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

E. L. Cecilio, A. P. M. Dumont, R. de B. Correia, L. F. Rust da C. e Luci Pirmez

Núcleo de Computação Eletrônica - Universidade Federal do Rio de Janeiro
Tel: 021 2598-3159 - Caixa Postal 2324, Rio de Janeiro, RJ, Brasil

cecilio@ime.eb.br, dumont@unisys.com.br,
reinaldo@posgrad.nce.ufrj.br, {rust, luci}@nce.ufrj.br

Abstract. *Performance management is done, typically, in a reactive way. This means a QoS failure is notified only after occurring. This paper proposes a pro-active performance management architecture. Variations of QoS parameters are monitored in real time and Tendency Alarms are generated when they show that, possibly, failures can occur. Then, actions are automatically executed to try to avoid the failures or minimize their effects. A distributed management infra-structure based on active technology and Java is used. Tests results allow efficacy and performance of monitoring and reaction time of the system to be evaluated.*

Resumo. *O gerenciamento de desempenho é tipicamente reativo, ou seja, notifica uma falha de QoS somente após a sua ocorrência. Esse artigo apresenta a proposta de uma arquitetura de gerenciamento de desempenho que é pró-ativa. Variações dos parâmetros de QoS são monitoradas em tempo real e Alarmes de Tendência são gerados quando elas indicam a possibilidade de ocorrência de falhas. Ações são então desencadeadas, automaticamente, para que se tente evitá-las ou minimizar seus efeitos. É utilizada uma infra-estrutura de gerenciamento distribuído baseada em tecnologia ativa e em Java. Os resultados dos testes realizados permitem que sejam avaliados o desempenho e a eficácia da monitoração e o tempo de reação do sistema.*

1. Introdução

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 antes da ocorrência de falhas 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.

A tecnologia ativa – redes ativas e agentes móveis – vem sendo considerada em recentes pesquisas para o provimento da flexibilidade e distribuição necessárias para a próxima geração de futuros sistemas de gerenciamento. Esse artigo 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 teve como objetivo investigar as possibilidades de monitoração e desencadeamento de ações em tempo real por um sistema de gerenciamento de desempenho pró-ativo que foi programado em Java e utiliza uma infra-estrutura de mobilidade de código baseado na mesma linguagem, 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 avaliadas as vantagens e desvantagens da arquitetura que é proposta.

O artigo contém mais quatro seções além dessa introdução. Na segunda seção são apresentados os conceitos básicos que são relevantes para o entendimento da proposta. A seção seguinte apresenta a arquitetura que foi concebida e o protótipo que foi implementado. A quarta seção apresenta os testes que foram realizados, os resultados que foram obtidos e a análise dos mesmos. Por fim, a última seção apresenta as conclusões e os trabalhos futuros que podem ser realizados tendo como base a arquitetura proposta.

2. Conceitos básicos

Nesta seção serão apresentados os conceitos mais relevantes para o entendimento da arquitetura que é proposta, que são a tecnologia ativa, as vantagens do uso da mesma no gerenciamento distribuído e o gerenciamento de desempenho pró-ativo.

2.1. Tecnologia ativa

Os paradigmas de redes ativas [9][10][11] e agentes móveis [12][13][14] 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 ou atualizadas com rapidez e flexibilidade. Embora os conceitos desses dois paradigmas tenham sido originados em diferentes comunidades de pesquisas, visando resolver diferentes problemas, eles se superpõem, fato que está popularizando o termo tecnologia ativa para referência a um ou ambos os paradigmas [8].

A 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 aplicações. 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 fluxos de 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 de uma rede ativa [14]. Ainda, agentes móveis podem migrar baseados em decisões autônomas, podem também gerar processos filhos ou *threads* para tratarem de problemas específicos e, por fim, podem trocar mensagens entre si. Essas funcionalidades não estão previstas nas redes ativas [8].

No plano de gerenciamento, pode-se considerar que as funcionalidades a serem executadas estão, principalmente, no nível das aplicações. No entanto, há casos, como no gerenciamento de falhas em nível de enlace, onde o trabalho é típico de redes ativas. É mais

aconselhável então, no que diz respeito a gerenciamento, empregar-se o termo tecnologia ativa, pois tanto redes ativas quanto agentes móveis podem ser utilizados.

2.2. Tecnologia ativa no gerenciamento distribuído

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 a esses elementos via software. O comportamento desses softwares e o seu nível de especialização podem ser modificados a qualquer momento. 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.

Além da facilidade de atualização de funcionalidades, esses softwares podem ser providos de capacidades avançadas no que diz respeito ao processamento local das informações de gerenciamento. Os tempos de detecção de problemas e de restabelecimento do funcionamento normal do elemento gerenciado podem então ser bastante reduzidos. 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. Pode haver filtragem e fusão de informações antes de as mesmas serem enviadas aos servidores de gerenciamento. O simples polling via rede em busca apenas de dados pode ser praticamente abolido. Granularidades mais finas para monitoração e controle dos parâmetros de QoS podem também ser utilizadas.

2.3. Gerenciamento de desempenho pró-ativo

Certos aspectos na monitoração de desempenho, especialmente quando medidas de tempo estão envolvidas, são difíceis de serem considerados nas abordagens centralizadas de gerenciamento. O retardo gerado pela transmissão dos dados pela rede torna a precisão de medidas questionável [13]. O envio de um software, como um agente móvel (ou uma aplicação ativa) permite a análise local do elemento a ser gerenciado.

Adicionalmente, softwares locais podem, por intermédio de cálculos sobre variações, detectar a tendência de comportamento de certos parâmetros. Ações podem ser desencadeadas localmente ou notificações podem ser enviadas ao servidor de gerenciamento quando limites dessa tendência forem ultrapassados. Dessa forma, é possível, mediante análise dessas tendências, se tentar prever a ocorrência de falhas de QoS, ou seja, realizar-se o gerenciamento pró-ativo em vez de reativo. É fundamental, no caso do gerenciamento pró-ativo, que ações possam ser desencadeadas automaticamente, fato que é possibilitado pela tecnologia ativa. 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.

2.4. Considerações sobre o gerenciamento pró-ativo usando tecnologia ativa

Apesar da flexibilidade proporcionada pelo uso da tecnologia ativa, ainda pode ser questionável a eficiência e a eficácia tanto do desencadeamento de ações de forma automática quanto da monitoração em tempo real quando se utiliza uma infra-estrutura de mobilidade, dada a presença de várias camadas de processamento. Caso essa infra-estrutura seja

baseada em uma linguagem independente de plataforma, como Java, ainda é adicionada a JVM a essas camadas. No entanto, teste preliminar sobre a viabilidade do funcionamento de um protótipo baseado nessa arquitetura, cujo resultado foi publicado em [18], já havia demonstrado que, mesmo assim, o gerenciamento pró-ativo é viável.

3. Gerenciamento de desempenho pró-ativo usando tecnologia ativa

A arquitetura que é proposta neste artigo é baseada na Arquitetura de Gerenciamento Distribuído usando Tecnologia Ativa (AGAD) que foi desenvolvida no NCE/UFRJ [3], estendendo-a para o gerenciamento de desempenho pró-ativo. Nesta seção serão apresentados um resumo da AGAD, o detalhamento da extensão para o gerenciamento de desempenho pró-ativo que é o foco do artigo e os detalhes do protótipo que foi implementado.

3.1. Arquitetura de Gerenciamento Distribuído usando Tecnologia Ativa (AGAD)

A AGAD [3], cuja visão geral é mostrada na Figura 1, provê a infra-estrutura que permite o gerenciamento distribuído das redes, serviços, estações e aplicações.

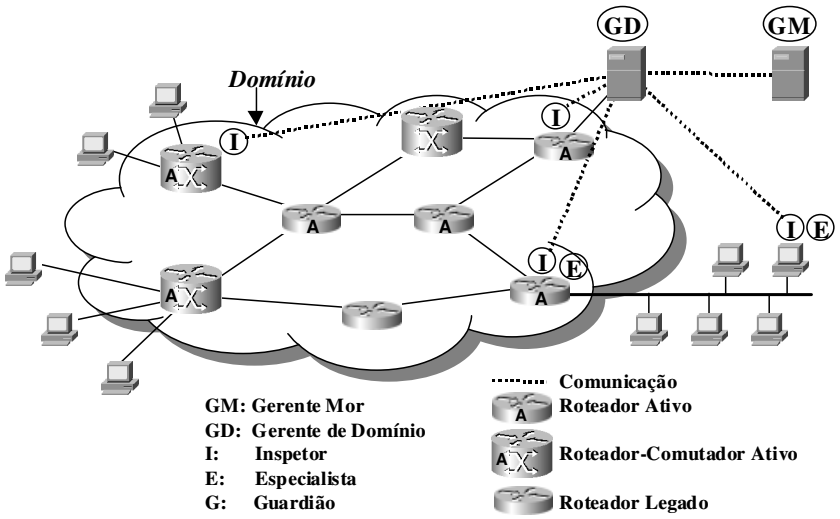

Figura 1 - Visão geral da AGAD

O GM é implementado de forma fixa e é o responsável pelo gerenciamento de um conjunto de domínios, via Gerentes de Domínio. 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. Um domínio equivale a um conjunto de elementos a serem gerenciados sob uma mesma autoridade administrativa. Da mesma forma que o GM, o GD disponibiliza uma interface gráfica para o gerente.

O GD cria e envia Inspetores e Especialistas aos elementos do domínio que devem ser gerenciados. Os Inspetores são os responsáveis pela obtenção, local, dos dados referentes às funcionalidades que são gerenciadas, ou seja, a monitoração dos parâmetros de QoS. A obtenção desses dados é feita com o uso das facilidades disponíveis localmente, como, por exemplo, acesso às MIB. Os dados podem então sofrer um processamento simplificado o suficiente para não sobrecarregar o elemento gerenciado em questão, para

que apenas um mínimo de informações sejam enviadas ao GD. 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 dos códigos em tempo de execução.

As informações de gerenciamento (consolidadas) recebidas dos Inspetores pelo GD são armazenadas e, após a análise e classificação das mesmas, Especialistas podem ser criados pelo GD para que sejam enviados para onde forem necessários para a realização automática de tarefas especializadas relativas ao gerenciamento. Exemplos de ações que podem ser realizadas por Especialistas são a alteração de políticas de escalonamento, a reconfiguração da utilização de memória, interagir com uma aplicação adaptativa, entregar uma mensagem para sinalizar a necessidade de que seja desencadeado um rerroteamento, etc.

3.2. Extensão à AGAD para o gerenciamento de desempenho pró-ativo

O gerenciamento de desempenho pode ter seus objetivos reunidos em duas diretrizes conflitantes: (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 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 [17], no modelo apresentado na Figura 2.

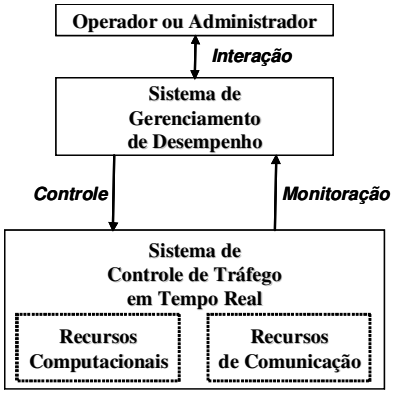

Figura 2 - O Sistema de gerenciamento de desempenho Segundo Pacifici

O Sistema de controle de tráfego em tempo real regula a competição pelos recursos disponíveis nas estações e nos nós de comutação, através de uma coleção de mecanismos, cada um com sua tarefa específica. São exemplos desses controles o gerenciamento de *buffers*, de escalonamento de processos ou *threads* no processador, de controle de fluxo, de roteamento e/ou re-roteamento e de controle de admissão, entre outros. A operação desses mecanismos pode ser sintonizada através de *parâmetros de controle*. 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.

A monitoração é realizada pelo Inspetor de Desempenho instalado no Inspetor (da

AGAD) associado ao elemento a ser gerenciado, enquanto as ações são desencadeadas por Especialistas. Durante a monitoração são coletados dados das fontes disponíveis, (MIBs, RTCP, sistema operacional, etc) na estação ou no nó de comutação, ou das próprias aplicações, sobre os parâmetros de controle que devem ser monitorados. As variações desses dados são calculadas para que sejam detectadas as tendências de ocorrência de falha de QoS, quando então Alarmes de Tendência são enviados ao Gerente de Desempenho de Domínio ou a outras aplicações. Mediante extrapolações, realizada pelo Gerente, são 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.

Visando um melhor entendimento, será usada como exemplo a monitoração da disponibilidade de um recurso, expressa percentualmente. Será utilizada uma amostra de teste com 50 observações (Figura 3). 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 um valor limite para o envio de alarmes, que no exemplo é de 80%. Dois limites máximos devem ser estabelecidos para a variação: o máximo para uma variação em um único intervalo (10%, no exemplo), e o máximo para a variação acumulada desde o último Alarme que foi enviado (12%), independentemente do número de observações. Ao longo da monitoração, um valor denominado de teto para cálculo da variação acumulada é verificado e, quando necessário, atualizado. 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 ao alarme, 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 este veio a ser maior ("melhor", portanto) do que aquela que motivou o envio de um Alarme, passa a ser considerado como o teto para o cálculo da variação acumulada.

O teto 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%.

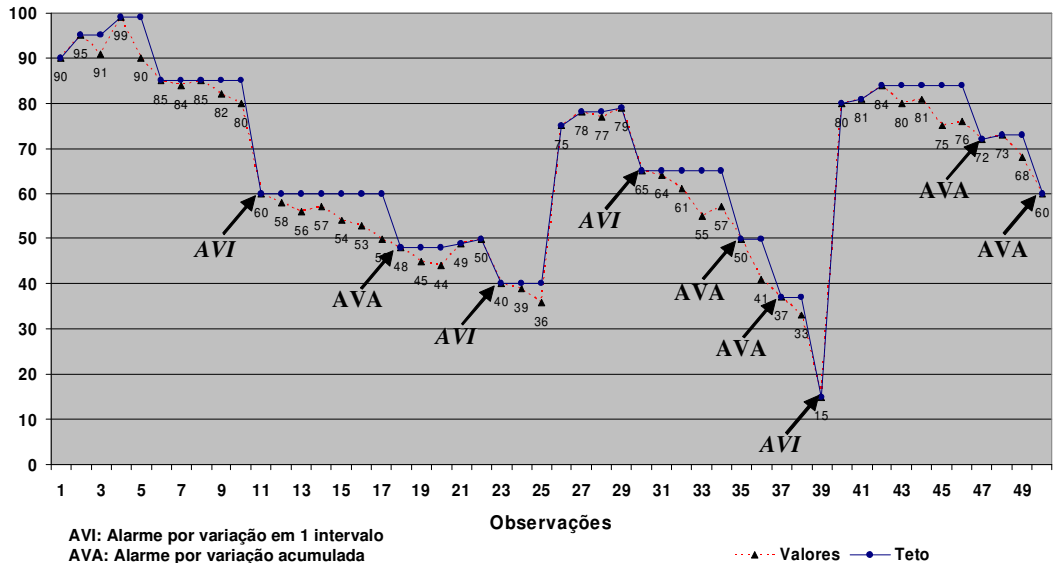

Figura 3 - 50 observações do parâmetro disponibilidade para teste

3.3. Protótipo implementado

O enfoque da implementação foi sobre o Inspetor de Desempenho, para que fosse possível investigar a viabilidade de se realizar tarefas de gerenciamento com fina granularidade de tempo, em tempo real, utilizando-se uma linguagem independente de plataforma, Java, e uma infra-estrutura de mobilidade, também baseada em Java.

A infra-estrutura de mobilidade utilizada foi o µCode [15]. Ela é baseada na presença de servidores, µServers, sendo executados sobre a JVM. 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 [2]. As operações básicas disponibilizadas pelos µServers são a cópia e a criação remota de *threads* e a realocação tanto de um conjunto de classes quanto do fechamento de uma classe, ou seja, a própria classe e todas que são por ela invocadas. 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 *threads* em execução e, principalmente, da fina granularidade disponibilizada para a mobilidade de código, além de ser gratuito.

Foi utilizada uma API de gerenciamento baseada no padrão SNMPv1, da empresa AdventNet [16]. Essa API foi escolhida por ser gratuita e oferecer um alto grau de encapsulamento, em Java, dos detalhes referentes à interação com agentes SNMP.

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. O Gerente de Desempenho de Domínio implementado trata somente da iniciação do gerenciamento, 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 a monitoração de um parâmetro de controle. Maiores detalhes, incluindo os diagramas da análise orientada a objetos realizada, podem ser encontrados em [1].

4. Testes e análise dos resultados obtidos

Foram realizados três testes: (i) execução isolada do Inspetor de Desempenho, (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. O objetivo geral dos testes era 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.

Os testes consistiram, quando necessário, na execução de até cinco rodadas do evento em questão, cada uma delas com pelo menos trinta experiências (cujos valores estão disponíveis em [1]), 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 [6][7]. Apenas as médias referentes a cada rodada serão apresentados neste artigo.

4.1. Testes do Inspetor de Desempenho isolado

O objetivo específico deste teste é a obtenção de dados referentes ao desempenho do funcionamento do próprio Inspetor de Desempenho em plataformas com diferentes capacidades de processamento.

Uma execução típica do laço de monitoramento envolve as seguintes fases: (i) obtenção de uma observação do valor do parâmetro monitorado; (ii) execução de um trecho de programa com cálculos e comparações simples; (iii) envio de um Alarme de Tendência, caso seja necessário, via pilha de protocolos do sistema operacional; (iv) armazenamento em memória secundária dos dados referentes à observação e ao alarme que foi enviado. Nesse teste, foi executado o pior caso, onde a obtenção do valor da observação é feita via SNMP, que é uma das formas mais demoradas de se fazê-lo, pois requer o preparo de uma PDU de requisição e a sua transmissão via UDP. Sempre era enviado um alarme. Os resultados obtidos estão apresentados na Tabela 1.

Tabela 1 – Execução de um laço de monitoração em seu pior caso

	AMD K6 475 MHz					*Athlon 1,3 MHz*				
Rodadas:	***1***	***2***	***3***	***4***	***5***	***1***	***2***	***3***	***4***	***5***
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

Caso seja necessário, é possível se diminuir o tempo de execução de um laço do Inspetor de Desempenho, através da utilização de *threads* de menor prioridade para o armazenamento de dados em disco. Ainda, dependendo do parâmetro monitorado, a obtenção do valor do mesmo pode colaborar para a melhora do desempenho da monitoração, como, por exemplo, no caso de ser necessária apenas uma chamada ao sistema. O esquema de *garbage collection* da JVM também pode ser cuidadosamente programado.

Pode-se afirmar que é viável a monitoração de valores de parâmetros de QoS em tempo

real via Java. 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 essa pior plataforma possuía capacidade de processamento que era quase insuficiente para a reprodução de um vídeo MPEG-1 de 400 kbps. Não se espera que esse tipo de plataforma seja utilizado em aplicações multimídia distribuídas. Na plataforma Athlon, o tempo de execução do laço foi menor do que 2 milissegundos, algo que permitiria, caso o tempo de execução desse laço fosse o único fator limitante, uma freqüência de monitoração de até cerca de 500 Hz.

4.2. 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.

O Inspetor de Desempenho e o Gerente de Desempenho foram executados em estações diferentes, ambas AMD K6-II 500 Mhz, 256 MB de RAM e 128 KB de cache, interligadas via *Fast Ethernet*. O Gerente Mor iniciava todo o processo, a partir de uma terceira estação. Um esquema simplificado do teste, é apresentado na Figura 4.

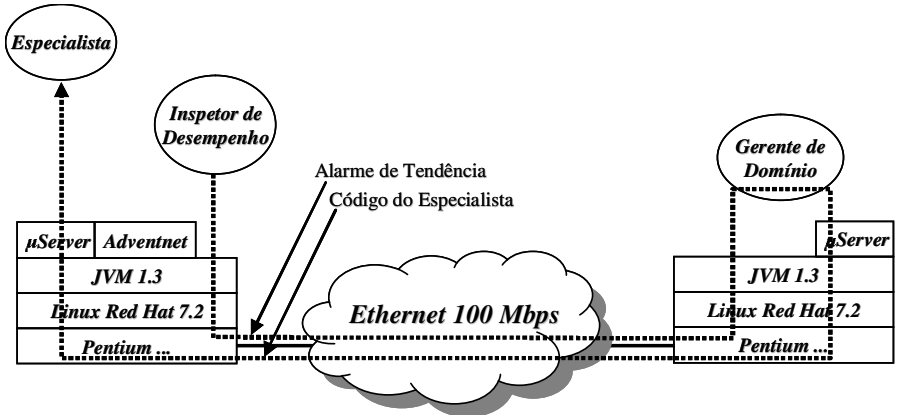

Figura 4 - Esquema simplificado da interação Inspetor – Gerente de Desempenho

Foram 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 ii (incluído no tempo i) 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. O Gerente envia primeiro o URL do código (do Especialista) que deve ser executado. O *class loader* do µServer, na estação do Inspetor de Desempenho, ao receber o URL, 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 De-

sempenho de Domínio. Caso um Especialista venha a ser executado muitas vezes, 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 três subtestes de desencadeamento da ação via Especialistas.

No subteste 1, todo o sistema foi reiniciado antes de cada uma das trinta experiências, para que o código do Especialista (com 1 KB de *bytecodes*) fosse sempre transferido. Os tempos obtidos estão apresentados na Tabela 2 (subteste 1).

Tabela 2 - Resultados dos testes de interação Inspetor - Gerente

	Subteste 1		Subteste 2		Subteste 3	
Tempos:	(i)	(ii)	(i)	(ii)	(i)	(ii)
Média:	**136,80**	**11,10**	**42,23**	**9,30**	**433,40**	**10,87**
Grau ajuste à normal (%):	99,99	95,27	95,48	99,79	95,30	96,30
Desvio padrão:	7,24	2,43	5,39	2,42	6,27	2,13
Int de conf 95%:	±2,59	±0,87	±1,93	±0,87	±2,24	±0,76

Obs: tempos em milissegundos

O subteste 2 repetiu o subteste 1, mas sem que o sistema fosse reinicializado. Nesse caso, o código do Especialista permanecia em cache, evitando a transmissão do pedido para o seu envio e a transferência do seu próprio *bytecode*, exceto na primeira execução (que não foi considerada).

O subteste 3 consistia na repetição do primeiro teste (com reinicialização) porém, com o Especialista tendo, em *bytecodes*, 10 KB.

O intervalo de tempo total (i), medido nesse teste, é decomposto nos processamentos e transmissões mostrados na Figura 5.

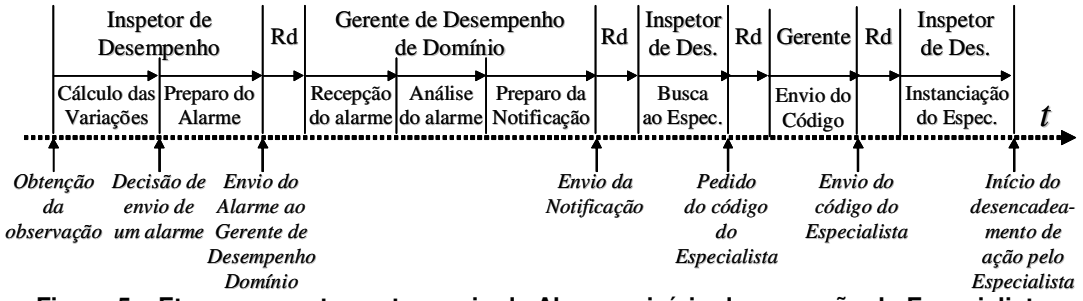

Figura 5 – Etapas e eventos entre envio de Alarme e início de execução do Especialista

O tempo de análise do Alarme 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 Desempenho de Domínio pode ser instalado em uma estação com elevada capacidade de processamento.

A distribuição dos valores do retardo na rede ao longo do dia pode ser razoavelmente bem conhecida através da consulta a um *baseline* atualizado do tráfego 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, não excedeu 0,4 ms, podendo este valor aumentar expressivamente. Os valores de retardo coletados para a

realização do terceiro teste (explicado a seguir), 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.

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 aos processamentos dos protocolos da arquitetura TCP/IP e das camadas de Enlace e Física e da infra-estrutura de mobilidade para a transferência desse código. 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, que normalmente são simples.

O tempo para o início de uma ação seria drasticamente reduzido, conforme pode-se deduzir, caso o Inspetor de Desempenho venha a enviar o Alarme de Tendência, por exemplo, diretamente a uma aplicação adaptativa.

Levando-se em conta todos os piores casos, ou seja, um Especialista com 10 KB não carregado 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 Especialista propriamente dito, embora esta deva ser, certamente, bastante rápida, em função da natureza das ações que, normalmente, serão desencadeadas.

4.3. Monitoramento do parâmetro de controle retardo

O terceiro teste teve como objetivo específico realizar o monitoramento do parâmetro de controle retardo experimentado por 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, antes da possível ocorrência da falha de QoS.

Os valores do retardo 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. O IME (7 departamentos de ensino independentes) é coberto por uma rede com cerca de 45 segmentos, todos baseados em Ethernet, e 500 estações de trabalho (Figura 6). Esse é um cenário típico não só para a utilização de uma aplicação multimídia distribuída, como também para um domínio de gerenciamento distribuído conforme o que é previsto pela AGAD. Visando obter maior realismo na obtenção dos dados referentes ao retardo no cenário idealizado, 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. A medição do retardo (a 0,2 Hz) foi realizada durante três dias consecutivos, sendo então selecionada uma amostra típica (das 15 as 17

horas, ou seja, 720 amostras) com vários picos de retardo e considerável variação, conforme pode ser observado na Figura 7. A linha horizontal de ordenada 90 representa o limite do retardo para a ocorrência de falha de QoS que foi utilizado nos testes.

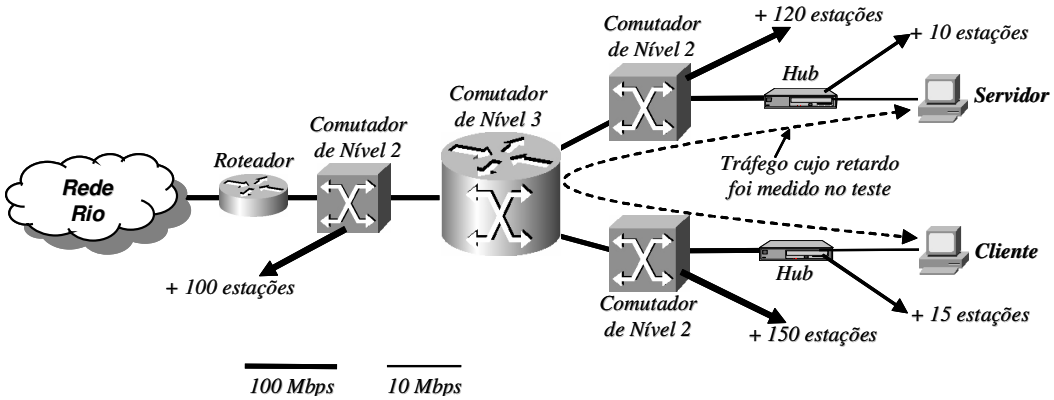

Figura 6 - Topologia da rede utilizada para obtenção do comportamento do retardo

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: (i) 65 ms como limite mínimo do retardo a partir do qual Alarmes devem ser gerados; (ii) 35 ms como variação máxima entre duas observações consecutivas e (iii) 40 ms como variação máxima acumulada. 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, da freqüência com a qual se pretende realizar a monitoração do retardo e da taxa(s) do(s) fluxo(s) em questão, entre outros. Uma vez que o foco dessa proposta é o mecanismo de gerenciamento de desempenho propriamente dito, esse problema foi deixado fora do escopo da mesma e é apontado como um dos possíveis trabalhos futuros.

Os triângulos da Figura 7 representam os Alarmes de Tendência que foram gerados, sendo que, quando na ordenada mais alta, o alarme foi gerado por desrespeito ao limite máximo de variação para duas amostras consecutivas (35 ms) e, quando na ordenada mais baixa, o alarme foi gerado por desrespeito à variação máxima acumulada (40 ms).

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, cause uma falha de QoS. Em contrapartida, se o envio de um alarme pode gerar o envio de Especialistas para a realização de ações e 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 é considerado falso.

A Figura 8 (superior) apresenta uma primeira seqüência de amostras que foi selecionada como exemplo para análise. Podem ser vistos casos onde as ações dos Especialistas teriam sido inúteis, como ocorre nas amostras 5, 13 e 20. Já na seqüência de baixo da figura, há Alarmes que podem ser considerados acertos, como aqueles das amostras 20, 31, 41 e 59. Nas amostras 37 e 49, os Alarmes também foram considerados falsos.

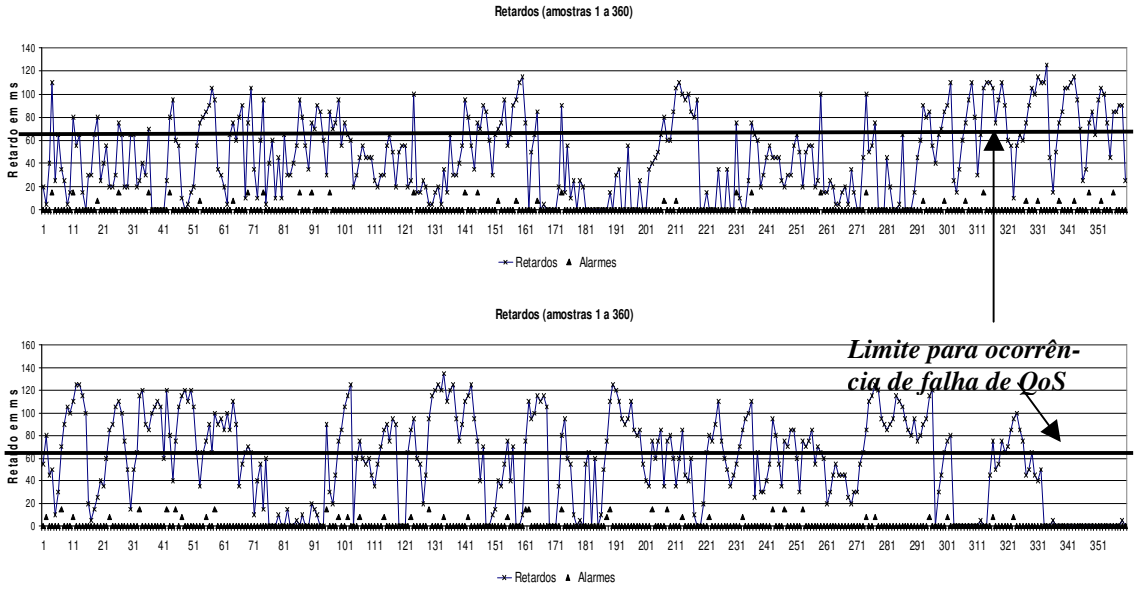

Figura 7 – Alarmes gerados na amostra de variação de retardo

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 em uma aplicação multimídia distribuída como, por exemplo, o ServiMídia, antes que ocorresse a falha de QoS. Esse tempo é mais do que suficiente para a execução das adptações, que demoram cerca de 2s [4].

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

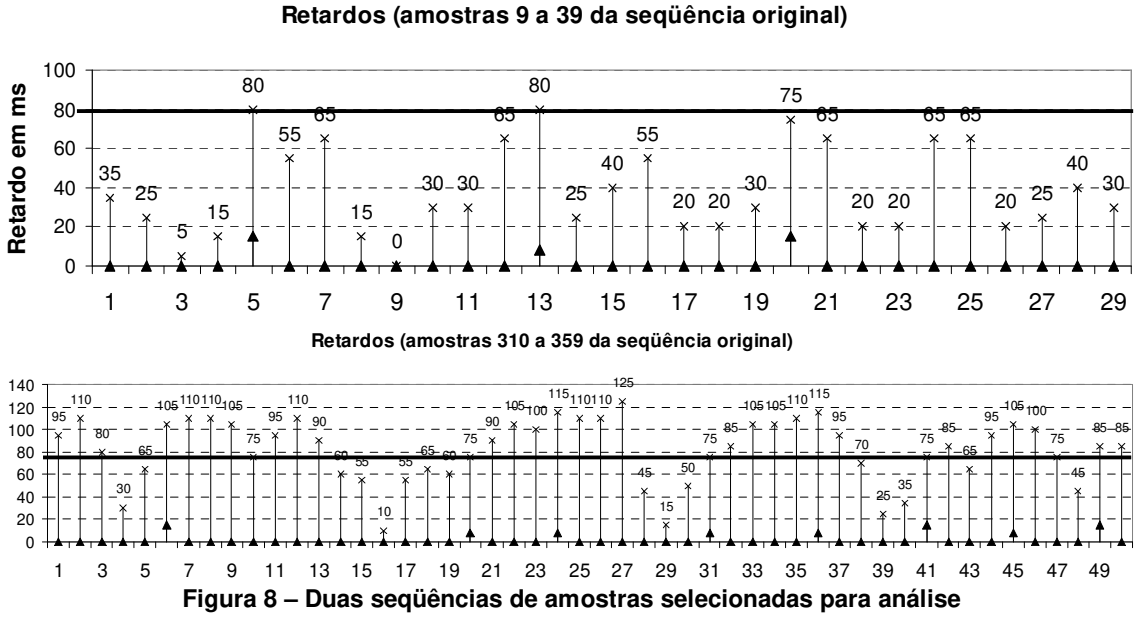

Figura 8 – Duas seqüências de amostras selecionadas para análise

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 8, as falhas de QoS fo-

ram precedidas de Alarmes de Tendência. Ao longo das 720 amostras (disponíveis em [1]) usadas no teste, 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, uma 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 causariam 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, 15, por serem causadas por variações grandes dentro de um único intervalo, não foram precedidas de Alarmes, que ocorreram no momento da falha apenas. Esses 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 ocorrência 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 49, 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 mais relevantes estão resumidas na Tabela 3.

Tabela 3 - Estatísticas consideradas relevantes no terceiro teste

Fato	*Quantidade*	
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*	*4,0 %*
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.

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.

A apreciação final que pode ser feita a partir do principal resultado desse teste, que foi a previsão de 49 % das falhas de QoS com um tempo entre 5 e 15 segundos de antecedência, é que, apesar da ocorrência de alarmes falsos, o mecanismo proposto pode ser efi-

caz, dependendo da configuração dos valores limites para a variação.

5. Conclusões e trabalhos futuros

O trabalho integrou áreas de conhecimento novas na computação e que estão sendo utilizadas em muitas pesquisas com a finalidade de realizar o gerenciamento de desempenho pró-ativo em tempo real. 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, já é possível.

As principais limitações que 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, a ocorrência de Alarmes falsos e a escalabilidade.

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 ou elementos críticos sejam gerenciados 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 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 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 proposta.

O uso da tecnologia ativa 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 também, portanto, um trabalho futuro, a definição de um modelo de segurança que permita que as funcionalidades previstas na arquitetura sejam desencadeadas sem riscos para proprietários das redes e estações quanto para os usuários.

6. Referências

[1] Cecilio, E. L. “Uma Arquitetura de Gerenciamento de Desempenho Pró-Ativo Distribuído Usando Tecnologia Ativa”, Dissertação de Mestrado, NCE/IM/UFRJ, Dezembro de 2002.

[2] Fuggeta, A., Picco, G.P. e Risso, F., “Understanding Code Mobility”, IEEE Transaction on Software Engineering, 24(5):146-155, Abril de 1998.

[3] Dumont, A. P. M, “AGAD: Uma Arquitetura de Gerenciamento Ativo Distribuído”, Dissertação de Mestrado, NCE/IM/UFRJ, Outubro de 2002.

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

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

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

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

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

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

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

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

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

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

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

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

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

[17] 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.

[18] Cecilio, E. L., Dumont, A. P., Delicato, F. C., Rust, L. F. e Pirmez, L., "A Viabilidade do Gerenciamento de Desempenho Pró-ativo Baseado em uma Arquitetura de Gerenciamento que usa Tecnologia Ativa", Anais do 20º SBRC, Búzios, RJ, Maio de 2002.