Rerroteamento de fluxos utilizando redes MPLS e tecnologia ativa

Reinaldo de Barros Correia

Dissertação de Mestrado

Luiz Fernando Rust da Costa Carmo, Dr. UPS, França, 1994
Luci Pirmez, D.Sc., COPPE/UFRJ, Brasil, 1996

Instituto de Matemática / Núcleo de Computação Eletrônica
Universidade Federal do Rio de Janeiro
Rio de Janeiro, Março de 2003

Rerroteamento de fluxos utilizando redes MPLS e tecnologia ativa

Reinaldo de Barros Correia

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:

_____

Prof°. Luiz Fernando Rust da Costa Carmo – Orientador

Dr., UPS, França

_____

Profª. Luci Pirmez – Orientadora

D. Sc., COPPE/UFRJ, Brasil

_____

Prof°. Oswaldo Vernet de Souza Pires - Membro da banca

D. Sc., COPPE/UFRJ, Brasil

_____

Prof°. Aloysio de Castro Pinto Pedrosa - Membro da banca

Dr., UPS, França

Rio de Janeiro – RJ
Março de 2003

FICHA CATALOGRÁFICA

CORREIA, REINALDO BARROS.

Rerroteamento de Fluxos Utilizando Redes MPLS e Tecnologia Ativa, [Rio de Janeiro], 2003.

xii, 84 p., 29,7 cm (IM/NCE/UFRJ, MSc., Informática, 2003)

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

1. Roteamento baseado em QoS 2. Infra-estrutura MPLS 3. Tecnologia Ativa

I. IM/NCE/UFRJ II. Título (série)

Agradecimentos

À minha mãe, Francelina, e à minha Irmã, Maria Cristina, pelo apoio irrestrito e, principalmente, por terem assumido minhas obrigações.

Ao meu cunhado, Carlos Alberto e à minha Irmã, Lucina, pelo entusiasmo e incentivo durante o curso.

Aos meus amigos Ignácio Tito e Wilson Marioni que pouparam me tempo precioso na reta final desse trabalho, assumindo muitas de minhas responsabilidades.

Aos meus novos amigos Edmundo Cecilio, Alexandre Lages, Ana Paula Dumont, Renata Corrêa, Noel Bastos, Werner Priess, José Coelho e Roberta Gomes que amenizaram o meu esforço para retornar às atividades acadêmicas após longo tempo.

Aos professores da banca Aloysio de Castro e Oswaldo Vernet pela dedicação na correção da Dissertação e sugestões apresentadas.

Aos meus orientadores Luci Pirmez e Luiz Fernando Rust pela paciência, oportunidade e, sobretudo, pela confiança depositada.

Ao NCE/UFRJ pela infra-estrutura, fruto da dedicação de seus funcionários.

Abstract

CORREIA, Reinaldo Barros. **Rerouting Application Flows on MPLS Networks Using Active Technology**. Advisors: Luiz Fernando Rust da Costa Carmo and Luci Pirmez. Rio de Janeiro. UFRJ/IM/NCE, 2003. Diss.

In the past few years the massive deployment of real-time-multimedia services in the Internet has motivated the research community to investigate new QoS mechanisms in order to overcome IP network limitations. Since assured level of service must be provided, these mechanisms should interact flexibly with performance management systems. In this way, it is possible to trigger actions precisely and effectively to recover from QoS failures and simultaneously to balance network load. Mechanisms for rerouting application flows may be employed to help performance management systems in supporting QoS. In addition, if both systems work pro-actively, actions can be taken before applications are affected by QoS faults. To meet theses goals, a pro-active rerouting architecture, which provides a means of establishing new routes with sufficient resources for application flows, is introduced. Our architecture stems on QoS routing concept, virtual circuit based networks and active technology. A prototype was implemented to reroute flows with bandwidth and delay constraints on an MPLS infrastructure. Test results show that the deployment of the proposed architecture is feasible considering the hardware processing capacities available today. It was also possible to establish the limit of our architecture in terms of virtual circuit length.

Resumo

Correia, Reinaldo de Barros. **Rerroteamento de Fluxos Utilizando Redes MPLS e Tecnologia Ativa**. Orientadores: Luiz Fernando Rust da Costa Carmo e Luci Pirmez. Rio de Janeiro. UFRJ/IM/NCE, 2003. Diss.

Nos últimos anos, a crescente implantação de serviços multimídia e, também, a grande diversidade dos serviços disponibilizados têm motivado o setor privado e a comunidade acadêmica a pesquisar novas arquiteturas e mecanismos para suplantar as limitações das redes IP atuais. Diante deste cenário em que níveis assegurados e rapidez na implantação de serviços são vitais, o gerenciamento de desempenho deve dispor de novos mecanismos com interfaces flexíveis para que possa atuar de maneira rápida e eficiente nos nós da rede a fim de manter os níveis da QoS das aplicações e simultaneamente otimizar a utilização de recursos. Em particular, o rerroteamento de fluxos é um desses novos mecanismos que pode ser disponibilizado ao gerenciamento de desempenho. Além disso, funcionalidades pró-ativas devem ser incorporadas tanto ao sistema de gerenciamento de desempenho quanto ao rerroteamento para que ações, a partir de detecção de tendências de falhas de QoS, sejam desencadeadas antes da ocorrência das falhas, tornando os seus efeitos o mais transparente possível às aplicações. Neste trabalho, é apresentada uma arquitetura de rerroteamento pró-ativo com o objetivo de redirecionar fluxos por rotas alternativas com recursos suficientes mediante solicitações externas (por exemplo pelo gerenciamento de desempenho). A arquitetura de rerroteamento considera como infra-estruturas de rede a tecnologia ativa, o roteamento baseado em QoS e redes baseadas em circuitos virtuais. Sob o protótipo, que foi implementado especificamente para o rerroteamento de fluxos com restrição de banda e retardo em um infra-estrutura de rede MPLS, executou-se uma bateria de testes a fim de averiguar a viabilidade da arquitetura de rerroteamento. Os testes serviram, também, para estabelecer os limites do emprego da arquitetura de rerroteamento proposta.

Sumário

Lista de figuras

Lista de figuras (continuação)

Lista de tabelas

Lista de Siglas e Abreviaturas

AGAD	*Arquitetura de Gerenciamento Ativa e Distribuída*
AS	*Autonomous System*
ATM	*Asynchronous Transfer Mode*
BGP	*Border Gateway Protocol*
CMIP	*Common Management Information Protocol*
CORBA	*Common Object Request Broker Architecture*
CR-LDP	*Constraint Routing Label Distribution Protocol*
FEC	*Forwarding Equivalent Class*
FIB	*Forwarding Information Base*
FIFO	*First in First out*
ICMP	*Internet Control Message Protocol*
IETF	*Internet Engineering Task Force*
IP	*Internet Protocol*
LDP	*Label Distribution Protocol*
LER	*Label Edge Router*
LIB	*Label Information Base*
LSA	*Link State Advertisements*
LSP	*Label Switched Path*
LSR	*Label Switching Router*
MPLS	*Multiprotocol Label Switching*
OSPF	*Open Shortest Path First*
QoS	*Quality of Service*
RIP	*Routing Information Protocol*
RMI	*Remote Method Invocation*
RSVP	*Resource ReServation Protocol*
SLA	*Service Level Agreement*
SNMP	*Simple Network Management Protocol*
SPF	*Shortest Path First*
TCP	*Transport Control Protocol*
TE	*Traffic Engineering*
WDM	*Wavelength Division Multiplex*

Capítulo 1 – Introdução

A crescente comercialização da Internet, que na visão de vários empreendedores é um instrumento de alavancamento de seus negócios, juntamente com a convergência de várias formas de mídia, vem exigindo dos provedores, além da diversificação, o fornecimento de novos serviços (telefonia, vídeo conferência, vídeo sob demanda, etc) com fortes requisitos de qualidade de serviço (QoS).

O protocolo IP foi idealizado com o objetivo de ser robusto, não sendo, portanto, capaz de atender as demandas advindas desse novo cenário de forma simples. O paradigma de entrega apenas por melhor esforço (*best-effort delivery*) não consegue atender a severas restrições de QoS porque é estabelecida uma comunicação sem garantia de entrega.

A política de encaminhamento por melhor esforço vem atendendo, até certo ponto, de forma eficiente, àqueles tráfegos oriundos de aplicações com baixos requisitos de QoS, tais como transferência de arquivos, correio eletrônico e acesso remoto. Todavia, a disponibilização em larga escala de aplicações multimídia exigirá que a infra-estrutura vigente evolua para um ambiente no qual deverão estar implementados esquemas de priorização de tráfegos, de controle de admissão de fluxos e de garantia de QoS, entre outros. Além disso, a futura arquitetura deverá ser flexível de maneira a possibilitar a rápida disponibilização de novos serviços, à medida que sejam concebidos, sem a longa espera dos processos de padronização.

1.1 – Motivações

Apesar de o esquema *best effort* propiciar o compartilhamento dos recursos da rede (*buffers*, banda passante, etc), requisitos de QoS fim a fim não podem ser garantidos. Outra dificuldade é a não garantia da ordem de chegada de pacotes de um mesmo fluxo, uma vez que caminhos distintos podem ser usados entre origem e destino. A utilização do protocolo TCP na camada de transporte não é apropriada devido à introdução de retardos inaceitáveis para aplicações multimídia. Logo, torna-se complexa a implantação de serviços, principalmente multimídia e em tempo real, para os quais severos requisitos de QoS devem ser garantidos.

A infra-estrutura de rede que fornecerá níveis garantidos de serviços às novas aplicações deverá não somente dispor de tecnologias de hardware mais complexas, mas também de arquiteturas de *software* com mecanismos que deverão atuar desde o nível de aplicação (gerenciamento de rede) até a camada de enlace (mecanismos de escalonamento de quadros com prioridade).

O roteamento baseado em restrições de QoS pode ser considerado como sendo um passo fundamental no processo de transição para uma infra-estrutura de comunicação com níveis garantidos de serviços. Nesse caso, as rotas são calculadas levando-se em conta restrições de QoS. Entretanto, esse tipo de roteamento impõe, em relação ao roteamento *best-effort*, uma maior sobrecarga de comunicação devido à maior quantidade de informações que devem ser trocadas e atualizadas. Além disso, como os algoritmos de roteamento são mais sofisticados, os roteadores ficam submetidos a uma maior sobrecarga computacional [1].

Um passo vital para o oferecimento de suporte à crescente demanda de aplicações multimídia é o emprego da arquitetura *Multiprotocol Label Switching* (MPLS) [2]. O MPLS traz para a Internet os benefícios inerentes às tecnologias de rede orientada a conexão, como ATM e *Frame Relay*, permitindo a implantação de serviços multimídia de forma escalável e segura. O gerenciamento dessas redes é simplificado.

A gerência de redes, conforme padronizada em [3], é divida em cinco áreas funcionais: configuração, segurança, desempenho, falha e contabilidade. O gerenciamento de desempenho é responsável por monitorar o estado da rede e os parâmetros de QoS dos fluxos das aplicações que possuem níveis de serviços assegurados. Em função dos resultados desse monitoramento, o gerenciamento deve desencadear ações nos diversos mecanismos existentes nos nós e estações da rede. Essas ações visam evitar ou minimizar os efeitos das falhas de QoS nos serviços que estão sendo prestados. A garantia de níveis de serviço é expressa em acordos de níveis de serviços (SLA – *Service Level Agreement*) em nível usuário e em contratos de tráfego em nível rede. Esses acordos e contratos quantificam limites para os parâmetros de QoS que devem ser monitorados. O gerenciamento de desempenho, ao constatar o desrespeito aos limites de QoS acordados, deve localizar os dispositivos responsáveis e desencadear ações com o intuito de reverter essa situação. Esta forma de atuação é dita reativa porque as ações são realizadas após a ocorrência da falha de QoS. Este tipo de abordagem tem a desvantagem de não ser transparente, ou seja, a qualidade da apresentação sofre degradações que, em geral, são perceptíveis aos usuários. Logo, é recomendável que o gerenciamento de desempenho desencadeie ações de forma pró-ativa. Nesse caso, ao serem detectadas tendências de ocorrência de falhas de QoS, o gerenciamento de desempenho deve atuar antecipadamente, tentando minimizar ou até mesmo eliminar os efeitos das falhas nas aplicações ou, devido ao desencadeamento dessas ações antecipadas, evitar que as falhas ocorram. Exemplos dessas ações são o disparo de adaptações em aplicações multimídia adaptativas, conforme previsto em [4] [5], a mudança de políticas de

escalonamento de pacotes e solicitações de rerroteamento de fluxos aos mecanismos da camada de rede.

O Rerroteamento vem sendo considerado um importante componente para dotar as redes IP com um maior grau de tolerância a falhas de enlaces e nós. Apesar de os protocolos IP (OSPF, RIP, BGP) propiciarem uma certa robustez, recalculando as tabelas de roteamento em função de queda de enlaces e nós, os tempos de recuperação de falhas são elevados. Esses tempos podem ser da ordem de dezenas de minutos [38], dependendo do tamanho da rede. Isto inviabiliza o rerroteamento de fluxos de aplicações multimídia em tempo real diante de falhas de QoS. Vários trabalhos têm sido apresentados [39] [40] [41] para minimizar esses tempos de recuperação.

Um dos grandes obstáculos na implementação do Gerenciamento de Desempenho Pró-ativo é o fato de que os mecanismos disponíveis em roteadores, comutadores ou estações são implementados de forma monolítica nos sistemas operacionais desses dispositivos. A interação com mecanismos externos é quase inexistente e, quando disponível, é difícil e precária.

Em [6] é proposto o uso da automação das atividades do Gerenciamento de Desempenho Pró-ativo para fazer frente às dificuldades que advêm do porte crescente das redes atuais e, também, dos requisitos funcionais cada vez mais severos que essas redes deverão atender no futuro. Essa automação somente poderá ser materializada se os dispositivos e mecanismos gerenciados oferecerem o suporte ao desencadeamento de ações pelo gerenciamento de desempenho. O ideal é que o gerenciamento de desempenho pró-ativo e os mecanismos dos dispositivos gerenciados interajam de forma flexível, síncrona ou assíncrona e bidirecional.

O emprego da tecnologia ativa[1] como base na concepção e implementação dos mecanismos das estações e dos dispositivos de redes disponibiliza o alto grau de flexibilidade necessário na interação com o Sistema de Gerenciamento Pró-ativo. Além disso, ao dotar os nós e estações com recursos computacionais e permitir a execução de programas móveis sob demanda, a tecnologia ativa permite que as tarefas de atualização de código e configuração dos dispositivos sejam efetuadas com maior rapidez.

[1] Conforme será explicado, tecnologia ativa consiste no uso dos paradigmas de redes ativas e agentes móveis.

1.2 – Objetivo

Este trabalho apresenta uma arquitetura genérica de rerroteamento pró-ativo cujo objetivo é redirecionar fluxos de aplicações mediante solicitações externas tais como aquelas geradas pelo Gerenciamento de Desempenho Pró-ativo [6]. A arquitetura proposta utiliza os princípios do roteamento baseado em QoS e tem como infra-estrutura uma tecnologia de enlace orientada a conexão e a tecnologia ativa.

Um protótipo especializado para fluxos com restrição de banda e retardo e para um infra-estrutura de rede MPLS (*Multiprotocol Label Switching*) foi desenvolvido com o intuito de verificar a viabilidade da arquitetura proposta no redirecionamento de fluxos de aplicações multimídia. Os resultados dos testes foram confrontados com os resultados obtidos em [6] para determinar os limites do emprego da arquitetura proposta. O Sistema ServiMídia [4] [5], desenvolvido no NCE para apresentação de aplicações multimídia adaptativas, serviu como cenário dos testes realizados no protótipo.

1.3 – Organização

O restante da dissertação está estruturado em cinco capítulos. No segundo capítulo, são apresentados os conceitos básicos necessários ao entendimento da Arquitetura de Rerroteamento Pró-ativo proposta neste trabalho e ao desenvolvimento do protótipo. Estão descritos os princípios que regem o funcionamento do roteamento *best-effort* e do baseado em QoS, assim como da arquitetura MPLS. O terceiro capítulo detalha a arquitetura proposta através da descrição de seu funcionamento, dos seus componentes e, também, da interação entre os mesmos. O quarto capítulo é dedicado ao ambiente de desenvolvimento escolhido para implementação do protótipo, cujo funcionamento é detalhado por meio de diagramas de seqüência. O quinto capítulo descreve os testes realizados no protótipo, abordando as medidas adotadas a título de simplificação e tecendo comentários sobre o impacto de tais medidas nos resultados. A análise dos resultados salienta as limitações da arquitetura de rerroteamento proposta. O trabalho é finalizado no sexto capítulo, onde são apresentadas as conclusões finais e os possíveis trabalhos futuros.

Capítulo 2 – Conceitos Básicos

Este capítulo apresenta os conceitos básicos que fundamentam a Arquitetura de Rerroteamento Pró-ativo proposta e o desenvolvimento do protótipo: gerenciamento distribuído usando tecnologia ativa, roteamento baseado em QoS, infra-estrutura MPLS e rerroteamento pró-ativo.

A seção 2.1 introduz de forma gradual os sistemas de gerenciamento. A seção 2.2 apresenta as principais abordagens de roteamento e as suas respectivas vantagens e desvantagens. Os principais componentes da arquitetura MPLS são apresentados na seção 2.3. A seção 2.4 define o conceito de rerroteamento e, por último, a seção 2.5 apresenta algumas considerações finais.

2.1 – Gerenciamento

As primeiras implementações de gerenciamento consistiam na coleta de informações dos elementos gerenciáveis por uma estação central para processamento. Quando o porte das redes aumentou, esse esquema centralizado utilizado no SNMP [7] e CMIP [8] [9] mostrou-se impróprio devido a vários fatores: (i) excesso de processamento na estação central; (ii) elevado consumo de banda por causa do excessivo tráfego de gerenciamento; (iii) saturação dos enlaces e nós próximos à estação central; (iv) retardo das ações e (v) imprecisão das informações, fruto da latência das mensagens de gerenciamento [10].

Com o objetivo de superar essas limitações da abordagem centralizada, vários esquemas de gerenciamento distribuídos vêm sendo propostos. A idéia é, ao invés de realizar a coleta das informações para posterior tratamento em um ponto central, levar o processamento para o mais próximo possível dos elementos a serem gerenciados. A hierarquização do gerenciamento [7] [11] [12], dividindo a rede em domínios com cada domínio possuindo uma estação de gerenciamento foi uma das primeiras tentativas. Estudos recentes apontam o uso da tecnologia ativa [13] como um meio eficaz de oferecer maior flexibilidade às tarefas de gerenciamento sem o ônus e complexidade impostas por abordagens que utilizam objetos distribuídos como CORBA [14] e RMI [15].

A tecnologia ativa [13] é a utilização de pelo menos um entre os paradigmas de agentes móveis, de redes ativas e de redes programáveis tanto na infra-estrutura de rede (roteadores, comutadores MPLS, *Switchers*, *hubs*) quanto nas estações, possibilitando a instalação de programas sob demanda. Esses programas são dotados de capacidade de processamento e

realizam o tratamento das informações localmente. Isto diminui o tráfego das mensagens de gerenciamento, economizando banda nos enlaces e propiciando maior rapidez nas ações.

A crescente adoção de serviços multimídia nas redes IP tornou o gerenciamento, em especial, aquele que contempla a área funcional de desempenho [16], um dos componentes mais importantes no provimento de níveis garantidos de QoS. O gerenciamento de desempenho tem a responsabilidade de monitorar os parâmetros de QoS, analisar os dados coletados, desencadear ações quando os níveis dos parâmetros monitorados excedem limites pré-estabelecidos, gerar relatórios, planejar expansão de capacidade instalada em função dos dados coletados e promover o balanceamento de carga para maximizar a utilização de recursos eliminando congestionamentos localizados de enlaces e nós (*hot spots*). Exemplos dessas ações são o desencadeamento do processo de adaptação em aplicações multimídia adaptativas [4] ou o redirecionamento de um fluxo através de rotas menos congestionadas.

O gerenciamento de desempenho pode adotar dois tipos de estratégias: a reativa e a pró-ativa. A reativa é aquela em que as ações corretivas são efetuadas após a falha de QoS ter ocorrido. A desvantagem é que as degradações na qualidade das aplicações são sentidas pelos usuários. Na estratégia pró-ativa [17] [18], as ações são efetivadas quando da detecção de tendência de ocorrência de falhas de QoS. O gerenciamento de desempenho pró-ativo é realizado tanto a partir de valores atuais obtidos durante o monitoramento dos parâmetros de QoS, quanto de valores passados que compõem o *baseline* da rede. Com essas informações, valores futuros para os parâmetros monitorados podem ser inferidos por intermédio de algum algoritmo. Caso esses valores futuros sejam maiores que os limites aceitáveis para uma determinada aplicação, um alarme de tendência é gerado. Neste caso, como as ações são realizadas antes da ocorrência da falha de QoS, os efeitos sobre as aplicações podem ser minimizados ou, até mesmo, eliminados. Outra conseqüência dessa abordagem pró-ativa é que as falhas podem não vir a ocorrer, devido à redução na utilização de recursos em virtude das ações de adaptação de apresentações multimídia ou, então, devido à utilização de rotas alternativas com recursos disponíveis após ações de rerroteamento de fluxos.

2.2 – Roteamento

O roteamento IP consiste, basicamente, na execução simultânea de duas tarefas ou processos nos roteadores. A primeira é a construção e atualização da tabela de roteamento. Esta tarefa exige o envio, processamento e recebimento de mensagens de atualização de estado. Essas mensagens contêm as informações necessárias à construção e atualização da tabela de

roteamento e são armazenadas em uma base de dados. O algoritmo de roteamento, com as informações dessa base de dados, calcula as rotas e constrói ou atualiza a tabela de roteamento. A segunda tarefa é o encaminhamento dos pacotes que chegam ao nó. Assim que um pacote chega, o sistema de roteamento extrai, entre outras informações, o endereço destino para então efetuar a busca na tabela de roteamento e determinar a interface por onde o pacote deve ser enviado.

A primeira tarefa é aquela que distingue o roteamento *best-effort* daquele baseado em QoS. O tipo *best-effort* considera no cálculo das rotas apenas o caminho mais curto. O roteamento baseado em QoS calcula os caminhos entre origem e destino considerando uma restrição ou conjunto de restrições de QoS como, por exemplo, retardo, banda e variação do retardo. Dependendo do algoritmo de roteamento utilizado, o caminho obtido pode ser ótimo ou não.

2.2.1 – Roteamento IP best-effort

A tarefa de construção e manutenção das tabelas de roteamento é executada pelos protocolos[2] de roteamento que estão instalados nos roteadores. O protocolo de roteamento intra-domínio mais utilizado na Internet é *Open Shortest Path First* (OSPF) [19]. O OSPF é um protocolo de roteamento hierárquico criado pelo *Internet Engineering Task Force* (IETF) em substituição ao *Routing Information Protocol* (RIP). O OSPF define um Sistema Autônomo (AS – *Autonomous System*) como sendo uma coleção de redes subordinadas a uma mesma administração e que compartilham a mesma estratégia de roteamento. Um AS é dividido em áreas ou domínios. O OSPF é um protocolo de estado de enlace, que envia por inundação (limitada) para todos os roteadores pertencentes à mesma área, mensagens de estado chamadas de *Link State Advertisements* (LSA), contendo informações das interfaces e métricas utilizadas. Essas informações são mantidas em uma base de dados. Os roteadores usam o algoritmo *Shortest Path First* (SPF) para calcular a melhor rota para todos os destinos, assumindo como origem o próprio roteador.

O sistema possui duas estruturas de dados para o tratamento das informações necessárias ao roteamento: a ***R**outing **I**nformation **B**ase* (RIB) e a ***F**orwarding **I**nformation **B**ase* (FIB). Na RIB estão as informações das rotas calculadas a partir das informações topológicas da rede obtidas por meio das mensagens de atualização de estado do protocolo de roteamento. A FIB

[2] Na realidade, o termo mais apropriado seria sistema de roteamento. No entanto, popularizou-se o emprego do termo protocolo de roteamento para referência a todas as operações realizadas por um sistema de roteamento.

contém as mesmas informações que a RIB, entretanto, os dados são organizados de maneira a otimizar o encaminhamento.

2.2.2 – Roteamento IP baseado em QoS

O roteamento baseado em QoS é um caso particular do roteamento baseado em restrição, que também pode ser particularizado em roteamento baseado em política. A Figura 1 apresenta um cenário no qual pode-se fazer uma clara distinção entre esses dois tipos de roteamento. Caso o roteador "A" estivesse executando um protocolo *best-effort* e, considerando que a métrica utilizada fosse o número de saltos (*hops*), o caminho escolhido para alcançar o roteador "H" seria através do enlace AH (zero saltos). Entretanto, se o algoritmo de roteamento fosse baseado em QoS e estabelecesse restrição de banda de 10 Mbit/s, os caminhos AEH, ABDH e ACFGH poderiam ser escolhidos, mas não o caminho AH, pois este não atende a restrição de banda mínima. Entre os caminhos possíveis, o AEH é o de menor custo (considerando a métrica salto), embora a sua escolha dependesse do algoritmo utilizado, porque nem todos selecionam o mais curto dentre os possíveis. Para uma aplicação com uma restrição de retardo fim a fim (de A para H) de, no máximo, 40ms, a rota selecionada seria ACFGH, apesar de apresentar o maior numero de saltos. Finalmente, o caminho AEH poderia ser selecionado ao invés do caminho AH (mais curto) devido a questões de segurança, caso o roteamento fosse baseado em política.

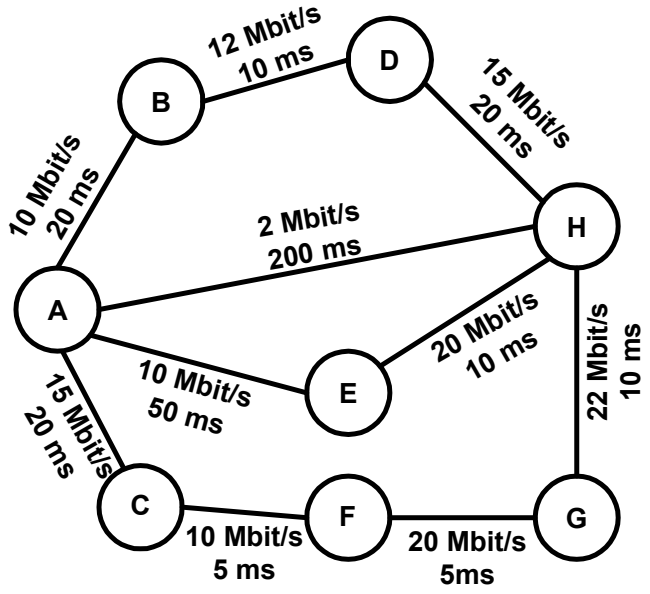


Figura 1 – Cenário para comparação dos vários tipos de roteamento.

O roteamento baseado em QoS necessita de informações de topologia da rede que já são disponibilizadas pelos protocolos de roteamento tradicionais. Entretanto, as informações adicionais do estado dos nós de comutação e dos enlaces devem ser fornecidas por intermédio de novos protocolos de roteamento ou protocolos de roteamento *best-effort* estendidos. A extensão do OSPF para o roteamento baseado em QoS [22] é um exemplo deste último.

Essas informações de estado são os valores das métricas de QoS tais como banda passante disponível nos enlaces, retardo, variação do retardo, taxa de erro nos enlaces, taxa de ocupação de CPU, etc. Todos esses valores locais são disponibilizados em todos os nós da rede por intermédio das mensagens de atualização de estado dos protocolos de roteamento. Desta forma, cada nó com as informações globais de estado da rede pode individualmente calcular a tabela de roteamento, utilizando algum algoritmo de roteamento.

Outras informações necessárias ao algoritmo de roteamento baseado em QoS são as restrições de QoS dos fluxos que devem ser fornecidas por algum protocolo de sinalização como, por exemplo, o RSVP [20] ou CR–LDP [21].

As rotas podem ser calculadas sob demanda ou pré-computadas. Os prós e contras de uma ou outra abordagem estão descritos em [1]. A vantagem do cálculo sob demanda está no fato de que as informações topológicas e de estado são mais atuais, levando a resultados mais precisos e maximizando assim a eficiência do protocolo de roteamento. Todavia, caso a taxa de requisições (mensagens de sinalização) seja elevada, o nó estará submetido a uma carga computacional elevada mesmo que execute algoritmos de baixa complexidade. Em [23] e [24] é proposto o uso de *path caching*, que consiste no reuso de caminhos previamente computados no cálculo das rotas de forma a diminuir a complexidade do algoritmo.

A abordagem de pré-computar as rotas em intervalos de tempo regulares tem o inconveniente de impor à CPU do nó uma maior carga computacional, porque toda a tabela tem que ser recalculada. A abordagem sob demanda, por sua vez, calcula somente um único caminho sob alguma restrição de QoS. Outro inconveniente de pré-computar as rotas é que o valor das restrições de QoS (banda passante, retardo) solicitadas nas requisições não são conhecidas a priori. A proposta apresentada em [22] define vários caminhos para cada destino. Esses caminhos estão associados aos valores, por exemplo, de banda passante possíveis nas requisições constantes nas mensagens do protocolo de sinalização. Neste caso, a quantidade de informações na tabela de roteamento (FIB) é substancialmente maior, infligindo não somente um aumento da quantidade de memória como também do custo computacional. Um

ponto favorável é que a freqüência de recálculo da tabela é um parâmetro que pode ser controlado pelo roteador enquanto que a freqüência de solicitações de conexão é função do número de aplicações que estão utilizando os serviços da rede. Logo, a freqüência de computação da tabela de roteamento pode ser ajustada de maneira que se obtenha um bom compromisso entre a eficiência do protocolo de roteamento e o custo computacional.

Uma questão relevante é o aumento do tráfego na rede e da carga computacional devido às mensagens de sinalização, principalmente se a freqüência de requisições de conexão for elevada em função de um grande número de fluxos de pequena duração. Várias propostas tem sido elaboradas para aliviar o seu impacto: [25] sugere limitar o uso do roteamento baseado em QoS para fluxos de longa duração; já em [26], o uso do roteamento baseado em QoS está restrito à agregação de fluxos, enquanto [27] e [28] demonstram como tornar os protocolos de sinalização mais eficientes.

O controle das mensagens de atualização do protocolo de roteamento é mais crítico quando restrições de QoS são incorporadas no cálculo de rotas, já que algumas métricas de QoS são muito dinâmicas e as mensagens são enviadas através da técnica de inundação. O ideal seria enviar mensagens de atualização para qualquer alteração de valor das métricas, uma vez que as informações de estado ficam mais precisas, tornando o protocolo de roteamento mais eficiente. Todavia, os benefícios seriam anulados pelas sobrecargas computacional e de comunicação impostos à rede.

As funções de disparo têm a responsabilidade de controlar a freqüência do envio das mensagens de atualização de topologia e de estado aos outros nós da rede de maneira que o consumo de banda passante nos enlaces e carga computacional para, respectivamente, o transporte e processamento dessas mensagens, sejam limitados em níveis aceitáveis.

Várias funções de disparo são implementadas para se alcançar um equilíbrio entre o desempenho do protocolo de roteamento e a carga computacional nos nós e de comunicação nos enlaces: (i) uso de um temporizador para limitar a freqüência das mensagens de atualização a fim de se ter um controle do volume total de mensagens enviadas; (ii) uso de detectores de níveis que disparam as mensagens de atualização em função da variação do valor, ou seja, sempre que o valor medido for maior que um percentual pré-estabelecido em relação ao valor anunciado anteriormente, ocorrerá o disparo das mensagens; e (iii) temporizador de curta duração (*hold-down-timer*) que garante um tempo mínimo entre atualizações para eliminar os efeitos de sobrecarga devido a flutuações rápidas de tráfego.

2.3 – Comutação de Rótulos – MPLS

A tecnologia MPLS cria um circuito virtual, o *Label Switched Path* (LSP) para o encaminhamento de pacotes de um fluxo usando apenas a comutação baseada na troca de rótulos. Dessa forma, é estabelecida uma transmissão orientada à conexão sob uma infra-estrutura IP, que é sem conexão. Uma vez construído o LSP, os pacotes são submetidos à disciplina FIFO de escalonamento de pacotes em todos os roteadores/comutadores (LSR – *Label Switch Routers*) do LSP, chegando em ordem ao destino.

No esquema IP tradicional, cada roteador toma decisões de encaminhamento independentemente, baseadas apenas nas informações contidas no cabeçalho do pacote IP. Na arquitetura MPLS, os pacotes são tratados de acordo com a posição do comutador no LSP. Os LSR de borda ou *Label Edge Routers* (LER), em particular, o de entrada, classifica os pacotes de acordo com alguma regra pré-estabelecida, utilizando as informações de nível 3 e/ou de nível 4 do cabeçalho. A classificação consiste em atribuir ao pacote uma classe (FEC – *Forwarding Equivalent Class*) e associar um número de rótulo que deve ser anexado à frente do cabeçalho IP. A FEC corresponde a um grupo de pacotes que têm o mesmo tratamento nos comutadores pertencentes ao LSP associado a esta FEC.

Como as regras de associação podem ser tão complexas quanto se deseje ou se necessite, este processamento pode tornar-se oneroso ao LSR. Vale ressaltar que mesmo os pacotes já rotulados podem passar por este processo de classificação, porque a arquitetura MPLS [2] permite a criação de hierarquias de domínios MPLS. Isto acarreta em um número variável de rótulos (pilha de rótulos) anexados à frente do pacote IP, dependendo do número de domínios existentes na rede.

Os LSR intermediários, ao receberem os pacotes rotulados, realizam o encaminhamento baseados única e exclusivamente no número do rótulo. É realizada a consulta a uma tabela de rótulos, a *Label Information Base* (LIB) para determinar a interface de saída em direção ao próximo LSR e também o rótulo de saída que substitui aquele recebido.

O LER de Saída quando recebe um pacote rotulado, retira o rótulo e, em seguida, faz o encaminhamento da forma usual.

Os rótulos podem ser criados baseados nas tabelas de roteamento (OSPF, BGP), nas requisições de conexão (RSVP) ou nas informações dos pacotes durante a recepção. Após

serem criadas, as associações de rótulos e FEC's devem ser anunciadas aos demais LSR para que o LSP seja criado.

A distribuição de rótulos pode ser efetuada por intermédio de protocolos existentes e modificados como, por exemplo, BGP e OSPF ou, então, através de novos protocolos especialmente definidos como é o caso do LDP (*Label Distribution Protocol*) do IETF. Existem extensões do LDP para suportar roteamento explícito baseado em QoS (CR–LDP).

O esquema utilizado nas redes MPLS traz diversas vantagens: melhor desempenho no encaminhamento porque os comutadores intermediários não levam em consideração as informações do cabeçalho IP, os pacotes podem ser encaminhados baseados em informações que não estão disponíveis no roteamento IP, garantia de QoS pode ser estabelecida atribuindo prioridades à FEC e suporte à engenharia de tráfego [30] [31].

2.4 – Rerroteamento

O rerroteamento em redes baseadas em circuito virtual pode ser definido como o elenco de operações realizadas junto aos comutadores e roteadores para redirecionar fluxos previamente estabelecidos através de caminhos redundantes. O rerroteamento é comumente utilizado para alcançar três objetivos [42]: (i) atendimento a políticas administrativas da rede; (ii) estabelecimento de novos perfis de tráfego; (iii) aumento do nível de tolerância a falhas.

A principal vantagem do rerroteamento é o tempo reduzido para o restabelecimento da nova rota em relação aos protocolos de roteamento tradicionais (OSPF, RIP, IGRP, etc). Apesar desses protocolos dotarem a rede com níveis de tolerância a falhas elevados, os tempos de convergência inviabilizam aplicações de missão crítica e multimídia em tempo real. Outra vantagem é o aproveitamento de recursos ociosos, uma vez que os melhores caminhos estabelecidos pelos protocolos de roteamento podem não ser necessariamente aqueles que melhor atendem os requisitos das aplicações.

O rerroteamento só é possível caso existam recursos ociosos no momento da solicitação. A carência de recursos pode ocorrer de forma localizada ou generalizada na rede. Caso a solicitação de rerroteamento seja devida a causas localizadas, o rerroteamento provavelmente poderá ser efetuado. Caso a razão seja uma insuficiência de recursos generalizada, a solicitação de rerroteamento certamente será rejeitada, a menos que seja efetuada a preempção de fluxos ou grupos destes, liberando assim recursos para o fluxo que está sendo rerroteado.

O aumento acentuado de falhas de QoS indica uma condição de saturação da infra-estrutura da rede para a carga de tráfego submetida ou, então, uma má distribuição deste tráfego produzindo áreas sobrecarregadas e outras com recursos ociosos. Logo, uma elevada taxa de rejeição de solicitações de rerroteamento de fluxos, ou um grande número de preempções de fluxos sugere que a rede está saturada e que é necessária uma ampliação de capacidade.

2.4.1 – Rerroteamento pró-ativo

A abordagem tradicional adotada no rerroteamento é reativa, uma vez que as ações são realizadas após a ocorrência da falha. No rerroteamento pró-ativo, todas as operações possíveis são realizadas antes de a falha ocorrer. Desta forma, é possível evitar os efeitos negativos produzidos nas aplicações que estão utilizando os serviços de comunicação da rede. No pior caso, esses efeitos são minimizados com tempos de duração reduzidos.

O rerroteamento pró-ativo e o Gerenciamento de Desempenho Pró-ativo (GDPA) [6] podem atuar conjuntamente em prol de uma aplicação multimídia com o objetivo de manter a QoS durante sua apresentação. O GDPA diante de tendências de falhas de QoS, dispara antecipadamente o pedido para reformatação de aplicações multimídia adaptativas. Esse esquema compensa o retardo introduzido por essas adaptações, eliminando ou minimizando os seus efeitos. Entretanto, estas falhas de aplicação podem até mesmo ser eliminadas caso o GDPA solicite o rerroteamento do fluxo, corrigindo as falhas de QoS no serviço de comunicação e evitando assim a necessidade de reformatar a apresentação multimídia

O rerroteamento pró-ativo é a ação de substituir um circuito virtual por outro com recursos suficientes para atender os requisitos de QoS do fluxo para o qual foi solicitada a mudança de rota. O caráter pró-ativo da abordagem reside no fato de que todas as ações possíveis são executadas antes de uma solicitação externa de rerroteamento. Essas ações antecipadas reduzem significativamente a latência do rerroteamento, porque no momento da solicitação só resta efetivar a troca de rótulos para desviar o fluxo para a nova rota.

A estratégia básica do rerroteamento pró-ativo é: (i) identificar o maior número possível de caminhos alternativos durante o monitoramento do fluxo; (ii) gerar todos os identificadores locais do circuito virtual nos nós pertencentes aos caminhos alternativos descobertos e (iii) efetuar a troca de rota.

As operações a serem efetuadas dependem da abordagem adotada, que pode ser plena ou parcial. O rerroteamento pleno ocorre quando todo o circuito virtual é substituído, enquanto que o parcial é aquele em que o alvo da substituição é apenas um trecho do circuito virtual.

A abordagem parcial de rerroteamento pressupõe, a priori, a existência de um trecho a ser substituído, o trecho crítico. Esse trecho é aquele que contribui com o maior peso percentual do valor fim a fim da métrica de QoS monitorada. Caso a métrica seja o retardo, o trecho crítico é aquele cujo somatório dos retardos dos seus enlaces tenham um valor tal que, percentualmente, em relação ao retardo fim a fim do circuito virtual, seja significativo.

O uso do rerroteamento parcial pode ser justificado pelos resultados publicados por G. Apostolopoulos [1], um dos quais está reproduzido na Figura 2.

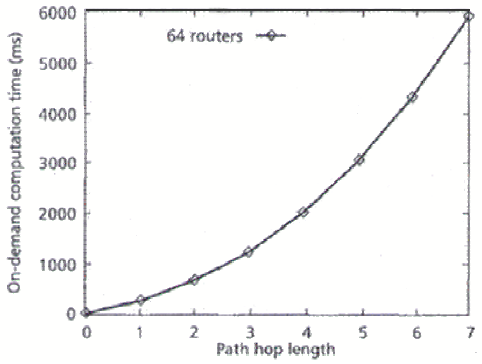


Figura 2 – Custo do cálculo de rota sob demanda versus saltos entre origem e destino

O gráfico apresenta o tempo de processamento do cálculo (sob demanda) de rotas em função da distância em saltos entre a origem e destino. Nota-se que o tempo aumenta exponencialmente. A estratégia no rerroteamento parcial é restringir a busca de caminhos alternativos, a fim de reduzir o tempo de processamento. A vantagem de tal abordagem torna-se acentuada quando os circuitos virtuais são longos. O surgimento de trechos críticos (*hot spot*) em redes IP não é incomum. Eles são o resultado da concentração de tráfego em alguns pontos da rede, que ocorre devido à topologia da rede, ao padrão e à distribuição do tráfego.

2.4.2 – Uso da Tecnologia Ativa em rerroteamento Pró-ativo

O emprego da tecnologia Ativa, dotando os nós com inteligência, flexibilidade e adaptabilidade na forma de agentes móveis, tem o mérito de tornar a tarefa de rerroteamento de fluxos independente do protocolo de roteamento. Porém, é necessário restringir a difusão de agentes necessários para estabelecimento de caminhos alternativos. A definição do

tamanho da área de atuação ou área de busca é um compromisso entre a profundidade desejada para a busca de caminhos alternativos e os recursos consumidos. Uma maior área, se por um lado, aumenta as chances de localizar mais trechos alternativos, por outro, aumenta o consumo de recursos computacionais dos nós e banda dos enlaces da rede.

Outra questão que deve ser apreciada atentamente é a mudança da posição do trecho crítico no circuito virtual durante o tempo de existência do fluxo. Isto pode ocorrer devido à topologia da rede e à variabilidade do tráfego. A mudança da posição do trecho crítico durante o monitoramento do fluxo traz severas complicações na implantação do rerroteamento parcial. Este fenômeno pode ser visualizado através do movimento do centro da área de busca ao longo do circuito virtual. Este movimento, ao mudar a posição da área de busca, exige o desencadeamento da inundação de agentes em regiões diferentes a todo momento, provocando um consumo de recursos acentuado.

O rerroteamento de fluxos de aplicações também apresenta o problema da escalabilidade. Isto ocorre porque, se o número de solicitações de rerroteamento for elevado, há um consumo excessivo de recursos na rede devido ao aumento de mensagens de controle e dos deslocamentos de agentes para o descobrimento e o estabelecimento de novas rotas.

2.4.3 – Trabalhos relacionados

O uso de agentes móveis em tarefas relacionadas ao roteamento dinâmico é investigado em alguns trabalhos recentes [43] [44] [45]. [46] emprega agentes móveis na construção de rotas multiponto/ponto (árvores) com restrições de QoS para implementar a agregação de FEC's prevista na arquitetura MPLS. [47] confronta o desempenho dos algoritmos de roteamento *shortest path first* (SPF) e *Bellman-Ford* (BF) com um algoritmo baseado em agentes móveis chamado AntNet. Em [48], é proposta uma aplicação utilizando agentes móveis e CORBA para o gerenciamento das operações de roteamento. O objetivo é estabelecer rotas em função da QoS previamente estabelecida em contratos de Serviço (SLA).

O rerroteamento tem sido pesquisado com o objetivo de tornar as redes de comutação de pacotes mais resistentes a falhas de enlaces e nós. As abordagens adotadas visam também reduzir a latência de recuperação da rede após a ocorrência de falhas. Em [49], é desenvolvido um trabalho teórico que se baseia na concatenação de LSP's menores para compor o LSP principal entre o comutador de ingresso e egresso. Este esquema simplifica o rerroteamento, reduzindo o tempo de recuperação de LPS's.

O esquema proposto em [39] utiliza vários caminhos multiponto/ponto para prover balanceamento de carga e recuperação no caso de falhas de nós. As árvores são construídas de tal forma que o rerroteamento seja efetuado pelo nó de ingresso. Este esquema acelera o rerroteamento devido à ausência de sinalização para o redirecionamento do fluxo.

Scott [50] apresenta um algoritmo para fornecer serviços robustos com garantias de QoS sobre uma infra-estrutura de rede IP com nós e enlaces suscetíveis a falhas. O algoritmo proposto calcula múltiplos caminhos com o menor número possível de elementos (nós e enlaces) em comum, minimizando assim o impacto das falhas nos serviços.

Em [40], é salientado que o rerroteamento pode ser empregado tanto na camada de rede quanto na camada física e de enlace. O rerroteamento na camada de rede tem a vantagem de poder ser realizado com uma menor granularidade, permitindo que serviços específicos sejam atendidos. A desvantagem é um tempo de recuperação elevado. Nas camadas física e de enlace, em especial as de redes orientadas a circuito, os tempos de recuperação são reduzidos. A contrapartida é que o rerroteamento, quando efetuado, atinge uma grande quantidade de fluxos simultaneamente. [41] propõe que o rerroteamento para redes IP sobre WDM seja efetuado tanto na camada IP quanto na WDM. O esquema proposto tenta otimizar as ações de rerroteamento considerando restrições de topologia, consumo de recurso na rede e características do rerroteamento nas duas camadas.

2.5 – Considerações finais

Neste capítulo, foram apresentados os conceitos básicos que são necessários para o entendimento de uma arquitetura de rerroteamento pró-ativo. Foi descrita de forma resumida a evolução das implementações de gerenciamento, começando pela abordagem centralizada, passando pela distribuída e orientada a objetos até chegar àquelas que se utilizam da tecnologia ativa como infra-estrutura básica para a distribuição de código, móvel ou não. Por estar intimamente ligado às atividades de rerroteamento, as características e funcionalidades do gerenciamento de desempenho, em especial o pró-ativo sob o paradigma da tecnologia ativa, foram salientadas. Em seguida, o funcionamento do roteamento *best-effort* foi descrito, bem como os requisitos e os objetivos do roteamento baseado em QoS, detalhando-se o impacto causado nos roteadores e na rede. Os principais componentes da arquitetura MPLS foram apresentados e o rerroteamento pró-ativo foi conceituado de forma genérica, apontado os seus objetivos, vantagens e limitações. Por último, conceituou-se o rerroteamento pleno e

parcial, justificando o porquê da preferência pelo uso deste último na seqüência deste trabalho.

Capítulo 3 – Arquitetura de Rerroteamento pró-ativo

As tarefas realizadas por um sistema de gerenciamento de desempenho são basicamente monitoração e controle. A monitoração verifica continuamente se parâmetros de QoS relevantes (banda passante, taxa de erros, retardo intra-fluxo e inter-fluxos, taxa de ocupação de CPU, custo monetário dos enlaces, etc) estão dentro das suas faixas de valores aceitáveis. Os valores limites são configurados manualmente pelo administrador (gerente da rede) ou, automaticamente pelo próprio sistema de gerenciamento de desempenho. As ações de controle são normalmente desencadeadas em decorrência de simples violações de QoS, ou de previsões de violação que possam vir a ocorrer. As primeiras são classificadas como ações reativas e as últimas como ações pró-ativas. Os sistemas de gerenciamento tradicionais são normalmente apenas reativos. O rerroteamento de fluxos no serviço de comunicação é uma das possíveis ações de controle que pode ser efetuada pelo sistema de gerenciamento.

Neste capítulo, é proposta uma arquitetura de rerroteamento pró-ativo de forma a possibilitar o redirecionamento de fluxos de aplicações críticas através de rotas alternativas. A Arquitetura proposta fundamenta-se nos conceitos do roteamento baseado em QoS e pressupõe o uso da tecnologia Ativa e de uma infra-estrutura de rede baseada em circuitos virtuais.

A seção 3.1 deste capítulo descreve resumidamente a Arquitetura de Gerenciamento de Desempenho Pró-ativo AGDPA [6], a qual é a base deste trabalho. A seção 3.2 apresenta a descrição geral da arquitetura de rerroteamento. Na seção 3.3, os componentes da arquitetura de rerroteamento são detalhados. A seção 3.4 descreve as diversas fases envolvidas durante um processo de rerroteamento e a seção 3.5 apresenta as considerações finais.

3.1 – Introdução à Arquitetura de Gerenciamento de Desempenho Pró-ativo

A Arquitetura de Gerenciamento de Desempenho Pró-ativo (AGDPA) é uma extensão à Arquitetura de Gerenciamento Ativo Distribuído (AGAD) [29] proposta por [6], de forma a atender especificamente a área funcional da gerência de desempenho. A sua natureza pró-ativa advém de mecanismos que permitem gerar alarmes de tendências de falha de QoS. A AGAD [29], cuja visão geral é mostrada na Figura 3, provê a infra-estrutura que permite o gerenciamento distribuído das redes, serviços, estações e aplicações.

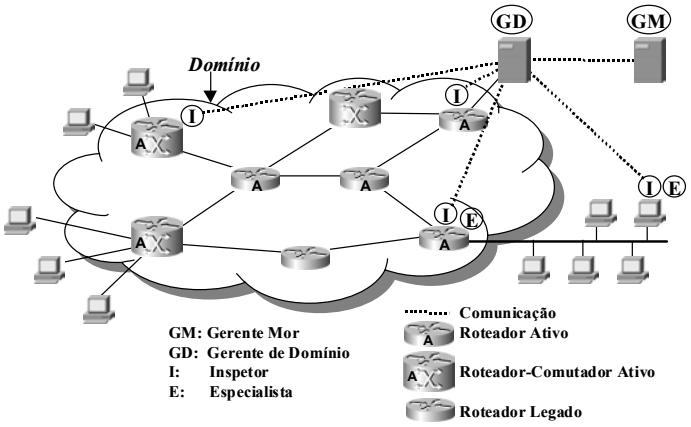


Figura 3 – Visão geral da AGAD [29]

O Gerente Mor (GM) é o responsável pelo gerenciamento de um conjunto de domínios, via Gerentes de Domínio. O GM cria os Gerentes de Domínio (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 [29], passa a ser monitorada pelos Guardiões. Um domínio equivale a um conjunto de elementos a serem gerenciados sob uma mesma autoridade administrativa.

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.

Na Arquitetura de Gerenciamento de Desempenho Pró-ativo (AGDP), é criada a figura do Inspetor de Desempenho (ID) para a coleta e processamento de dados que apontem tendências de ocorrência de falha de QoS, de forma que Alarmes de Tendência possam ser enviados ao Gerente de Desempenho de Domínio (GDD). O ID é também responsável por disponibilizar uma previsão do instante em que a falha irá ocorrer.

Baseado nas informações recebidas dos ID's, o GDD pode despachar Especialistas para, por exemplo, alterar as políticas de escalonamento, reconfigurar a utilização de memória, sinalizar a necessidade de rerroteamento, etc.

3.2 – Descrição Geral da Arquitetura de Rerroteamento

A Arquitetura de Rerroteamento Pró-ativo é constituída de três agentes: AgenteNóEntrada (ANE), AgenteNóIntermediário (ANI) e AgenteRotaAlternativa (ARA). O agrupamento das funcionalidades necessárias nesses três agentes visa minimizar o tamanho dos seus códigos e conseqüentemente os seus tempos de migração e instalação assim como o consumo de memória nos comutadores do circuito virtual ou trechos alternativos.

- **AgenteNóEntrada (ANE)** é o primeiro a iniciar o seu processamento no comutador de entrada do circuito virtual. Esse componente efetua o gerenciamento do rerroteamento dos fluxos pertencentes ao mesmo circuito virtual e é o elemento que interage com algum sistema externo.

- **AgenteNóIntermediário (ANI)** possui dois objetivos: (i) identificar os nós pertencentes ao circuito virtual do fluxo a ser rerroteado; (ii) enviar ao componente ANE hospedado no nó de entrada do circuito virtual os valores das métricas de QoS relevantes ao fluxo dos nós e enlaces do circuito virtual.

- **AgenteRotaAlternativa (ARA)** é especializado na descoberta de trechos alternativos em torno do trecho crítico. O tamanho da região (área de busca) na qual são efetuados os processamentos para a descoberta de trechos alternativos é definido por um parâmetro chamado de raio da área de busca (RAB). O valor desse parâmetro é passado quando criado pelo componente ANE. O componente ARA possui dois outros objetivos: (i) definir o melhor trecho alternativo e (ii) realizar a mudança de rota.

Inicialmente, os três agentes devem ser enviados ao primeiro nó ou comutador do circuito virtual já estabelecido tão logo a monitoração do fluxo da aplicação é iniciada. O primeiro nó é o de entrada do circuito virtual, ou seja, é o primeiro a receber o fluxo proveniente da fonte (servidor de vídeo, etc).

As atividades de rerroteamento iniciam no momento em que os agentes instalam-se no nó de entrada. Isto caracteriza a abordagem pró-ativa da arquitetura proposta porque essas atividades são realizadas durante a monitoração do fluxo e não após a ocorrência da falha ou

tendência de falha de QoS. Quando for sinalizada a falha ou tendência de falha de QoS, a única operação a ser efetuada será a mudança de rota do fluxo através do trecho alternativo previamente descoberto.

O agente ANE é fixado no primeiro nó do circuito virtual visando amenizar o problema de escalabilidade de rerroteamento de fluxos individuais. A idéia é torná-lo um elemento gerenciador do rerroteamento de fluxos que pertençam ao mesmo circuito virtual. Desta forma, a migração do agente ANI e a difusão do agente ARA só são efetuadas para a primeira solicitação de rerroteamento de fluxos que estejam associados a um mesmo circuito virtual.

A arquitetura estabelece que o agente ANE deve implementar uma função de disparo do agente ARA. Nesta função estão configurados os critérios para a criação de novos agentes ARA. Desta forma, é possível evitar ou amenizar os efeitos produzidos devido à mudança da posição do trecho crítico no circuito virtual.

Os agentes ANI foram incluídos com o objetivo de obter as informações dos estados locais dos comutadores do circuito virtual e enviá-las ao agente ANE instalado no nó de entrada. Essas informações de estado são necessárias para que o agente ANE calcule a posição do trecho crítico no circuito virtual.

O rerroteamento parcial necessita que o trecho crítico pertencente ao circuito virtual seja definido (tarefa efetuada pelo agente ANE) e que trechos alternativos em torno desse trecho crítico sejam descobertos (tarefa efetuada pelo agente ARA). A estratégia de identificação dos trechos alternativos é a difusão limitada de réplicas desse agente na região (área e busca) em torno do trecho crítico. O agente ARA também executa a mudança de rota mediante solicitação do agente ANE.

3.3 – Detalhamento dos Componentes

Os componentes da Arquitetura de Rerroteamento Pró-ativo apresentados na Figura 4 podem estar localizados fisicamente no mesmo nó de comutação ou em nós distintos. O componente Gerente de Desempenho Pró-ativo (GDPA), que não faz parte da arquitetura proposta, deve estar em uma estação específica devido a uma maior exigência de capacidade computacional conforme especificado em [29]. Esse componente foi incluído na Figura 4 para mostrar a interação com elementos externos.

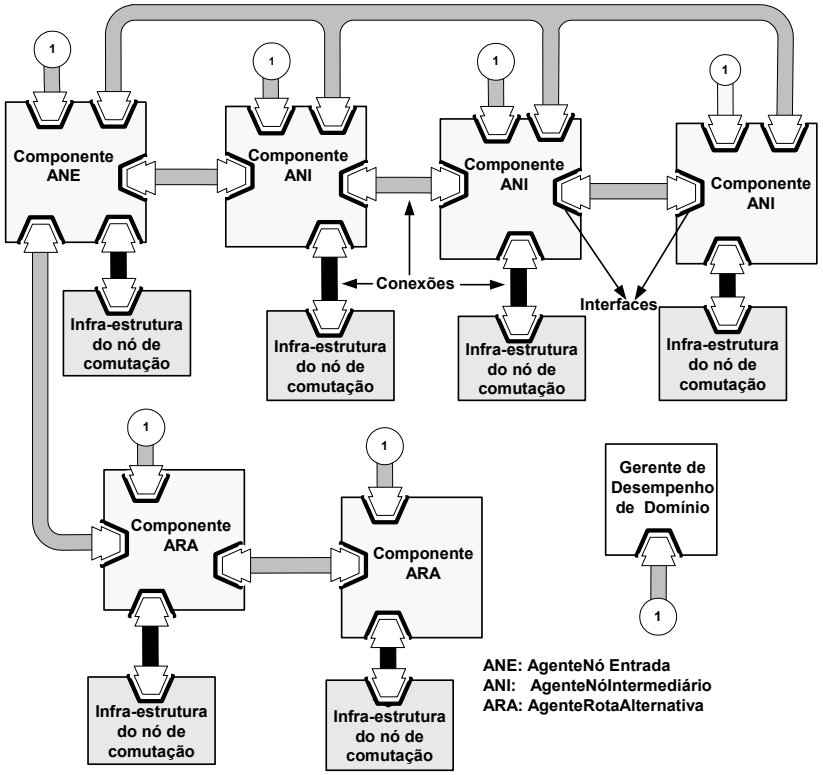


Figura 4 – Componentes da arquitetura e seus relacionamentos.

As conexões representadas na Figura 4 são abstrações dos canais de comunicação entre as interfaces dos componentes. Todos os componentes da arquitetura possuem interfaces externas que permitem o completo monitoramento e controle das atividades dos componentes da arquitetura. No exemplo apresentado na Figura 4, o sistema externo é o GDPA. Exemplos desses controles são a interrupção do processamento ou a migração de algum agente de rerroteamento para aliviar a carga da CPU de algum nó de comutação.

Interfaces internas estão também presentes em todos os componentes da arquitetura para a interação com os mecanismos dos nós de comutação. Por meio dessas interfaces, os agentes executam mudanças de rota, consultam tabelas de roteamento e rótulos, obtêm os estados locais dos nós, etc.

Os componentes ANI e ARA possuem interfaces para a comunicação com o componente ANE, permitindo que este último efetue o gerenciamento de todo o processo de rerroteamento.

O número de componentes ANI que estão simultaneamente em atividade depende do comprimento do circuito virtual assim como o número de componentes ARA depende do comprimento e número de trechos alternativos.

AgenteNóEntrada (ANE)

O seu funcionamento pode ser detalhado por meio das operações que deve executar durante o seu ciclo de vida:

- ser iniciado por algum componente externo (por exemplo, o GDPA);
- criar o componente ANI;
- tratar as mensagens de estado do circuito virtual que são geradas pelos componentes ANI instalados nos nós do circuito virtual;
- obter os valores de métricas de QoS do nó em que está instalado e dos enlaces;
- calcular a posição do trecho crítico no circuito virtual em função dos valores das métricas de QoS obtidas localmente e nas mensagens de estado;
- executar a função de disparo;
- criar o componente ARA de acordo com o resultado da função de disparo;
- solicitar mudança de rota ao agente ARA instalado no primeiro nó do trecho crítico ao receber notificação externa.

AgenteNóIntermediário (ANI)

O componente ANI deve ser capaz de executar as seguintes operações durante o seu ciclo de vida:

- migrar para o próximo nó do circuito virtual assim que for criado pelo componente ANE no nó de entrada;
- a partir desse nó, instalar réplicas de seu código em todos os nós restantes do circuito virtual;
- informar ao componente ANE o comprimento do circuito virtual assim que instalar-se no nó de saída;
- iniciar o envio periódico das mensagens de estado do circuito virtual ao componente ANE.

AgenteRotaAlternativa (ARA)

Deve realizar as seguintes operações durante o seu ciclo de vida:

- migrar para o primeiro nó do trecho crítico assim que for criado pelo componente ANE no nó de entrada do circuito virtual;
- iniciar o processo de difusão a partir do primeiro nó do trecho crítico, instalando réplicas de seu código em todos os nós dos trechos alternativos;
- parar a difusão das réplicas assim que alcançar o nó destino (último nó do trecho crítico) ou atingir o limite da área de busca;
- iniciar o envio periódico de mensagens de estado dos trechos alternativos quando instalado no nó destino;
- tratar as mensagens de estado quando instalado no nó intermediário ou no primeiro nó do trecho crítico;
- medir o estado local (valores das métricas de QoS);
- reenviar as mensagens de estado após atualizar as informações de estado quando instalado nos nós intermediários;
- escolher o melhor trecho alternativo em função das informações de estado das mensagens quando instalado no primeiro nó do trecho crítico;
- mediante solicitação do componente ANE, efetuar a mudança de rota quando instalado no primeiro nó do trecho crítico.

3.4 – Fases do processo de rerroteamento

O funcionamento da Arquitetura de Rerroteamento Pró-ativo pode ser explicado estabelecendo cinco fases para as operações necessárias ao rerroteamento de um fluxo:

- Instalação dos agentes (IA)
- Monitoração do Circuito Virtual (MCV)
- Descoberta de Trechos Alternativos (DTA)
- Monitoração dos Trechos Alternativos (MTA)
- Mudança de Rota (MR)

O diagrama da Figura 5 apresenta a seqüência de execução das fases detalhando a relação temporal entre as fases e eventos externos.

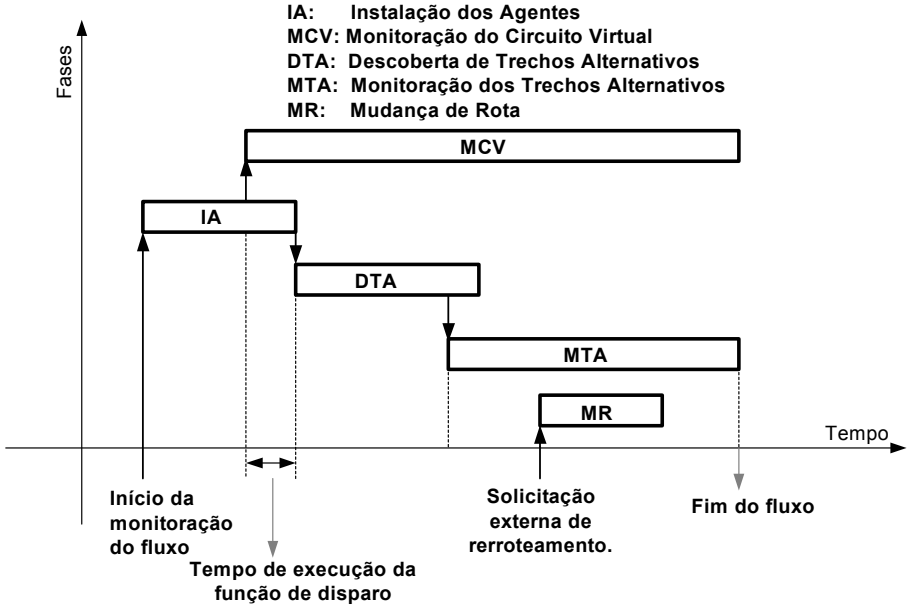


Figura 5 – Fases do rerroteamento

A partir do início do processamento do primeiro agente (ANE) no nó de entrada do circuito virtual, todas as operações referentes a todas as fases, excetuando a fase de mudança de rota (MR), são executadas seqüencialmente sem a necessidade de outros eventos externos. Essas fases são chamadas de pró-ativas porque suas operações são concluídas ou iniciadas antes da ocorrência da falha ou tendência de falha de QoS do fluxo. O adiantamento dessas operações é responsável pela redução da latência do rerroteamento. A fase MR é considerada reativa uma vez que sua execução só será efetuada após solicitação de rerroteamento do sistema externo. Apesar de reativa, considerando como referencial o rerroteamento, a fase MR pode ser executada de forma pró-ativa em relação ao fluxo caso a solicitação de rerroteamento externa seja gerada devido à detecção de uma tendência de falha de QoS. Esta situação ocorre

quando o sistema externo que está utilizando o rerroteamento tiver funcionalidades pró-ativas como é o caso do Gerenciamento de Desempenho Pró-ativo (GDPA) [6].

A Figura 5 mostra que as fases MCV e MTA, uma vez iniciadas, são executas durante a existência do fluxo. Essas fases não influenciam no tempo de chaveamento do fluxo.

A superposição das fases MCV e IA ocorre porque o componente ARA somente é criado após o cálculo da posição do trecho crítico e a confirmação da função de disparo. Além disso, essas operações, para serem realizadas, dependem das informações de estado que são disponibilizadas ao componente ANE pelas mensagens de estado geradas na fase MCV.

Outra superposição detalhada na Figura 5 é a das fases DTA e MTA. Neste caso, a superposição ocorre porque os agentes ARA chegam ao último nó do trecho crítico em momentos diferentes. A fase MTA inicia quando o primeiro agente ARA alcança este nó, enviando a primeira mensagem de estado ao agente ARA localizado no primeiro nó do trecho crítico. A fase DTA somente encerra assim que a última cópia do agente ARA alcança o nó destino da difusão (último nó do trecho crítico). Logo, o tempo de superposição dessas fases é igual ao tempo transcorrido entre a chegada do primeiro e do último agente ARA ao nó destino da difusão.

3.4.1 – Instalação dos agentes (IA)

A Figura 6 detalha a seqüência das operações durante a instalação dos agentes de rerroteamento no circuito virtual. O exemplo da Figura 6 contempla um circuito virtual com seis comutadores.

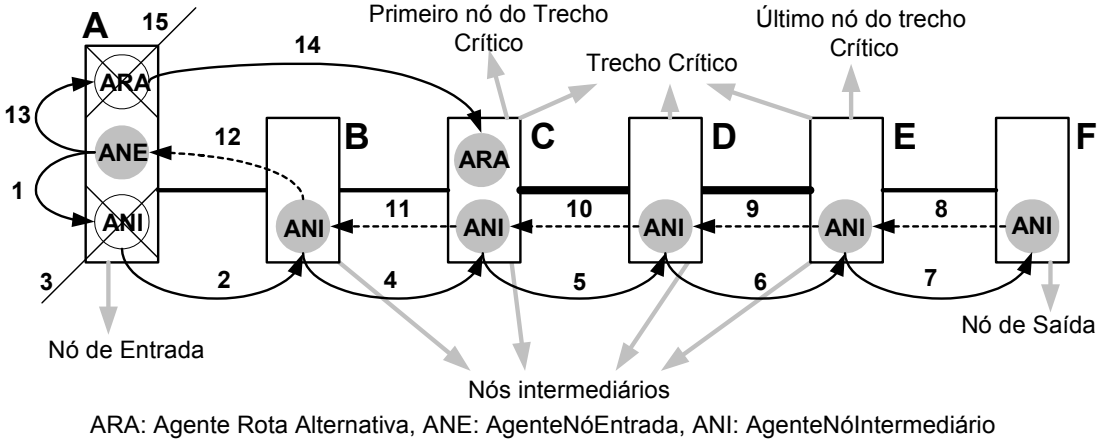


Figura 6 – Seqüência de operações da fase IA

Assim que o agente ANE inicia o seu processamento, cria o agente ANI (operação 1, Figura 6). O agente ANI, recém criado, migra (operação 2, Figura 6) para o próximo nó do circuito virtual (nó B, Figura 6). A operação 3 da Figura 6 indica que o agente ANI após deslocar-se para o próximo nó, interrompe o seu processamento naquele nó (nó A da Figura 6). O agente ANI, no nó B (operação 4, Figura 6), gera uma réplica de seu código que migra para o próximo nó (nó C, da Figura 6). As operações 5, 6 e 7 são similares à operação 4. As operações 8 a 12, que não fazem parte da fase de instalação dos agentes, correspondem às mensagens de estado do circuito virtual. Essas operações foram mostradas na Figura 6 porque são os eventos que causam a criação do agente ARA pelo agente ANE. O agente ARA, no nó A, migra (operação 14, Figura 6) imediatamente para o primeiro nó do trecho crítico (nó C, Figura 16). A operação 15 indica que o agente ARA após migrar para o primeiro nó do trecho crítico encerra seu processamento naquele nó (nó A, Figura 6).

3.4.2 – Monitoração do Circuito Virtual (MCV)

A fase MCV inicia assim que o agente ANI instala-se no nó de saída do circuito virtual (nó F, Figura 6). O seu término se dá juntamente com o término do fluxo (Figura 5). Durante essa fase, são realizadas as operações necessárias para disponibilizar as informações de estado do circuito virtual ao agente ANE hospedado no nó de entrada do circuito virtual (nó A, Figura 6). Essas informações de estado são utilizadas pelo agente ANE no cálculo da posição do trecho crítico no circuito virtual.

O esquema utilizado consiste no envio dessas mensagens a partir do agente ANI do nó de saída até ao agente ANE, passando por todos os agentes ANI instalados no nós intermediários. Desta forma, os agentes ANI dos nós intermediários, após receber a mensagem de estado e obter as informações de estado local, incorpora essas informações na mensagem recebida. Em seguida, reenvia essa nova mensagem ao agente ANI hospedado no nó vizinho anterior em relação ao fluxo (operações 8 a 12, Figura 6). O processo repete-se até que a mensagem alcance o agente ANE. O envio dessas mensagens pelo agente ANI do nó de saída é efetuado periodicamente.

3.4.3 – Descoberta de Trechos Alternativos (DTA)

A fase de descoberta de trechos alternativos inicia quando o agente ARA instala-se no primeiro nó do trecho crítico. O encerramento desta fase se dá quando a última cópia do agente ARA instala-se no último nó do trecho crítico, indicando a descoberta do último trecho

alternativo. A Figura 7 detalha o posicionamento dos agentes nos diversos comutadores e a seqüência de operações da fase DTA em um cenário em que existem dois trechos alternativos na área de busca.

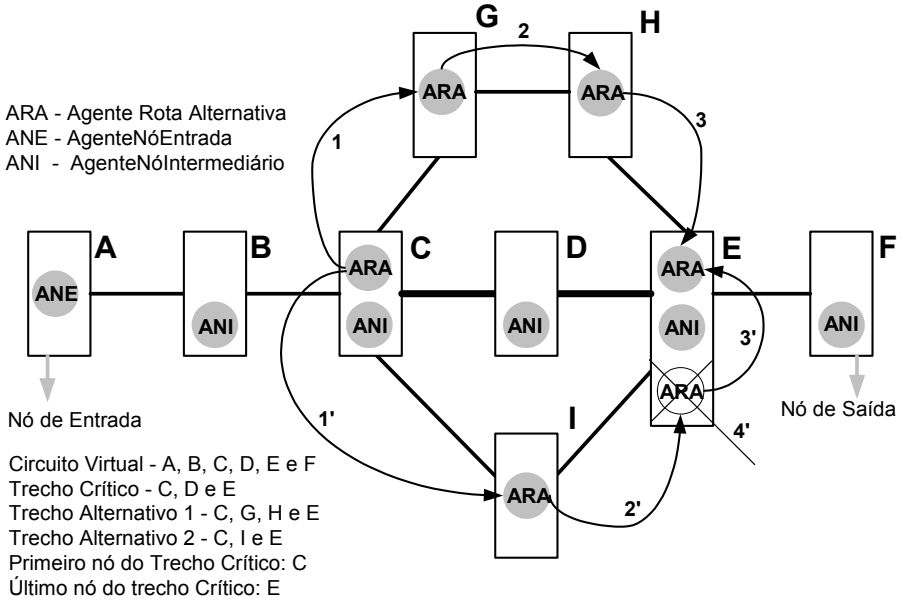


Figura 7 – Seqüência de operações da fase DTA

O agente ARA no primeiro nó do trecho crítico (nó C, Figura 7) cria uma cópia do seu código que migra para o vizinho pertencente ao trecho alternativo 1 (nó G, Figura 7). Esse processo de criação e migração de réplicas para os respectivos nós vizinhos ocorre até que uma das cópias do agente ARA atinja o último nó do trecho alternativo (nó E, Figura 7).

No trecho alternativo 2, um processo semelhante também ocorre. A primeira diferença é que este trecho alternativo possui comprimento menor do que o primeiro, resultando em um menor número de operações de criação e de migração de réplicas do agente ARA. A segunda diferença corresponde às operações adicionais executadas por este agente ARA quando alcança o último nó do trecho crítico. Essas operações adicionais são efetuadas porque o outro agente ARA (referente ao trecho alternativo 1) tinha instalado-se anteriormente neste nó. Ao detectar a presença deste outro agente ARA (referente ao trecho alternativo 1) no último nó do trecho crítico, o agente ARA do trecho alternativo 2 envia uma mensagem ao agente ARA já instalado (operação 3', Figura 7), informando a interface pela qual chegou e o comprimento do trecho alternativo 2. Em seguida, encerra o seu processamento (operação 4', Figura 7). A ordem de chegada dos agentes ARA ao último nó do trecho crítico foi escolhida arbitrariamente. Os tempos de execução dessas operações dependem de vários fatores tais como velocidade dos enlaces, capacidade de processamento dos comutadores, retardo de

enfileiramento na saída das interfaces e ordem de envio das réplicas pelo agente ARA do primeiro nó do trecho crítico.

3.4.4 – Monitoração dos Trechos Alternativos (MTA)

A fase MTA contém as operações necessárias para a obtenção do estado local dos nós dos trechos alternativos e ao envio periódico das mensagens de estado. A origem dessas mensagens é o agente ARA do último nó do trecho crítico (nó E, Figura 8) e o destino é o agente ARA do primeiro nó do trecho crítico (nó C, Figura 8). Essas mensagens são enviadas através de todos os trechos alternativos descobertos, passando por todos os nós desses trechos. As informações de estado contidas nessas mensagens permitem ao agente ARA do último nó do trecho crítico escolher o melhor trecho alternativo com recursos suficientes para atender as restrições de QoS do fluxo a ser rerroteado. A Figura 8 apresenta o esquema adotado para o envio das mensagens durante essa fase.

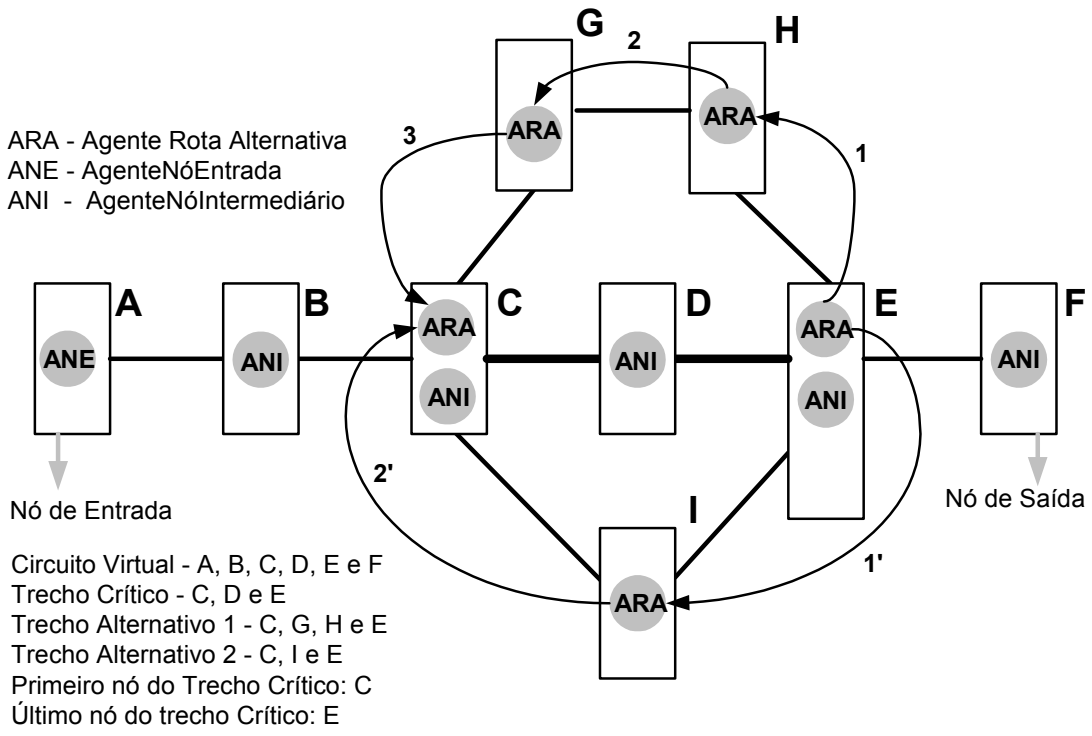


Figura 8 – Seqüência de operações da fase MTA

A fase MTA inicia no momento em que o primeiro agente ARA alcança o último nó do trecho crítico e seu término se dá juntamente com o do fluxo. O agente ARA, antes de enviar a mensagem (operação 1, Figura 8) ao seu vizinho anterior do trecho alternativo 1 (nó H, Figura 8), obtém as informações locais de estado e gera o identificador local do novo circuito virtual. O novo circuito virtual é aquele após o rerroteamento, ou seja, após a substituição do trecho

crítico pelo trecho alternativo. O nó vizinho anterior (nó H, Figura 8), ao receber essa mensagem (operação 1, Figura 8), efetua as mesmas operações antes de repassá-la ao seu vizinho anterior. Esse procedimento é efetuado de nó em nó desse trecho alternativo até que a mensagem (operação 3, Figura 8) alcance o agente ARA do primeiro nó do trecho crítico. As operações referentes ao trecho alternativo 2 (operações 1' e 2', Figura 8) são similares às operações do trecho alternativo 1. Vale ressaltar que os identificadores locais do novo circuito virtual somente são gerados quando a primeira mensagem é enviada considerando a abordagem sob demanda de geração dos identificadores locais do circuito virtual.

3.4.5 – Mudança de Rota (MR)

A fase mudança de rota compreende o conjunto de operações executadas entre o recebimento da solicitação externa de rerroteamento pelo agente ANE e o redirecionamento do fluxo através do trecho alternativo. O redirecionamento é efetuado pelo agente ARA hospedado no primeiro nó do trecho crítico.

As operações envolvidas durante a fase de mudança de rota dependem da abordagem adotada na geração e associação dos identificadores locais do circuito virtual. As abordagens podem ser antecipadas ou sob demanda. Na antecipada, esses identificadores são criados durante a passagem da primeira mensagem de estado pelos nós dos trechos alternativos, conforme descrito na seção 3.4.4. Esta abordagem minimiza o tempo de chaveamento porque a fase MR fica praticamente restrita a uma operação de associação de identificadores. Entretanto, a desvantagem é que impõe um consumo elevado de identificadores locais, podendo tornar a operação dos comutadores crítica.

A Figura 9 apresenta as operações quando a abordagem antecipada é adotada. O agente ANE, ao receber uma mensagem externa de rerroteamento, envia uma mensagem de chaveamento (operação 1, Figura 9) ao agente ARA localizado no primeiro nó do trecho crítico. Esse agente finalmente executa a associação (operação 2, Figura 9) entre o identificador de entrada com o de saída. O identificador local de saída é aquele que corresponde ao trecho alternativo escolhido.

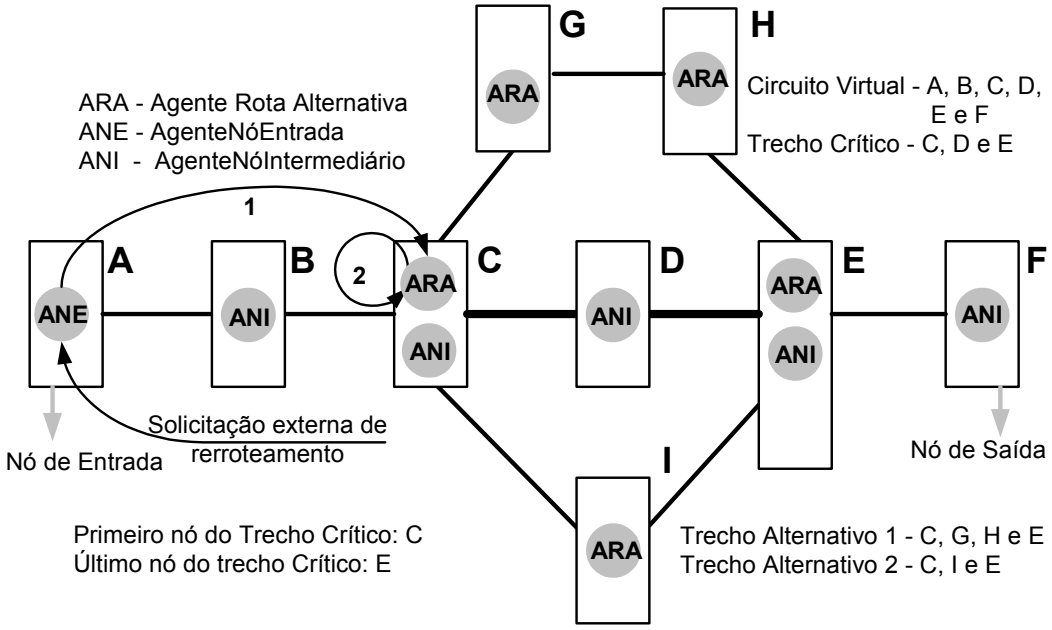


Figura 9 – Seqüência de operações da fase MR – abordagem antecipada

Para contornar a desvantagem do consumo excessivo de identificadores, embora aumentado o tempo de execução da fase MR, a segunda abordagem pode ser adotada. Nesse caso, a geração e associação dos identificadores locais são feitas sob demanda, ou seja, os nós pertencentes aos trechos alternativos só geram os identificadores quando ocorre a solicitação externa de rerroteamento. A Figura 10 detalha o desencadeamento das operações para essa abordagem.

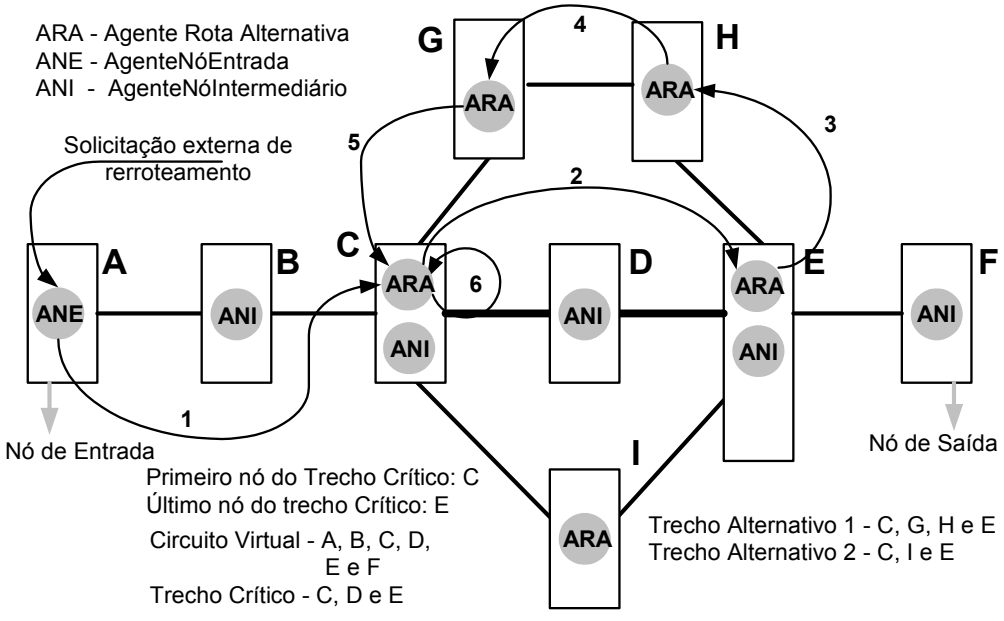


Figura 10 – Seqüência de operações da fase MR – abordagem sob demanda

Após receber a mensagem de chaveamento proveniente do agente ANE (operação 1, Figura 10), o agente ARA do primeiro nó do trecho crítico envia uma mensagem de geração de identificador local (operação 2, Figura 10) ao agente ARA do último nó do trecho crítico. Em seguida, após gerar o seu rótulo, este agente (nó E, Figura 10) envia ao agente ARA instalado no nó vizinho anterior do trecho alternativo escolhido uma mensagem de fechamento de caminho (operação3, Figura 10) com a identificação do caminho alternativo e o número do identificador gerado localmente. A medida que essa mensagem é processada pelos demais agentes ARA instalados nos nós intermediários do trecho alternativo escolhido (operações 4 e 5, Figura 10), os identificadores são criados com as devidas associações. O agente ARA do primeiro nó do trecho crítico recebendo essa mensagem com o identificador de saída do seu vizinho posterior, cria a associação dos rótulos para redirecionar o fluxo (operação 6, Figura 10).

3.5 – Considerações finais

Neste capítulo, o rerroteamento é caracterizado como uma das ações do gerenciamento de desempenho pró-ativo em decorrência de falhas ou tendência de falhas de QoS nos fluxos de aplicações. A Arquitetura proposta leva em conta os problemas inerentes de um rerroteamento pró-ativo parcial baseado em agentes, tal como a possível mudança da posição do trecho crítico em um circuito virtual. O funcionamento da Arquitetura de Rerroteamento Pró-ativo é explicado através da caracterização de cinco fases distintas de operações necessárias ao rerroteamento de um fluxo.

Capítulo 4 – Implementação de um Protótipo

Este capítulo descreve um protótipo para a implementação da Arquitetura de Rerroteamento Pró-ativo. A principal motivação é o seu emprego no ambiente cliente/servidor para aplicações multimídia adaptativas ServiMídia desenvolvido no NCE [4] [5]. No ServiMídia, a ocorrência de falhas no serviço de comunicação desencadeia um processo de adaptação no cliente para reestruturação dinâmica de uma apresentação multimídia. A reestruturação da apresentação consiste em substituir as mídias que estavam sendo apresentadas por outras similares, porém, com requisitos de QoS mais brandos, de forma a manter o conteúdo informacional e, ao mesmo tempo, reduzir o consumo de recursos da rede. Entretanto, a latência da adaptação documentada em [4], momentaneamente, interrompe a apresentação. A capacidade de detectar a tendência de falha de QoS na abordagem de gerenciamento pró-ativo pode antecipar o processo de adaptação, amenizando ou eliminando o impacto na apresentação.

A outra alternativa é o emprego do rerroteamento no serviço de comunicação, ou seja, redirecionar os fluxos através de caminhos alternativos com recursos suficientes, visando evitar o processo de adaptação da aplicação ServiMídia.

O objetivo deste protótipo é verificar a viabilidade do rerroteamento de fluxos de aplicações ServiMídia sobre uma infra-estrutura de rede MPLS mediante solicitação do Sistema de Gerenciamento de Desempenho Pró-ativo [6]. Neste cenário, os três agentes da arquitetura proposta são especializados para o rerroteamento de fluxos com restrições de banda e retardo.

A seção 4.1 introduz o cenário de implementação enfocado. A seção 4.2 descreve o ambiente de desenvolvimento, a estratégia adotada para transformar computadores pessoais em comutadores MPLS (LSR) e, também, o ambiente para prover mobilidade aos agentes, **μCode**. A seção 4.3 descreve uma proposta de protocolo para a troca de mensagens entre os agentes de rerroteamento, enquanto a seção 4.4 apresenta uma proposta de implementação destes agentes. Por último, a seção 4.5 apresenta as considerações finais do capítulo.

4.1 – Cenário de aplicação

A Figura 11 apresenta o esquema do cenário de aplicação da arquitetura proposta, bem como o ambiente no qual ela atua, os seus elementos e os relacionamentos entre os mesmos.

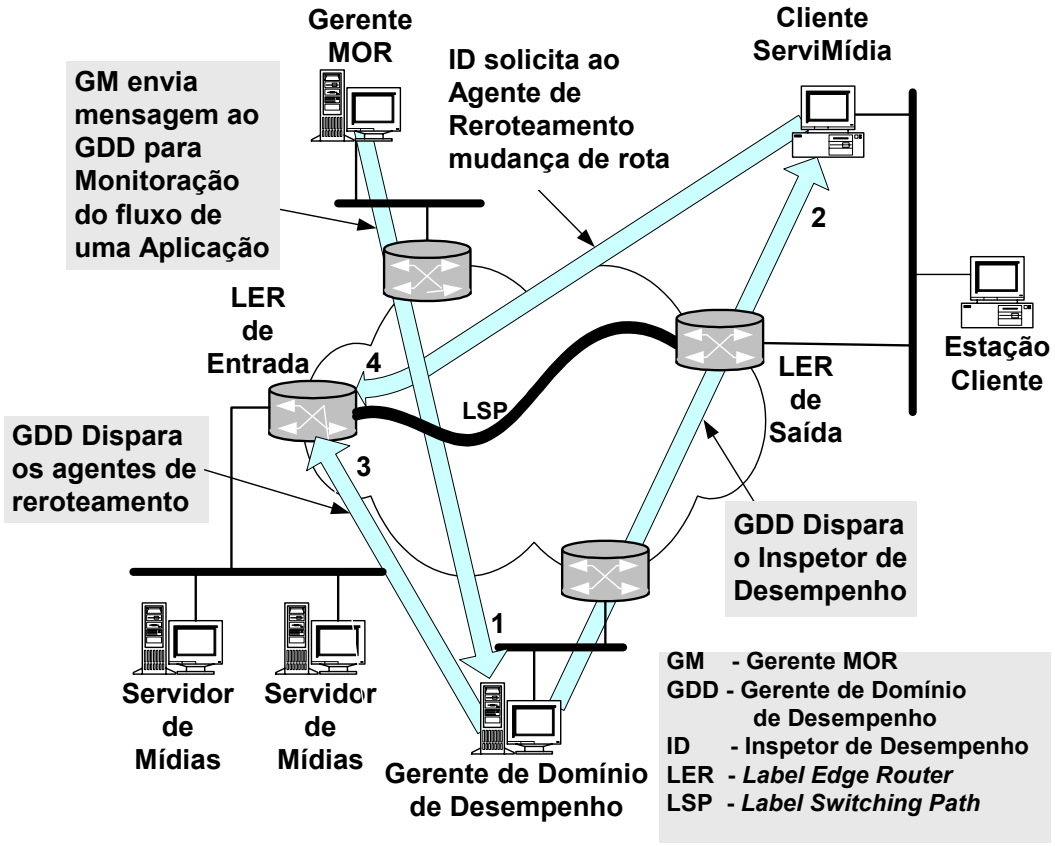


Figura 11 – Cenário de aplicação do rerroteamento pró-ativo

A solicitação de monitoramento de um fluxo é determinada pelo operador da rede ou, automaticamente, quando o serviço é solicitado pelo usuário, para o Gerente de Domínio de Desempenho conforme indica a seta 1 da Figura 11. Este envia um Inspetor de Desempenho à estação Cliente ServiMídia (seta 2, Figura 11) e um grupo de agentes de rerroteamento ao LER de entrada do LSP ao qual o fluxo pertence (seta 3, Figura 11). A seta 4 da Figura 11 representa o envio, pelo Inspetor de Desempenho, de um alarme, quando foi detectada uma tendência de falha, para que seja desencadeado o processo de rerroteamento daquele fluxo.

A seqüência das ações da Figura 11 diverge um pouco daquela prevista na Arquitetura AGAD [29], mas é justificada porque o rerroteamento deve ocorrer em tempo exíguo. Os agentes de rerroteamento instalam-se no LER de entrada (nó de entrada do circuito virtual), nos LSR (nós intermediários do circuito virtual) e LER de saída (nó de saída do circuito virtual) durante o monitoramento do fluxo pelo Inspetor de Desempenho. Eles identificam o maior número possível de trechos alternativos e ficam preparados para efetuar a troca de rótulos (e, conseqüentemente, da rota do fluxo) quando do recebimento de um alarme de tendência no LER de Entrada.

Para esse cenário ServiMídia/MPLS, os componentes da arquitetura devem ser especializados para rerrotear fluxos com restrições de banda e retardo. Então, os componentes AgenteNóEntrada (ANE), AgenteNóIntermediário (ANI) e AgenteRotaAlternativa (ARA) passam a ser designados LerEntradaBandaRetardo (LEBR), LsrBandaRetardo (LBR) e RotaAlternativaBandaRetardo (RABR) respectivamente.

A implementação deste protótipo visa determinar a latência do rerroteamento neste cenário e confrontar com os resultados obtidos em [6]. Desta forma, é possível constatar se a abordagem de rerroteamento adotada elimina a necessidade de adaptações das aplicações ServiMídia diante de falhas de QoS.

4.2 – Ambiente de desenvolvimento

As estações utilizadas tanto no desenvolvimento dos agentes da arquitetura quanto na fase de testes foram PC's (a configuração está descrita no Capítulo 5) com o sistema operacional Linux Red Hat 7.2 com o Kernel 2.4.19. A escolha foi motivada pela gratuidade da distribuição, pela existência de ampla documentação disponível e, principalmente, pela facilidade da configuração dessa distribuição do Linux como roteador. As funcionalidades necessárias à transformação da estação em um comutador MPLS foram incorporadas pela atualização do Kernel através do *patch* desenvolvido por James R. Leu [51]. Essa atualização permite o estabelecimento de LSP's de forma estática através de chamadas ao sistema que criam entradas nas tabelas de rótulos, associando rótulos de entrada, de saída e interfaces de entrada e de saída, respectivamente. Os agentes de rerroteamento utilizam-se dessas chamadas para geração de rótulos, troca de rota do LSP e identificação do LSP do fluxo.

Os enlaces que interligam os comutadores MPLS são segmentos *Ethernet* de 10 Mbit/s dedicados, ou seja, não há compartilhamento.

A implementação do protótipo da arquitetura foi feita em Java e seguiu os princípios da orientação a objetos e da orientação a componentes. O pacote JDK versão 1.3.1 foi instalado em todas as estações que compõem a rede de teste. O uso da linguagem Java, além de proporcionar independência de plataforma, robustez e segurança, também disponibiliza abstrações de programação adequadas para o desenvolvimento de sistemas distribuídos baseados em código móvel. A segurança é intrinsecamente disponibilizada através de políticas de acesso, limitando ou negando o acesso aos recursos locais das estações por programas não autorizados.

Os agentes são implementados na forma de *threads* móveis. A multi-programação quando realizada por *threads* oferece uma sobrecarga computacional ao processador menor do que quando implementada por processos devido ao menor impacto causado pelas trocas de contexto. Entretanto, a multi-programação tem a desvantagem de exigir a sincronização no acesso simultâneo a recursos compartilhados, aumentando a complexidade do código e propiciando a introdução de erros difíceis de serem detectados e corrigidos. A máquina virtual Java (JVM) oferece um ambiente de multi-programação através de *threads* em nível usuário com primitivas de sincronização disponibilizadas em classes específicas para esse fim.

A infra-estrutura de mobilidade adotada foi o μCode [32], por possuir características não encontradas em outros ambientes como, por exemplo, o *Aglets Software Development Kit* (ASDK) [33]. O nível de segurança na movimentação de código, provimento de primitivas básicas de mobilidade, mobilidade gradativa de código e baixa complexidade relativa são características que tornam o ambiente escolhido propício para o desenvolvimento de aplicações da camada de rede.

A alta portabilidade propiciada pela linguagem Java, aliada à mobilidade disponibilizada pela infra-estrutura μCode, oferece a vantagem de tornar a configuração e a adição de novas funcionalidades na infra-estrutura para o atendimento de novos serviços uma tarefa automática, diminuindo a possibilidade de introdução de erros.

Infra-estrutura de mobilidade μCode

A infra-estrutura de mobilidade oferecida pelo ambiente μCode [32] é baseada no modelo cliente/servidor. O elemento que cria o ambiente computacional sobre a Máquina Virtual Java para a execução de *threads* móveis (agentes móveis) é chamado de μServer. Os μServer's devem estar instalados nas estações de origem e destino quando do envio dos agentes móveis. Eles são responsáveis, no envio, pela serialização dos agentes e pela remontagem destes na recepção. No destino, as *threads*, que são os agentes de rerroteamento, são executadas pelos μServer's. O μCode implementa a mobilidade fraca [34] de agentes, ou seja, o estado dos dados é mantido enquanto o estado de execução se perde após a migração, provocando o reinício do processamento na primeira instrução.

O fator determinante na seleção do μCode como infra-estrutura de mobilidade é a fina granularidade de mobilidade, permitindo a realocação de código na medida exata para atender as necessidades específicas do programador. Isto minimiza a carga de comunicação no

transporte e o consumo de memória no armazenamento dos códigos dos agentes envolvidos nas tarefas do rerroteamento Pró-ativo.

Para efeito de mobilidade, objetos e classes podem estar associados a um grupo. É possível enviar somente as classes necessárias para executar funções específicas. O compartilhamento de classes entre μServer's instalados em máquinas distintas se dá via registro das mesmas no μServer da máquina onde estão hospedadas essas classes. Na ausência do registro, essas classes têm significado apenas local, não sendo acessíveis remotamente.

O μServer disponibiliza para os programas de usuário uma coleção de métodos na classe MuServer. Esses métodos permitem dotar os códigos com mobilidade através de abstrações que escondem grande parte da complexidade inerente à implementação dessas funcionalidades. Dois esquemas de mobilidade foram utilizados. O primeiro, idêntico ao utilizado na Arquitetura AGAD [29], usa o método *spawnThread* da classe *Relocator* para enviar as classes dos agentes de rerroteamento ao LER de Entrada pelo Gerente de Domínio de Desempenho quando da solicitação do monitoramento do fluxo. O segundo esquema utiliza o método *copyThread*, também da classe *Relocator*, que replica o código do agente e o envia ao destino especificado. Esse método envia também, além das classes, os estados dos agentes.

4.3 – Comunicação entre os agentes

Os agentes da Arquitetura de Rerroteamento Pró-ativo podem ser configurados e ter seus processamentos controlados através de troca de mensagens. A especificação do formato e a semântica dos campos das mensagens foram concebidos de acordo com a Arquitetura AGAD [29] visando completa compatibilidade com o Sistema de Gerenciamento de Desempenho Pró-ativo [6]. Os dois tipos de mensagens previstos são os de controle e de configuração.

Na Figura 12, estão especificados os campos da parte fixa e da parte variável. A parte fixa da mensagem possui seis campos sendo quatro deles de um octeto e os outros dois com quatro octetos correspondendo aos endereços origem e destino. O campo *tipo de mensagem* define se a mensagem é de controle de configuração ou operacional ou, ainda, de estado. O campo *área de gerência* para mensagens internas entre os elementos da arquitetura não tem significado e portanto pode ter atribuído um valor qualquer diferente daqueles especificados pela Arquitetura AGAD [29]. O campo *remetente* define qual a entidade que enviou a mensagem. No presente trabalho, pode ser o AgenteNóEntrada (ANE), AgenteNóIntermediário (ANI) ou

AgenteRotaAlternativa (ARA). Este campo só é considerado nas mensagens geradas externamente e destinadas a um dos componentes. O campo *tipo de Elemento Gerenciado* é preenchido de acordo com o tipo do dispositivo em que o agente emissor da mensagem está hospedado. Em relação ao rerroteamento em redes MPLS pode ser um LER de Entrada, LER de Saída, LSR intermediário ou um comutador de rótulos fora do caminho MPLS. O conteúdo dos campos *endereço destino* e *endereço origem* são, respectivamente, os identificadores dos nós destinos e de origem da mensagem.

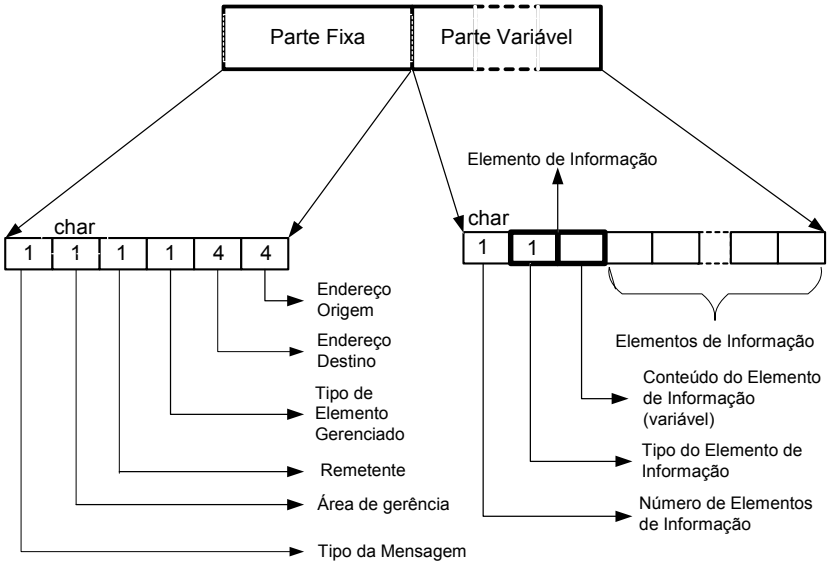


Figura 12 – Formato das mensagens

A parte variável é composta por um octeto e um número variável de elementos de informação. O octeto informa o número de elementos contidos na mensagem. Um elemento possui um campo de um octeto e outro de tamanho variável dependendo do tipo de informação. O primeiro campo define o tipo de informação que pode ser o retardo ou banda disponível dos enlaces.

4.4 – Implementação dos agentes de rerroteamento

As funcionalidades necessárias ao rerroteamento pró-ativo de fluxos com restrição de banda e retardo estão distribuídas nos três agentes da arquitetura de rerroteamento. A modelagem desses agentes foi realizada utilizando a *Unified Model Language* (UML) [35] [36]. Entretanto, certas funcionalidades devem ser necessariamente disponibilizadas para qualquer tipo de agente específico. Logo, a modelagem contemplou as funcionalidades gerais e específicas através de uma estrutura de classes apropriada, tornando a implementação de novos agentes para atender diferentes restrições de QoS uma tarefa simplificada e menos

sujeita a erros. O diagrama de classes e as descrições do seus respectivos métodos estão apresentados no anexo A.

Diagramas de seqüência

As dependências entre os diversos componentes e os encadeamentos de mensagens previstos na arquitetura estão apresentados no diagrama de seqüência da Figura 13. Por simplicidade, não estão representadas as computações referentes à criação das mensagens de estado e controle. O diagrama da Figura 13 descreve a seqüência dos eventos que ocorrem durante a instalação dos componentes de rerroteamento ao longo de um LSP com cinco comutadores, conforme mostrado na Figura 14.

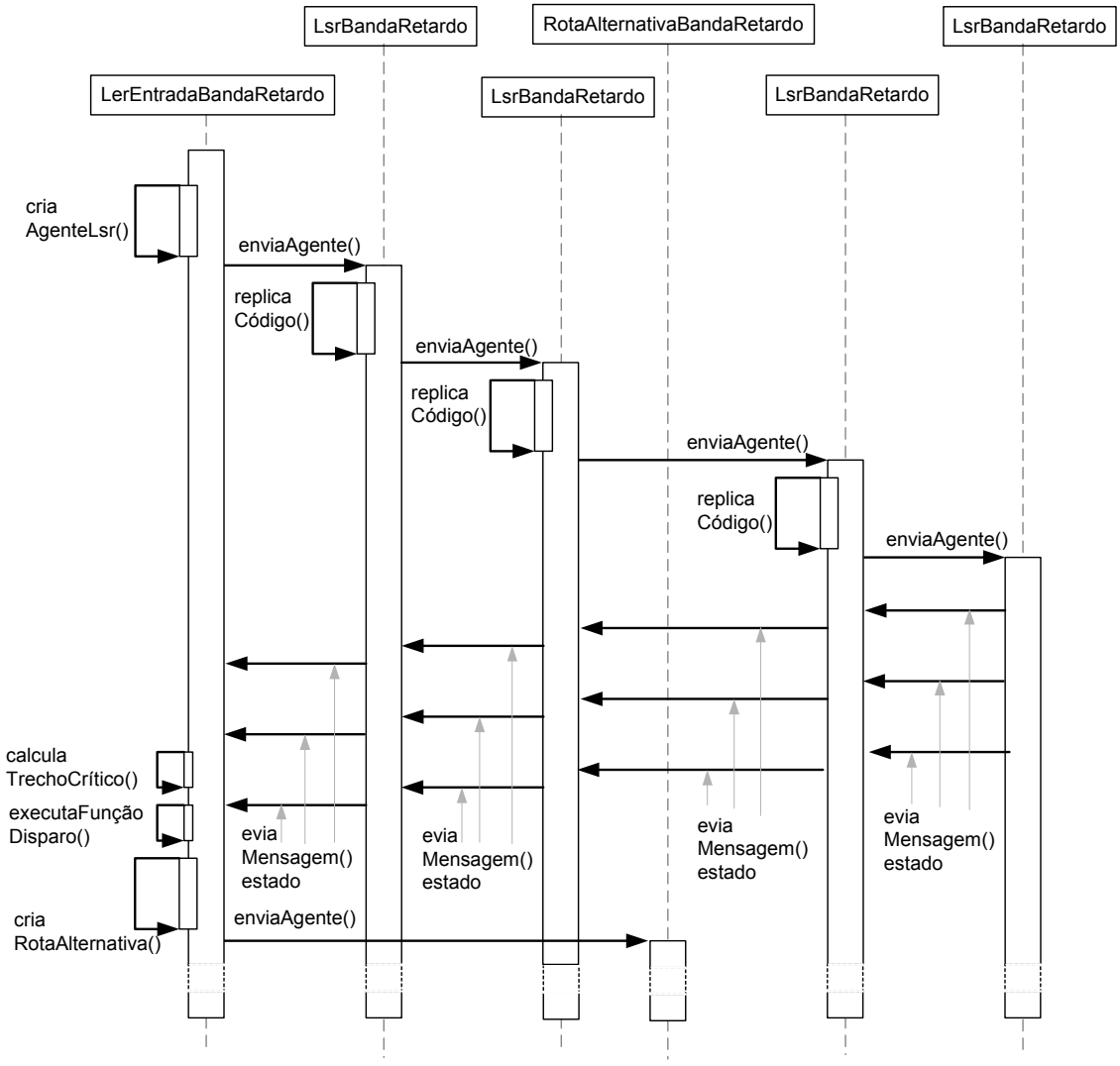


Figura 13 – Diagrama de seqüência da fase de instalação dos agentes

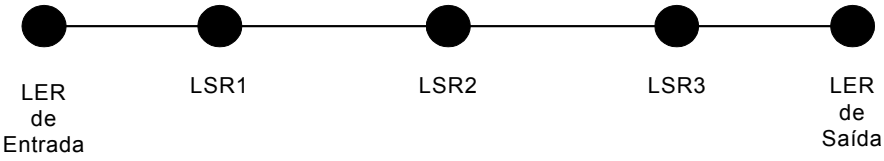


Figura 14 – LSP – exemplo

No caso ilustrado no diagrama de seqüência de instalação dos agentes (Figura 13), o primeiro nó do trecho crítico é o LSR2, uma vez que o componente LerEntradaBandaRetardo enviou a este LSR o componente RotaAlternativaBandaRetardo. Está, também, claro no diagrama que o método *executaFunçãoDisparo()* autorizou o envio do componente RotaAlternativaBandaRetardo depois de verificar que o método *calculaTrechoCrítico()* retornou o mesmo resultado após três mensagens de estado consecutivas.

O diagrama de seqüência da Figura 15 ilustra o processo de difusão de agentes para a descoberta de caminhos alternativos em torno do trecho crítico.

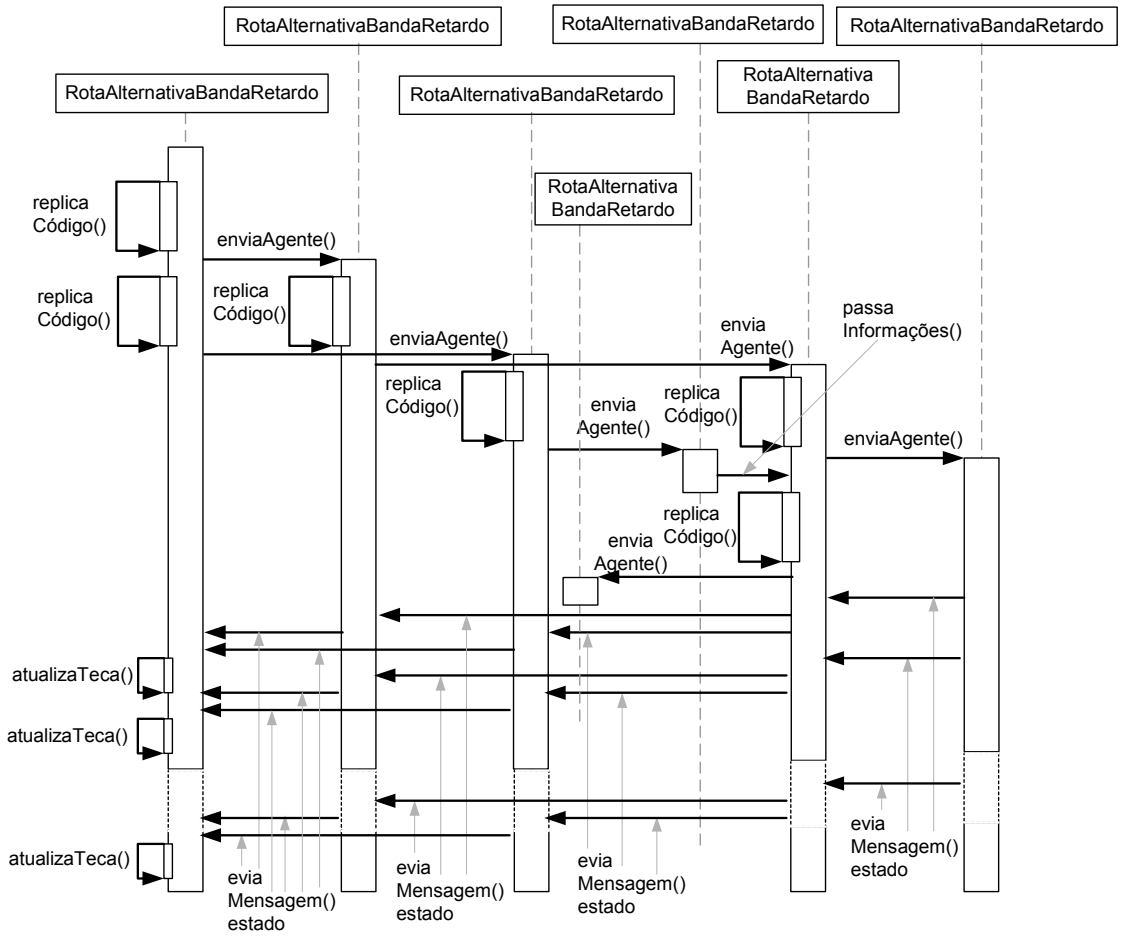


Figura 15 – Diagrama de seqüência da fase de descoberta de trechos alternativos

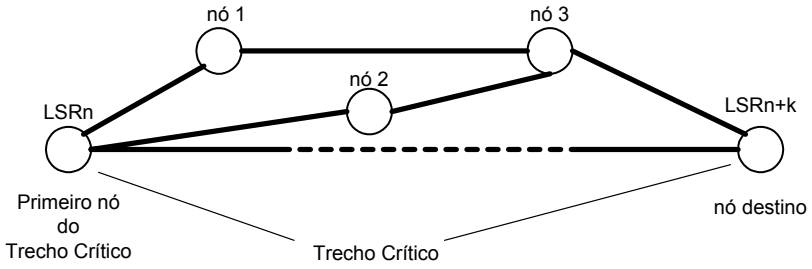


Figura 16 – Topologia da área de busca – exemplo

A forma final do diagrama de seqüência de difusão depende da topologia existente em torno do trecho crítico, ou seja, da topologia da área de busca. A Figura 16 mostra a topologia da área de busca adotada como exemplo.

Constata-se na Figura 15 a existência de componentes *RotaAlternativaBandaRetardo* (RABR) instalados em vários LSR. Alguns desses componentes possuem tempo de vida curto como, por exemplo, o segundo componente a instalar-se no nó 2. O seu processamento é interrompido quando constata que a distância percorrida da origem até alcançar o nó corrente é igual ao comprimento do raio da área de busca e que o nó corrente não é o destino da difusão.

O segundo componente RABR a instalar-se no nó 3, antes de parar o seu processamento, passa ao componente RABR já hospedado o número de saltos para alcançar o nó 3 e a interface por onde chegou. Neste caso, o segundo componente chegou ao nó 3 através de uma rota válida como caminho alternativo (com número de saltos menor que o do raio da área de busca e não tendo já passado no nó 3). Então, o primeiro componente RABR registra as informações recebidas na tabela de retorno e prossegue seu processamento normal. Entretanto, se o número de saltos do segundo componente RABR para chegar ao nó 3 fosse menor do que aqueles já registrados na tabela de retorno, o primeiro componente RABR difundiria cópias de seu código por todas as interfaces ativas, excetuando aquela pela qual chegou o segundo componente RABR.

4.5 – Considerações finais

Neste capítulo, foi apresentado um protótipo de implementação de uma arquitetura de rerroteamento pró-ativo visando o ambiente de aplicação ServiMídia/MPLS. O protótipo foi desenvolvido especificamente para a realização do rerroteamento de fluxos com restrições de banda e retardo. Este protótipo consistiu na implementação de três agentes: LerEntradaBandaRetardo, LsrBandaRetardo e RotaAlternativaBandaRetardo. Foi salientado

que a organização das funcionalidades dos agentes da arquitetura em classes facilita a implementação de novos agentes para atender outras restrições de QoS. Finalmente, dois diagramas de seqüência detalharam o funcionamento da arquitetura e as principais funções executadas pelos agentes de rerroteamento.

Capítulo 5 – Testes e análise dos resultados

Os testes foram efetuados em uma topologia na qual estavam presentes apenas os LSR referentes ao LSP e aos trechos alternativos. Os agentes de rerroteamento foram especificamente projetados e implementados para atender fluxos com restrição de banda e retardo. Foi assumido que o LSP já estava estabelecido e a sua rota não era conhecida pelo Gerente de Desempenho de Domínio (LSP estabelecido sob demanda). As únicas informações fornecidas aos agentes de rerroteamento eram o endereço IP do destino do fluxo, onde executava um cliente ServiMidia, o identificador do fluxo, o retardo e banda exigidos pelo fluxo.

Foi também assumido que os valores do retardo considerados pelo Inspetor de Desempenho [6] para geração dos alarmes de tendências são iguais ao valores do retardo fim-a-fim do LSP, ou seja, o LSP liga cliente e servidor do fluxo, ambos do ServiMídia.

Os testes foram realizados com o objetivo de verificar a viabilidade do rerroteamento de fluxos de forma pró-ativa no cenário de emprego do sistema ServiMídia e em função dos resultados obtidos pelos testes realizados na Arquitetura de Gerenciamento de Desempenho Pró-ativo [6]. Além disso, os testes permitiram que fossem avaliadas as conseqüências do uso de uma linguagem interpretada como Java executando operações na camada de rede. Finalmente, os testes possibilitaram, através de extrapolações, determinar os limites dentro dos quais é possível, em tempo hábil, concretizar o chaveamento de fluxos.

A seção seguinte, seção 5.1, define a metodologia utilizada nos testes e o tratamento estatístico adotado. Os testes realizados com os seus respectivos resultados, bem como as topologias utilizadas estão detalhados na seção 5.2. A seção 5.3 apresenta a análise dos resultados que foram obtidos. A seção 5.4 finaliza o capítulo apresentando as considerações finais.

5.1 – Metodologia

Para cada teste, foi realizada uma rodada com no mínimo cinqüenta medidas. Como um dos principais objetivos dos testes é constatar a viabilidade do rerroteamento pró-ativo proposto no atendimento a solicitações do Sistema de Gerenciamento de Desempenho Pró-ativo, foi adotado o mesmo tratamento estatístico utilizado em [6]. Para cada rodada foram computados a média, o desvio padrão, o intervalo de confiança de 95% e o índice de ajuste à distribuição normal, obtido pelo teste Qui-quadrado [37].

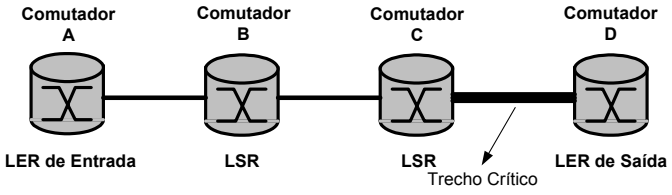
5.2 – Testes realizados e resultados obtidos

O protótipo de rerroteamento pró-ativo que foi implementado foi submetido a cinco tipos de teste correspondendo cada um deles às fases descritas no Capítulo 3: (i) Instalação dos agentes (IA); (ii) Monitoração do Circuito Virtual (MCV); (iii) Descoberta de Trechos Alternativos (DTA); (iv) Monitoração dos Trechos Alternativos (MTA) e (v) Mudança de Rota (MR).

5.2.1 – Instalação dos agentes (IA)

Este teste consiste em determinar o tempo decorrido entre o instante anterior à instanciação do agente LerEntradaBandaRetardo (LEBR) no LER de Entrada e o instante em que o agente RotaAlternativaBandaRetardo (RABR), já instalado no primeiro nó do trecho crítico, executa a sua primeira instrução. Os resultados permitem determinar o tempo mínimo de duração dos fluxos monitorados que podem ser atendidos pelo rerroteamento pró-ativo. O agente LerEntradaBandaRetardo (LEBR), conhecendo o tempo de instalação dos agentes de rerroteamento em função do comprimento do LSP, pode prontamente aceitar ou rejeitar solicitações prematuras de rerroteamento de fluxos que venham a ocorrer durante a fase de instalação ou mesmo de difusão.

A topologia do LSP utilizado para esse teste está apresentada na Figura 17. O trecho crítico está assinalado e também é apresentada uma tabela com as características das plataformas utilizadas na topologia. Os trechos alternativos presentes na topologia foram omitidos por não serem relevantes nesse teste.



Comutador	CPU	Clock (MHz)	Memória Cache (Kbytes)	Memória RAM
A	Pentium II	350	-	256
B	K6-II	500	-	256
C	K6-II	500	-	256
D	K6-II	500	512	256

Figura 17 – Topologia do teste IA e configuração das plataformas

Os LSR foram configurados manualmente de forma a estabelecer um LSP entre os comutadores A (LER de Entrada) e D (LER de Saída). Para as mensagens de estado, não é

necessária a configuração de rotas estáticas porque essas mensagens devem necessariamente ser processadas em todos os nós intermediários antes de chegar ao LER de Entrada. Logo, as redes destinos dessas mensagens são sempre redes diretamente conectadas aos comutadores.

As operações necessárias durante a difusão dos agentes foram agrupadas em fases com objetivo de identificar aquelas que são executadas em função do comprimento do LSP. Desta forma, foi possível determinar os limites do esquema de rerroteamento adotado. As operações desta fase foram agrupadas em 4 subfases conforme apresentado na Figura 18.

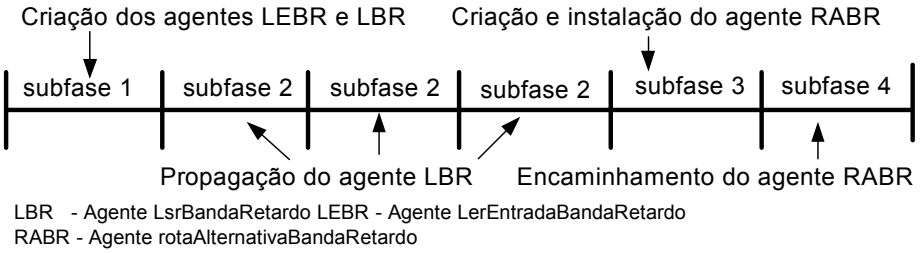


Figura 18 – Subfases do teste IA

As subfases ilustradas na Figura 18 não ocorrem sempre de maneira seqüencial. Dependendo do comprimento do LSP, algumas subfases podem ser executadas tantas vezes quantos forem o número de saltos entre o primeiro LSR após o LER de Entrada e o LER de Saída, enquanto outras podem não ser executadas. A subfase 2, por exemplo, não existiria caso o LSP possuísse somente um LER de Entrada e um LER de Saída. A seqüência da Figura 18 refere-se ao LSP da Figura 17. As operações executadas em cada uma das fases estão detalhadas no anexo B.

A Tabela 1 apresenta de maneira resumida os tempos das diversas subfases, ou seja, estão somente indicados os tempos máximos, mínimos e médios de cada subfase. Todos os valores medidos estão disponíveis no anexo C. Os tempos das subfases correspondem a um único salto.

Tabela 1 – Resultados do teste da fase IA (ms)

Medida	Subfase 1	Subfase 2	Subfase 3
Média	846	580	321
Desvio Padrão	41	20	4,58
Mínimo	698	544	312
Máximo	941	632	330
Int. conf. 95%	± 11,39	± 5,44	± 1,27
Ajuste à normal (%)	99,99	99,99	99,99

5.2.2 – Monitoração do Circuito Virtual (MCV)

Esta fase inicia quando o agente LsrBandaRetardo (LBR) instala-se no LER de saída e seu término se dá juntamente com o fluxo a ser rerroteado. O tempo de execução dessa fase tem também impacto no tempo mínimo de duração dos fluxos monitorados que podem ser atendidos pelo rerroteamento pró-ativo. Isto ocorre porque, durante a fase de instalação dos agentes, é necessário que o agente LEBR receba três mensagens de estado consecutivas para criar o agente RABR.

Esta fase consiste basicamente na obtenção do estado local (valores do retardo dos enlaces) e envio das mensagens com essas informações ao nó vizinho anterior. A topologia de teste é a mesma do teste anterior (Figura 17). A Tabela 2 apresenta os valores médio, máximo e mínimo. Todos os valores medidos estão no anexo C.

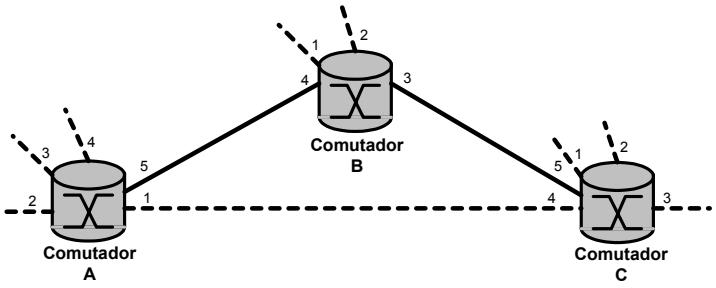
Tabela 2 – Resultados do teste da fase MCV (ms)

Medida	Fase MCV
Média	33,5
Desvio Padrão	0,35
Mínimo	32,5
Máximo	35,0
Int. de Conf. 95%	± 0,10
Ajuste à normal (%)	99,99

5.2.3 – Descoberta de Trechos Alternativos (DTA)

O teste DTA foi executado para obter o tempo total gasto entre o início da execução do agente RotaAlternativaBandaRetardo (RABR) no primeiro nó do trecho crítico e o início da execução de sua réplica no próximo nó do trecho alternativo.

A Figura 19 mostra a topologia que foi utilizada para o teste. Os demais LSR do LSP foram omitidos por não serem relevantes no teste. As linhas pontilhadas que saem dos comutadores representam as interfaces que, em uma situação normal, seriam também utilizadas para a difusão do agente RABR.



Comutador	CPU	Clock (MHz)	Memória Cache (Kbytes)	Memória RAM
A	K6-II	500	-	256
C	K6-II	500	-	256
D	K6-II	500	512	256

Figura 19 – Topologia do teste DTA e configuração das plataformas

Neste teste, os LSR não foram configurados para estabelecer um LSP porque este só é estabelecido quando ocorre a primeira passagem das mensagens de estado (abordagem antecipada) ou chega uma solicitação de rerroteamento (abordagem sob demanda).

As principais operações desta fase são a criação e o envio das cópias do agente RABR. A Tabela 3 resume os tempos obtidos no teste de difusão. Somente os valores máximo e mínimo, assim como a média, estão apresentados. Todos os valores medidos estão disponíveis nas tabelas do anexo C.

Tabela 3 – Resultados do teste da Fase DTA (ms)

Medida	Fase DTA
Média	326
Desvio Padrão	15
Mínimo	303
Máximo	366
Int. de Conf. 95%	± 4,24
Ajuste à normal (%)	99,99

5.2.4 – Monitoração dos Trechos Alternativos (MTA)

A fase MTA tem como objetivo a coleta de informações de estado (banda e retardo) pelo agente RABR do primeiro nó do trecho crítico para registro dos trechos alternativos descobertos durante a fase DTA. É necessário que, depois da instalação do agente RABR no primeiro nó do trecho crítico, as cópias (clones) deste agente sejam instaladas em todos os nós de todos os trechos alternativos (fase DTA), incluindo os nós destinos de difusão, para que as

mensagens de estado sejam geradas e retransmitidas de volta por esses agentes até o primeiro nó do trecho crítico.

Este teste consiste em medir a latência de todas as operações necessárias para o envio e recebimento das mensagens de estado. Esta fase inicia quando o primeiro agente RABR instala-se no nó destino da difusão e termina assim que o fluxo deixa de existir.

A topologia de teste é a mesma do teste da fase DTA (Figura 19). Foram definidas duas subfases para identificar as operações necessárias à abordagem de geração de rótulos antecipada e sob demanda. A Figura 20 apresenta a seqüência de execução das duas subfases para a topologia de teste (Figura 19) e abordagem antecipada de geração de rótulos. Caso a abordagem fosse sob demanda, a subfase 2 não existiria na Figura 20. O detalhamento das operações das duas subfases estão no anexo B.

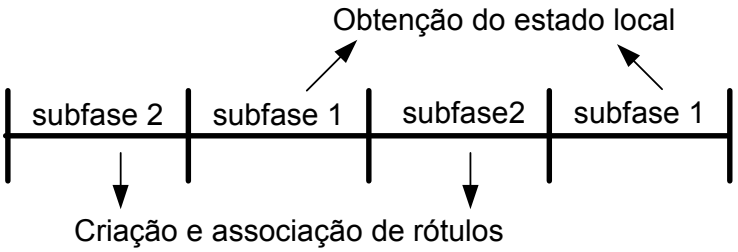


Figura 20 – Subfases do teste MTA

A Tabela 4 apresenta os tempos obtidos durante os testes. Os tempos médios, máximos e mínimos estão apresentados. Os tempos de todas as medidas podem ser consultados no anexo C.

Tabela 4 – Resultados do teste da fase MTA (ms)

Medida	Subfase 1	Subfase 2
Média	27,9	74,7
Desvio Padrão	0,71	3,89
Mínimo	27,0	72,0
Máximo	29,0	78,0
Int. de Conf. 95%	± 0,20	± 1,08
Ajuste à normal (%)	99,99	99,99

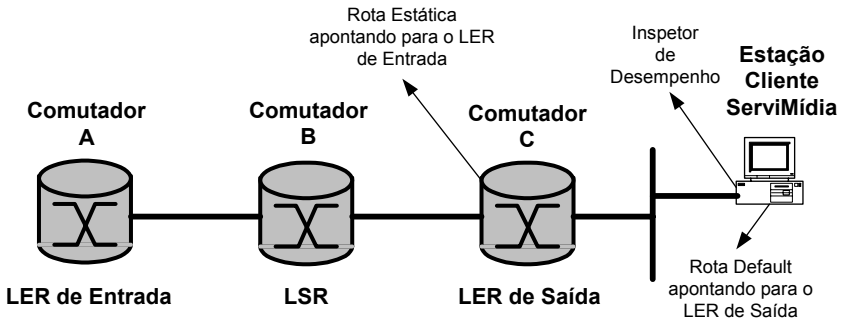
5.2.5 – Mudança de rota

Este teste consiste na obtenção do tempo decorrido entre a solicitação de rerroteamento pelo Inspetor de Desempenho hospedado na estação cliente ServiMídia e a troca do rótulo correspondente ao novo LSP executada pelo agente RABR instalado no primeiro nó do trecho crítico. Cabe salientar que as operações envolvidas neste teste são aquelas efetivadas somente após a ocorrência do evento de solicitação de rerroteamento, sendo, portanto, as mais críticas em termos de tempo de execução.

Este teste foi dividido em três subtestes: (i) notificação de rerroteamento do Inspetor de Desempenho ao agente LEBR hospedado no LER de Entrada; (ii) chaveamento do fluxo na abordagem antecipada e (iii) chaveamento do fluxo na abordagem sob demanda.

Subteste 1: Notificação de rerroteamento do Inspetor de Desempenho

Este subteste consiste em medir o tempo entre o início do envio da notificação pelo Inspetor de desempenho e o fim do tratamento dessa mensagem pelo agente LEBR no LER de entrada. O esquema de interligação e a tabela com a configuração dos comutadores estão apresentados na Figura 21.



Dispositivo	CPU	Clock (MHz)	Memória Cache (Kbytes)	Memória RAM
Comutador A	K6-II	350	-	256
Comutador B	K6-II	500	-	256
Comutador C	K6-II	500	-	256
Estação Cliente ServiMídia	Pentium II	500	512	256

Figura 21 – Topologia do teste MR e configuração de plataformas – Subteste 1

Considerou-se que a estação cliente ServiMídia a partir da qual o Inspetor de Desempenho gerou a notificação de rerroteamento estava em uma sub-rede diretamente conectada ao LER de Saída. Além disso, o próprio LER de Saída foi configurado como o roteador de saída desta sub-rede. Foi também estabelecida como rota da mensagem de notificação com destino ao

LER de Entrada o próprio LSP através da inclusão de uma rota estática no comutador C, que é o LER de Saída.

O teste foi dividido em duas subfases para verificar o impacto causado pela distância, em saltos, entre a estação cliente ServiMídia e o LER de Entrada. Esta distância, de acordo com a Figura 21, é o comprimento do LSP do fluxo monitorado. A primeira é executada uma única vez não importando o comprimento do LSP. A segunda subfase é executada tantas vezes quanto for o número de nós do LSP. As duas subfases estão apresentadas na Figura 22.

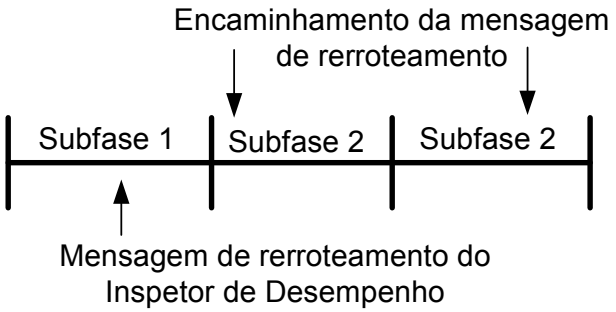


Figura 22 – Subfases do teste MR – Subteste 1

Os valores medidos estão apresentados na Tabela 5. Somente os valores máximos, mínimos e médios constam nesta tabela. Todos os valores medidos estão no anexo C.

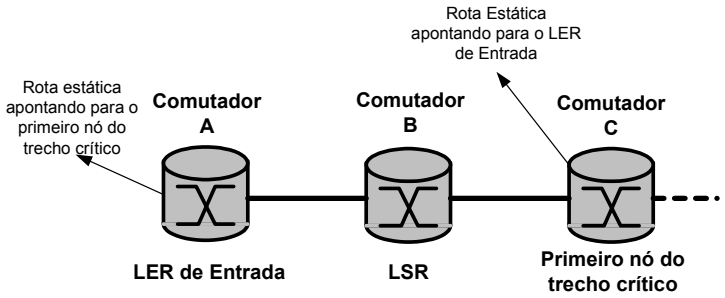
Tabela 5 – Resultados do teste da fase MR – Subteste 1 (ms)

Medida	Subfase 1
Média	4
Desvio Padrão	0
Mínimo	4
Máximo	4
Int. de Conf. 95%	0
Ajuste à normal (%)	100

Subteste 2: chaveamento do fluxo na abordagem antecipada

O subteste 2 constitui-se das operações de sinalização de chaveamento entre o agente LEBR e o RABR instalado no primeiro nó do trecho crítico e a troca de rótulo efetuada pelo agente RABR. Foi medido, então, o tempo decorrido entre o envio da mensagem de chaveamento, após o tratamento da mensagem de rerroteamento, e a troca de rótulo realizada pelo agente RABR do primeiro nó do trecho crítico.

As operações que constituem o subteste 2 foram agrupadas em três subfases de forma a identificar qual o grupo de operações que são executadas em função do número de saltos entre o LER de Entrada e o primeiro nó do trecho crítico. A Figura 23, mostra a topologia de teste e, também, as especificações dos comutadores envolvidos.



Dispositivo	CPU	Clock (MHz)	Memória Cache (Kbytes)	Memória RAM
Comutador A	K6-II	350	-	256
Comutador B	K6-II	500	-	256
Comutador C	K6-II	500	-	256

Figura 23 – Topologia do teste MR e configuração de plataformas – Subteste 2

Das três fases existentes, conforme mostra a Figura 24, somente a subfase 2 é executada tantas vezes quantos forem os saltos entre o LER de Entrada e primeiro nó do trecho crítico.

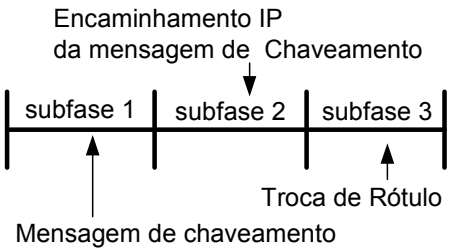


Figura 24 – Subfases do teste MR – Subteste 2

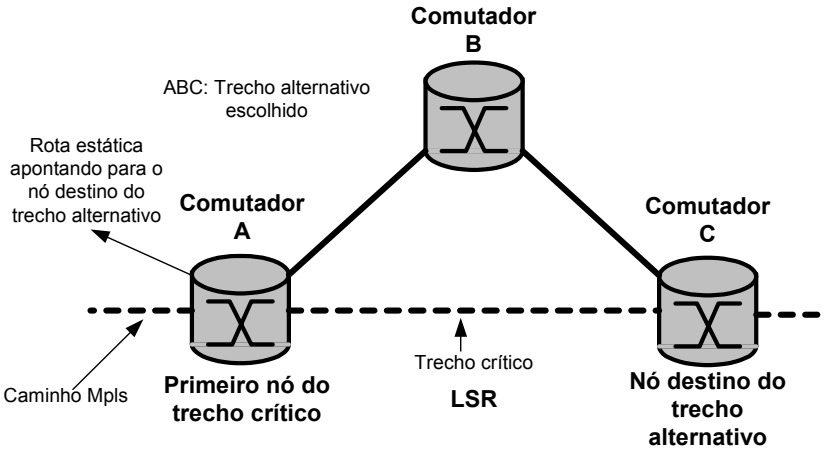
A Tabela 6 apresenta os tempos médios, mínimos e máximos das subfases. Os tempos de todas as medidas efetuadas estão no anexo C.

Tabela 6 – Resultados do teste da fase MR – Subteste 2 (ms)

Medida	Subfase 1	Subfase 3
Média	4	90
Desvio Padrão	0	2,87
Mínimo	4	87
Máximo	4	97
Int. de Conf. 95%	0	± 0,80
Ajuste à normal (%)	100	99,99

Subteste 3: chaveamento do fluxo na abordagem sob demanda

Este teste consiste em medir a latência das operações adicionais necessárias quando a abordagem de geração de rótulos é feita sob demanda. A topologia de teste juntamente com as especificações das estações estão apresentadas na Figura 25.



Dispositivo	CPU	Clock (MHz)	Memória Cache (Kbytes)	Memória RAM
Comutador A	K6-II	350	-	256
Comutador B	K6-II	500	-	256
Comutador C	K6-II	500	-	256

Figura 25 – Topologia do teste MR e configuração de plataformas – Subteste 3

As operações adicionais foram agrupadas em subfases de maneira a identificar aquelas que são executadas em função do comprimento dos trechos crítico e alternativos. Essas subfases, conforme pode ser constatado na Figura 26, são executadas entre as subfases 2 e 3 do subteste 2.

A Figura 26 contém o diagrama das fases adicionais cujas operações estão no anexo B. A fase B é executada um número de vezes igual ao número de saltos entre o primeiro nó do trecho crítico e o nó destino do trecho alternativo enquanto a fase C é executada em função do número de enlaces do trecho alternativo.

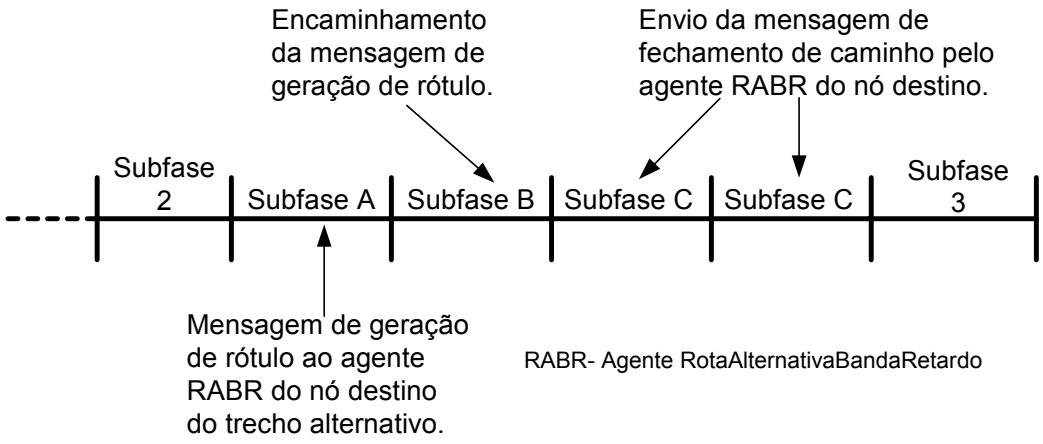


Figura 26 – Subfases do teste MR – Subteste 3

Os tempos referentes ao subteste 3 estão resumidos na Tabela 7, onde estão apresentados somente os valores máximos, mínimos e médios das fases adicionais. No anexo C, estão apresentados todos os valores medidos.

Tabela 7 – Resultados do teste da fase MR – Subteste 3 (ms)

Medida	Subfase A	Subfase C
Média	4,5	122
Desvio Padrão	0,07	3,95
Mínimo	4,5	116
Máximo	5,0	132
Int. de Conf. 95%	± 0,02	± 1,09
Ajuste à normal (%)	99,99	99,99

5.3 – Análise dos Resultados

Nesta seção, os resultados foram analisados considerando três pontos distintos: os diversos fatores que influenciam nos tempos das diversas fases; o impacto da interação dos agentes da arquitetura de rerroteamento com os sistemas operacionais dos nós e os limites do uso do protótipo implementado.

5.3.1 – Tempos de execução das fases

Vários fatores causam impacto no tempo total gasto para a execução das diversas fases. Em relação à fase de instalação dos agentes, constata-se que somente as subfases 2 e 4 são executadas em função do número de nós do LSP. Essas fases consistem na operação de encaminhamento IP tradicional que tem que ser obrigatoriamente realizada para o envio dos

agentes LBR e RABR assim como para o envio das mensagens de estado e controle (fases MCV, MTA e MR). O tempo de busca da interface de saída na tabela de roteamento (FIB) depende do número de entradas existentes que, por sua vez, é função do tamanho da rede[3]. Em todos os testes, as tabelas de roteamento (FIB) foram povoadas artificialmente com rotas estáticas para verificar o impacto na propagação dos agentes. Entretanto, a metodologia adotada nos testes não permitiu medir esses tempos. Empregando o utilitário *ping* para estimar esse tempo com as tabelas de roteamento povoadas com 1000 rotas estáticas, obteve-se um retardo (RTT) da ordem de 1,8 ms para uma distância de um salto. Esta é a razão por não terem sido apresentados os tempos das fases que envolvem encaminhamento IP referente ao envio de mensagens ou agentes. Então os tempos dessas fases foram desprezados em virtude de seus valores serem bastante inferiores aos das outras fases (menores que 1 %).

O tamanho das mensagens de estado é um fator que contribui no tempo da fase de monitoração do circuito virtual. As mensagens de estado do circuito virtual possuem tamanhos variáveis porque as informações de estado (banda disponível e retardo dos enlaces) são agregadas à mensagem à medida que as mesmas vão passando pelos LSR, fazendo com que o seu tamanho aumente. Para simplificar a análise dos resultados dos testes, adotou-se um tamanho fixo para todas as mensagens de estado independentemente do comprimento do LSP utilizado. O tamanho adotado corresponde ao tamanho da mensagem que o agente LEBR receberia caso o comprimento do LSP fosse de 10 saltos (87 octetos). Esta simplificação acarreta tempos maiores pois, em situação real para o caso de um comprimento de 10 saltos, o tamanho das mensagem variaria de um mínimo de 23 octetos para o primeiro comutador até um máximo de 87 no LER de Entrada. É esperado que o impacto desta simplificação seja maior no processamento do que na transmissão, porque a diferença teórica entre os tempos totais de transmissão na situação real e na situação simplificada de teste para a taxa de 10 Mbit/s dos enlaces é de apenas 51,2 μs. As mensagens de estado dos trechos alternativos nos testes executados possuíam tamanhos fixos porque a informação de banda e retardo eram valores acumulados referentes ao trecho alternativo em questão.

As mensagens de controle que são utilizadas na fase de mudança de rota devem ser o mais curtas possíveis para minimizar o seu tempo de transmissão e, principalmente, evitar ou eliminar a fragmentação em nível IP e enlace. A fragmentação possui a desvantagem da carga

[3] O tamanho da rede é o número de nós que possui [1].

computacional adicional produzida pelos processos de segmentação na origem e de remontagem no destino. Como o tamanho das mensagens de controle, de acordo com o formato descrito no Capítulo 4, é de 18 octetos, não ocorreu fragmentação durante os testes.

A operação de cálculo do trecho crítico da subfase 3 do teste de instalação dos agentes depende, também, do comprimento do LSP, devido ao número de iterações do algoritmo variar em função do número de enlaces e quantidade de métricas envolvidas. Estabeleceu-se que o algoritmo foi sempre executado para o caso no qual existem 10 saltos no LSP, independentemente do comprimento real do LSP do teste.

O tempo gasto para a execução da fase DTA depende do número de interfaces que os nós dos trechos alternativos possuem. Observando a Figura 19, conclui-se que, no pior caso, as cópias dos agentes RABR, ao migrarem pela interface correta (aquela que alcança o nó destino) na última tentativa em todos os nós (seguindo a numeração das interfaces da Figura 19), levam um tempo maior para alcançar o nó destino. Isto ocorre porque a difusão dos agentes através das interfaces é feita seqüencialmente. Cada envio mal sucedido corresponde ao tempo gasto para replicar o código e enviar o mesmo pela interface incorreta. Considerando a Figura 19, o tempo adicional para todo o trecho crítico seria 6 vezes o tempo de replicar um único agente e enviar a réplica por uma única interface (Tabela 7), totalizando 463,8 ms. O tempo de envio corresponde ao tempo de execução do método *enviaAgente()* (77,3 ms – Tabela 8). Este método invoca o método *copyThread* da classe *Relocator* do μCode para serializar e enviar a cópia do agente RABR.

O tempo de execução da subfase 1 (846 ms) do teste IA (Tabela 1) é cerca de 45,8 % superior ao da subfase 2 (580 ms) do mesmo teste. A razão desta diferença é que, na subfase 1, são efetuadas duas operações (instanciação do agente LEBR e LBR) com alto consumo de tempo de CPU enquanto, na subfase 1, é realizada somente uma (remontagem do agente LBR). Entretanto, o impacto da subfase 1 é atenuado porque só é executada uma única vez ao passo que as operações da subfase 2 são efetuadas em função do comprimento do LSP.

5.3.2 – Interação com o sistema operacional

Existem basicamente dois tipos de subfases: aquelas envolvidas na migração de agentes e aquelas relacionadas com o envio de mensagens.

No primeiro tipo, a criação dos agentes (instanciação), a remontagem dos agentes no destino e o acesso ao sistema operacional do nó são as operações que mais contribuem para as latências dessas fases. Foi constatado que o tempo de reconstrução das estruturas de dados dos agentes no destino é significativamente maior do que o tempo de serialização e de envio. A Tabela 8 apresenta os tempos de execução do método *enviaAgente()* quando é invocado pelos agentes LBR e RABR.

Tabela 8 – Resultados do teste de execução do método enviaAgente() (ms)

Medida	enviaAgente()	
	Agente LBR	Agente RABR
Média	73,8	77,3
Desvio Padrão	0,71	0,92
Mínimo	71,5	76
Máximo	74,5	80
Int. de Conf. 95%	± 0,20	± 0,26
Ajuste à normal (%)	99,99	99,99

Os métodos que interagem com os sistemas operacionais das estações utilizadas nos testes executam scripts desenvolvidos em Perl para obter informações das tabelas de roteamento e rótulos, das interfaces locais ou para efetuar a geração e associação de rótulos. A Tabela 9 apresenta os tempos de execução desses métodos.

Tabela 9 – Resultados do teste de execução dos métodos de sistema (ms)

Medidas	proxSaltoCircuitoVirtual()	retornaInterfacesAtivas()
Média	61,3	67
Desvio Padrão	1,37	1,01
Mínimo	59,0	65
Máximo	65,5	70
Int. de Conf. 95%	± 0,38	± 0,28
Ajuste à normal (%)	99,99	99,99

As cópias do agente LBR durante as subfases 1 e 2 do teste de instalação dos agentes invocam o método *proxSaltoCircuitoVirtual()* para descobrir o endereço IP do próximo nó do LSP do fluxo, migrando em seguida. Esse método recebe como parâmetro o endereço IP destino do fluxo monitorado (LER de Entrada) ou o rótulo de entrada (LSR ou LER de Saída). O script associado consulta as tabelas de roteamento e rótulos para depois retornar um vetor contendo o endereço IP da interface de saída local, o endereço IP do próximo nó e o rótulo de saída. Caso o valor do rótulo seja zero, o agente LBR conclui que está no LER de Saída.

Considerando o tempo de execução da subfase 2 do teste IA (580ms), esse método tem um peso elevado correspondendo a mais de 10 % do tempo total desta fase. Os 518,7 ms restantes são referentes principalmente à serialização e remontagem do agente LSR.

O método *retornaInterfacesAtivas()* é invocado pelo agente RABR independentemente em que nó esteja instalado. Esse método retorna todas as interfaces ativas do nó corrente em um vetor, utilizando um script que extrai os endereços IP das interfaces locais das informações geradas pelo comando */sbin/ifconfig -a*. A relação de interfaces locais obtidas por esse método permite ao agente RABR identificar que tipo de nó está instalado (LER de Entrada, primeiro nó do trecho crítico, último nó do trecho crítico ou nó intermediário do trecho crítico) e adequar o seu comportamento. O tempo médio de execução desse método (67 ms) é bastante elevado porque corresponde a 20 % do tempo total da fase DTA (326 ms).

As subfases do segundo tipo também possuem operações que interagem com o sistema operacional. A fase MCV, por exemplo, consiste basicamente na obtenção do retardo do enlace e envio da mensagem. Nessa operação, o método *obtemEstadoLocal()* executa um script para ler os valores de banda ou retardo de um arquivo texto contendo esses valores para todas as interfaces do nó. Esse arquivo foi gerado manualmente de forma que a primeira, segunda e terceira colunas correspondessem respectivamente aos endereços IP das interfaces locais, aos retardos e às bandas disponíveis dos enlaces associados a essas interfaces locais. Em situação normal, processos locais e específicos para medir esses parâmetros seriam os responsáveis pela criação e atualização desse arquivo. A contribuição dessa operação no tempo médio da fase MCV (33,5 ms – Tabela 2) é de 88 %, considerando que o tempo médio para enviar uma mensagem de estado é de 4 ms. Nas subfases dos testes MCV e MR, os percentuais de participação desse tipo de operação são também similares.

5.3.3 – Limites do rerroteamento

O tempo mínimo necessário para que seja possível efetuar o rerroteamento de um fluxo é o tempo transcorrido entre o inicio da execução do agente LEBR no LER de Entrada (início da monitoração do fluxo pelo inspetor de desempenho) e o registro do primeiro trecho alternativo válido realizado pelo agente RABR instalado no primeiro nó do trecho crítico. Esse tempo corresponde ao somatório da latência da fase de Instalação dos Agentes, do tempo de execução da fase de Descoberta de Trechos Alternativos (DTA), do tempo necessário para o recebimento de três mensagens de estado do circuito virtual na fase MCV e para o

recebimento da primeira mensagem de estado do trecho alternativo na fase MTA (considerando que a primeira mensagem corresponde a um trecho alternativo válido). Esse valor define o tempo de duração mínimo do fluxo para que o rerroteamento possa ser realizado.

Conforme mencionado anteriormente, algumas subfases são executadas em função dos comprimentos do LSP e trechos alternativos, tornando o tempo mínimo para rerroteamento também dependente desses parâmetros. A expressão $t_{mr}= \Sigma t_{fl}\ c_{lsp} + \Sigma t_{fa}\ c_{ta} + \Sigma t_{fi}$ estabelece a relação entre o tempo mínimo para rerroteamento e os comprimentos do LSP e do trecho alternativo onde:

- t_{mr} é o tempo mínimo para efetuar o rerroteamento.
- Σt_{fl} é o somatório dos tempos das subfases que são executadas em função do comprimento do LSP.
- c_{lsp} é o comprimento do LSP
- Σt_{fa} é o somatório dos tempos das subfases que são executadas em função do comprimento do trecho alternativo
- c_{ta} é o comprimento do trecho alternativo.
- Σt_{fi} é o somatório dos tempos das subfases que independem dos comprimentos do LSP e do trecho alternativo.

A Tabela 9 apresenta os valores dos coeficientes da expressão acima de acordo com os resultados dos testes (Tabelas 1 a 4) e a abordagem de geração de rótulos. Foi também considerado no cálculo desses coeficientes que o agente RABR só é criado pelo agente LEBR após receber três mensagens de estado do circuito virtual consecutivas. O período dessas mensagens de estado foi arbitrariamente fixado em 500 ms.

Tabela 10 – Parâmetros da equação do tempo mínimo

Antecipada			**Sob demanda**		
Σt_{fl} (ms/salto)	**Σt_{fa} (ms/salto)**	**Σt_{fi} (ms)**	**Σt_{fl} (ms/salto)**	**Σt_{fa} (ms/salto)**	**Σt_{fi} (ms)**
613,5	428,6	1587	613,5	353,9	1587

De posse desses coeficientes é possível relacionar o tempo mínimo para o rerroteamento e os comprimentos do LSP e do trecho alternativo. Então, fixando-se o comprimento do trecho alternativo obtém-se uma equação de reta, tornando o tempo mínimo para rerroteamento função somente do comprimento do LSP. A Figura 27 apresenta graficamente as equações de retas para diversos valores de comprimento do trecho alternativo, considerando a geração de rótulo antecipada. As retas são paralelas porque os seus coeficientes angulares independem do comprimento do trecho alternativo enquanto os seus coeficientes lineares são função do comprimento do trecho alternativo. Vale ressaltar que os pontos de descontinuidade do gráfico ocorrem porque quando o trecho alternativo coincide com o LSP (o trecho crítico é o próprio LSP) as fases de instalação dos agentes (IA) e de monitoração do caminho crítico (MVC) deixam de existir.

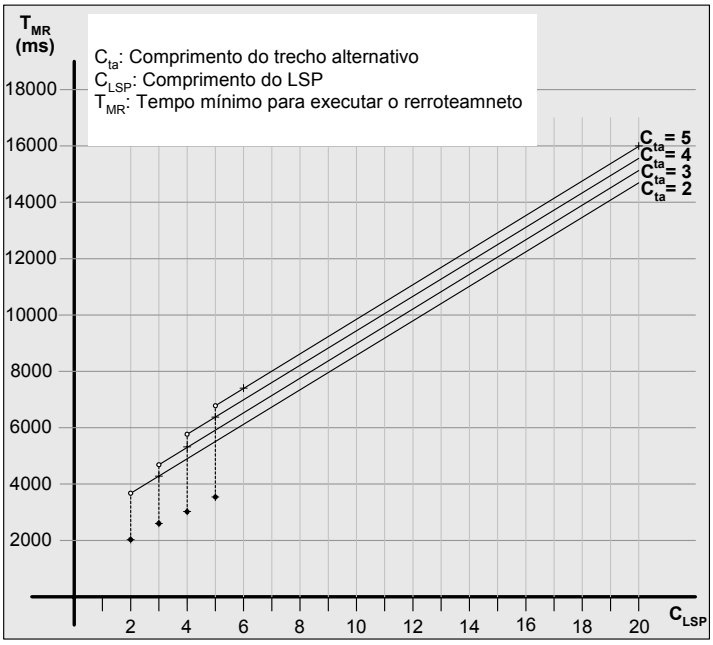


Figura 27 – Tempo mínimo para rerroteamento (geração de rótulos antecipada)

Nestes casos, o agente RABR assim que é criado pelo agente LEBR no LER de Entrada, inicia imediatamente a fase de descoberta de trechos alternativos (DTA) porque o LER de Entrada passa a ser também o primeiro nó do trecho crítico (rerroteamento pleno). Observando a Figura 27 é possível determinar qual é o tempo mínimo necessário para efetuar o rerroteamento ou o tempo mínimo de duração do fluxo que pode ser rerroteado na abordagem antecipada. Por exemplo, em um LSP de comprimento igual a 14 saltos, somente fluxos acima de 11,0, 11,5, 11,9 e 12,3 s podem ser rerroteados, considerando comprimentos de trechos alternativos iguais a 2, 3, 4 e 5 saltos respectivamente. A Figura 28 apresenta os

gráficos para a abordagem de geração de rótulos sob demanda. Os valores são menores porque a subfase 2 da fase MTA só é executada após solicitação externa e durante a fase MR.

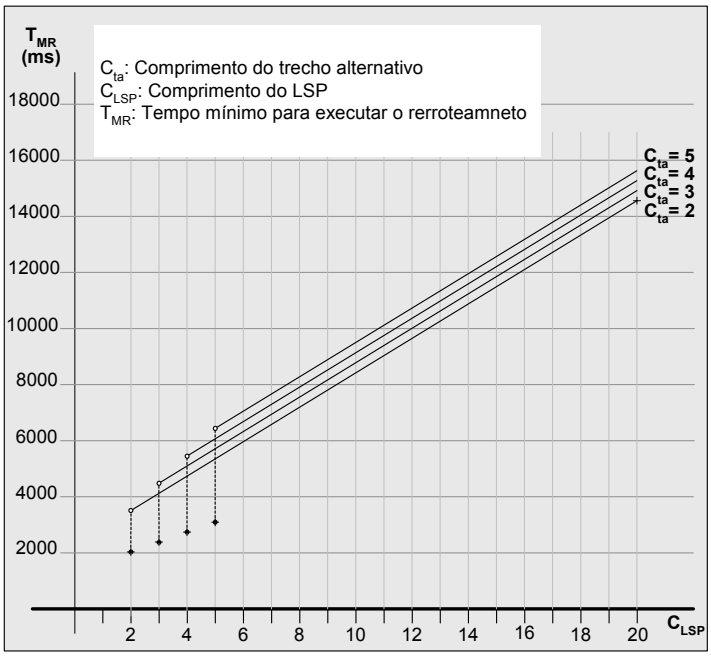


Figura 28 – Tempo mínimo para rerroteamento (geração de rótulos sob demanda)

A fase de mudança de rota é executada após a solicitação de rerroteamento do Gerenciamento de Desempenho Pró-ativo. Esta fase é considerada reativa porque sua execução é efetuada após a falha de QoS ou a detecção de tendência de falha. No caso da geração de rótulo antecipada, o tempo total para esta fase é o somatório dos tempos dos subtestes 1 e 2, totalizando 98 ms. Quando a geração de rótulos sob demanda é adotada, os tempos das subfases adicionais devem ser considerados. Entretanto, o número de execuções dessas subfases depende do comprimento do trecho alternativo. A expressão $t_{cv} = \Sigma t_{fa}\ c_{ta} + \Sigma t_{fi}$ define essa dependência. Os coeficientes são:

- Σt_{fa}: somatório dos tempos das subfases adicionais que são executadas de acordo com o comprimento do trecho alternativo.

- Σt_{fi} somatório dos tempos das subfases adicionais que independem do comprimento do trecho alternativo

Utilizando os valores médios das Tabelas 6 e 7, obtém-se o gráfico da Figura 29. Neste gráfico, os valores dos tempos de chaveamento são inferiores a 834,5 ms para comprimentos de trechos alternativos menores que 6 saltos, comprovando que é possível o redirecionamento

de fluxos mediante solicitação do Gerenciamento de Desempenho Pró-ativo (GDPA) antes da ocorrência de falha de QoS. O GDPA [6] é capaz de detectar tendências de falhas com 5 s segundos em média de antecedência à falha e portanto acima dos 834,5 ms necessários ao rerroteamento. Desta forma, é viável que ações de rerroteamento sejam efetuadas evitando o processo de adaptação nas aplicações ServiMídia.

A Arquitetura de Rerroteamento Pró-ativo é ainda capaz de rerrotear fluxos dependendo dos comprimentos do LSP e trechos alternativos nos casos de solicitações prematuras, ou seja, solicitações que são efetuadas antes da instalação dos agentes e descoberta dos trechos alternativos (durante o tempo t_{MR}). Considerando, por exemplo, a geração de rótulos sob demanda, comprimentos do LSP e do trecho alternativo iguais a 5 saltos, o tempo total para efetuar o rerroteamento ($t_{MR} + t_{cv}$) é de 3,8 s e portanto inferior aos 5 s que é o tempo decorrido entre o momento da detecção da tendência de falha de QoS realizada pelo Gerenciamento de Desempenho Pró-ativo (GDPA) [6] e a ocorrência da falha.

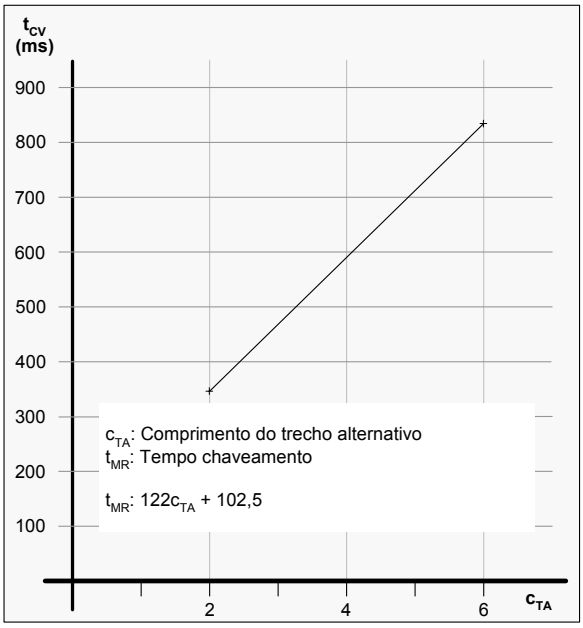


Figura 29 – Tempo de chaveamento (geração de rótulos sob demanda)

5.4 – Considerações finais

Os objetivos dos testes e as considerações de execução dos mesmos em um cenário de emprego do ServiMídia foram apresentados no início deste capítulo. A primeira seção descreveu a metodologia adotada nos testes. Os testes foram organizados em função das fases de operação descritas no Capítulo 3. Os resultados obtidos foram analisados considerando inicialmente os fatores gerais que influenciaram os tempos das diversas fases e subfases. O

impacto da interação com os sistemas operacionais dos comutadores dos testes foi salientado. Os resultados obtidos indicaram que o protótipo implementado é capaz em tempo hábil de redirecionar o fluxo mediante solicitação do gerenciamento de desempenho pró-ativo.

Capítulo 6 – Conclusão

O propósito principal da Arquitetura de Rerroteamento Pró-ativo apresentada é permitir o redirecionamento de fluxos de aplicações críticas mediante solicitações externas. Esses eventos externos podem ser oriundos de detecção de falhas de QoS ou de tendência de falhas. A arquitetura proposta viabiliza o rerroteamento pró-ativo uma vez que todos os procedimentos necessários à descoberta de rotas alternativas podem ser efetuados em tempo de monitoramento do fluxo, ou seja, somente as operações relativas ao chaveamento do fluxo são realizadas após a solicitação de redirecionamento externo. O rerroteamento é realizado sob uma infra-estrutura de rede baseada em circuitos virtuais, de rede com roteamento baseado em QoS e de rede com tecnologia ativa.

Três são os componentes que compõem a arquitetura de rerroteamento: AgenteNóEntrada (ANE), AgenteNóIntermediário (ANI) e AgenteRotaAlternativa (ARA). O componente ANE é o elemento gerenciador do rerroteamento dos fluxos pertencentes ao mesmo circuito virtual. Este componente se encarrega de criar o componente ANI, tratar as mensagens de atualização, construir a tabela de estado do circuito virtual, calcular a posição do trecho crítico e criar o componente ARA.

O componente ANI tem como função instalar cópias de seu próprio código em todos os nós intermediários e nó de saída do circuito virtual para executar a monitoração dos estados locais desses nós. As informações locais de estado são periodicamente enviadas ao componente ANE para que este possa desempenhar as suas funções.

O componente ARA é o elemento incumbido de descobrir os trechos alternativos em torno do trecho crítico. A técnica utilizada foi a difusão limitada restringindo o processo de descoberta dos trechos alternativos à área de busca. Após a difusão, cópias desse componente permanecem instaladas em todos os nós de todos os trechos alternativos descobertos para a monitoração desses trechos. O componente ARA no nó destino efetua o envio periódico de mensagens de estado pelos enlaces diretamente conectados que fazem parte dos trechos alternativos. No nó origem, recebe as mensagens de estado e constrói a tabela de estado dos trechos alternativos. Além disso, efetua a troca de rótulo para redirecionar o fluxo. A troca de rótulo ocorre após o recebimento da mensagem de controle gerada pelo componente ANE e escolha do trecho alternativo.

A Arquitetura de Rerroteamento Pró-ativo adotou dois princípios: (i) limitar a difusão e instalação de agentes somente aos nós onde fosse necessário capacidade de processamento para efetuar as operação de rerroteamento e, quando possível, optou pelo uso de mensagens. Este princípio visou a minimização no uso de recursos da rede uma vez que o custo computacional nos nós e banda nos enlaces no tratamento e transporte de mensagens é menor comparado aos custos para a mobilidade dos agentes; (ii) o uso de uma infra-estrutura de mobilidade e de execução de programas móveis e independente de plataforma baseada na linguagem Java. Apesar de impor uma sobrecarga maior, torna as tarefas de atualização de código e instalação de novos protocolos imediatas sem a necessidade de padronização.

O cenário de aplicação para a Arquitetura de Rerroteamento Pró-ativo foi o Sistema ServiMídia operando sobre uma infra-estrutura de rede MPLS e o Sistema de Gerenciamento de Desempenho Pró-ativo (GDPA). Para este ambiente, os componentes foram desenvolvidos de forma especializada de maneira a minimizar o tamanho de seus códigos. A especialização dos componentes implementados foi para atender aos fluxos com restrição de banda e retardo. A implementação dos componentes obedeceu os princípios da orientação a objetos. O diagrama de classes apresentado facilitou identificar as funcionalidades mínimas necessárias e estruturar o código no desenvolvimento do protótipo. A vantagem adicional ao utilizar o paradigma da orientação a objetos é permitir o reuso de código referente à funcionalidades comuns a qualquer tipo de agente de rerroteamento que venha a ser implementado.

O protótipo foi submetido a uma série de testes com a finalidade de investigar a viabilidade da arquitetura proposta no cenário ServiMídia/MPLS. Os impactos da linguagem Java, da infra-estrutura de mobilidade adotada (μCode) e de mobilidade de código na forma de agentes nas ações de rerroteamento de fluxos sob uma infra-estrutura de comutação de rótulos MPLS puderam também ser avaliados.

Os testes foram estruturados de maneira que fosse possível determinar os limites da arquitetura proposta em função do comprimento do circuito virtual, do trecho alternativo e dos resultados obtidos em [6]. Então, os resultados obtidos foram apresentados de acordo com as fases identificadas no Capítulo 3: (i) Instalação dos agentes (IA); (ii) Monitoração do Circuito Virtual (MCV); (iii) Descoberta de Trechos Alternativos (DTA); (iv) Monitoração dos Trechos Alternativos (MTA) e (v) mudança de rota.

Os resultados obtidos evidenciam que a mobilidade dos componentes da arquitetura é o fator de maior impacto na latência das operações de rerroteamento. Entretanto, considerando a

acentuada discrepância entre as capacidades de processamento das estações utilizadas nos testes e as já existentes no mercado, é possível afirmar que já é factível o uso dessas tecnologias em operações de rerroteamento.

Um outro fator que degradou consideravelmente o desempenho do protótipo é a interação dos agentes de rerroteamento com os sistemas operacionais dos nós. Por exemplo, o tempo mínimo (T_{MR}) necessário para efetuar o redirecionamento de um fluxo é igual a 9,1 s em LSP's de 12 saltos com trecho crítico de 2 saltos na abordagem de geração de rótulos sob demanda. Desse valor, 1979 ms são devido à operações que envolvem interações com os sistemas operacionais dos nós, ou seja, cerca de 20 %. Um outro exemplo é o tempo de execução da fase Mudança de Rota (MR) quando a abordagem adotada para a geração dos rótulos é sob demanda e o comprimento do trecho alternativo é igual a 3 saltos. Dos 468 ms para redirecionar o fluxo, 442 ms são gastos na interação com os sistemas operacionais, ou seja, em torno de 94 %. Logo, é necessário que outras formas de interação com o sistema operacional sejam investigadas para minimizar esse impacto.

Os gráficos das Figuras 27, 28 e 29 do Capítulo 5 definem claramente os limites da aplicabilidade da arquitetura de Rerroteamento em função dos comprimentos do circuito virtual e do trecho alternativo. Constata-se que a adoção de formas mais eficientes de interação com os sistemas operacionais dos nós estenderão esses limites, permitindo que fluxos com tempos de duração menores possam ser rerroteados. Além disso, requisições externas de rerroteamento poderão ser atendidas para circuitos virtuais mais longos e mesmo que ocorram durante as fases de Instalação dos Agentes (IA) e Descoberta de Trechos alternativos (DTA).

Confrontando os tempos obtidos na fase de mudança de rota – entre 98 e 834,5 ms (antecipada ou sob demanda), que é executada após a solicitação externa de rerroteamento, e o tempo entre a detecção da tendência de falha realizada pelo GDPA [6] e o momento da efetiva ocorrência da falha (5 s), constata-se que as ações de rerroteamento podem ser efetuadas em tempo hábil. Logo, estas ações devem ser preferencialmente adotadas pelo GDPA com o objetivo de evitar os processos de adaptação das aplicações ServiMídia. Essas ações de adaptação deveriam ser desencadeadas caso não fosse possível o rerroteamento do fluxo monitorado. A adoção desse procedimento possui as vantagens de tornar as ações do GDPA mais transparentes às aplicações e de otimizar a utilização de recursos da rede.

A mudança de posição do trecho crítico, a escalabilidade da arquitetura proposta, a extensão da funcionalidade pró-ativa da arquitetura e a viabilidade da implementação da arquitetura proposta para outras tecnologias de enlace podem ser apontadas como sugestões para trabalhos futuros.

Conforme mencionado anteriormente, a posição do trecho crítico pode sofrer mudanças durante a monitoração do fluxo devido às condições momentâneas de tráfego nos enlaces e nós, provocando a criação e difusão de cópias do agente RABR em regiões distintas a todo momento. Neste trabalho, foi adotado o critério de enviar o agente RABR ao primeiro nó do trecho crítico (função de disparo) após constatar que as informações de estado contidas em três mensagens de estado consecutivas levam a três resultados iguais, indicando que o trecho crítico está estável. Entretanto, dependendo da intensidade e padrão da variação do tráfego nos enlaces, esse esquema pode levar a resultados insatisfatórios. Um aumento do número de mensagens recebidas para disparo do agente RABR tem o inconveniente de introduzir maior latência na instalação dos agentes e na descoberta de trechos alternativos, diminuindo a eficácia do rerroteamento. A concepção de algoritmos que considerem o *baseline* da rede poderia ser um caminho para solucionar ou amenizar esse problema.

A escalabilidade é um fator limitante da arquitetura proposta apesar de ter sido definido um agente (ANE) gerenciador do rerroteamento de fluxos pertencentes a caminhos virtuais com o mesmo nó de entrada. O estudo do impacto na banda passante dos enlaces e processamento nos nós devido às operações de rerroteamento em diferentes topologias de rede seria importante para determinar o número máximo de fluxos que podem ser atendidos pelo rerroteamento para uma determinada sobrecarga adicional.

Outra limitação da arquitetura de rerroteamento proposta é o fato de não considerar as tendências dos valores das métricas de QoS dos enlaces e nós para escolha do trecho alternativo. Devido a essa limitação, não é garantido que, em curto prazo, o trecho alternativo escolhido terá os recursos necessários após o rerroteamento. Então, outras solicitações de rerroteamento para o mesmo fluxo podem ser efetuadas, podendo ocorrer instabilidades. Logo, a funcionalidade pró-ativa da arquitetura poderia ser estendida dotando o agente RABR com algumas das funcionalidades dos Inspetores de Desempenho da AGDPA [6] ou solicitando ao Sistema de Gerenciamento de Desempenho que Inspetores de Desempenho Pró-ativo (GDPA) sejam instalados nos nós dos trechos alternativos para efetuar os cálculos de tendência.

Finalmente, o estudo de viabilidade do uso da Arquitetura de Rerroteamento Pró-ativo para redes baseadas em outras tecnologias de enlace como ATM e Frame Relay, considerando as peculiaridades de cada uma dessas tecnologias, permitiria aumentar a abrangência de aplicação da arquitetura de rerroteamento proposta neste trabalho.

Referências

[1] G. Apostolopoulos et al, *Intradomain QoS Routing in IP Networks: A Feasibility and Cost/Benefit Analysis*, IEEE Network Magazine, 1999.

[2] Rosen, E. et al, *Multiprotocol Label Switching Architecture*, RFC–3031, IETF, Janeiro de 2001.

[3] Y. Yemini, *The OSI Network Management Model*, IEEE Communications Magazine, vol.31, pp.20–29, maio de 1993.

[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] Cecilio, E.L, *Uma Arquitetura de Gerenciamento de Desempenho Pró-ativo Distribuído Usando Tecnologia Ativa, Dissertação de Mestrado*, NCE/IM/UFRJ, 2002.

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

[8] Sluman, C., *A Tutorial on OSI Management, Computer Networks Magazine*, Vol 17, n^{os} 4 e 5, outubro de 1989.

[9] 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.

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

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

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

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

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

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

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

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

[18] 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.

[19] Moy, J., *OSPF Version 2*, RFC–2328, IETF, abril de 1998.

[20] R. Braden, Ed. et al, *Resource ReSerVation Protocol – Version 1 Functional Especification*, RFC–2205, IETF, setembro de 1997.

[21] Jamoussi, B. et al, *Constraint–based LSP Setup Using LDP*, RFC–3212, janeiro de 2002.

[22] Apostolopoulos, G. et al, *QoS Routing Mechanisms and OSPF Extensions*, RFC–2676, IETF, agosto de 1999.

[23] G. Apostolopoulos et al, *On Reducing the Processing Cost of On–Demand QoS Path Computation*, In Proceedings of ICNP'98, páginas 80–89, outubro de 1998.

[24] M. Peyravian e A. D. Kshemkalyani, *Path Caching: Issues, Algorithms and a Simulation Study, Comp. Commun.*, vol. 20, pp. 605–614, 1997.

[25] A. Shaikh, J. Rexford e K. G. Shin, *Load–Sensitive Routing of Long–Lived IP Flows*, In Proceedings of SIGCOM 99, setembro de 1999.

[26] F. Baker et al, *Aggregation of RSVP for IPv4 and IPv6 Reservations*, RFC–3175, IETF, setembro de 2001.

[27] P. Pan e H. Schulzrine, *YESSIR: A simple Reservation Mechanism for the Internet, ACM Computer Communications Review*, vol. 29, no. 2, páginas 89–101, abril de 1999.

[28] W. Almesberger et al, *Scalable resource reservation for the Internet*, In Proceedings of PROMS–MmNet 97, novembro de 1997.

[29] Dumont, A.P.M., *AGAD: Arquitetura de Gerenciamento de Rede Ativa Distribuída*, Dissertação de Mestrado, NCE–UFRJ, outubro de 2002.

[30] D. Awduche et al, *Overview and Principles of Internet Traffic Engineering*, RFC–3272, IETF, maio de 2002.

[31] D. Awduche et al, *Requirements for Traffic Engineering Over MPLS*, RFC–2702, IETF, setembro de 1999.

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

[33] Oshimm, M., Karjoth, G., *Aglets Specification*, disponível na INTERNET via www.url: www.trl.ibm.co/aglets/spec_alpha.html

[34] A.Fuggeta, G.P. Picco, G. Vigna, *Understanding Code Mobility*, IEEE Transactions on Software Engineering, Vol. 24, 1998.

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

[36] Rumbauch, J. e Jacobson, I., UML Guia do Usuário, 2000.

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

[38] Labovitz, C. et al, *Delayed internet routing convergence,* In Proc. ACM SIGCOMM '00 pp. 175–187, Estocolmo, Suécia, 2000.

[39] H. Saito, Y. Miyao e M. Yoshida, *Traffic Engineering using Multiple Multipoint–to–Point LSPs*, Info– com'2000.

[40] Autenrieth, A. e Kirstdter, A. *Fault–Tolerance and Resilience Issues in IP–Based Networks*, Second International Workshop on the Design of Reliable Communication Networks (DRCN), abril de 2000.

[41] Andrea Fumagalli et al, *IP Restauration vs. WDM Protection: Is there an Optimal Choice?*, revista IEEE Network, novembro/dezembro de 2002.

[42] Courtiat, J., *Aplicações MPLS*, SBRC'2002 – Mini-curso, Búzios, Brasil, maio 2002.

[43] K. Oida e M. Sekido, *ARS: An Efficient Agente–Based Routing System for Qos Guarantees*, Comp. Commun., vol. 23, 2000, pp. 1437–47.

[44] M. Dorigo e G. Di Caro, *Mobile Agentes for Dynamic Routing*, Proc. 31st Int'l. Conf. Sys. Hawaii, janeiro de 1998.

[45] N. Minar et al., *Cooperating Mobile Agents for Dynamic Network Routing*, Software Agents for Future Communications Systems, Springer–Verlag, 1999.

[46] Sergio Gonzalez e Victor C. M. Leung, *QoS Routing for MPLS Networks Employing Mobile Agents*, revista IEEE Network, Maio/Junho de 2002.

[47] Gianni Di Caro e M. Dorigo, *Mobile Agents for Adaptive Routing*, Proc. 31st Hawaii International Conference on System Sciences, IEEE Computer Society Press, Los Alamitos, CA, pp. 74–83, 1998.

[48] Angélica Reyes et al, *Routing Managment Application Based on Mobile Agents on the Internet2*, EUNICE 2000, setembro 2002.

[49] Anat Bremler et al, *Fast Recovery of MPLS Paths*, AT&T Labs–Research, 2001.

[50] Scott Seongwook Lee and Mario Gerla, *Fault tolerance and load balancing in QoS provisioning with multiple MPLS paths*, Lecture Notes in Computer Science, vol. 2092, 2001.

[51] James R. Leu, http://sf.net/projects/mpls-linux/.

ANEXO A

Hierarquia de Classes da Arquitetura de Rerroteamento Pró-ativo

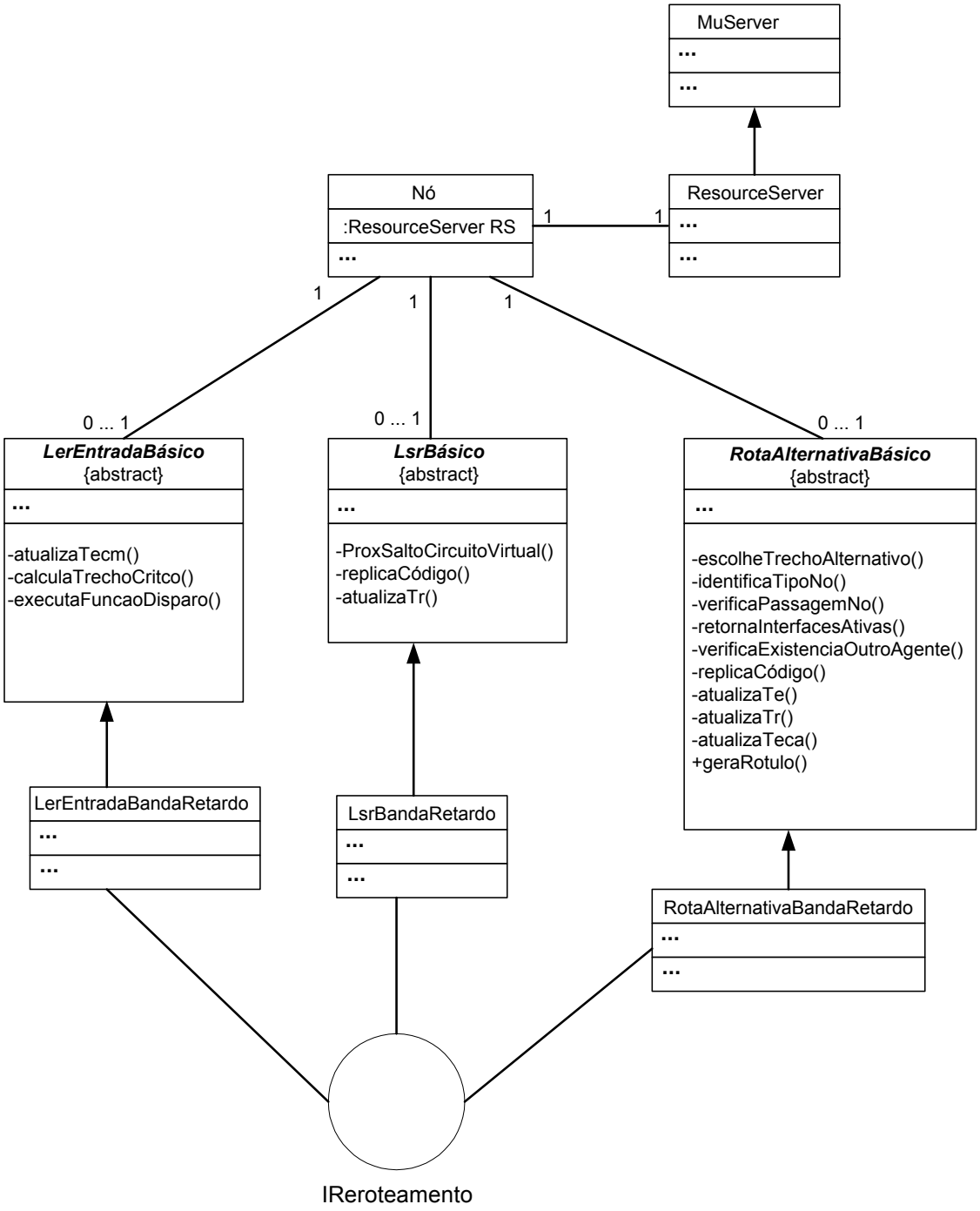


Figura 22 – Diagrama de Classes dos Agentes de Rerroteamento

O diagrama apresenta as classes com seus respectivos métodos que são necessários na implementação dos agentes de rerroteamento de fluxos que estejam submetidos a qualquer

tipo de restrição de QoS, embora esteja exemplificado as classes referente aos agentes de rerroteamento de fluxos com restrição de banda e atraso.

Classe LerEntradaBásico

A classe LerEntradaBásico reúne todos os métodos que definem o comportamento que todo o AgenteNóEntrada deve possuir independentemente das métricas de QoS para as quais foi projetado. Os principais métodos estão descritos à seguir:

atualizaTecm() – Atualiza a tabela de estado do caminho MPLS (TECM) de acordo com as informações extraídas das mensagens de atualização de estado pelo método *trataMensagem*. É executado sempre que chegam mensagens de atualização de estado ao LER de Entrada.

calculaTrechoCritco() – Determina os identificadores do primeiro e último nó do trecho crítico no LSP.

executaFuncaoDisparo() – Identifica o surgimento de um novo trecho crítico evitando o disparo desnecessário de AgenteRotaAlternativa para outros nós do caminho MPLS.

Classe LsrBasico

Todos os tipos de agentes de monitoramento dos nós do circuito virtual (AgenteNóIntermediário) apresentam as funcionalidades implementadas na classe LsrBasico cujos principais métodos estão descritos a seguir:

proxSaltoCircuitoVirtual() – Determina o próximo nó do circuito virtual. No MPLS, consulta a LIB para determinar qual a interface correspondente ao próximo salto do caminho MPLS do fluxo monitorado. Verifica também qual é o tipo de nó no qual o agente instalou-se. Os tipos de nó podem ser um Ler de Saída e LSR intermediário.

atualizaTr() – Atualiza a tabela de retorno (TR) para o reenvio das mensagens de atualização de estado ao longo do caminho MPLS no sentido *upstream*.

Classe RotaAlternativaBásico

Dentre as três classes, esta é que apresenta o maior número de funcionalidades. A sua maior complexidade se deve ao fato de que os AgenteRotaAlternativa são os responsáveis pela

descoberta dos trechos alternativos em torno do trecho crítico. Suas funcionalidades obrigatórias estão implementadas nos diversos métodos abaixo:

escolheTrechoAlternativo() – Executa o algoritmo de escolha do trecho alternativo caso exista vários disponíveis. Este método é executado antes de redirecionar o fluxo, invocando o método **chaveiaFluxo()**.

identificaTipoNo() – Verifica qual é o tipo de nó no qual o agente instalou-se. O agente RotaAlternativaBandaRetardo, através deste método, determina se é um nó intermediário de algum trecho alternativo ou o nó destino da difusão.

verificaPassagemNo() – determina se o agente RotaAlternativaBandaRetardo já passou pelo nó no qual acabou de instalar-se. Este método faz parte das medidas adotadas na Arquitetura de Roteamento Pró-ativo para eliminar rotas circulares.

RetornaInterfacesAtivas() – Retorna todas as interfaces válidas do nó para a difusão dos agentes RotaAlternativaBandaRetardo.

verificaExistenciaOutroAgente() – Retorna a referencia do objeto agente RotaAlternativaBandaRetardo já instalado ao agente RotaAlternativaBandaRetardo que acabou de instalar-se no nó. Caso esta função retorne nulo, conclui que é o primeiro a instalar-se no nó.

atualizaTe() – Registra na tabela de entrada por qual interface o agente RotaAlternativaBandaRetardo chegou e o número de saltos para alcançá-lo.

atualizaTr() – Atualiza a tabela de retorno (TR) para o reenvio das mensagens de atualização de estado dos trechos alternativos encontrados no sentido do nó destino da difusão ao primeiro nó do trecho crítico.

atualizaTeca() – Registra as informações de estado das mensagens de estado na tabela de estado dos trechos alternativos (TECA). Este método é executado pelo agente RotaAlternativaBandaRetardo instalado no primeiro nó do trecho crítico.

geraRótulo() – Gera e associa rótulos para o redirecionamento de fluxos.

Interface IRerroteamento

Além das funcionalidades específicas para cada um dos três diferentes tipos de agentes envolvidos no rerroteamento, existe um conjunto de funcionalidades que todo o agente de rerroteamento obrigatoriamente deve disponibilizar. Os métodos que implementam tais funcionalidades de rerroteamento foram agrupados em uma interface devido a razões organizacionais e, também, para garantir que serão implementadas nos novos agentes de rerroteamento. Esses métodos além de possuir implementações distintas dependendo em quais classes são implementadas, também apresentam semânticas diferentes. Os métodos dessa interface são os seguintes:

migraAgente() – Transfere de uma máquina para outra um dos agentes da arquitetura de rerroteamento à pedido, por exemplo, do Gerente de Desempenho de Domínio Pró-ativo (GDPA).

páraProcessamento() – Interrompe a execução do objeto à partir de solicitação do Gerente de Desempenho de Domínio.

configuraParâmetros() – Permite configurar os parâmetros de monitoramento das métricas de QoS como, por exemplo, a freqüência do envio das mensagens de estado.

passaParâmetros() – Retorna os valores dos parâmetros solicitados. O agente LerEntradaBandaRetardo usa esse método para obter os valores dos parâmetros que foram configurados nos agentes LsrBandaRetardo hospedados nos comutadores do circuito virtual.

enviaMensagem() – Monta e envia as mensagens de acordo com o formato descrito na seção 4.3.

obtemEstadoLocal() – Obtém do sistema local os valores das métricas de QoS relevantes para o fluxo que está sendo monitorado pelo Sistema de Gerenciamento de Desempenho Pró-ativo. No presente trabalho, as métricas relevantes são o retardo e a banda disponível.

trataMensagem() – Analisa os campos de controle das mensagens identificando o método apropriado para o tratamento das suas informações de dados.

enviaAgente() – Envia agente pela interface definida pelos métodos **proxSaltoCircuitoVirtual** ou **retornaInterfacesAtivas**.

ANEXO B

Tabela 11 – Operações da fase de Instalação dos Agentes

Subfases	Operações	Dispositivos
1	(i) Instanciação do agente LEBR. (ii) Instanciação do agente LBR (iii) Criação da réplica do agente LBR. (iv) Serialização do agente LBR. (v) Encaminhamento IP origem de saída. (vi) Transmissão do agente LBR.	Comutador A
	(vii) Encaminhamento IP destino de entrada. (viii) Remontagem do agente LBR. (ix) Instalação do agente LBR.	Comutador B
2	(i) Criação da réplica do agente LBR. (ii) Serialização do agente LBR. (iii) Encaminhamento IP origem de saída. (iv) Transmissão do agente LBR.	Comutadores B e C
	(v) Encaminhamento IP destino Entrada. (vi) Remontagem do agente LBR. (vii) Instalação do agente LBR.	Comutadores C e D
3	(i) Obtenção do atraso do enlace *downstream*. (ii) Atualização da tabela TECM. (iii) cálculo do trecho crítico (iv) Instanciação do agente RABR no LER de Entrada. (v) Criação da réplica do agente RABR. (vi) Serialização do agente RABR. (vii) Encaminhamento IP origem de saída. (viii) Transmissão do agente RABR.	Comutador A
	(ix) Encaminhamento IP destino de entrada. (x) Remontagem do agente RABR. (xi) Instalação do agente RABR.	Comutador C
4	(i) Encaminhamento IP Intermediário do agente RABR. (ii) Transmissão do agente RABR.	Comutador B

Tabela 12 – Operações da fase de Monitoração do Circuito Virtual

Fase	Operações	Dispositivos
MCV	(i) Obtenção do atraso do enlace *downstream*. (ii) Montagem da mensagem de estado. (iii) Encaminhamento IP origem de saída. (iv) Transmissão da mensagem de estado.	Comutadores B, C e D
	(v) Encaminhamento IP destino de entrada. (vi) Tratamento da mensagem de estado.	

Tabela 13 – Operações da fase de Descoberta de Trechos Alternativos

Fase	Operações	Dispositivos
DTA	(i) Identificação das interfaces ativas. (ii) Criação da réplica do agente RABR. (iii) Serialização do agente RABR. (iv) Encaminhamento IP origem de saída. (v) Transmissão do agente RABR.	Comutadores A e B
	(vi) Encaminhamento IP destino de entrada. (vii) Remontagem do agente RABR. (viii) Instalação do agente RABR.	Comutadores B e C

Tabela 14 – Operações da fase de Monitoração dos Trechos Alternativos

Subfases	Operações	Dispositivos
1	(i) Obtenção do atraso e banda do enlace *downstream*.	Comutadores A, B e C
2	(i) Criação e associação dos rótulos. (ii) Montagem da mensagem de estado. (iii) Encaminhamento IP origem de saída. (iv) Transmissão da mensagem de estado. (v) Encaminhamento IP destino de entrada. (vi) Tratamento da mensagem de estado.	Comutadores B e C

Tabela 15 – Operações da fase de Mudança de Rota – Subteste 1

Subfases	Operações	Dispositivos
1	(i) Envio da mensagem de rerroteamento. (ii) Encaminhamento IP origem de saída. (iii) Transmissão da mensagem de rerroteamento.	Estação ServiMídia
	(iv) Encaminhamento destino de entrada. (v) Tratamento da mensagem de rerroteamento.	Comutador A
2	(i) Encaminhamento IP intermediário. (ii) transmissão da mensagem.	Comutadores B e C

Tabela 16 – Operações da fase de Mudança de Rota – Subteste 2

Subfases	Operações	Dispositivos
1	(i) montagem e envio da mensagem de chaveamento. (ii) encaminhamento IP origem de saída. (iii) transmissão da mensagem de chaveamento.	Comutador A
	(iv) encaminhamento IP destino de entrada. (v) tratamento da mensagem de chaveamento.	Comutador C
2	(i) encaminhamento IP intermediário. (ii) transmissão da mensagem de chaveamento.	Comutador B
3	(i) troca de rótulos.	Comutador C

Tabela 17 – Operações da fase de Mudança de Rota – Subteste 3

Subfases Adicionais	Operações	Dispositivos
A	(i) Envio da mensagem de geração e associação de rótulo. (ii) Encaminhamento IP origem de saída. (iii) Transmissão da mensagem de geração e associação de rótulo.	Comutador A
	(iv) Encaminhamento IP destino de Entrada. (v) Tratamento da mensagem de geração e associação de rótulo.	Comutador C
B	(i) Encaminhamento IP intermediário. (ii) Transmissão da mensagem de geração e associação de rótulo.	Comutador B
C	(i) Geração e associação de rótulo local. (ii) Montagem da mensagem de fechamento de caminho. (iii) Envio da mensagem de fechamento de caminho. (iv) Encaminhamento IP origem de saída. (v) Transmissão da mensagem de fechamento de caminho. (vi) Encaminhamento IP destino de Entrada.	Comutador B e C
	(vi) Tratamento da mensagem de fechamento de caminho.	Comutador A e B

Anexo C

Tabela 18 – Tempos de execução do método enviaAgente()

Medida	enviaAgente()		Medida	enviaAgente()	
	Agente LBR	Agente RABR		Agente LBR	Agente RABR
1ª	73,5	78,0	26ª	74,0	76,5
2ª	73,0	77,5	27ª	73,0	76,5
3ª	74,5	76,0	28ª	74,0	76,5
4ª	73,0	76,5	29ª	72,0	77,0
5ª	72,5	78,0	30ª	73,5	79,0
6ª	74,0	77,0	31ª	73,0	76,5
7ª	73,5	77,0	32ª	73,5	79,5
8ª	74,0	76,5	33ª	73,0	77,5
9ª	72,0	77,0	34ª	73,5	77,0
10ª	74,5	77,0	35ª	73,5	78,5
11ª	72,0	78,5	36ª	74,0	76,5
12ª	72,5	79,0	37ª	72,5	76,5
13ª	73,0	77,5	38ª	72,5	77,0
14ª	73,0	76,5	39ª	72,5	76,5
15ª	73,5	78,0	40ª	73,5	78,5
16ª	71,5	77,5	41ª	74,0	77,0
17ª	73,0	77,0	42ª	73,5	76,5
18ª	72,5	78,0	43ª	73,5	76,5
19ª	73,5	78,5	44ª	73,0	76,5
20ª	74,0	77,0	45ª	73,5	78,0
21ª	73,5	77,5	46ª	74,0	78,0
22ª	73,0	77,0	47ª	72,5	76,5
23ª	73,0	80,0	48ª	72,5	76,5
24ª	73,5	78,0	49ª	72,0	77,0
25ª	73,0	77,5	50ª	74,5	77,5

Tabela 19 – Tempos de execução dos métodos de sistema

Medida	ProxSalto CircuitoVirtual()	Retorna InterfacesAtivas()	Medida	ProxSalto CircuitoVirtual()	Retorna InterfacesAtivas()
1ª	60,5	67	26ª	61,5	66
2ª	60,5	68	27ª	61,5	65
3ª	60,5	66	28ª	59,5	67
4ª	62,0	67	29ª	60,5	68
5ª	59,5	67	30ª	61,0	67
6ª	60,0	67	31ª	61,5	65
7ª	59,5	67	32ª	61,0	68
8ª	60,5	67	33ª	60,5	68
9ª	59,5	68	34ª	62,0	67
10ª	61,0	68	35ª	62,5	69
11ª	60,5	69	36ª	61,0	66
12ª	61,0	67	37ª	62,0	68
13ª	60,0	68	38ª	60,5	69
14ª	61,0	68	39ª	62,5	68
15ª	61,5	67	40ª	64,0	70
16ª	59,5	66	41ª	63,5	66
17ª	59,0	67	42ª	62,5	66
18ª	60,0	68	43ª	62,5	66
19ª	61,0	68	44ª	63,5	66
20ª	61,5	66	45ª	65,5	67
21ª	60,0	67	46ª	61,5	66
22ª	61,0	67	47ª	61,5	68
23ª	60,0	66	48ª	63,0	67
24ª	61,0	66	49ª	63,0	66
25ª	60,0	67	50ª	64,0	67

Tabela 20 – Tempos da fase de Instalação dos Agentes

Medida	Subfase 1	Subfase 2	Subfase 3	Medida	Subfase 1	Subfase 2	Subfase 3
1ª	799	579	315	**26ª**	841	571	320
2ª	850	574	317	**27ª**	823	566	328
3ª	842	565	315	**28ª**	856	569	322
4ª	862	581	323	**29ª**	835	559	328
5ª	836	619	323	**30ª**	906	551	329
6ª	895	574	328	**31ª**	845	559	325
7ª	927	608	325	**32ª**	832	577	323
8ª	849	627	314	**33ª**	864	558	325
9ª	902	594	325	**34ª**	822	556	323
10ª	879	610	321	**35ª**	807	561	322
11ª	899	587	323	**36ª**	828	551	324
12ª	941	587	322	**37ª**	808	544	316
13ª	881	581	324	**38ª**	807	572	320
14ª	830	605	330	**39ª**	844	578	322
15ª	698	584	322	**40ª**	802	577	325
16ª	862	594	320	**41ª**	795	579	319
17ª	892	587	322	**42ª**	811	563	314
18ª	909	591	321	**43ª**	812	559	316
19ª	860	594	325	**44ª**	826	572	318
20ª	875	632	326	**45ª**	806	577	317
21ª	868	598	313	**46ª**	844	558	312
22ª	827	590	326	**47ª**	806	569	315
23ª	888	586	325	**48ª**	826	612	317
24ª	862	585	319	**49ª**	842	565	315
25ª	831	581	324	**50ª**	850	574	317

Tabela 21 – Tempos da fase de Monitoração do Circuito Virtual

Medida	FASE MCV	Medida	FASE MCV
1ª	34,0	**26ª**	34,0
2ª	34,0	**27ª**	32,5
3ª	34,0	**28ª**	34,0
4ª	34,0	**29ª**	33,0
5ª	34,0	**30ª**	33,5
6ª	34,0	**31ª**	33,0
7ª	34,0	**32ª**	33,0
8ª	33,5	**33ª**	33,0
9ª	34,0	**34ª**	33,5
10ª	34,0	**35ª**	33,0
11ª	33,5	**36ª**	33,0
12ª	34,0	**37ª**	34,0
13ª	33,5	**38ª**	33,5
14ª	33,5	**39ª**	33,0
15ª	33,5	**40ª**	32,5
16ª	33,5	**41ª**	33,0
17ª	33,5	**42ª**	33,5
18ª	33,0	**43ª**	33,0
19ª	33,5	**44ª**	33,0
20ª	33,5	**45ª**	32,5
21ª	33,0	**46ª**	33,0
22ª	33,5	**47ª**	33,0
23ª	33,5	**48ª**	35,0
24ª	33,5	**49ª**	33,0
25ª	34,0	**50ª**	33,5

Tabela 22 – Tempos da fase de Descoberta de Trechos Alternativos

Medida	FASE DTA	Medida	FASE DTA
1ª	366	**26ª**	318
2ª	345	**27ª**	344
3ª	353	**28ª**	312
4ª	334	**29ª**	309
5ª	340	**30ª**	315
6ª	320	**31ª**	314
7ª	317	**32ª**	311
8ª	332	**33ª**	303
9ª	345	**34ª**	303
10ª	342	**35ª**	311
11ª	331	**36ª**	306
12ª	316	**37ª**	316
13ª	324	**38ª**	338
14ª	320	**39ª**	355
15ª	316	**40ª**	344
16ª	324	**41ª**	322
17ª	349	**42ª**	322
18ª	325	**43ª**	334
19ª	320	**44ª**	361
20ª	332	**45ª**	327
21ª	325	**46ª**	325
22ª	319	**47ª**	330
23ª	311	**48ª**	318
24ª	366	**49ª**	309
25ª	345	**50ª**	328

Tabela 23 – Tempos da fase de Monitoração de Trechos Alternativos

Medida	Subfase 1	Subfase 2	Medida	SubFase 1	SubFase 2
1ª	27,5	72,0	**26ª**	27,5	74,5
2ª	27,5	72,0	**27ª**	27,5	75,0
3ª	28,5	72,0	**28ª**	27,5	76,0
4ª	28,0	72,5	**29ª**	28,0	75,0
5ª	28,5	72,0	**30ª**	27,5	74,5
6ª	27,5	72,0	**31ª**	28,0	76,0
7ª	28,5	74,0	**32ª**	28,0	75,5
8ª	28,0	73,0	**33ª**	28,0	76,0
9ª	28,0	73,0	**34ª**	27,5	76,0
10ª	28,0	72,0	**35ª**	28,0	76,0
11ª	28,0	73,0	**36ª**	27,5	76,5
12ª	28,5	72,5	**37ª**	27,5	76,0
13ª	27,5	73,0	**38ª**	28,0	75,5
14ª	28,0	73,5	**39ª**	28,0	75,5
15ª	28,0	73,0	**40ª**	28,5	76,5
16ª	27,0	73,5	**41ª**	28,0	76,5
17ª	27,5	74,0	**42ª**	28,0	76,0
18ª	28,5	74,0	**43ª**	27,5	76,5
19ª	27,5	74,5	**44ª**	27,5	76,5
20ª	27,5	74,0	**45ª**	28,0	77,0
21ª	28,0	75,5	**46ª**	28,0	77,0
22ª	27,5	74,5	**47ª**	29,0	78,0
23ª	27,5	74,5	**48ª**	28,5	77,5
24ª	28,0	74,0	**49ª**	27,5	77,0
25ª	27,5	74,5	**50ª**	28,5	77,5

Tabela 24 – Tempos da fase de Mudança de Rota – Subteste 1

Medida	Subfase 1	Medida	Subfase 1
1ª	4	26ª	4
2ª	4	27ª	4
3ª	4	28ª	4
4ª	4	29ª	4
5ª	4	30ª	4
6ª	4	31ª	4
7ª	4	32ª	4
8ª	4	33ª	4
9ª	4	34ª	4
10ª	4	35ª	4
11ª	4	36ª	4
12ª	4	37ª	4
13ª	4	38ª	4
14ª	4	39ª	4
15ª	4	40ª	4
16ª	4	41ª	4
17ª	4	42ª	4
18ª	4	43ª	4
19ª	4	44ª	4
20ª	4	45ª	4
21ª	4	46ª	4
22ª	4	47ª	4
23ª	4	48ª	4
24ª	4	49ª	4
25ª	4	50ª	4

Tabela 25 – Tempos da fase de Mudança de Rota – Subteste 2

Medida	Subfase 1	Subfase 3	Medida	Subfase 1	Subfase 3
1ª	4	94,0	26ª	4	87,0
2ª	4	95,5	27ª	4	89,0
3ª	4	93,0	28ª	4	87,5
4ª	4	93,0	29ª	4	87,0
5ª	4	89,5	30ª	4	87,5
6ª	4	89,0	31ª	4	87,5
7ª	4	88,5	32ª	4	88,0
8ª	4	89,0	33ª	4	89,5
9ª	4	88,5	34ª	4	88,5
10ª	4	88,0	35ª	4	89,0
11ª	4	88,0	36ª	4	88,5
12ª	4	88,0	37ª	4	88,0
13ª	4	87,0	38ª	4	89,5
14ª	4	87,5	39ª	4	91,0
15ª	4	87,0	40ª	4	91,5
16ª	4	89,0	41ª	4	90,0
17ª	4	88,5	42ª	4	95,5
18ª	4	87,0	43ª	4	93,5
19ª	4	88,0	44ª	4	93,5
20ª	4	87,5	45ª	4	94,0
21ª	4	87,0	46ª	4	93,5
22ª	4	86,5	47ª	4	93,0
23ª	4	94,0	48ª	4	97,0
24ª	4	95,5	49ª	4	93,5
25ª	4	93,0	50ª	4	93,0

Tabela 26 – Tempos da fase de Mudança de Rota – Subteste 3

Medida	Subfase A	Subfase C	Medida	Subfase A	Subfase C
1ª	4,5	117	**26ª**	4,5	123
2ª	4,5	118	**27ª**	4,5	123
3ª	4,5	117	**28ª**	4,5	126
4ª	4,5	117	**29ª**	4,5	126
5ª	4,5	118	**30ª**	4,5	123
6ª	4,5	118	**31ª**	4,5	125
7ª	4,5	117	**32ª**	4,5	123
8ª	4,5	118	**33ª**	5,0	123
9ª	4,5	118	**34ª**	4,5	124
10ª	4,5	117	**35ª**	4,5	125
11ª	4,5	117	**36ª**	4,5	125
12ª	4,5	116	**37ª**	4,5	121
13ª	4,5	119	**38ª**	4,5	127
14ª	4,5	120	**39ª**	4,5	124
15ª	4,5	118	**40ª**	4,5	123
16ª	4,5	118	**41ª**	4,5	125
17ª	4,5	122	**42ª**	4,5	125
18ª	4,5	120	**43ª**	4,5	128
19ª	4,5	121	**44ª**	4,5	127
20ª	4,5	121	**45ª**	4,5	127
21ª	4,5	120	**46ª**	4,5	125
22ª	4,5	119	**47ª**	4,5	125
23ª	4,5	117	**48ª**	4,5	125
24ª	4,5	118	**49ª**	4,5	124
25ª	4,5	117	**50ª**	4,5	127