código do Especialista.

O envio do Especialista de Desempenho pode necessitar de mais do que uma única transmissão via rede, diferentemente do que está sendo sugerido pela Figura 22. O Gerente de Desempenho do Domínio envia primeiro uma notificação com o *Uniform Resource Locator* (URL) que identifica o código (do Especialista) que deve ser carregado na estação do Inspetor de Desempenho. O *class loader* do µServer, na estação do Inspetor de Desempenho, ao receber essa notificação, verifica se o código que deve ser enviado já está presente nos espaços de classes locais. Caso não esteja, o mesmo é pedido para o µServer do Gerente de Desempenho de Domínio, que então o envia.

Uma situação alternativa, menos demorada, é aquela onde o código de um Especialista já esteja presente no elemento gerenciado. Nesse caso, não será necessária a transferência do código, reduzindo a latência até o início de sua execução. Caso um Especialista venha a ser executado com freqüência, poucas dessas transferências serão necessárias. Essa é a situação que tem a

maior probabilidade de acontecer, pois não é esperado que o código dos Especialistas seja atualizado com freqüência. Ambas as situações foram testadas.

Foram realizados então três subtestes de desencadeamento da ação via Especialistas.

No primeiro subteste, todo o sistema foi reiniciado antes de cada uma das trinta experiências, para que o código do Especialista fosse sempre transferido. Esse código consistia de um arquivo de *bytecodes* com 1483 bytes. Os tempos obtidos, (i) total, e (ii) de processamento no Gerente de Desempenho de Domínio, estão apresentados na Tabela 3.

Tabela 3 - Resultados do teste com reinicialização e Especialista de 1 KB

Tempos:	***(i)***	***(ii)***
1ª	137	12
2ª	159	8
3ª	133	9
4ª	132	10
5ª	148	14
6ª	137	14
7ª	131	9
8ª	134	11
9ª	161	11
10ª	139	16
11ª	137	14
12ª	135	12
13ª	133	10
14ª	138	15
15ª	135	11
16ª	140	13
17ª	133	10
18ª	132	9
19ª	134	10
20ª	140	16
21ª	134	10
22ª	133	10
23ª	131	8
24ª	133	9
25ª	132	9
26ª	133	9
27ª	139	15
28ª	133	9
29ª	135	11
30ª	133	9
Média	136,80	11,10
Grau ajuste à normal (%)	99,99	95,27
Desvio padrão	7,24	2,43
Int de conf 95%	±2,59	±0,87

Obs: tempos em milissegundos

O segundo subteste consistiu do mesmo experimento, no entanto, o sistema não foi reinicializado. Nesse caso, o código do Especialista (o mesmo do subteste anterior, de 1 KB) permanecia em memória cache, evitando a transmissão do pedido para o seu envio e a transferência do seu próprio *bytecode*, exceto na primeira execução.

Os tempos obtidos, (i) total, e (ii) de processamento no Gerente de Desempenho de Domínio, estão apresentados na Tabela 4.

Tabela 4 - Resultados do teste sem reinicialização do sistema e Especialista de 1 KB

Tempos:	***(i)***	***(ii)***
1ª	38	8
2ª	40	6
3ª	37	13
4ª	37	5
5ª	40	8
6ª	42	9
7ª	51	9
8ª	41	9
9ª	44	12
10ª	46	15
11ª	63	6
12ª	37	7
13ª	43	10
14ª	49	9
15ª	41	10
16ª	41	9
17ª	38	8
18ª	38	7
19ª	40	8
20ª	41	10
21ª	43	12
22ª	38	14
23ª	45	8
24ª	39	8
25ª	38	8
26ª	41	13
27ª	47	9
28ª	43	9
29ª	47	8
30ª	39	12
Média	42,23	9,30
Grau ajuste à normal (%)	95,48	99,79
Desvio padrão	5,39	2,42
Int de conf 95%	±1,93	±0,87

Obs: tempos em milissegundos

O terceiro subteste realizado consistia na repetição do primeiro teste (com reinicialização) porém, com o código de Especialista tendo, em *bytecodes*, 10488 bytes. O objetivo desse teste era verificar qual a conseqüência no aumento do tempo total do experimento com um Especialista 10 vezes maior. Os resultados obtidos estão apresentados na Tabela 5.

Tabela 5 - Resultados do teste com reinicialização do sistema e Especialista de 10 KB

Tempos:	(i)	(ii)
1ª	433	9
2ª	438	14
3ª	430	10
4ª	433	13
5ª	436	10
6ª	436	13
7ª	431	14
8ª	435	9
9ª	427	11
10ª	457	10
11ª	429	12
12ª	428	9
13ª	433	12
14ª	434	11
15ª	427	9
16ª	434	7
17ª	427	12
18ª	438	10
19ª	429	8
20ª	435	12
21ª	431	8
22ª	430	9
23ª	426	14
24ª	434	7
25ª	430	14
26ª	430	10
27ª	434	13
28ª	434	12
29ª	448	13
30ª	435	11
Média	433,40	10,87
Grau ajuste à normal (%)	95,30	
Desvio padrão	6,27	2,13
Int de conf 95%	±2,24	±0,76

Obs: tempos em milissegundos

5.2.3. Monitoramento do parâmetro de controle retardo

O terceiro teste tem como objetivo específico realizar o monitoramento do parâmetro de controle retardo em uma aplicação executada em um ambiente real e verificar quando e por que são gerados Alarmes de Tendência. A análise dos resultados desse teste permite a inferência de conclusões a respeito não só da eficácia do gerenciamento pró-ativo na previsão de falhas de QoS como também sobre o tempo disponível para o desencadeamento de ações, ou seja, o tempo que sobrará entre a instanciação do Especialista até a (possível) ocorrência da falha de QoS. Esse será o tempo que o Especialista terá para desencadear suas ações ou para a aplicação realizar adaptações.

Esse teste consistiu na execução do Inspetor de Desempenho sobre um conjunto de valores observados do parâmetro de controle retardo, que foram obtidos ao longo de um intervalo de duas horas, durante o pico de utilização, em um dia normal de trabalho, da rede do Instituto Militar de Engenharia (IME). O prédio do IME, que é uma faculdade de Engenharia com 11 cursos de gra-

duação organizados em 7 departamentos de ensino independentes, é coberto por uma rede com cerca de 45 segmentos, todos baseados em Ethernet, e 500 estações de trabalho. Esse é um cenário típico não só para a utilização de uma aplicação multimídia distribuída como o ServiMídia, como também para um domínio de gerenciamento distribuído conforme o que é revisto pela AGAD.

A topologia da rede, detalhada apenas em seus pontos relevantes para o teste que foi realizado, está apresentada na Figura 23.

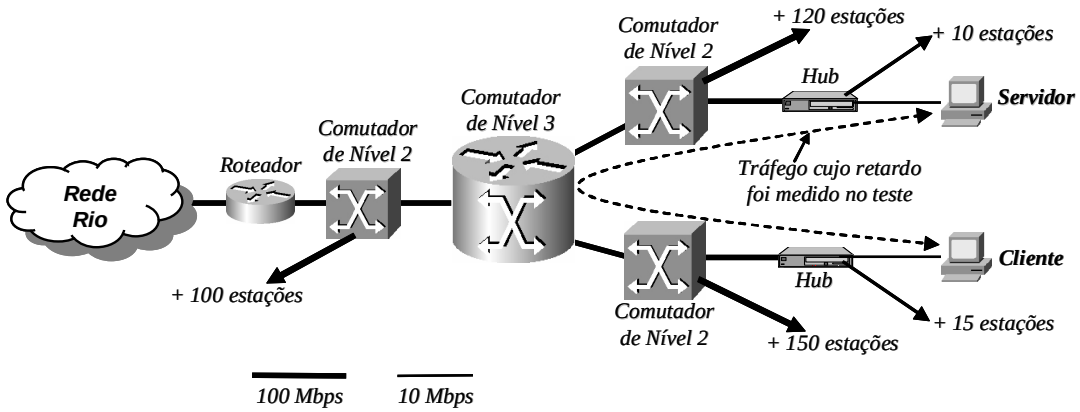
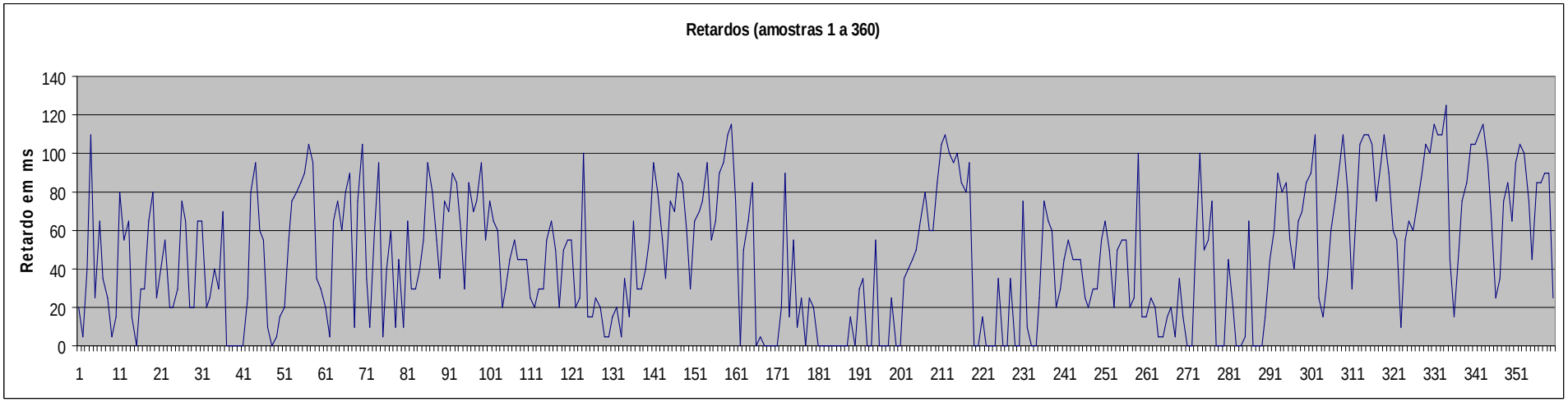


Figura 23 - Topologia da rede utilizada para obtenção do comportamento da variação do retardo

Cabe destacar que atualmente, não só na rede do IME, mas também na maioria das redes, mesmo de faculdades e universidades, ainda não são utilizadas aplicações multimídia distribuídas. Visando obter maior realismo na obtenção dos dados referentes ao retardo, foi inserida a transmissão de um fluxo de vídeo codificado em MPEG-1, com duas horas de duração e com taxa de 400 Kbps, do servidor para o cliente.

Os dados referentes ao retardo foram obtidos com o uso de uma aplicação cliente-servidor também programada em Java, para que o retardo equivalente ao processamento da JVM fosse considerado. O cliente enviava pacotes UDP a cada cinco segundos para o servidor, que os enviava de volta. O retardo então era medido e dividido pela metade.

A medição do retardo foi realizada durante três dias consecutivos, sendo então selecionada uma amostra típica com vários picos de retardo e considerável variação, conforme pode ser observado na Figura 24.



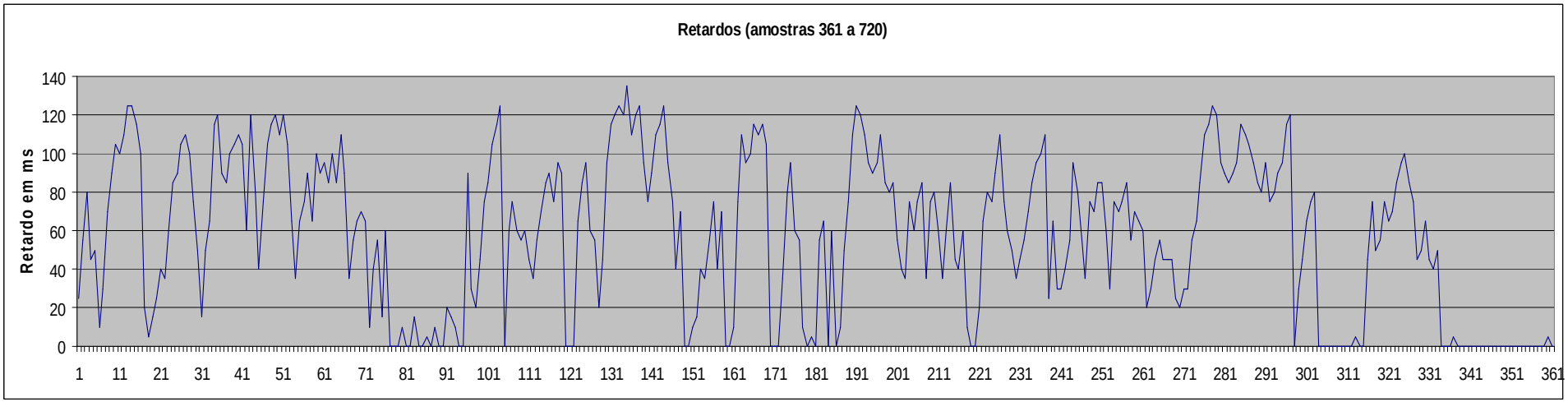


Figura 24 – Amostra da variação do retardo que foi utilizada no teste

O Inspetor de Desempenho foi então executado sobre essa amostra, para que, quando fosse o caso, Alarmes de Tendência fossem detectados. Os seguintes valores foram considerados como limites para a configuração do Inspetor de Desempenho:

- 65 ms como limite mínimo do retardo a partir do qual Alarmes devem ser gerados,
- 35 ms como variação máxima entre duas observações consecutivas;
- 40 ms como variação máxima acumulada.

Foi considerado que se o retardo chegasse a 90 ms ocorreria uma falha de QoS.

Cabe destacar que a lógica de escolha destes valores não é trivial e depende de fatores como, por exemplo, o tamanho dos *buffers* de recepção das aplicações multimídia e da freqüência com a qual se pretende realizar a monitoração do retardo. Uma vez que o foco dessa proposta é focada no mecanismo de gerenciamento de desempenho propriamente dito, esse problema foi deixado fora do escopo da mesma e será apontado como um dos possíveis trabalhos.

Os Alarmes que foram detectados na amostra de variação do retardo da Figura 24 e os Alarmes de Tendência que foram detectados estão apresentados no gráfico da Figura 25. Os triângulos representam os Alarmes, sendo que, quando na ordenada 15, o alarme foi gerado por desrespeito ao limite máximo de variação para duas amostras consecutivas (35 ms) e, quando na ordenada 8, o alarme foi gerado por desrespeito à variação máxima acumulada (40 ms).

Todos os valores do retardo (Ret, na tabela) de cada amostra (Am, na tabela) e os Alarmes (Al, na tabela) que foram gerados a partir da execução do Inspetor de Desempenho, estão apresentados no Anexo A.

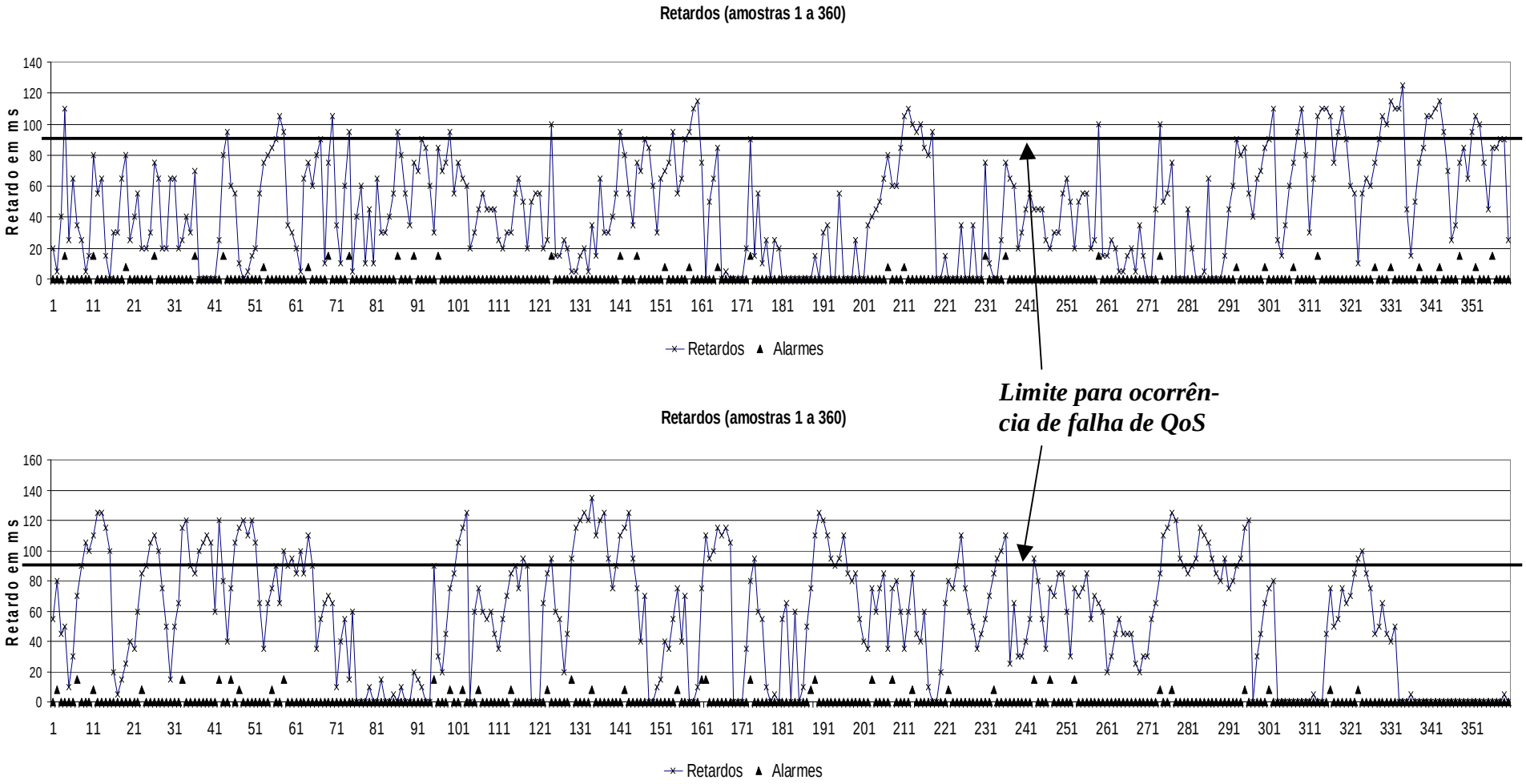


Figura 25 – Alarmes gerados na amostra de variação de retardo

5.3. Análise dos resultados

Nesta seção serão analisados, inicialmente, os resultados de cada um dos testes separadamente. Em seguida, serão realizadas considerações sobre os testes como um todo. Em ambos os casos, as análises serão feitas com o objetivo de se analisar a eficiência, a eficácia, a viabilidade e as limitações do gerenciamento de desempenho pró-ativo que é proposto nesta dissertação.

5.3.1. Inspetor de Desempenho isolado

A necessidade de se executar o Inspetor de Desempenho com alta freqüência, dependendo do parâmetro de QoS que se deseja monitorar, impõe o conhecimento de quanto tempo leva a execução completa de um laço do mesmo. Essa execução completa envolve, além dos cálculos referentes aos valores dos limites máximos (ou mínimos) tolerados, o acesso a outros processos, quer sejam do sistema operacional, para a obtenção de referências de tempo, quer sejam de outras aplicações, como, por exemplo, um agente SNMP sendo executado na mesma máquina. Adicionalmente, o registro dos cálculos referentes àquela iteração devem ser armazenados, normalmente, em unidades de disco. Conforme pode ser observado na Tabela 2, o tempo de execução completo do Inspetor de Desempenho varia, na plataforma mais fraca, de 9,97 até 10,77 milissegundos, incluindo o envio de um alarme e o armazenamento de dados.

Caso seja necessário, é possível se diminuir ainda mais o tempo de execução de um laço do Inspetor de Desempenho, através da utilização de *threads* de menor prioridade para a realização de tarefas relativas ao registro de dados em disco. Ainda, dependendo do parâmetro a ser monitorado, a forma de obtenção do valor do mesmo pode colaborar para a melhora do desempenho do Inspetor de Desempenho, uma vez que no caso testado, essa obtenção envolvia o acesso a uma MIB, via agente SNMP. O esquema de *garbage collection* da JVM também pode ser alterado, através de cuidadosa programação do Inspetor de Desempenho, de forma a diminuir o tempo de execução desse último.

Em função do desempenho apresentado, pode-se afirmar que é viável a utilização de um Inspetor de Desempenho programado em Java para a monitoração de valores de parâmetros de QoS em temp real. A determinação da freqüência de monitoração do Inspetor de Desempenho deverá ser limitada apenas pelo tipo do parâmetro que se deseja monitorar e não pela capacidade de processamento do elemento gerenciado, caso essa seja parecida ou superior àquela da pior plataforma testada.

Cabe destacar que a plataforma que apresentou o pior resultado possuía capacidade de processamento que era quase insuficiente para a reprodução do vídeo que foi transmitido durante a captura dos valores do retardo. Na plataforma 2 (Athlon 1,3 GHz), o tempo de execução do laço

do Inspetor de Desempenho caiu para menos do que 2 milissegundos, algo que permitiria, caso o tempo de execução do Inspetor de Desempenho fosse o único fator limitante, uma freqüência de monitoração do parâmetro em questão de até cerca de 30 Hz (30 observações por segundo).

5.3.2. Interação entre Inspetor de Desempenho e Gerente de Desempenho de Domínio

O teste compreendeu a medição do intervalo de tempo decorrente desde a tomada de decisão, pelo Inspetor de Desempenho, após o cálculo da variação do valor observado do parâmetro em questão em relação à observação anterior, de enviar um Alarme de Tendência até o final da instanciação de um Especialista nesse mesmo elemento gerenciado. Esse intervalo de tempo é decomposto nos processamentos e transmissões mostrados na Figura 26.

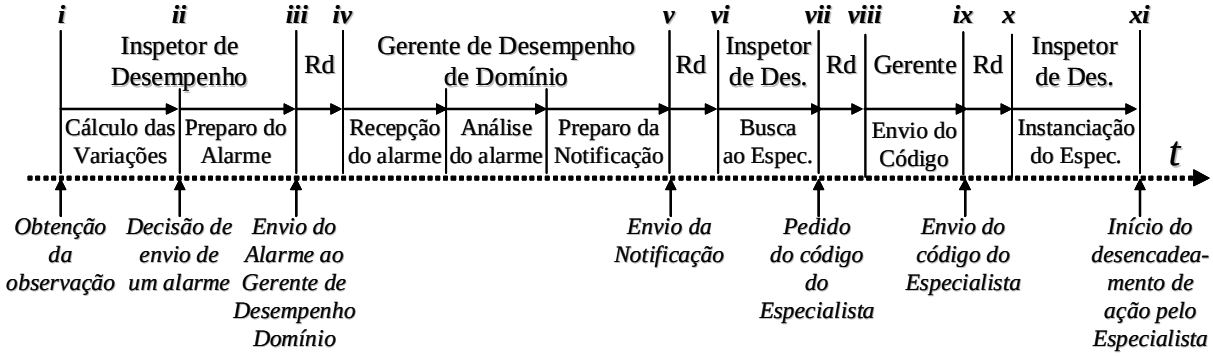


Figura 26 - Etapas entre a decisão de envio de Alarme de Tendência e início de execução do Especialista

O detalhamento das etapas mostradas nesta figura, com a descrição de cada processamento e de cada transmissão, é apresentado na Tabela 7. Os eventos numerados de *i* a *xi* na figura são usados como referência.

Os resultados que foram obtidos na execução desse teste podem ser resumidos de acordo com o que está apresentado na Tabela 6.

Tabela 6 - Resultados da interação entre Insp. de Desempenho e o Ger. de Desempenho

	Com reinicialização para cada experimento, Especialista de 1 KB	Sem reinicialização para cada experimento, Especialista de 1 KB	Com reinicialização para cada experimento, Especialista de 10 KB
Tempo total, entre os eventos (ii) e (xi).	**136,8 ms**	**42,23 ms**	**433,4 ms**
Tempo total, entre os eventos (iv) e (v).	**11,1 ms**	**9,30 ms**	**11,57 ms**

O tempo de análise do Alarme de Tendência no Gerente de Desempenho de Domínio pode ser grande, em função da complexidade das técnicas que forem empregadas para a sua realização. No entanto, o Gerente de Domínio (e, conseqüentemente, o Gerente de Desempenho de Domínio) pode ser instalado em uma estação com elevada capacidade de processamento, de forma que esse tempo de processamento venha a ser reduzido.

Tabela 7 – Detalhamento das etapas apresentadas na Figura 26

Evento	Processamento relacionado ao evento
i. Obtenção de uma observação do parâmetro de controle.	Cálculo das variações em um intervalo de tempo e acumulada. Comparação do valor da observação e das variações com os limites configurados.
ii. Decisão por envio de um Alarme de Tendência (início da medição).	Preparo do Alarme para transmissão via μServer.
iii. Envio do alarme via socket	Propagação via rede.
iv. Recepção do alarme pela estação do Gerente de Desempenho de Domínio (GDD).	Recepção do alarme pelo sistema operacional, processamento pelo μServer e entrega do alarme ao GDD. O GDD analisa o alarme e decide qual o Especialista que deve ser enviado e para qual elemento gerenciado. A notificação que envia a URL referente ao código do Especialista é então preparada.
v. Envio da notificação ao elemento gerenciado. No caso testado, este elemento é aquele onde está sendo executado Inspetor de Desempenho.	Propagação via rede.
vi. Recepção da notificação no elemento gerenciado.	Recepção da notificação pelo sistema operacional e processamento pelo μServer, buscando o código do Especialista em seus espaços de classe. No caso testado, por ser a primeira vez que esse Especialista será executado nesse elemento gerenciado, seu código não será encontrado. Um pedido do código é então preparado pelo μServer.
vii. Envio do pedido do código à estação do GDD.	Propagação via rede.
viii. Recepção do pedido de envio do código do Especialista.	Recepção do pedido pelo sistema operacional, entrega do mesmo ao μServer, que recupera o código e o prepara para ser enviado.
ix. Envio do código à estação do Inspetor de Desempenho (ID).	Propagação via rede.
x. Recepção do código na estação do ID.	Recepção do código pelo sistema operacional, entrega do mesmo ao μServer que fará o carregamento do mesmo.
xi. Instanciação do código do Especialista (fim da medição).	Execução da ação para a qual o Especialista foi programado.

A distribuição dos valores do retardo na rede ao longo do dia pode ser razoavelmente bem conhecida através da consulta a um *baseline* atualizado da rede. Dessa forma, pode-se não só configurar o tipo de análise a ser feito no Gerente de Desempenho de Domínio, como também determinar-se ações diferentes (Especialistas diferentes) de acordo com os retardos esperado para o momento.

O retardo da rede, medido em paralelo com a execução dos testes, através de execuções do programa *Ping*, não excedeu 0,4 ms. Esse valor pode aumentar expressivamente, principalmente quando o afastamento entre o Inspetor de Desempenho e o Gerente for grande ou existir um grande número de nós de comutação entre os mesmos. Os valores de retardo coletados para a realização do terceiro teste, onde o pior caso foi de 120 ms, podem ser usados como referência. Nesse caso, os retardos obtidos no teste analisado nesta seção seriam aumentados em cerca de 480 ms, devido às quatro transmissões via rede que seriam necessárias para a recuperação do código do Especialista.

O tamanho do Especialista a ser transmitido deve também ser considerado. Os testes mostraram que a transmissão do código de um Especialista de 10 KB levou à triplicação do tempo total do experimento em relação a um Especialista de 1 KB. Tal fato se deve ao processamento dos protocolos da arquitetura TCP/IP e das camadas de Enlace e Física para a transferência desse código. O TCP foi utilizado na camada de Transporte. Com Especialista de 1 KB, dois segmentos TCP foram utilizados, enquanto que no segundo caso, com Especialista de 10 KB, nove segmentos foram utilizados. No entanto, não é esperado que os Especialistas tenham códigos grandes, uma vez que os mesmos são desenvolvidos especificamente para as ações que devem realizar, sendo que essas ações devem envolver, conforme já foi citado, a simples alteração de um valor em uma tabela, a entrega de uma mensagem a uma aplicação ou a realização de uma chamada ao sistema operacional. Nesses casos, dificilmente o código do Especialista alcançaria os 10 KB.

O tempo para o início de uma ação seria drasticamente reduzido, conforme pode-se deduzir com a análise da Figura 27, caso o Inspetor de Desempenho venha a enviar o Alarme de Tendência, por exemplo, diretamente a uma aplicação adaptativa. Em relação ao retardo na rede, dois casos podem ocorrer, Inspetor de Desempenho e Aplicação na mesma estação ou em estações diferentes. O caso ilustrado na Figura 27, pode ser considerado o melhor, uma vez que o Inspetor de Desempenho e a Aplicação encontram-se sendo executados na mesma estação. Caso estivessem em estações diferentes, o retardo referente a apenas uma transmissão via rede deveria ser considerado.

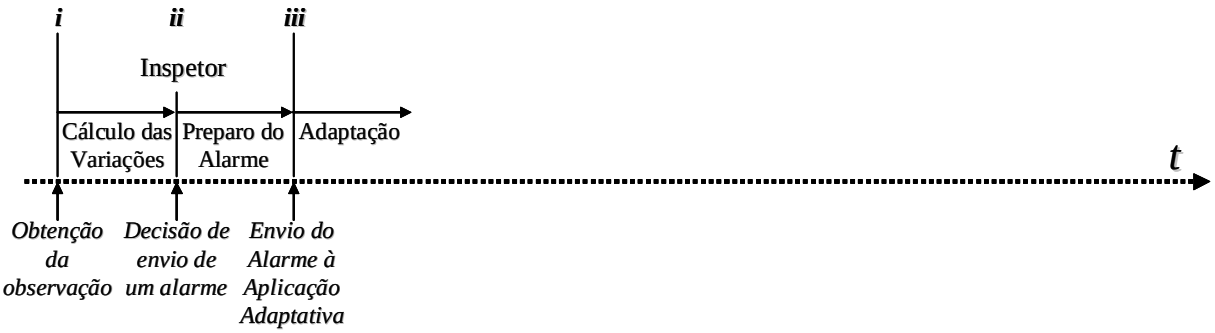


Figura 27 - Etapas entre a decisão de envio de Alarme e início da ação por uma aplicação

Em função do que foi apresentado nos parágrafos anteriores desta seção, levando-se em conta todos os piores casos, ou seja, um Especialista com 10 KB, a ausência do mesmo em cache e um retardo de 120 ms na rede, pode-se considerar que o tempo para que um Especialista inicie a execução de uma ação não deve ultrapassar cerca de 433 (subteste 3) + 480 ms (4 retardos da rede), ou seja, 913 ms, mais o tempo levado na análise do alarme no Gerente de Desempenho de Domínio. Ainda, nesse pior caso, deve ser considerado o tempo necessário à execução do Espe-

cialista propriamente dito, embora esta deva ser, certamente, bastante rápida, em função da natureza das ações que, normalmente, serão desencadeadas.

O tempo total que pode decorrer entre a decisão de envio de um Alarme de Tendência e o término da ação pelo Especialista deve ser, então, limitado a cerca de, pouco menos de um segundo.

No cenário de aplicação do gerenciamento de desempenho que está sendo explorado, em prol do ServiMídia, a melhor opção seria o envio do Alarme de Tendência diretamente ao cliente que está executando uma apresentação multimídia. Nesse caso, o tempo decorrido entre a detecção de uma variação que venha a gerar um alarme e o recebimento do mesmo pelo cliente é bastante pequeno. Num pior caso, se o parâmetro monitorado (o retardo) fosse obtido via SNMP, esse tempo consistiria de 10 ms (tempo de execução do laço de monitoração pelo Inspetor de Desempenho, do primeiro teste) mais o tempo necessário para o desencadeamento da comunicação entre o Inspetor de Desempenho e o cliente ServiMídia, que é desprezível, se ambos estiverem na mesma estação. Nesse mesmo cenário, o Alarme de Tendência poderia ser enviado também ao Servidor ServiMídia para que esse iniciasse os preparativos para a realização de uma adaptação. Nesse caso, o tempo entre a detecção da variação que gerará o alarme e a recepção do alarme pelo Servidor do ServiMídia incluiria ainda um tempo de propagação pela rede.

5.3.3. Execução do Inspetor de Desempenho em massa de dados real

A eficácia do sistema de Gerenciamento de Desempenho Pró-ativo é proporcional à taxa de acertos de suas previsões. Pode ser considerado como um acerto o envio de um Alarme de Tendência quando da detecção de uma variação que, se continuar, venha a causar a ocorrência de uma falha de QoS.

O envio de um alarme, dependendo da análise do mesmo pelo Gerente de Desempenho de Domínio, pode gerar o envio de Especialistas para a realização de ações. Caso a variação a partir da qual tenha sido gerado esse alarme não continue e, portanto, a falha de QoS não venha a ocorrer, as ações desencadeadas pelos Especialistas terão sido inúteis. Esse alarme será chamado de alarme falso.

A Figura 28 apresenta uma primeira seqüência de amostras (da 9 a 39 na seqüência original) que foi selecionada como exemplo para análise. Na Figura 28, podem ser vistos casos onde as ações dos Especialistas teriam sido inúteis, como ocorre nas amostras 5, 13 e 20. A linha horizontal de ordenada 90 representa o limite do retardo para a ocorrência de falha de QoS que foi utilizado nos testes.

Já na segunda seqüência de amostras exemplo, que está apresentada na Figura 29 e representa as amostras 310 a 359 da seqüência original, há Alarmes que podem ser considerados acertos, co-

mo os Alarmes das amostras 20, 31, 41 e 59. Nos casos das amostras 37 e 49, os Alarmes também foram considerados falsos.

A freqüência de monitoração do retardo era de 12 Hz, ou seja, o período entre a obtenção de cada amostra é de 5 segundos. Isso significa que, no caso de um acerto como na amostra 20, as ações dos Especialistas teriam cerca de 5 segundos para que, por exemplo, fossem desencadeadas adaptações no ServiMídia, antes que ocorresse a falha de QoS que levaria a uma falha de QoP. Esse tempo é mais do que suficiente para a execução das adptações, que levam, no máximo, cerca de 2 segundos [27].

A situação definida acima como “acerto” pode ser estendida para o caso no qual o Alarme de Tendência desencadeie ações que venham a evitar, ou até mesmo reduzir a valores toleráveis, a ocorrência de uma falha de QoP, ainda que a falha de QoS que foi prevista ocorra.

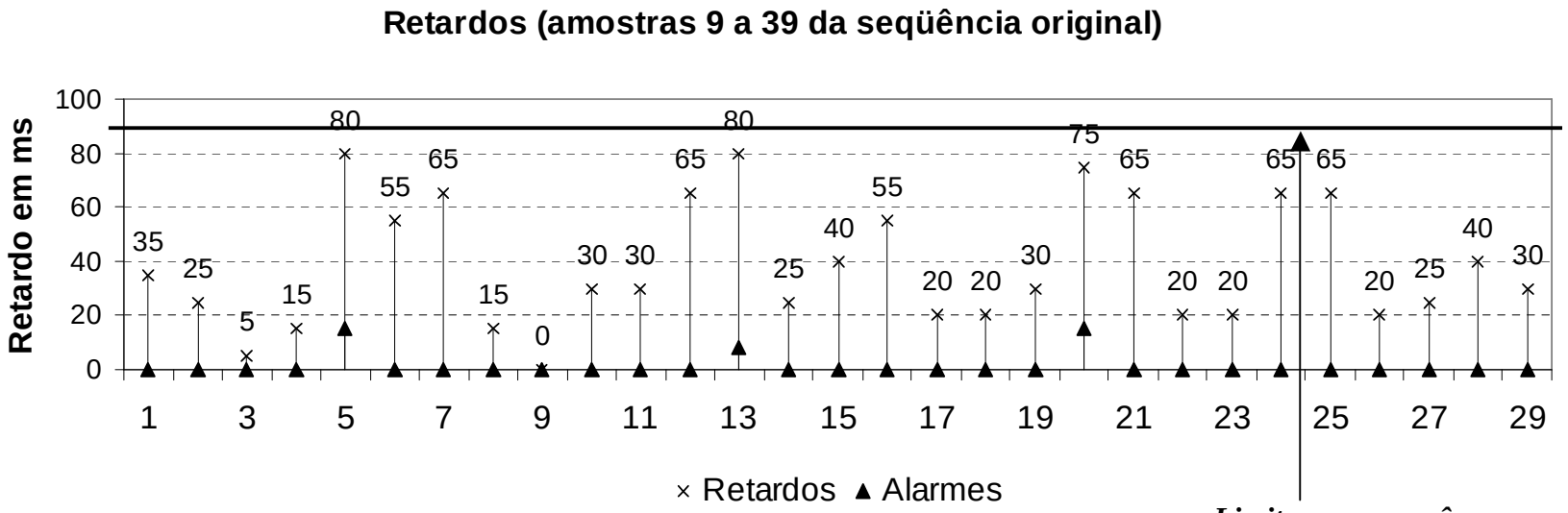


Figura 28 - Primeira seqüência de amostras selecionadas para análise

Limite para ocorrência de falha de QoS

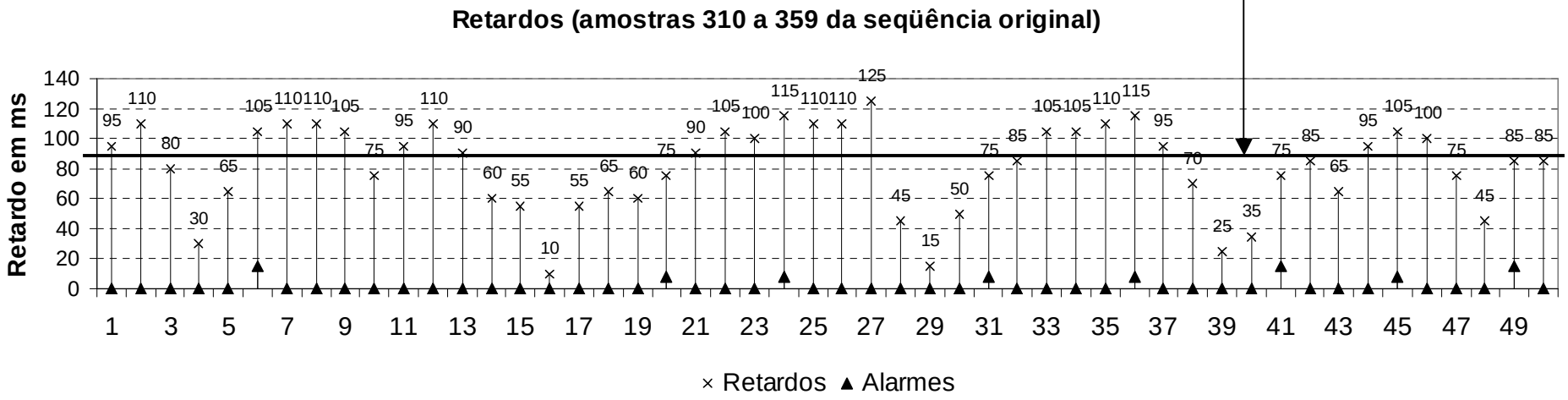


Figura 29 – Segunda seqüência de amostras selecionadas para análise

Excetuando-se as situações onde a variação do retardo é tal, que ao mesmo tempo que ultrapasse o limite para a variação acumulada e o limite para a variação em um intervalo entre duas observações consecutivas, venha a ultrapassar o próprio limite para a ocorrência de falha de QoS, caso que ocorre na amostra 6 da Figura 29, as falhas de QoS são precedidas de Alarmes de Tendência que foram enviados em observações anteriores. Ao longo das 720 amostras (apresentadas no Anexo A) usadas nesse teste equivalentes a duas horas de coleta do retardo, ocorreram 127 falhas de QoS (para o limite de 90 ms). Na realidade, por 127 vezes o retardo ultrapassou 90 ms. No caso da aplicação em questão, um fluxo de vídeo, a pouca variação do retardo, mesmo que com um alto valor absoluto (do retardo) não causa, necessariamente, falha de QoS. O que causa falha é o esvaziamento do *buffer* de recepção do cliente e isso ocorre quando o retardo varia bruscamente, aumentando. Isso significa que nem todas essas 127 amostras de retardo maiores do que 90 ms causarão falhas de QoS para a aplicação. Considerando que a apenas a primeira amostra que ultrapassa 90 ms de uma seqüência de amostras superiores a 90 ms, causa falha, 49 dessas 127 amostras poderiam, de fato, ter causado falha de QoS. Destas 49 amostras que estão sendo consideradas como falhas nesse teste, 15, por serem causadas por variações grandes dentro de um único intervalo, não foram precedidas de Alarmes de Tendência. O Alarme, nessas amostras, foi desencadeado no mesmo momento que ocorreu a falha. Esse poderiam ser considerados Alarmes inúteis. Em outras 8 amostras não foram gerados alarmes antes ou sequer na própria amostra. Isso se deveu ao fato do retardo estar alto, próximo ao limite para a detecção de falhas, variando abaixo e acima do limite, porém sem variar mais do que 35 ou 40 ms, que são os limites configurados como variações máximas toleradas. Do total de falhas consideradas, que é de 51, 24 das mesmas foram precedidas de Alarmes com antecedências entre 5 e 20 segundos. Do total de alarmes que foram gerados, que é de 76, 24 dos mesmos foram Alarmes falsos, ou seja, não houve falha de QoS em até 20 segundos após os mesmos. As estatísticas sobre esse teste que foram consideradas mais relevantes estão apresentadas de forma resumida na Tabela 8.

Tabela 8 - Estatísticas consideradas relevantes no terceiro teste

Fato	*Quantidade*	
Quantidade de amostras em 2h	*720*	
Falhas (Amostras que ultrapassaram 90 ms)	*127*	*Percentual*
Falhas não precedidas de outra falha	*49*	*100 % (Referência)*
Falhas precedidas de alarme	*24*	*49,0 %*
Alarmes gerados 5 s antes da falha	*10*	*20,4 %*
Alarmes gerados 10 s antes da falha	*7*	*14,3 %*
Alarmes gerados 15 s antes da falha	*5*	*10,2 %*
Alarmes gerados 20 s antes da falha	*2*	*4 %*
Alarmes gerados no momento da falha	*15*	*30,6 %*
Falhas não precedidas por alarmes	*8*	*16,3 %*

Essas análises foram feitas a partir da determinação, inicialmente, feita de forma arbitrária, dos limites para ocorrência de falhas, para envio de Alarmes e para as variações máximas toleradas. Após a obtenção dos primeiros resultados deste teste, esses valores limite foram variados, também de forma arbitrária, para que fossem exploradas as possibilidades e limitações do sistema proposto nesse trabalho. A determinação precisa destes limites é, conforme já foi citado, complexa e deve envolver um conhecimento do *baseline* da rede, de forma a se reduzir a ocorrência de Alarmes falsos e se evitar transiências nas ações.

Neste teste, não houve o desencadeamento de ações. Sendo assim, não é possível analisar a conseqüência das mesmas, como, por exemplo, adaptações, no comportamento do retardo. É possível que as próprias ações venham a reduzir o tráfego na rede e, conseqüentemente, o retardo diminua ou, ainda, venham a alterar as exigências de QoS (por causa das adaptações) de forma que o limite do retardo que venha a causar uma falha seja aumentado. Um exemplo desta última situação seria a diminuição da resolução de um fluxo de vídeo, que levaria a um aproveitamento diferente dos *buffers* do cliente de forma que um maior retardo fosse tolerado.

Outra conseqüência da falta do desencadeamento de ações é a impossibilidade de se avaliar o grau de transiência que pode vir a ocorrer por causa das mesmas. Um exemplo dessa transiência seria a realização de adaptações freqüentes pelas aplicações, fato que, sem dúvida, prejudicaria a QoP percebida pelo usuário.

5.3.4. Considerações sobre os resultados

Os resultados obtidos nos testes que foram realizados indicam que a realização de gerenciamento pró-ativo utilizando-se não só de uma linguagem interpretada, porém, totalmente independente de plataforma, mas também de uma infra-estrutura de mobilidade, é viável. O mecanismo principal, o Inspetor de Desempenho, apresentou um baixo tempo de execução, ainda que interpretada, e o tempo entre a detecção de uma variação e a execução de uma ação também é pequeno. A evolução do hardware computacional só virá a melhorar o desempenho dessa execução.

Algumas limitações de um sistema de gerenciamento desse tipo já podem ser visualizadas. A possibilidade de ocorrer a redução da exigência de QoS, com possíveis perdas para os usuários, sem que, de fato, falhas de QoS venham a ocorrer, por conta do envio de um alarme falso, é uma dessas limitações. A transiência nas ações devido à necessidade de se monitorar muitos parâmetros de controle também é uma limitação dessa proposta.

A escalabilidade dessa proposta pode ser questionada, caso o número de elementos a serem gerenciados ou a freqüência do envio de Alarmes sejam grandes. No entanto, espera-se que apenas

aquelas aplicações que são críticas sejam as escolhidas para serem gerenciadas de forma pró-ativa. A escolha dos parâmetros de controle que devem ser monitorados e a sua freqüência também deve ser feita de forma cuidadosa. Apenas aqueles que são relevantes para a garantia de QoS devem escolhidos. A determinação da quantidade de elementos gerenciados em um domínio também deve ser cuidadosamente considerada, para evitar que o Gerente de Desempenho de Domínio seja sobrecarregado de Alarmes.

5.4. Resumo do capítulo

Este capítulo apresentou, inicialmente, a metodologia que foi utilizada para a realização dos testes. Na seção seguinte, foram apresentados os testes realizados e os resultados obtidos, sem que os mesmos fossem analisados. Os testes compreenderam a execução do Inspetor de Desempenho isolada, a interação entre o Inspetor de Desempenho e o Gerente de Desempenho de Domínio para o envio de Especialistas e execução do Inspetor de Desempenho em uma amostra de observações do retardo em uma rede real. Na terceira seção os resultados foram analisados, com o objetivo de que fossem avaliadas a eficiência, a eficácia, as vantagens e as limitações da proposta.

Capítulo 6 - Conclusões e trabalhos futuros

A primeira seção deste capítulo apresenta algumas considerações sobre a arquitetura, resumindo-a e apontando as suas principais vantagens e limitações. Os resultados obtidos a partir da implementação do protótipo são também considerados. A seção seguinte discorre sobre os trabalhos futuros que podem ser realizados a partir da arquitetura de gerenciamento de desempenho pró-ativo. Por fim, na seção 6.3, são apresentadas as considerações finais do trabaho.

6.1. Considerações sobre a arquitetura proposta

A arquitetura que é proposta nesta dissertação tem como objetivo realizar o gerenciamento de desempenho pró-ativo (GDPA). Ela foi desenvolvida tendo como base a AGAD [19], que é uma arquitetura de gerenciamento distribuída que se utiliza de tecnologia ativa.

Os principais componentes da arquitetura são um Gerente de Desempenho de Domínio, que é o responsável pelo gerenciamento de desempenho em um domínio, um Inspetor de Desempenho, que é quem realiza a monitoração de parâmetros de um sistema de controle, calculando a variação dos valores dos mesmos e enviando Alarmes de Tendência quando essas variações ultrapassam limites configurados; e de Especialistas, que são enviados pelo Gerente de Desempenho de Domínio aos elementos gerenciados para desencadearem ações com o objetivo de minimizar as conseqüências ou evitar a ocorrência da possível falha de QoS alertada pelo Alarme de Tendência.

O GDPA utiliza-se da tecnologia ativa para que capacidades de processamento local sejam instaladas nos elementos a serem gerenciados, reduzindo assim o tráfego na rede. Esse tráfego é limitado apenas a Alarmes, códigos de Especialistas e pequenas mensagens. A tecnologia ativa viabiliza ainda que essas capacidades de processamento local, em particular do Inspetor de Desempenho e do Gerente de Desempenho de Domínio, possam ser atualizadas ou substituídas sem que seja necessária a espera de longos processos de padronização.

Uma flexibilidade do GDPA é ser independente de plataforma, por ser inteiramente programado em linguagem Java.

O seu desenvolvimento utiliza-se do que é previsto no desenvolvimento de software orientado a componentes. Esta funcionalidade, aliada ao uso de Java e da tecnologia ativa viabiliza a possibilidade de que os diversos componentes de software da arquitetura possam ter o seu código substituído em tempo real, respeitando-se apenas restrições momentâneas de sincronização de forma a se evitar inconsistências nos atributos, variáveis e métodos que estão sendo executados.

O emprego de um sistema que utilize a arquitetura de GDPA não se limita apenas ao gerenciamento dos elementos da rede. Foram citados casos onde a utilização do sistema, em razão de suas características, pode trazer benefícios para os usuários através da economia de recursos e da redução do tráfego na rede. Esses casos estendem a abrangência do gerenciamento para aplicações, serviços de comunicação, nós de comutação e enlaces de comunicação.

Visando obter dados referentes ao desempenho que mostrassem a viabilidade de um sistema de gerenciamento pró-ativo programado em Java, foi implementado um protótipo que incluía o Gerente de Desempenho de Domínio, o Inspetor de Desempenho e dois Especialistas. O desenvolvimento desse protótipo foi realizado de acordo com o que prevê a análise orientada a objetos. Foram apresentados os casos de uso, o modelo conceitual, os diagramas de seqüência e a hierarquia de classes da análise e da implementação. O protótipo explorou apenas um dos paradigmas disponíveis na tecnologia ativa, os agentes móveis.

Utilizando-se o protótipo, três tipos de testes foram realizados: (i) execução isolada do Inspetor de Desempenho, (ii) interação entre o Inspetor de Desempenho e o Gerente de Desempenho de Domínio o tratamento de Alarmes de Tendência, e (iii) execução do Inspetor de Desempenho em uma amostra com a variação do retardo em uma rede real.

Os resultados obtidos nos testes permitiram a constatação de que, mesmo com o uso de uma linguagem interpretada, como Java, e de uma infra-estrutura de mobilidade, já é viável, com as capacidades de processamento disponíveis nos dias de hoje, o gerenciamento de desempenho pró-ativo de forma eficiente. Esses resultados omitiram o tempo necessário à análise do Alarme de Tendência pelo Gerente de Desempenho de Domínio, que, em função das técnicas que forem utilizadas (como, por exemplo, Inteligência Artificial), pode vir a ser significativo. Nesse caso, uma estação com maior capacidade de processamento seria necessária. Os resultados que amparam a afirmação de viabilidade do gerenciamento de desempenho pró-ativo são aqueles detalhados na seção 5.2, que podem assim ser resumidos:

- a execução laço de monitoração do Inspetor de Desempenho dura, em um AMD K6 de 475 MHz, cerca de 10 milissegundos. Já em um AMD Athlon de 1,3 GHz, esse tempo cai para menos de 2 milissegundos.
- o tempo de reação, entre a detecção de uma variação que, caso continue, causará uma falha de QoS, e a instanciação de um Especialista para a realização de uma ação é, no pior caso, quando usadas estações AMD K6 II de 500 MHz e uma rede cujo retardo seja de 120 milissegundos, menor do que 1 segundo.

- através da precisa configuração dos valores limites para a variação de um parâmetro a ser monitorado, a eficácia do sistema na previsão de falhas de QoS pode ser grande, prevendo uma grande parte das mesmas, por intermédio dos alarmes que são “acertos”, ou seja, que, de fato, prevêm falhas de QoS.
- O envio do Alarme de Tendência diretamente às aplicações interessadas no comportamento do parâmetro de controle que é monitorado foi considerado como uma forma de se diminuir o retardo para o início do desencadeamento de ações com vistas a minimizar os efeitos da falha de QoS que pode acontecer, uma vez que, nesse caso, não existiria a interação entre o Inspetor de Desempenho e o Gerente de Desempenho de Domínio.

Esses valores permitem a inferência de que, com o aumento da capacidade de processamento disponível no hardware computacional e da taxa de transferência da rede, a eficiência da arquitetura será bem maior, pois serão ainda menores o consumo de tempo de CPU e o retardo na rede. Assim, a freqüência de monitoramento dos parâmetros, bem como o número destes que podem ser monitorados, poderão ser maiores.

As principais limitações que já podem ser visualizadas para o emprego da arquitetura são a transiência no desencadeamento de ações quando a variação do parâmetro monitorado oscilar com grande amplitude; e a ocorrência de Alarmes falsos, quando houver variações grandes.

A eficácia pode ser melhorada e o efeito das limitações citadas pode ser diminuído com a cuidadosa escolha dos limites a serem monitorados, problema que foi deixado fora do escopo desta dissertação.

6.2. Trabalhos futuros

A análise do Alarme de Tendência pelo Gerente de Desempenho de Domínio e a determinação dos valores limites para a configuração do Inspetor de Desempenho podem ser apontados como os principais trabalhos futuros.

A análise do alarme deve, ao mesmo tempo, ser rápida, e levar em consideração o máximo de informações possível para que seja eficaz no tempo que ainda estiver disponível até a provável falha de QoS. O conhecimento do *baseline* da rede é essencial. O uso de técnicas de Inteligência Artificial já foi considerado [24] e pode aumentar a eficácia dessa análise através do aprendizado ao longo do próprio funcionamento do gerenciamento de desempenho pró-ativo.

A determinação dos valores limite é importante para que se tente minimizar a ocorrência de Alarmes falsos. O *baseline* da rede também será necessário para essa determinação. Um conhecimento, ou possível configuração, dos *buffers* disponíveis nas estações ou outros elementos ge-

renciados e das capacidades de processamento disponíveis, poderá ajudar no cálculo desses limites. A análise da influência da oscilação na variação do parâmetro monitorado no desencadeamento de ações também deverá ser considerada na determinação desses limites ou mesmo impor um novo limite, como, por exemplo, um número máximo de desencadeamento de ações por certo intervalo de tempo.

Outro trabalho que é necessário é a análise do desempenho do Gerente de Desempenho no tratamento de um número grande de Alarmes de Tendência em um pequeno intervalo de tempo.

A extensão do uso do gerenciamento pró-ativo em outras áreas funcionais de gerenciamento como, por exemplo, segurança e falhas, também pode ser considerado um trabalho futuro relacionado à arquitetura que é proposta nesta dissertação.

O uso da tecnologia ativa, seja das redes ativas ou dos agentes móveis, sempre deverá estar subordinado a condições severas de segurança, uma vez que o envio e a execução automáticos de programas para nós de comutação e estações é o objetivo desta tecnologia. O gerenciamento de desempenho pró-ativo que é proposto nesse trabalho prevê ainda a atualização dos componentes de software da arquitetura em tempo de execução. É, portanto, um trabalho futuro, a definição de um modelo de segurança que permita que as funcionalidades previstas na arquitetura sejam desencadeadas sem riscos para os tanto para proprietários das redes e estações quanto para os usuários.

6.3. Considerações finais

O trabalho integrou áreas de conhecimento novas na computação e que estão sendo utilizadas em muitas pesquisas, como agentes móveis, Java na independência de plataforma e distribuição de processamento, com a finalidade de realizar o gerenciamento de desempenho pró-ativo em tempo real.

O protótipo implementado mostrou que a construção de um sistema baseado na arquitetura já pode ser considerada viável e também que, com os aumentos da capacidade de processamento e da necessidade por garantias de QoS, um melhor aproveitamento dos recursos da rede e melhores níveis de serviços podem ser prestados aos usuários com o uso da arquitetura.

Referências

[1] Bohoris, C., Pavlou, G., Cruickshank, H., *Using Mobile Agentes fos Network Performance Management*, IEEE/IFIP Network Operations and Management Symposium (NOMS´00), Honolulu, Hawaii, 2000.

[2] BRISA (Sociedade Brasileira para Interconexão de Sistemas Abertos), *Arquiteturas de Redes de Computadores OSI e TCP/IP*, Makron Books, 1994.

[3] Keshav, S. e Sharma, R., *Achieving Quality of Service through Network Performance Management*, Proceedings of Network and Operating System Support for Digital Audio and Video, Cambridge, 1998 .

[4] Rubinstein, M.G., Duarte, O.C.M. e Pujolle, G., *Evaluating the Performance of Mobile Agents in Network Management*, Proceedings of IEEE Global Telecommunications Conference, 1999.

[5] Bieszczad, A., Pagurek, B. e White, T., *Mobile Agents for Network Management*, IEEE Communication Surveys, Fourth Quarter, 1999.

[6] Huston, G., *Internet Performance Survival Guide*, Wiley Computer Publishing, 2000.

[7] Bradshaw, J. et al, *Software Agents*, AAAI Press / The MIT Press, Menlo Park, California, 1997.

[8] Raz, D. e Shavitt, Y., *Active Networks for Efficient Distributed Network Management*, IEEE Communications Magazine, Março de 2000.

[9] Bush, S. F., Active *Virtual Network Management Protocol*, 13th Workshop on Parallel and Distributed Simulation (PADS´99), maio de 1999.

[10] Kawamura, R. e Stadler, R., *Active Distributed Management for IP Networks*, IEEE Communications Magazine, Abril de 2000.

[11] Psounis, K., *Active Networks: Applications, Security, Safety and Architectures*, IEEE Communications Surveys, Abril de 1999.

[12] Sabata, B, Chatterjee, S., Davis, M. Sydir, J. J. e Lawrence, T. F., *Taxonomy for QoS Specifications*, WORDS´97, Newport Beach, California, 1997.

[13] Schwartz, B., Jackson, A.W., Strayer, T., Zhou, W., Rockwell, D. e Partridge, C., *Smart Packets: Applying Active Networks to Network Management*, ACM Transactions on Computer Systems, Fevereiro de 2000.

[14] Hu, C., Chen, W., *A Mobile Agent-based Active network Architecture*, International Conference on Parallel and Distributed Systemas, ICPADS´00, IEEE, 2000.

[15] Ferreira, A.B.H., *Dicionário Básico da Língua Portuguesa*, Folha de São Paulo, 1995.

[16] Gomes, A.T.A., *Um Framework para Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação*, Dissertação de mestrado, Departamento de Informática, PUC-Rio, Rio de Janeiro, 1999.

[17] Goldszmidt, G. e Yemini, Y., *Distributed Management by Delegation*, 15a conferência internacional sobre sistemas de computação distribuídos, 1995.

[18] Tennenhouse, D.L. et al, *A Survey of Active Network Research*, IEEE Communications Magazine, janeiro de 1997.

[19] Dumont, A.P.M., *Arquitetura de Gerenciamento de Rede Ativa Distribuída*, Dissertação de Mestrado, NCE-UFRJ, outubro de 2002.

[20] Wheterall, D.J., *Active Network Vision and Reality: Lessons from a Capsule-based system*, 17º Simpósio em Princípios de Sistemas Operacionais da ACM, SOSP´99, dezembro de 1999.

[21] Object Management Group (OMG), *Unified Modeling Language Specification (draft), v 1.4*, fevereiro de 2001.

[22] Brown, A. e Wallnau, K.C., *The Cuurent State of CBSE*, IEEE Software Magazine, outubro de 199Acum.

[23] Data Communications, *Proactive LAN Management*, Data Communications Magazine, março de 1993.

[24] Franceschi, A.S.M., Rocha, M.A., Weber, H.L. e Westphall, C.B., *Proactive Network Management Using Remote Monitoring and Artificial Intelligence Techniques*, Proceedings of the 2nd IEEE Symposium and Communications (ISCC´97), junho de 1997.

[25] Pacifici, G. e Stadler, R., *An Architecture for Performance Management of Multimedia Networks*, Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management, maio de 1995.

[26] Ghinea, G., Thomas, J.P., Fish, R.S. *Quality of Perception to Quality of Service Mapping Using a Dynamically Reconfigurable Communication System*, IEEE Global Telecommunications Conference, Rio de Janeiro, Brazil, 1999.

[27] Gomes, R.L., *Autoria e Apresentação de Documentos Multimídia Adaptativos em Redes*, Dissertação de Mestrado, NCE/IM/UFRJ, 2001.

[28] Cunha, E.C., *Uma Estratégia de Criação e Apresentação de Documentos Multimídia Adaptativos em Rede*, Dissertação de Mestrado, NCE/IM/UFRJ, 2000.

[29] ITU-T, *Rec X.739, Information Technology – Open Systems Interconnection, Systems Management Functions – Metric Objects and Attributes*, ITU-T, 1992.

[30] Martin-Flatin, J.P., Znaty, S. e Hubaux, J.P., *A Survey of Distributed Enterprise Network and Systems Management*, Journal of Network and Systems Management, 1999.

[31] Object Management Group, *The Common Object Request Broker: Architecture amd Specification (CORBA)*, Versão 2.0, 1995.

[32] Deitel, H.M. e Deitel, P. J.. *Java, How to Program*, Ed. Prentice Hall, quarta edição, 2001.

[33] Stevens, W.R., *Unix Networking Programming*, Ed. Prentice Hall, segunda edição, 1998.

[34] Rubinstein, M.G., Duarte, O.C.M., *Evaluating Tradeoffs of Mobile Agents in Network Management*, Networking and Information Systems Journal, 1999.

[35] Wetherall, D.J. e Tennenhouse, , D.L., *Tha Active IP Option*, Telemedia Networks and Systems Group, Laboratoty for Computer Science, Massachusetts Institute of Technology, 1996.

[36] Alexander, S. et al, *Active Network Encapsulation Protocol*, RFC draft, Julho de 1997.

[37] Sluman, C., *A Tutorial on OSI Management*, Computer Networks Magazine, Vol 17, nºs 4 e 5, outubro de 1989.

[38] Reiser, H. e Voigt, G., *Security Requirements for Management Systems using Mobile Agents*, Proceedings of the Fifth Symposium on Computers and Communications: ISCC, Antibes, Frnaça, 2000, p.160-165.

[39] Vicente, J. e Campbell, A.T., *Genesis: Toward Programmable Virtual Networking*, Workshop on Open Signaling for ATM, Internet and Mobile Networks, OPENSIG 98, Outubro de 1998.

[40] Fuggeta, A., Picco, G.P. e Vigna, G., *Understanding Code Mobility*, IEEE Transactions on Software Engineering, IEEE, maio de 1998.

[41] Picco, G.P., *µCode: A Lightweight and Flexible Mobile Code Toolkit*, Proceedings of the 2nd International Workshop on Mobile Agents 98, September de 1998.

[42] Larman, C., *Applying UML and Patterns*, Ed, Prentice Hall, primeira edição, 1997.

[43] AdventNet, *SNMPv1, v2 and V3 Java API*, disponível em http://www.adventnet.com, AdventNet, 2002.

[44] Hardaker, W., *NET-SNMP Project*, disponível em http://www.net-snmp.org/, Carnegie Mellon University, 2002.

[45] Davie, B. S. e Rekhter, Y., MPLS: *Technology and Applications*, Ed Morgan Kaufmann Publishers, Primeira edição, 2001.

[46] Ball, B. *Red Hat Linux 7.2 Unleashed*, Ed Sams, Primeira edição, 2001.

[47] El-Kharashi, M. W., ElGuibaly, F. e Li, K. F., *Quantitative Analysis for Java Microprocessor Architectural Requirements:Instruction Set Design*, Workshop on Hardware Support for Objects And Microarchitectures for Java (no ICCD'99), Texas, 1999.

[48] Ploog, H. et all, *A Two Step Approach in the Development of a Java Silicon Machine (JSM) for Small Embedded Systems*, Workshop on Hardware Support for Objects And Microarchitectures for Java (no ICCD'99), Texas, 1999.

[49] Black, D. P., *Building Switched Networks*, Ed Addison Wesley Longman, 1999.

[50] Rekhter, Y. e Davie, B., *MPLS Technology and Applications*, Ed Academic Press & Morgan Kaufmann, 2000.

[51] Schmidt, K. J. e Maura, D., SNMP Essencial, Ed Campus, 2001.

[52] Montgomery, D. C. e Runger, G. C., *Applied Statistics and Probability for Engineers*, Ed John Wiley & sons, segunda edição, 1999.

[53] Sun Microsystems, *Java 2 Platform, Standard edition, v 1.3.1*, disponível em http://java.sun.com, Sun Microsystems, 2002.

[54] ITU-T, *Rec X.739, Information Technology – Open Systems Interconnection, Systems Management Functions – Summarization Function*, ITU-T, 1992.

[55] StatSoft, *Statistica v5*, disponível em http://www.statsoft.com/, 2001.

Anexo A – Valores referentes ao terceiro teste

Am	Ret	Al	Am	Ret	Al	Am	Ret	Al	Am	Ret	Al	Am	Ret	Al	Am	Ret	Al
1	20	Não	71	35	15	141	95	0	211	105	8	281	45	0	351	95	0
2	5	Não	72	10	0	142	80	0	212	110	0	282	20	0	352	105	0
3	40	Não	73	60	0	143	55	15	213	100	0	283	0	0	353	100	0
4	110	Int	74	95	0	144	35	0	214	95	0	284	0	0	354	75	8
5	25	Não	75	5	0	145	75	0	215	100	0	285	5	0	355	45	0
6	65	Não	76	40	0	146	70	0	216	85	0	286	65	0	356	85	0
7	35	Não	77	60	0	147	90	15	217	80	0	287	0	0	357	85	0
8	25	Não	78	10	0	148	85	0	218	95	0	288	0	0	358	90	15
9	5	Não	79	45	0	149	60	0	219	0	0	289	0	0	359	90	0
10	15	Não	80	10	0	150	30	0	220	0	0	290	15	0	360	25	0
11	80	Int	81	65	0	151	65	0	221	15	0	291	45	0	361	55	0
12	55	Não	82	30	0	152	70	0	222	0	0	292	60	0	362	80	0
13	65	Não	83	30	0	153	75	0	223	0	0	293	90	0	363	45	0
14	15	Não	84	40	0	154	95	8	224	0	0	294	80	0	364	50	8
15	0	Não	85	55	0	155	55	0	225	35	0	295	85	8	365	10	0
16	30	Não	86	95	0	156	65	0	226	0	0	296	55	0	366	30	0
17	30	Não	87	80	0	157	90	0	227	0	0	297	40	0	367	70	0
18	65	Não	88	55	15	158	95	0	228	35	0	298	65	0	368	90	0
19	80	Não	89	35	0	159	110	0	229	0	0	299	70	0	369	105	15
20	25	Não	90	75	0	160	115	8	230	0	0	300	85	0	370	100	0
21	40	Não	91	70	0	161	75	0	231	75	15	301	90	0	371	110	0
22	55	Não	92	90	15	162	0	0	232	10	0	302	110	8	372	125	0
23	20	Não	93	85	0	163	50	0	233	0	0	303	25	0	373	125	8
24	20	Não	94	60	0	164	65	0	234	0	0	304	15	0	374	115	0
25	30	Não	95	30	0	165	85	0	235	25	0	305	35	0	375	100	0
26	75	Int	96	85	0	166	0	0	236	75	15	306	60	0	376	20	0
27	65	Não	97	70	0	167	5	8	237	65	0	307	75	0	377	5	0
28	20	Não	98	75	15	168	0	0	238	60	0	308	95	0	378	15	0
29	20	Não	99	95	0	169	0	0	239	20	0	309	110	8	379	25	0
30	65	Não	100	55	0	170	0	0	240	30	0	310	80	0	380	40	0
31	65	Não	101	75	0	171	0	0	241	45	0	311	30	0	381	35	0
32	20	Não	102	65	0	172	20	0	242	55	0	312	65	0	382	60	0
33	25	Não	103	60	0	173	90	0	243	45	0	313	105	0	383	85	0
34	40	Não	104	20	0	174	15	0	244	45	0	314	110	0	384	90	0
35	30	Não	105	30	0	175	55	15	245	45	0	315	110	15	385	105	8
36	70	Não	106	45	0	176	10	0	246	25	0	316	105	0	386	110	0
37	0	Não	107	55	0	177	25	0	247	20	0	317	75	0	387	100	0
38	0	Não	108	45	0	178	0	0	248	30	0	318	95	0	388	75	0
39	0	Não	109	45	0	179	25	0	249	30	0	319	110	0	389	50	0
40	0	Não	110	45	0	180	20	0	250	55	0	320	90	0	390	15	0
41	0	Não	111	25	0	181	0	0	251	65	0	321	60	0	391	50	0
42	25	Não	112	20	0	182	0	0	252	50	0	322	55	0	392	65	0
43	80	Int	113	30	0	183	0	0	253	20	0	323	10	0	393	115	0
44	95	Não	114	30	0	184	0	0	254	50	0	324	55	0	394	120	0
45	60	Não	115	55	0	185	0	0	255	55	0	325	65	0	395	90	15
46	55	Não	116	65	0	186	0	0	256	55	0	326	60	0	396	85	0
47	10	Não	117	50	0	187	0	0	257	20	0	327	75	0	397	100	0
48	0	Não	118	20	0	188	0	0	258	25	0	328	90	0	398	105	0
49	5	Não	119	50	0	189	15	0	259	100	15	329	105	8	399	110	0
50	15	Não	120	55	0	190	0	0	260	15	0	330	100	0	400	105	0
51	20	Não	121	55	0	191	30	0	261	15	0	331	115	0	401	60	0
52	55	Não	122	20	0	192	35	0	262	25	0	332	110	0	402	120	0
53	75	Não	123	25	0	193	0	0	263	20	0	333	110	8	403	80	0
54	80	Não	124	100	0	194	0	0	264	5	0	334	125	0	404	40	15
55	85	Não	125	15	0	195	55	0	265	5	0	335	45	0	405	75	0
56	90	Não	126	15	15	196	0	0	266	15	0	336	15	0	406	105	0
57	105	Acum	127	25	0	197	0	0	267	20	0	337	50	0	407	115	15
58	95	Não	128	20	0	198	0	0	268	5	0	338	75	0	408	120	0
59	35	Não	129	5	0	199	25	0	269	35	0	339	85	0	409	110	8
60	30	Não	130	5	0	200	0	0	270	15	0	340	105	8	410	120	0
61	20	Não	131	15	0	201	0	0	271	0	0	341	105	0	411	105	0
62	5	Não	132	20	0	202	35	0	272	0	0	342	110	0	412	65	0
63	65	Não	133	5	0	203	40	0	273	45	0	343	115	0	413	35	0
64	75	Não	134	35	0	204	45	0	274	100	15	344	95	0	414	65	0
65	60	Não	135	15	0	205	50	0	275	50	0	345	70	8	415	75	0
66	80	Não	136	65	0	206	65	0	276	55	0	346	25	0	416	90	0
67	90	Não	137	30	0	207	80	0	277	75	0	347	35	0	417	65	8
68	10	Não	138	30	0	208	60	0	278	0	0	348	75	0	418	100	0
69	75	Não	139	35	0	209	60	8	279	0	0	349	45	0	419	90	0
70	105	Não	140	10	0	210	85	0	280	0	0	350	20	15	420	95	15

Anexo A (cont) – Valores referentes ao terceiro teste

Am	Ret	Al	Am	Ret	Al	Am	Ret	Al	Am	Ret	Al	Am	Ret	Al
421	85	0	491	120	0	561	40	0	631	30	0	701	0	0
422	100	0	492	125	0	562	35	0	632	55	0	702	0	0
423	85	0	493	120	0	563	75	15	633	65	0	703	0	0
424	110	0	494	135	8	564	60	0	634	85	8	704	0	0
425	90	0	495	110	0	565	75	0	635	110	0	705	0	0
426	35	0	496	120	0	566	85	0	636	115	0	706	0	0
427	55	0	497	125	0	567	35	0	637	125	8	707	0	0
428	65	0	498	95	0	568	75	15	638	120	0	708	0	0
429	70	0	499	75	0	569	80	0	639	95	0	709	0	0
430	65	0	500	90	0	570	60	0	640	90	0	710	0	0
431	10	0	501	110	0	571	35	0	641	85	0	711	0	0
432	40	0	502	115	8	572	60	0	642	90	0	712	0	0
433	55	0	503	125	0	573	85	8	643	95	0	713	0	0
434	15	0	504	95	0	574	45	0	644	115	0	714	0	0
435	60	0	505	75	0	575	40	0	645	110	0	715	0	0
436	0	0	506	40	0	576	60	0	646	105	0	716	0	0
437	0	0	507	70	0	577	10	0	647	95	0	717	0	0
438	0	0	508	0	0	578	0	0	648	85	0	718	0	0
439	10	0	509	0	0	579	0	0	649	80	0	719	5	0
440	0	0	510	10	0	580	20	0	650	95	0	720	0	0
441	0	0	511	15	0	581	65	0	651	75	0			
442	15	0	512	40	0	582	80	8	652	80	0			
443	0	0	513	35	0	583	75	0	653	90	0			
444	0	0	514	55	0	584	90	0	654	95	0			
445	5	0	515	75	8	585	110	0	655	115	8			
446	0	0	516	40	0	586	75	0	656	120	0			
447	10	0	517	70	0	587	60	0	657	0	0			
448	0	0	518	0	0	588	50	0	658	30	0			
449	0	0	519	0	0	589	35	0	659	45	0			
450	20	0	520	10	0	590	45	0	660	65	0			
451	15	0	521	75	15	591	55	0	661	75	8			
452	10	0	522	110	15	592	70	0	662	80	0			
453	0	0	523	95	0	593	85	8	663	0	0			
454	0	0	524	100	0	594	95	0	664	0	0			
455	90	15	525	115	0	595	100	0	665	0	0			
456	30	0	526	110	0	596	110	0	666	0	0			
457	20	0	527	115	0	597	25	0	667	0	0			
458	45	0	528	105	0	598	65	0	668	0	0			
459	75	8	529	0	0	599	30	0	669	0	0			
460	85	0	530	0	0	600	30	0	670	0	0			
461	105	0	531	0	0	601	40	0	671	0	0			
462	115	8	532	35	0	602	55	0	672	5	0			
463	125	0	533	80	15	603	95	15	673	0	0			
464	0	0	534	95	0	604	80	0	674	0	0			
465	60	0	535	60	0	605	55	0	675	45	0			
466	75	8	536	55	0	606	35	0	676	75	8			
467	60	0	537	10	0	607	75	15	677	50	0			
468	55	0	538	0	0	608	70	0	678	55	0			
469	60	0	539	5	0	609	85	0	679	75	0			
470	45	0	540	0	0	610	85	0	680	65	0			
471	35	0	541	55	0	611	60	0	681	70	0			
472	55	0	542	65	0	612	30	0	682	85	0			
473	70	0	543	0	0	613	75	15	683	95	8			
474	85	8	544	60	0	614	70	0	684	100	0			
475	90	0	545	0	0	615	75	0	685	85	0			
476	75	0	546	10	0	616	85	0	686	75	0			
477	95	0	547	50	0	617	55	0	687	45	0			
478	90	0	548	75	8	618	70	0	688	50	0			
479	0	0	549	110	15	619	65	0	689	65	0			
480	0	0	550	125	0	620	60	0	690	45	0			
481	0	0	551	120	0	621	20	0	691	40	0			
482	65	0	552	110	0	622	30	0	692	50	0			
483	85	8	553	95	0	623	45	0	693	0	0			
484	95	0	554	90	0	624	55	0	694	0	0			
485	60	0	555	95	0	625	45	0	695	0	0			
486	55	0	556	110	0	626	45	0	696	5	0			
487	20	0	557	85	0	627	45	0	697	0	0			
488	45	0	558	80	0	628	25	0	698	0	0			
489	95	15	559	85	0	629	20	0	699	0	0			
490	115	0	560	55	0	630	30	0	700	0	0			