SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 23.09.2002

Assinatura:

Projeto de um serviço de redes ativas para QoS

Ricardo José Martines Ribeiro

Orientador: *Prof. Dr. Edson dos Santos Moreira*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Setembro/2002

A Comissão Julgadora:

Prof. Dr. Edson dos Santos Moreira

A stylized handwritten signature in black ink, consisting of a large, sweeping 'M' followed by a horizontal line.

Prof. Dr. Edmundo Roberto Mauro Madeira

A handwritten signature in black ink that reads 'Edmundo R. M. Madeira' in a cursive script.

Profa. Dra. Regina Borges de Araujo

A handwritten signature in black ink that reads 'Regina B. de Araujo' in a cursive script.

Dedicatória

Dedico este trabalho aos meus pais, Adão e Vera pela paciência demonstrada, apoio nos momentos difíceis e alegria compartilhada nos momentos felizes. Sem eles, nada disso seria possível, sequer a minha própria existência.

Agradecimentos

Inicialmente a Deus, pela vida, graças concedidas e força obtida.

Ao meu orientador Edson dos Santos Moreira, pelo trabalho realizado.

Ao amigo Rudinei, pela ajuda prestada durante todo este trabalho.

Aos amigos e colegas do laboratório Intermídia: Milagres, Dalton, Gustavo, Martins, Brandão.

Ao ICMC/USP, pela oportunidade de realizar este trabalho.

A Radiumsystems.com, pelo apoio oferecido.

A minha namorada Roberta, pela compreensão demonstrada nos momentos de mau humor e pelas horas em que aceitou ser trocada por este trabalho.

Aos atuais colegas e ex-companheiros da Radiumsystems.com, pelo ótimo ambiente de trabalho.

Ao amigo de trabalho Pedro, pelo apoio oferecido sempre que foi necessário.

Aos amigos de turma de Ciências da Computação 96, especialmente DG, Marquinhos e Percival pela amizade demonstrada.

A todos os amigos de São Carlos, pelo companheirismo compartilhado.

A SSVP (Sociedade de São Vicente de Paulo) que possibilitou-me sempre manter os pés no chão e perceber que o nosso problema sempre é pequeno quando comparado às dificuldades enfrentadas por tantos cidadãos e filhos de Deus como cada um de nós.

Aos amigos da SSVP, especialmente o Fernandinho, pela ajuda na impressão desta dissertação, por partilharem comigo momentos de dificuldades, compaixão, caridade, alegria e companheirismo.

E a todos que contribuíram direta ou indiretamente para a realização deste trabalho, ou que simplesmente conviveram comigo nestes últimos anos...

Índice

1	Introdução	1
1.1	Contexto	1
1.2	Motivação	3
1.3	Objetivo	3
1.4	Estrutura	3
2	Televisão Interativa	4
2.1	Considerações Iniciais	4
2.2	Vídeo sob Demanda	5
2.3	Padrões para a TV Interativa	6
2.3.1	ATVEF	7
2.3.2	TV Anytime Forum	8
2.3.3	W3C	9
3	Qualidade de Serviço	10
3.1	Introdução	10
3.2	Aspectos Gerenciais	12
3.3	Tópicos relacionadas à QoS	13
3.4	Mapeamento de QoS	14
3.4.1	Camada de QoS do Usuário	14
3.4.2	Camada de QoS da Aplicação	15
3.4.3	Camada de QoS do Sistema	15
3.4.4	Camada de QoS da Rede	15
3.4.5	Transformação de Parâmetros de QoS	16
3.5	Tecnologias que Implementam QoS	17
3.5.1	A tecnologia ATM	18
3.5.1.1	Conceitos gerais	18
3.5.1.2	Parâmetros de QoS	18
3.5.2	Serviços Integrados	23
3.5.2.1	Conceitos gerais	23
3.5.2.2	RSVP	26
4	Redes Ativas	30
4.1	Introdução	30
4.1.1	Duas Escolas de Pensamento	31
4.1.2	Conceitos Básicos	33
4.1.2.1	Interfaces de Programação	34
4.1.2.2	Nós Ativos	35
4.1.2.3	Pacote Ativo	35
4.1.2.4	Redes Programáveis x Redes Ativas	35
4.1.2.5	Redes Ativas x Agentes Móveis	36
4.1.2.6	ANEP - <i>Active Network Encapsulation Protocol</i>	36
4.1.3	Localização do Código	37
4.1.3.1	Código apenas nos pacotes	38

4.1.3.2	Código apenas nos nós ativos	38
4.1.3.3	Código nos pacotes e nós da rede	39
4.2	Mecanismos de Funcionamento	39
4.2.1	Conceitos Gerais	39
4.2.2	Fluxo nos nós ativos	41
4.3	Implementações de Ambiente de Execução	42
4.3.1	SwitchWare	43
4.3.2	Smartpackets	43
4.3.3	Plataforma Joust e o Software Líquido	44
4.3.4	Netscript	44
4.4	Segurança em Redes Ativas	45
4.4.1	SANE (Security Active Network Environment)	47
4.5	Experiências com uma Implementação de Redes Ativas	48
5	Serviço Ativo para Provimento de Parâmetros de QoS	49
5.1	Introdução	49
5.2	Parâmetros de QoS	49
5.3	Parâmetros de QoS Estudados	50
5.4	Especificação do Serviço	52
5.4.1	Objetivos	52
5.4.2	Parâmetros Disponibilizados	53
5.4.2.1	Regras de Composição dos Parâmetros	55
5.4.3	A Interface Administrativa do Serviço	56
5.4.3.1	Administração de Parâmetros de Ajuste de Medida	56
5.4.3.2	Administração dos Parâmetros de Aplicação	56
5.4.3.3	Administração de Parâmetros de Usuário	57
5.4.3.4	Análise da QoS Oferecida pela Rede	58
5.4.4	Aplicações	59
5.5	Projeto do serviço	59
5.5.1	Escolha da Rede Ativa	59
5.5.2	Parâmetros de ajuste	61
5.5.3	Descrição do Ambiente	62
5.5.4	Coleta dos Parâmetros Locais	62
5.5.4.1	Parâmetros Envolvidos	63
5.5.4.2	Informações Básicas para Cálculo dos Parâmetros	64
5.5.4.3	Cálculo dos Parâmetros de QoS de Rede	65
5.5.4.4	Cálculo dos Parâmetros QoS de Sistema	70
5.5.4.5	Armazenamento dos Dados Locais	71
5.5.5	Coleta dos Parâmetros de um Caminho	72
5.5.5.1	A Cápsula Ativa	72
5.5.5.2	Estratégia de Funcionamento	73
5.5.5.3	Armazenamento dos Dados de um Caminho	73
5.5.5.4	Forma de Utilização dos Parâmetros	75
5.5.6	Interface Administrativa	75
5.5.6.1	Administração dos Parâmetros de Ajuste	76
5.5.6.2	Administração dos Parâmetros de QoS de Aplicação e Usuário	77
5.5.6.3	Análise da QoS oferecida pela rede	84

5.6	Modificações Necessárias ao ANTS	86
6	Conclusões	87
6.1	Considerações Iniciais	87
6.2	Resultados Alcançados	87
6.3	Contribuições	87
6.4	Limitações Observadas	88
6.5	Trabalhos Futuros	88
6.6	Considerações Finais	89
	Referências Bibliográficas	90
	Apêndice: ANTS - Active Network Transport System	98
A.1	Introdução	98
A.2	Instalação e Configuração	98
A.3	Estrutura de Diretórios	100
A.4	A Arquitetura das Aplicações ANTS	100
A.4.1	Ping.start	103
A.4.2	Nodesetup.sh	103
A.4.2.1	DefaultEnv	107
A.4.2.2	Antsvm	107
A.4.3	ConfigurationManager	108
A.4.3.1	Ping.config	109
A.4.3.2	Ping.Routes	110
A.4.4	Exemplos de Aplicações	110
A.5	Executando ANTS em Múltiplas Máquinas	111
A.6	Segurança e ANTS 2.0.0	112
A.7	Componentes de um Serviço ANTS	114

Índice de Figuras

Figura 2.1 - Componentes de Um Sistema de Vídeo Sob Demanda	6
Figura 3.1 - Necessidade de QoS x Aplicação [San99]	10
Figura 3.2 - Questões de Qualidade de Serviço [Cha00]	13
Figura 3.3 - Um modelo conceitual de QoS [Nah95]	14
Figura 3.4 - Tolerância na variação do delay entre células - CDVT [Rob94]	20
Figura 3.5 - Função densidade de probabilidade do tempo de chegada de células [Roc00]	22
Figura 3.6 - Componentes Básicos da Arquitetura Intserv [Mag97]	25
Figura 4.1 - Exemplo de funcionamento de Redes Ativas [Bec00]	31
Figura 4.2 - Cabeçalho ANEP [Ane97]	37
Figura 4.3 - Componentes da arquitetura para redes ativas [Tei00]	40
Figura 4.4 - Fluxo de pacotes através de uma rede ativa [Gon00]	42
Figura 5.1 - Projeto da Tela inicial da Interface Administrativa	76
Figura 5.2 - Projeto da Interface de Administração dos Parâmetros de Ajuste	76
Figura 5.3 - Projeto da Interface de Administração dos Parâmetros de Aplicação e Usuário	78
Figura 5.4 - Projeto da Interface de Administração dos Parâmetros de Aplicação	80
Figura 5.5 - Projeto da Interface de Administração dos Parâmetros de Usuário	82
Figura 5.6 - Projeto da Interface de Análise da QoS Oferecida pela Rede	84
Figura A.1 - Estrutura de Diretórios - ANTS [ANT02]	100
Figura A.2 - Topologia da Rede Virtual para Aplicação Ping - [ANT02]	101
Figura A.3 - Processo de Funcionamento da Aplicação Ping - [ANT02]	102
Figura A.4 - Topologia da Rede Ativa Virtual para Três Máquinas [ANTS 01]	111

Índice de Tabelas

<i>Tabela 3.1 - Características e requisitos das diversas mídias.....</i>	<i>12</i>
<i>Tabela 3.2 - Exemplos de mapeamento usuário/aplicação [Taw93]</i>	<i>17</i>
<i>Tabela 3.3 - Exemplos de mapeamento aplicação/sistema(rede) [Nah94]</i>	<i>17</i>
<i>Tabela 5.1 - Parâmetros de QoS a considerar pelas aplicações [Arg98].....</i>	<i>50</i>
<i>Tabela 5.2 - Parâmetros de QoS estudados</i>	<i>51</i>
<i>Tabela 5.3 - Parâmetros de QoS Disponibilizados.....</i>	<i>53</i>
<i>Tabela 5.4 - Regras de Composição dos Parâmetros de QoS Disponibilizados</i>	<i>55</i>
<i>Tabela 5.5 - Parâmetros de QoS e Funções de Transformação Fictícios I.....</i>	<i>58</i>
<i>Tabela 5.6 - Parâmetros de QoS e Funções de Transformação fictícios II</i>	<i>58</i>
<i>Tabela 5.7 - Parâmetros de Sistema e de Rede Disponibilizados.....</i>	<i>62</i>
<i>Tabela A.1 - Exemplo do script .start em 3 máquinas</i>	<i>112</i>

Resumo

Qualidade de Serviço (*Quality of Service - QoS*) é um tema bastante explorado na literatura atualmente. Contudo, as potenciais vantagens da tecnologia de redes ativas, como forma de facilitar a obtenção e o gerenciamento de parâmetro de QoS, é pouco estudada. Então, auxiliando o preenchimento dessa lacuna, este trabalho visa a especificação e o projeto de um serviço ativo para o provimento de parâmetros de QoS. O serviço permite ao administrador da rede criar e refinar o cálculo de parâmetros de QoS por meio de uma interface Web amigável. Os parâmetros são disponibilizados em locais e formatos pré-definidos, possibilitando assim que outros serviços possam acessar os dados. O Serviço projetado neste trabalho deve servir como base para outros serviços, por exemplo, o serviço de reserva de recursos, criando-se subsídios para o desenvolvimento de estratégias, baseadas nos parâmetros, de controle da rede e de negociação com aplicações.

Abstract

Quality of Service (QoS) is a well-explored issue in the network area. However, the potential advantages of the active networks technology as a way to ease the QoS parameters management need more studies. Helping to fulfill this gap this work aims to specify and to design an active service that provides QoS parameters. The service allows the network administrator to build and to refine QoS parameters through a Web interface. The parameters are made available at pre-defined locals and in pre-defined formats, making possible the data access by others services. The service designed in this work can be used by others services like the resource reservation service, creating the necessary bases to the development of QoS parameters based strategies to control the network and the negotiation with applications.

1 Introdução

1.1 Contexto

Uma das principais tendências que se observam atualmente na Internet é uma demanda crescente por uma classe de aplicações, conhecidas como aplicações multimídia. Esta classe de aplicações distingue-se das outras pela necessidade de transmissão de mídias contínuas, como vídeo ou áudio. Nessas condições exige-se uma certa Qualidade de Serviço (*Quality of Service* - QoS) da infra-estrutura de rede para que as aplicações possam funcionar adequadamente.

Provisão de QoS na Internet é uma das áreas que vêm recebendo muito incentivo dentre os principais organismos de pesquisa e padronização da Internet. Na sua maneira de funcionamento atual, a Internet não oferece nenhum suporte no que se refere a compartilhamento e orquestração de recursos. Para solucionar esses problemas, é necessária uma mudança profunda na maneira de funcionamento da Internet, por exemplo, definindo novas árvores de protocolos ou paradigmas de arquitetura de rede para utilização por aplicações com esse tipo de necessidade.

Além de serviços que demandam QoS, é importante lembrar que o serviço de melhor esforço é satisfatório na maioria dos casos. Tendo em vista esse cenário, a integração de diferentes tipos de tráfegos, em uma rede comutada de pacotes de forma escalável e flexível, é uma solução muito almejada. O que dificulta essa integração é que aplicações multimídia têm necessidades diferentes das aplicações convencionais [Kuo98]. O serviço de melhor esforço não é adequado para tráfego sensível a atrasos, como o vídeo, já que podem ocorrer eventos como perdas de pacotes e variações no atraso. Esses tipos de distorções podem diminuir severamente a qualidade da transmissão de tempo real, podendo até mesmo inviabilizá-la [Bla98] [Bra94].

A fim de se atender às necessidades de uma aplicação de tempo real como vídeo sob demanda (*Video on Demand* - VoD), vídeo conferência, TV Interativa (*Interactive TV* - ITV), entre outros, faz-se necessária a inserção de novas características à infra-estrutura para oferecer QoS [Kuo98]. Em linhas gerais, isso pode ser obtido através da inclusão de itens como: controle de tráfego, alocação e gerência de recursos, policiamento, controle de admissão, entre outros [Bla98] [Bra94] [Bra97].

Roteadores de protocolo IP se multiplicaram nas redes de comunicação de dados existentes hoje com o serviço de melhor esforço. Sendo assim, as soluções devem atender às novas demandas de tráfego mantendo as características originais e adicionando novos recursos necessários ao tráfego de dados de mídia contínua [Bla98] [Bra94] [Xia99].

Diante desses pontos, a ITV surge como um exemplo de serviço que necessita integrar Internet, vídeo digital interativo e QoS. Essa aplicação demanda certas condições especiais sobretudo da arquitetura de rede. A rede deve ser capaz de suportar interatividade do usuário em tempo real, lidar com a questão da qualidade da transmissão e, ainda, ser escalável para atender às demandas. Nesse sentido, um grupo do laboratório Intermídia ICMC /USP vem trabalhando desde o ano 2000 em um protótipo de ITV. Esse protótipo procura trabalhar diversas questões, desde a padronização XML (*eXtensible Markup Language*) do conteúdo de ITV a ser transportado, até a infra-estrutura de rede e software necessária para a implementação.

Este trabalho utiliza a tecnologia de redes ativas (*Active Network* - AN) como infra-estrutura para o protótipo de ITV. Essa escolha baseia-se, entre outros pontos, na possibilidade de coexistência de fluxos com garantia de QoS e outros em que o melhor esforço é suficiente.

Trabalhos que procuram pesquisar QoS juntamente com redes ativas não são muito encontradas atualmente. A Universidade Federal do Rio de Janeiro (UFRJ), nos últimos anos tem publicado alguns trabalhos nesse contexto. Dentre eles, pode-se destacar um serviço para distribuição de vídeo multiponto [Gon00], onde o usuário pode escolher a qualidade do vídeo que deseja receber. A rede ativa, implementada com o pacote ANTS, é responsável por fazer chegar o vídeo nos respectivos destinos e com a qualidade acordada, sempre que possível. Outro trabalho relacionado, da mesma universidade, é uma arquitetura de roteamento com QoS baseada em redes ativas [Jun97]. Nesse trabalho o foco está no gerenciamento da rede para a garantia de QoS.

Existem outros trabalhos relacionados à gerência de redes e rede ativas. Tais trabalhos não focalizam QoS, porém contribuem para melhorar a qualidade do serviço a medida que conseguem otimizar a utilização dos recursos da rede. Na UFRJ foi desenvolvida uma proposta para o emprego da tecnologia de redes ativas no gerenciamento de redes [Tei00]. Esse trabalho projetou um cenário onde o emprego da tecnologia de redes ativas juntamente com JAVA e CORBA poderia se conseguir uma gerência da rede mais eficiente. Na Unicamp foi desenvolvido um sistema para gerência de serviços em redes ativas virtuais [Ver02]. Esse sistema utiliza redes ativas em conjunto com outras tecnologias como agentes móveis. Tal

trabalho coleta e analisa dados além de permitir tomar determinadas ações, visando sempre a melhora do desempenho global da rede. O pacote ANTS também é utilizado por esse trabalho para implementar o componente referente a rede ativa.

1.2 Motivação

A qualidade de serviço é uma necessidade cada vez mais recorrente para certos tipos de aplicação, como TV Interativa, sobretudo com a massificação do uso da Internet e o aumento da utilização de mídias como vídeo e áudio. Redes IP convencionais que oferecem qualidade de serviço não conseguem um rápido diagnóstico de problemas com elementos pertencentes a essa rede; dessa forma fica comprometida a QoS oferecida às aplicações.

1.3 Objetivo

Utilizar a tecnologia de redes ativas para construir uma infra-estrutura de rede capaz de propiciar QoS aos usuários finais é o objetivo do projeto desse serviço. Para isso, este trabalho deve compor uma solução para a arquitetura de rede IP convencional, utilizando uma implementação de AN, que dará suporte ao protótipo de ITV. Essa arquitetura deve ser projetada para oferecer QoS aos usuários finais e, também, permitir o diagnóstico de problemas com elementos da rede. Dessa forma, é viabilizada a adoção de políticas que minimizem o impacto na QoS oferecida.

1.4 Estrutura

Este capítulo apresentou o contexto no qual este trabalho se insere: a motivação para seu desenvolvimento, objetivos propostos e resultados alcançados. O capítulo 2 descreve uma visão geral da tecnologia de ITV e os principais padrões trabalhados nos últimos anos. O capítulo 3 apresenta os principais conceitos de QoS e descreve algumas tecnologias que implementam QoS, em aspectos relevantes ao projeto de uma infra-estrutura de rede para ITV. O capítulo 4 descreve uma visão geral da tecnologia de AN e apresenta algumas implementações já existentes. O capítulo 5 apresenta a especificação e o projeto de um serviço utilizando AN para ser utilizado na infra-estrutura da ITV. Por fim, o capítulo 6 apresenta as conclusões, resultados, contribuições, limitações e os trabalhos futuros.

2 Televisão Interativa

2.1 Considerações Iniciais

Serviços de ITV requerem alta velocidade na transmissão de dados audiovisuais, assim como um certo nível de QoS. Oferecer tais requisitos torna-se difícil sobre uma rede de baixa velocidade como a Internet, devido aos serviços do tipo melhor esforço, sem garantias de atraso, vazão ou perda de pacotes.

A televisão conhecida atualmente possui como características:

- Linearidade;
- Transmissão por difusão;
- Veículo de massa;
- Unidirecionalidade;
- Escolha de programação limitada;
- Pouca personalização.

A ITV terá características distintas das mencionadas anteriormente sobre a TV analógica, sobretudo no que se refere à interatividade. Rever cenas, avançar e retroceder filmes, saber mais informações sobre determinado produto ou mesmo comprá-lo sem sair de casa. Além disso, com o advento da WWW, o usuário também deseja ter a possibilidade de navegar na Internet e ler seus e-mails, via TV.

Pretende-se integrar a TV à Internet e à TV Digital, através de *Set-Top Boxes* (STBs) e *Personal Digital Recorders* (PDRs). Uma das primeiras empresas a vislumbrar esse mercado foi a WebTV. Em 1995, ela desenvolveu um STB capaz de agregar à TV a função de navegação na Web. Com o sucesso, em 1997 a Microsoft comprou-a e, ao longo do tempo, tem feito contínuas atualizações de software e de hardware. Atualmente, diversas empresas brigam por esse mercado [Far00].

Na seção 2.2 deste capítulo, são discutidos os conceitos de VoD, que é uma das técnicas básicas empregadas em transmissão de conteúdo de ITV. Na seção 2.3, são apresentados os principais organismos empenhados em padronizar as tecnologias envolvidas bem como seus padrões.

2.2 Vídeo sob Demanda

Um importante serviço de ITV é o VoD. Ele oferece um serviço de aluguel de vídeo eletrônico através da rede pelo qual o usuário pode selecionar programas de um servidor de vídeo e assistir em sua casa. O serviço de VoD oferece funções estilo videocassete, como retrocesso rápido (*fast-forward*) e avanço rápido (*rewind*), através das quais o usuário pode interagir com o servidor.

O VoD pode ser considerado a evolução virtual das locadoras de vídeo. Ele possui a mesma função de “entregar” ao usuário o filme desejado: uma diferença para a videolocadora está na necessidade do transporte do meio de armazenamento físico do filme, a fita de vídeo. Usando-se VoD, o filme é transportado no formato digital através de redes de computadores.

Existem diversos modos de implementar o VoD, porém um ponto imprescindível que sempre deve-se levar em conta é a largura de banda e os demais requisitos de QoS necessários para sua implementação. Esse ponto é especialmente relevante para este trabalho devido ao fato de ser um fator que motivou os estudos realizados.

Por exemplo, se for permitido ao usuário interromper, retroceder e avançar o filme, pode ser-lhe necessário ter uma conexão exclusiva, gerando assim maior necessidade de banda. Porém, se esses recursos não forem permitidos, pode-se utilizar outro tipo de implementação, chamado de similar ao vídeo sob demanda (*near video on demand* - NVoD), no qual um mesmo vídeo é iniciado em canais diferentes a cada t minutos. Nesse caso, o usuário pode retroceder ou avançar sempre de t em t minutos. A banda utilizada nesse caso é menor, uma vez que uma mesma conexão é utilizada por múltiplos usuários.

Existem pesquisas em sincronização de vídeos para a otimização do uso da banda de transmissão de dados [Agg96a] [Agg96b] [Fon97] [Cri99]. Tais pesquisas enumeram algumas possibilidades existentes para evitar a necessidade de criação de uma conexão por usuário, dentre outros problemas relacionados. Maiores detalhes podem ser obtidos nos próprios trabalhos.

Um sistema de VoD contém muitos componentes [Lia97], os quais são mostrados na Figura 2.1: o servidor de vídeo, a rede de transporte, uma unidade terminal subscrita e um serviço de *gateway*.

Figura 2.1 - Componentes de Um Sistema de Vídeo Sob Demanda

O servidor de vídeo armazena um grande número de vídeos digitalizados e serve um número considerável de requisições de vídeo simultâneas para o mesmo ou para diferentes usuários. As funções básicas de um servidor de vídeo incluem manipulação de requisições, acesso randômico a dados, por exemplo, o próprio vídeo, interações de usuários, além de controle de admissão de requisições e garantias de QoS.

A rede de transporte transmite os programas de vídeo do servidor para um ou mais usuários. Ela deve possuir velocidade de transferência e demais requisitos de QoS suficientes para satisfazer as restrições de tempo real do tráfego de vídeo. As funções da unidade terminal subscrita, ou STBs, incluem o recebimento dos fluxos de vídeo de entrada, demodulação, demultiplexação e decodificação de sinais, realizando a conversão do sinal necessária, por exemplo, a transformação de um sinal digital em analógico para a reprodução (*playback*) no monitor de TV.

O serviço de *gateway* pode ser integrado com um modo de acesso ou pode ser um elemento separado na rede. As principais funções realizadas por um serviço de *gateway* incluem: serviço de diretório; mapeamento da identificação do serviço para locação correspondente; controle, coordenação e sinalização do estabelecimento de uma sessão multimídia, mantendo-a e desconectando-a conforme a necessidade; administração do sistema, incluindo administração de falhas, configuração, administração de recursos e administração de desempenho.

2.3 Padrões para a TV Interativa

A padronização de produtos na área de TV Interativa é uma necessidade devido ao grande número de empresas disputando o mercado e à sua estratificação. Existem diversos órgãos e empresas buscando essa padronização, porém aqui serão apresentados somente os

três mais importantes, e que, graças aos seus históricos de padronização, merecem ser estudados. São eles:

- ATVEF – *Advanced Television Enhancement Forum* – Liderado por empresas como Microsoft, Liberate, NBC e Warner Bros.
- TV-Anytime Forum – Intimamente ligado ao MPEG (*Motion Pictures Experts Group*) e a grandes empresas japonesas como Sony, Toshiba e Hitashi, americanas como Microsoft e IBM, e européias como Nokia e Philips.
- W3C – *World Wide Web Committee* - Conhecido pela especificação de tecnologias relacionadas à WWW. Diferentemente dos anteriores, pretende somente especificar uma forma de integrar a *Web* e a TV Interativa [Far00].

2.3.1 ATVEF

O *Advanced Television Enhancement Forum* (ATVEF) é um grupo formado por empresas que atuam nos setores de programação e transmissão de TV, eletrônicos e empresas de informática. Seu objetivo é criar uma televisão interativa independente de plataforma.

Essa aliança já criou uma especificação para programas de TV Interativa baseados em HTML (*HyperText Markup Language*) intitulada *ATVEF Specification for Interactive Television 1.1* [Atv00]. As principais características desse padrão são:

- Uso de HTML 4.0 [HTM02];
- *Cascading Style Sheets, Level1* (CSS1) – Permite a aplicação de estilos a documentos HTML [CSS02];
- *Document Object Model* (DOM 0) – É uma estrutura de representação hierárquica que permite a programas e *scripts* o acesso dinâmico e a atualização de conteúdo, estrutura e estilo dos documentos [DOM02];
- ECMAScript – Uma linguagem de programação do tipo *script* independente de plataforma [ECM02];
- Suporte ao Javascript 1.1, equivalente ao uso do ECMAScript e do DOM 0.

A integração da TV com as páginas WWW é feita através das URIs “tv:” [Vic00] [Fin00] e “lid:” [Bla00]. A URI “tv:” tem como função inserir conteúdo de televisão em uma página HTML; já a URI “lid:”, *local identifier*, é um identificador local de um arquivo. O ATVEF especifica que os STBs devem possuir no mínimo 1 MB para cache simultâneo, e que é necessário especificar, antes da transmissão, a quantidade de memória cache para cada programa (essa quantidade é o pico e não o total) [Far00].

O modelo do ATVEF é baseado no conceito de *triggers* (gatilhos) [Atv00] - as páginas HTML da TV Interativa devem possuir um objeto capaz de recebê-los e interpretá-los. Os *triggers* são eventos em tempo real transmitidos para programas da TV Interativa. As implementações dos receptores permitirão a criação de diferentes políticas para o recebimento dos *triggers*, por exemplo, a utilização do aviso de sua chegada para notificar os usuários a disponibilidade de conteúdo extra. Eles devem sempre possuir uma *Universal Resource Locator* (URL) e podem possuir opcionalmente nome, validade e um *script*.

O ATVEF definiu dois tipos de transportes:

- A - para transporte dos *triggers*;
- B - destinado ao transporte de dados, o vídeo propriamente dito.

O primeiro tipo deve ser implementado de acordo com cada padrão de televisão - há documentos mostrando como o ATVEF deve ser implementado tanto para o padrão *National Television Systems Committee* (NTSC) quanto o padrão *Phase Alternate Line* (PAL). Tem-se trabalhado para implementar para o padrão *Digital Video Broadcast* (DVB) e o padrão *Advanced Television Systems Committee* (ATSC).

O segundo tipo de transporte, ao contrário do A, faz que a TV digital seja independente de conexões Internet auxiliares, por exemplo, o telefone, já que ele é feito para receber dados também.

O ATVEF desenvolveu um protocolo especial para cuidar do *broadcast* dos dados, seja *trigger* ou vídeo, quando necessário, o *Unidirectional Hypertext Transfer Protocol* (UHTTP) que é uma versão simplificada do *Hypertext Transfer Protocol* (HTTP), desenvolvida para ser eficiente na transmissão de dados *broadcast*.

O ATVEF tem cerca de 130 empresas afiliadas. Muitos dos atuais projetos de TV Interativa estão sendo feitos de acordo com sua especificação.

2.3.2 TV Anytime Forum

O *TV Anytime Forum* foi criado em setembro de 1999 com o objetivo de mapear os diferentes requisitos da TV Interativa. Visava-se também, a partir desses, criar especificações de equipamentos para geração e distribuição de conteúdo.

Em setembro de 2000, foi escrita a versão final dos requisitos da TV Interativa (*Requirements Series*), que foi dividida em 5 documentos :

- 1º documento [Tva00a] sugere os modelos de TV Interativa que a TV-Anytime permitirá;
- 2º documento [Tva00b] mostra como a TV Interativa pode ser implementada tecnicamente;
- 3º documento [Tva00c] mostra como os metadados, os quais podem estar presentes em uma transmissão, devem ser tratados;
- 4º documento [Tva00d] apresenta modos de criar referências ao conteúdo. Essas referências servem para auxiliar a busca em um determinado documento;
- Por fim, o 5º documento [TVA2000e] diz respeito ao gerenciamento dos direitos autorais.

Hoje em dia o *TV-Anytime Forum* trabalha na especificação da TV Interativa (*Specification Series*). Essa especificação, também, está dividida em cinco documentos; o único já pronto é o quarto, documento sobre referências ao conteúdo [Tva00f].

O sistema da TV-Anytime é centrado na figura do PDR (*Personal Digital Recorders*), que poderá ser incorporado em aparelhos de DVD, DVD-R, videogames, TVs entre outros.

2.3.3 W3C

Em junho de 1998, foi realizado o *workshop "Television and the Web"*¹, durante o qual ficou evidente a necessidade da padronização de tecnologias para integrar a Web e a televisão. Com isso, foi criado o *Television and Web Interest Group* dentro da W3C (*World Wide Web Committee*)

Na comunidade da televisão existia um desejo de adaptar certas especificações da W3C antes de usá-las em aplicações TVWeb. Pretendia-se definir que somente uma parte das especificações da W3C fosse suportada e que novas características fossem implementadas. Com isso, membros desse *Interest Group* representaram os interesses das televisões nos *Working Groups* do HTML (*Hypertext Markup Language*) e do CSS (*Cascading Style Sheets*).

O grupo trabalhou durante um ano (setembro de 1998 até setembro de 1999) e gerou a especificação da URI "tv:" [Vic00, Fin00], além de gerar mais discussões e idéias sobre a TV Interativa.

¹ W3C Workshop on "Television and the Web" - realizado dias 29 e 30 de junho de 1998 em Sophia-Antipolis na França. Estiveram presentes pesquisadores e representantes de diversas empresas de várias partes do mundo. Estiveram representadas NEC Corp.[®], Microsoft[®], IBM[®] e Philips[®] entre outras.

3 Qualidade de Serviço

3.1 Introdução

QoS em redes IP é um aspecto operacional fundamental para o desempenho fim-a-fim das novas aplicações como voz sobre IP (*Voice Over IP* - VoIP), multimídia, ITV entre outras. Assim sendo, é importante o entendimento dos seus princípios, parâmetros, mecanismos, algoritmos assim como protocolos desenvolvidos e utilizados para a obtenção de QoS. A seguir, serão comentados seus aspectos gerais e parâmetros. Dentre as várias definições existentes para o termo QoS, podem-se destacar as seguintes:

- "O efeito coletivo de desempenho que determina o grau de satisfação do usuário de um serviço específico" [Itu93];
- "Qualidade de Serviço é um requisito que uma classe de aplicações possui. Isso exige que uma série de parâmetros esteja dentro de uma faixa de valores bem definidos, para o seu perfeito funcionamento." [She97]

Abaixo, na Figura 3.1, pode-se observar quanto algumas aplicações necessitam de QoS. Por meio da Figura 3.1, é possível observar que nem todas as aplicações necessitam de QoS. A garantia de uma certa qualidade é muito mais importante para aplicações de base de dados que para projetos de engenharia, por exemplo.

Figura 3.1 - Necessidade de QoS x Aplicação [San99]

As aplicações que necessitam de QoS devem solicitar tal serviço através de uma requisição. Do ponto de vista da infra-estrutura de rede, requisições de uma aplicação em que se deseja utilizar de QoS são vistas como uma solicitação, que será traduzida em um “contrato de serviço”. Solicitações são feitas através da negociação de parâmetros cujos valores devem estar bem definidos. A solicitação de QoS da aplicação é denominada tipicamente de SLA (*Service Level Agreement*) [Job99] [Jam98].

A SLA deve definir claramente quais requisitos devem ser garantidos para que a aplicação possa executar com qualidade. Um exemplo de SLA para uma aplicação de VoIP com algumas centenas de canais de voz simultâneos numa rede IP WAN (*Wide Area Network*) poderia ser:

- Vazão: 2Mbps;
- Atraso: 250mseg
- Disponibilidade: 99,5%

Uma vez garantida tal SLA, tem-se como resultado que a aplicação VoIP em questão poderá executar garantindo a qualidade de voz acordada. Isso com os seus usuários se comunicando simultaneamente através da rede IP.

Do ponto de vista dos usuários, normalmente a qualidade obtida para uma aplicação pode ser variável e, a qualquer momento, pode ser alterada ou ajustada, para melhor ou pior qualidade. Por exemplo, pode-se assistir a um vídeo com uma qualidade de 32fps (*Frames per Second*) ou 4fps e, fundamentalmente, isso depende da qualidade de vídeo esperada pelo usuário final. Embora esse comportamento possa ser dinâmico do ponto de vista dos usuários finais (seres humanos), do ponto de vista das redes, as SLAs são estáticas e, eventualmente, podem ser alteradas. Uma alteração numa SLA implica, normalmente, uma nova solicitação de QoS à rede em questão [Mar00b]. A tabela 3.1 mostra algumas características das mídias mais utilizadas atualmente.

Tabela 3.1 - Características e requisitos das diversas mídias

Mídia	Tráfego	Vazão média dos dados	Tolerância a Perdas	Atraso máximo e Variação de atraso
Texto	Rajada	Poucos Kbps a alguns Mbps	Não tolerável	Não constituem problemas
Imagem Gráfica	Rajada	Até dezenas de Mbps	Compromete imagem final	Não constituem problemas
Áudio	Tráfego contínuo com taxa constante. Introduzindo silêncio, tráfego em rajadas. Com técnicas de compactação, tráfego com taxa variável	Depende da qualidade do sinal e codificação utilizada	Pode ser alta, não causando problemas	Crítico em aplicações que exigem comunicação interativa em tempo real
Vídeo	Tráfego contínuo com taxa constante. Introduzindo-se técnicas de compactação, o tráfego passa a caracterizar-se por ter taxas variáveis.	Varia com a qualidade do sinal e algoritmos de codificação usados	Tolerável, mas com restrições	Deve ser compensada e obedecer aos requisitos da mídia de áudio

Para caracterizar uma comunicação faz-se uso de uma entidade abstrata denominada fluxo (*flow*), termo que será utilizado em vários tópicos a seguir. Um fluxo é uma cadeia de unidades de informação (pacotes, células, etc) com os mesmos requisitos de desempenho e segue uma rota simplex fixa entre a fonte e o destino.

3.2 Aspectos Gerenciais

Do ponto de vista de um gerente ou administrador de redes, a percepção da QoS é orientada para a utilização de mecanismos, algoritmos e protocolos em benefício de seus clientes, e suas aplicações. Esse gerente preocupa-se em como efetivamente a rede e seus componentes podem garantir as inúmeras SLAs definidas para diversos usuários e aplicações.

Cada SLA irá definir um certo conjunto de recursos da rede a serem utilizados para atender à demanda de QoS. Dessa forma, fica evidente a necessidade de uma adequada administração dos recursos – os quais são finitos – de rede; caso contrário, torna-se impossível satisfazer as necessidades dos usuários com a qualidade desejada.

Outros dois aspectos são muito importantes do ponto de vista gerencial, a escalabilidade e flexibilidade da solução de garantia de qualidade de serviço.

A escalabilidade dos protocolos, algoritmos e mecanismos de QoS é um assunto de ampla pesquisa atualmente e se torna particularmente relevante quando se considera a possibilidade de estender a garantia de QoS através de múltiplos domínios administrativos IP [Jam98].

A flexibilidade dos mecanismos de controle de QoS é um fator determinante na sua aceitabilidade pela comunidade. A infra-estrutura de Internet existente atualmente já implantada e amplamente utilizada é bastante heterogênea, por esse motivo mecanismos pouco flexíveis praticamente inviabilizam sua eficácia e utilização.

3.3 Tópicos relacionadas à QoS

De uma maneira geral, as questões relacionadas QoS podem ser visualizadas, em suas diversas vertentes através da seguinte figura 3.2.

Figura 3.2 - Questões de Qualidade de Serviço [Cha00]

Existem três abordagens importantes quando são tratadas questões de QoS:

- Níveis:
 - Host - Hardware e software de cada máquina pertencente a um fluxo;
 - Rede - Tecnologia de comunicação visitada durante o percurso de um fluxo excetuando os *hosts*.
- Procedimentos envolvidos:
 - Gerenciamento de recursos - Sua função é administrar a utilização dos recursos existentes em uma rede, segundo políticas definidas;
 - Reserva de recursos - Realiza a reserva de recursos necessários para uma transmissão tendo como base os parâmetros de QoS desejados para a transmissão;

- Estabelecimento de rota - Dependendo das restrições de QoS e das condições da rede entre as máquinas origem e destino é estabelecida uma rota. Para tal, um algoritmo é utilizado;
- Compartilhamento de *link* - Conforme as rotas são estabelecidas, determinadas transmissões utilizam a mesma porta de saída de um roteador. Esse procedimento não deve desrespeitar os parâmetros de QoS acertados para a transmissão;
- Controle de congestionamento - Procedimentos que visam evitar e também agir de forma corretiva na ocorrência de congestionamento;
- Mapeamento de QoS - Esse item será mais bem descrito no subtópico a seguir.

3.4 Mapeamento de QoS

Os parâmetros de QoS podem ser divididos conceitualmente em diversas camadas, que podem ser observadas na Figura 3.3.

Figura 3.3 - Um modelo conceitual de QoS [Nah95]

3.4.1 Camada de QoS do Usuário

Os parâmetros de qualidade relacionados diretamente aos usuários são necessariamente menos técnicos e mais subjetivos, normalmente mensurados de forma qualitativa. Uma interface apropriada deve ser oferecida para facilitar a escolha dos parâmetros [Vog95].

A essência dessa camada é ocultar, tanto quanto possível, os parâmetros de QoS internos do sistema (freqüentemente sem significado para o usuário). Para isso, devem-se

apresentar exemplos que permitam realizar escolhas como imagens de tamanho e resolução diferentes, ou áudio com qualidade de CD, ou telefone. Em um sistema, cada escolha impacta nas exigências de QoS da rede. Por exemplo, a seleção de uma imagem de alta resolução pode resultar em um aumento de banda necessária e atraso de transferência.

3.4.2 Camada de QoS da Aplicação

Nessa camada são descritos os requisitos para a aplicação em termos de qualidade da mídia e relacionamento entre mídias (sincronização intermídia e intramídia). Escolhas realizadas pelo usuário final, na camada de QoS do usuário, são mapeadas em um conjunto de parâmetros de aplicação, os quais devem atender os seus anseios.

Nesse nível, os parâmetros são associados a quadros de vídeo (tamanho do quadro, taxa de quadros/segundo) ou amostras (taxa de amostragem, bits/amostragem). Relacionamentos entre as mídias também podem ser especificados. Como exemplo de áudio com qualidade de telefone, o mapeamento pode resultar em uma taxa de amostragem de 8KHz com 8bits por amostragem.

3.4.3 Camada de QoS do Sistema

Parâmetros dessa camada representam características de cada *host* ou roteador visitado, as quais podem ser especificadas em termos de critérios quantitativos e qualitativos. Critérios quantitativos são aqueles que podem ser avaliados em termos de medidas concretas, como bits por segundo, tempo de processamento de uma tarefa, e tamanho da unidade de dados. Critérios qualitativos especificam os serviços esperados, como sincronização, mecanismos de escalonamento e recuperação de erros.

Os parâmetros, nesse nível, podem ter um grande impacto na qualidade percebida pelo usuário. Por exemplo, a velocidade do barramento, memória disponível e tempo de CPU (*Central Process Unit*) podem limitar a taxa de quadros da apresentação de um vídeo, bem como uma tela preta e branca não pode mostrar imagens coloridas.

3.4.4 Camada de QoS da Rede

Abordam-se os requisitos dos serviços de rede, por exemplo o desempenho, associados a propriedades de pacotes ou bits, tais como, atraso de pacotes, taxa de bits. Nesse nível, parâmetros fim-a-fim (atraso, variação de atraso e perda de pacotes) são especificados.

Para aplicações multimídia, por exemplo, a rede deve satisfazer os requisitos de transmissão de dados multimídia. O tráfego de mídia discreta (transferência de arquivos ou recuperação de imagens) requer serviços livres de erros, mas é tolerante a atrasos. O tráfego de mídia contínua requer tempo real e transmissão em alta velocidade. Essa mídia é sensível a atrasos e suas variações, mas é tolerante a perdas ocasionais de pacotes. A rede deve suportar os requisitos especificados pela aplicação.

3.4.5 Transformação de Parâmetros de QoS

Para alcançar a satisfação do cliente no que se refere a um determinado serviço, é essencial a correta transformação dos parâmetros de QoS desde o usuário final até a rede de transmissão. Com esse intuito surgiu o conceito de mapeamento de parâmetros de QoS, o qual pode ser definido como: "Transformação da especificação de qualidade de serviço entre os diferentes níveis de qualidade de serviço" [Hua97].

Vale a pena ressaltar que, para uma correta e eficiente transformação de parâmetros de QoS, é necessário levar em conta o tipo da aplicação em questão. A técnica empregada para a transformação de parâmetros de QoS de uma aplicação multimídia não será necessariamente a mesma para uma aplicação de telerrobótica, por exemplo. Dentre as técnicas para realizar o mapeamento de parâmetros de QoS, existem duas formas tradicionais. São elas:

- Mapeamento baseado em tabela [Fuk98] - Esse tipo de mapeamento é estático e para qualquer adequação ou novo parâmetro, é necessária a alteração direta da tabela de mapeamento de classes de parâmetros de QoS. Para esse caso, podem existir determinadas parâmetros não contemplados nessa tabela, e o sistema irá simplesmente recusá-lo. Entretanto esse mapeamento é de simples implementação e, além disso, é bastante ágil: apenas se consulta a tabela e caso seja encontrado o parâmetro, retorna-se o valor; caso contrário, apenas é ignorado.
- Mapeamento baseado em função [Hua97] - Esse mapeamento necessita de uma função matemática que irá realizar o mapeamento de parâmetros de QoS. Qualquer parâmetro de entrada resultará em um parâmetro de saída. A escolha adequada da função de mapeamento passa a ser a questão-chave. Deve-se escolher uma função que corresponda à realidade do público alvo do serviço e, como consequência, deve ser adequada ao escopo da aplicação.

Perante essas técnicas de mapeamento de QoS, torna-se difícil escolher apenas uma, pois ambas possuem suas vantagens e desvantagens, conforme já mencionado. Por isso, foi

desenvolvido um terceiro método que procura reunir o que ambos possuem de vantagem. Esse é um método híbrido, conhecido como "*Spline QoS Mapping*" [Yam99].

A tabela 3.2 mostra um exemplo de mapeamento onde um parâmetro de QoS de usuário é convertido em três parâmetros de QoS da aplicação.

Tabela 3.2 - Exemplos de mapeamento usuário/aplicação [Taw93]

QoS de Usuário	QoS de Aplicação		
Qualidade do Áudio	Frequência de amostragem	Tamanho da amostra	Canais
Muito Bom	44KHz	16bits	Estéreo
Bom	22KHz	8bits	Mono
Aceitável	11KHz	8bits	Mono
Ruim	8KHz	8bits	Mono

QoS de Usuário	QoS de Aplicação		
Qualidade do Vídeo	Taxa (Apresentação)	Tamanho do Quadro	Número de Cores
Muito Bom	25 – 30fps	320 x 240	256
Bom	15 – 24fps	320 x 240	256
Aceitável	6 – 14fps	160 x 120	16
Ruim	1 – 5fps	160 x 120	16

A tabela 3.3 mostra um exemplo de mapeamento de parâmetros de QoS de aplicação para parâmetros de QoS do sistema de rede. Essa conversão é realizada através de funções de conversão ou mapeamento.

Tabela 3.3 - Exemplos de mapeamento aplicação/sistema(rede) [Nah94]

QoS de Aplicação (mídia)	QoS de Rede	Funções de conversão
Ta = Tamanho do quadro	Tb = Tamanho do pacote	$Cb = (Ta/Tb) * Ca$
Ca = Taxa de apresentação	Cb = Throughput	$PPb = Ppa * (Ta/Tb)$
PPa = Taxa de perda de Quadro	PPb = Taxa de perda de pacote TEPb = Tempo entre pacotes	$TEPb = (1/Ca) * (Ta/Tb)$
Aa = Atraso fim-a-fim	Ab = Atraso fim-a-fim	$Ab = (Aa - 2TPT) * (Ta/Tb)$
TPT = Tempo necessário para a aplicação processar um quadro		

O fator essencial desse tópico e relevante para essa pesquisa é a necessidade da realização de um mapeamento ou transformação de parâmetros entre os diversos níveis de QoS. Essa transformação acontece desde a interface com o usuário até a interface com a rede. Esta última é responsável pela transmissão dos dados entre as partes interessadas.

3.5 Tecnologias que Implementam QoS

O foco dos estudos das tecnologias descritas posteriormente está nos parâmetros trabalhados por cada uma delas. A tecnologia ATM possui um grupo razoável de parâmetros e garante QoS para as aplicações que utilizam essa infra-estrutura. O serviço integrado visa

garantir QoS em redes IP, sobretudo com o protocolo RSVP. Dessa forma, esse estudo demonstra-se vital para o projeto do serviço ativo descrito no capítulo 5.

3.5.1 A tecnologia ATM

O modo de transmissão assíncrono (*Asynchronous Transfer Mode* - ATM) é uma tecnologia baseada na transmissão de pequenos pacotes de tamanho fixo denominados células. Essas células são transmitidas através de conexões de circuitos virtuais, sendo que sua entrega e comutação são feitas pela rede, com base na informação de seu cabeçalho. Essa tecnologia se adapta facilmente às exigências de um grande número de tráfegos diferenciados, suportando, com isto, diferentes tipos de serviços [Tan96].

3.5.1.1 Conceitos gerais

Antes de descrever os parâmetros existentes numa conexão do tipo ATM, é importante citar algumas considerações relevantes ao contexto:

- Parâmetros de tráfego da rede - referem-se à fase de transferência de dados;
- Parâmetros relacionados ao momento da conexão - referem-se à sinalização da condição momentânea da rede bem como seu posterior controle;
- Rajadas de células - flutuações da taxa de transmissão das células ATM;

3.5.1.2 Parâmetros de QoS

Os parâmetros de QoS são negociados na fase de estabelecimento de uma conexão, muito embora os procedimentos de sinalização permitam que a negociação possa ser feita depois do estabelecimento. O tipo de serviço a ser fornecido pela conexão no ATM é estabelecido através de um contrato que inclui um descritor da conexão, que informa as características do tráfego gerado e uma especificação de QoS para cada sentido.

A seguir, serão descritos os parâmetros que influenciam na QoS de uma transmissão ATM. Esses parâmetros podem ser negociados no momento do estabelecimento da conexão ou apenas servem para descrever o nível de QoS da transmissão.

Descritores de tráfego ATM

O descritor de tráfego constitui-se de um conjunto mínimo de parâmetros definidos pelo usuário, no sentido de fornecer ao gerente da rede as informações que permitam um controle do tráfego e utilização dos recursos (banda e *buffers*) da rede eficientes. Os parâmetros de

QoS relacionados ao tráfego da rede podem ser divididos em dois subgrupos, cada um com seus respectivos parâmetros, que são:

Descritor de tráfego da fonte:

- *PCR (Peak Cell Rate)*

Taxa de pico de transmissão. Número máximo de células por segundo durante uma rajada de dados. Também pode ser entendida como a taxa máxima na emissão de células pela fonte. O inverso da taxa de pico T ($T=1/PCR$) representa o intervalo teórico mínimo entre as chegadas das células de uma conexão (Figura 3.4). O fluxo de células de uma conexão contido em um enlace de acesso físico pode ser representado através de um eixo de tempo segmentado em intervalos. Esses intervalos representam o tempo de transmissão T_t de uma célula, na taxa de transmissão máxima do enlace. Assim, uma conexão que contratou um PCR igual a um terço da taxa de transmissão do enlace poderá transmitir uma célula a cada três intervalos.

- *SCR (Sustained Cell Rate)*

A taxa de células sustentável de um conexão ATM representa um limite superior da taxa média de transmissão. Seu valor é igual à média da taxa de células transmitidas durante um intervalo de tempo maior que o usado para definir o PCR, levando-se em conta as rajadas. O inverso de SCR ($T_s = 1/SCR$) representa um limite superior para a média do tempo entre chegadas teóricas de células em relação à taxa de longo prazo do enlace.

- *MBS (Maximum Burst Size)*

Tamanho máximo da rajada de dados associado à conexão. Especifica o número máximo de células que pode ser transmitido pela fonte, na taxa de pico, e ao mesmo tempo ainda estar de acordo com a SCR negociado.

- *BT (Burst Tolerance)*

A tolerância a rajadas é um parâmetro de tráfego da fonte e especifica o intervalo entre rajadas de células consecutivas tolerado. O parâmetro BT mais o valor T_s definem o limite superior em relação ao MBS transmitido em conformidade com a taxa de pico da conexão.

- *MCR (Minimum Cell Rate)*

Taxa de transferência de dados mínima aceitável por parte do usuário.

- *MFS (Maximum Frame Size)*

Tamanho máximo do quadro de dados a ser transmitido pela camada de enlace.

Descritor de tráfego da conexão

Esse parâmetro pode incluir, além dos parâmetros do descritor de tráfego da fonte, um outro parâmetro, negociado no momento do estabelecimento da conexão. O descritor de tráfego da conexão integra o contrato de tráfego de uma conexão ATM. A cada conexão são associados dois descritores de tráfego, um para cada sentido do fluxo (ida e volta). O conjunto dos descritores de tráfego varia de acordo com a categoria do serviço. O parâmetro que compõe o descritor de tráfego da conexão é definido como:

- CDVT (*Cell Delay Variance Tolerance*)

Tolerância na variação do atraso das células (tempo entre chegadas de células), ou a quantidade de “*jitter*” aceito pela rede. Esse parâmetro leva em conta as variações de atraso que o fluxo sofre nos equipamentos do ambiente de usuário ao longo do caminho da rede. Seu valor corresponde à diferença entre o tempo de chegada de uma célula e o tempo esperado de chegada (o tempo esperado de chegada é chamado TAT – “*Theoretical Arrival Time*”). Veja a Figura 3.4.

Figura 3.4 - Tolerância na variação do delay entre células - CDVT [Rob94]

O CDVT não é um parâmetro do descritor de tráfego da fonte, mas, sim, um parâmetro introduzido nos mecanismos de UPC (*User Policy Control*) para levar em conta o fenômeno do *jitter*. Sendo assim, esse fenômeno será tolerado dentro dos limites definidos por CDVT, sem que o fluxo seja julgado como não de acordo.

Descritores relacionados a transferência dos dados

Para a fase de transferência de informação, existem seis parâmetros de QoS. Três deles podem ser negociados no momento da conexão e três não. Os três parâmetros negociados no momento da conexão descrevem o perfil de QoS da conexão ATM. Eles são observáveis no

destinatário e medem a performance da rede, por isso também são chamados de QoS-NP (*QoS network parameters*), ou descritor de QoS da rede, e são definidos por:

- CLR (*Cell Loss Ratio*)

Fração de células perdidas causada principalmente pelo tamanho insuficiente de *buffers*, e o conseqüente descarte de células. Também pode ocorrer devido a falhas de enlace. São levadas em conta tanto células que não chegam ao destino como aquelas que são descartadas devido a cabeçalho inválido, atraso excessivo, ou que fazem parte de conteúdo corrompido por erros.

- CTD (*Cell Transfer Delay*)

Representa o atraso de transferência das células pela rede que vai desde o instante em que são emitidas pelo sistema terminal de origem, até a chegada no sistema final de destino. Esse atraso é composto de uma parte fixa, que corresponde aos tempos de propagação dos enlaces, portanto proporcional ao seus comprimento e uma parte variável (CDV). A parte variável surge principalmente por causa da natureza estatística do ATM. Essa natureza faz que o atraso de enfileiramento das células varie de uma célula para outra, dependendo do estado dessas filas. Esse parâmetro foi dividido em dois para uma melhor análise da condição da rede. São eles:

- max-CTD (*Maximum Cell Transfer Delay*) - O atraso máximo na transferência de uma célula pode ser representado como $(1 - \frac{\infty}{\infty})$ da função densidade de probabilidade de CTD (Figura 3.5). As células que excedem esse valor máximo de CTD podem, por exemplo, não ser aproveitadas em tráfego de tempo real. A fixação do parâmetro ∞ é específico da rede e leva em conta o tamanho das filas encontradas ao longo do caminho.
- pp-CDV (*Peak-to-Peak Cell Delay Variation*) - O valor de pico-a-pico do CDV (pp-CDV) representa a diferença entre o valor máximo e o valor mínimo (parte fixa) do CTD (Figura 3.5). Essa medida permite avaliar o atraso máximo entre duas células consecutivas no destino, que na origem foram emitidas com um espaçamento determinístico.

A Figura 3.5 apresenta um exemplo de função densidade de probabilidade do atraso das células em relação à taxa de pico de um tráfego tempo real, portanto não representativo para

tráfego que não seja tempo real. Pode ser observada, na figura, a relação que existe entre os parâmetros max-CTD e pp-CDV, onde este sempre será menor ou igual àquele.

Figura 3.5 - Função densidade de probabilidade do tempo de chegada de células [Roc00]

Existem parâmetros de QoS que não são negociados como parte integrante do contrato de tráfego. Esses parâmetros descrevem a qualidade da rede. São eles:

- CER (*Cell Error Ratio*)

Fração de células com erro. Uma célula é considerada com erro quando possui um erro irreversível, seja no cabeçalho, seja no campo de informação. Erros são causados principalmente nos meios de transmissão devido às suas características físicas.

- SECBR (*Severely - Errored Cell Block Ratio*)

Fração de blocos severamente errados. Um bloco de células é uma sequência de N células transmitidas consecutivamente em uma conexão. Um bloco é considerado severamente errado se ocorreram mais de M células erradas, incluindo-se aqui células perdidas, erradas e mal inseridas.

- CMR (*Cell Misinsertion Ratio*)

Fração de células mal inseridas. Uma célula é considerada mal inserida quando ela é transportada em uma conexão virtual à qual ela não pertence. A taxa de CMR é provocada principalmente por erros de cabeçalho ocorridos ao longo da conexão virtual e que não foram detectados pelo mecanismo de correção de erro de cabeçalho (*head error correction* - HEC) nos comutadores.

Parâmetros específicos da classe de Serviço ABR

- **PCR** - já mencionado anteriormente;
- **MCR** - (*Minimum cell rate*) - taxa de envio de dados mínima garantida pela fonte de inserção de dados rede;
- **ICR** - (*Initial Cell Rate*) - taxa de envio de células pela fonte após período de inatividade;
- **RIF** - (*Rate Increase Factor*) - utilizado para calcular o aumento da taxa de envio de células, caso seja necessário, após o recebimento de uma célula RM (*Resource Management*). Esse parâmetro pode ser encarado como um aditivo $(AIR)=PCR \cdot RIF$;
- **RDF** - (*Rate Decrease Factor*) - utilizado para calcular a diminuição da taxa de envio de células quando necessário;
- **TBE** - (*Transient Buffer Exposure*) - número negociado de células que a fonte deve enviar durante períodos de *start-up*;
- **FRTT** - (*Fixed Round-Trip Time*) - soma dos valores de atraso de propagação e fixo de uma fonte até o destino mais afastado de uma rede local e seu retorno.

3.5.2 Serviços Integrados

A alternativa técnica IntServ foi definida pelo IETF e corresponde a um conjunto de recomendações RFCs (*Request for Comments*), sendo o [Bra94] o principal. Visa à implantação de uma infra-estrutura robusta para a Internet capaz de suportar o transporte de áudio, vídeo e dados em tempo real, com qualidade de serviço, na infra-estrutura atual. O conjunto de recomendações proposto é denominado de "Arquitetura de Serviços Integrados" (*Integrated Services Architecture*) [Bra94].

3.5.2.1 Conceitos gerais

A arquitetura IntServ tem um princípio básico de concepção (Princípio arquitetural):

- A QoS é garantida através de mecanismos de reserva de recursos na rede.

Existem quatro suposições sobre o modelo IntServ:

- Os recursos podem ser gerenciados explicitamente de maneira a atender às requisições;
- As garantias de serviço para serviço tempo-real não podem ser ativadas sem reservas;
- Embora certas aplicações possam adaptar-se dinamicamente às mudanças da rede, o atraso fim-a-fim é limitado;
- Tráfego é dividido entre aplicações de tempo-real e de não-tempo-real, criando-se uma infra-estrutura para essa primeira.

Na definição da arquitetura IntServ dois aspectos operacionais precisam ser definidos:

- Como as aplicações solicitam sua necessidade de QoS à rede, ou seja, como elas fazem suas reservas [Bla94];
- Como os elementos da rede, por exemplo, roteadores, devem proceder para garantir a QoS solicitada (Garantir a reserva) [Bla94].

As aplicações solicitam suas necessidades de QoS à rede através de um protocolo de sinalização, por exemplo, o RSVP (*Resource Reservation Protocol*) onde:

- A aplicação cliente identifica sua necessidade de QoS;
- A aplicação cliente solicita à rede a garantia da QoS que lhe é conveniente (Reserva) através de um protocolo como o RSVP;
- A rede (Equipamentos, roteadores e chaves de roteamento) aceita eventualmente a solicitação e "tenta garantir" a reserva solicitada;
- Uma vez aceita a reserva, os fluxos de dados (*streams*) correspondentes à aplicação são identificados e roteados segundo a reserva feita para os mesmos.

O RSVP habilita dois tipos de serviços integrados [Mag97]:

- Garantido: próximo a um circuito dedicado virtual. Provê limites de atraso de enfileiramento fim-a-fim. Esse limite é obtido pela combinação de parâmetros dos elementos de rede existentes no caminho do fluxo de dados juntamente com a certificação de disponibilidade de largura de banda disponível. Essas estimativas são obtidas a partir dos parâmetros Tspec e Rspec, que serão mais bem descritos no subtópico RSVP.
- Carga Controlada: próximo do serviço de melhor esforço sob condições de rede não congestionada. Utiliza os parâmetros Tspec, porém apenas para garantia probabilística de QoS, ou seja, é melhor que o serviço de "melhor-esforço", mas não garante QoS.

Ambos os serviços devem configurar caminhos e reservar recursos antes de transmitir seus dados [Bla98]. Além das classes de serviços citadas anteriormente, também é oferecida a classe melhor esforço. Essa classe é o serviço padrão oferecido na internet atualmente. O Modelo de Serviços Integrados é implementado por quatro componentes básicos:

- Rotina de controle de admissão - Controla os recursos associados a cada conexão estabelecida. A admissão de uma nova conexão depende da disponibilidade da quantidade de recursos suficiente para atender a requisição nos nós locais da rede. Isso busca prevenir essa nova conexão de interferir no desempenho das conexões já

estabelecidas. Para tal, um algoritmo de controle de admissão deve existir para determinar se um novo fluxo pode ser admitido ou não;

- Classificação de pacotes - Inspecciona múltiplos campos em cada pacote de entrada para determinar a classe do pacote e, assim, o nível de serviço a ser concedido para ele. Tendo obtido os dados necessários, os pacotes são enviados para as filas adequadas conforme o serviço;
- Escalonamento de pacotes - Aplica um ou mais mecanismos de administração de tráfego, tal como escalonamento de filas avançado, como *Weighted Fair Queuing*¹, para assegurar que o pacote seja transmitido dentro da rede em tempo para satisfazer as restrições de largura de banda e atraso do fluxo;
- Protocolo de sinalização (por exemplo, RSVP);

Esses componentes podem ser mais bem entendidos na Figura 3.6 abaixo:

Figura 3.6 - Componentes Básicos da Arquitetura Intserv [Int01]

O RSVP, na Figura 3.6, é o protocolo de sinalização responsável pela reserva de recursos para cada fluxo de transmissão existente no roteador. As rotinas de controle de admissão decidirão se uma requisição por recursos pode ser garantida. Existindo condições para garantir a QoS desejada, o módulo classificador é informado pelo RSVP sobre as condições e exigências do novo fluxo. Quando o roteador recebe um pacote, este é passado ao classificador e, após seu trabalho, o pacote é enviado à sua respectiva fila. Em seguida, o escalonador de pacotes recebe-o e realiza seu trabalho de forma a satisfazer as exigências de QoS estabelecidas para a transmissão.

¹ *Weighted Fair Queuing* marca pacotes individuais de um fluxo com um *timestamp* baseado na taxa de chegada ao roteador [MET99]. A fila de saída do escalonador é reordenada quando um novo pacote chega, assim pacotes com um tempo menor são transmitidos antes.

3.5.2.2 RSVP

ReSerVation Protocol é um protocolo de sinalização que provê controle e configuração de reserva de recursos. Uma das possibilidades de utilização desse protocolo é na implantação de serviços integrados e seu objetivo é estabelecer reservas na rede para fluxos particulares [Bra97].

Esse protocolo é uma das tecnologias mais complexas dentre as tecnologias que visam prover QoS para aplicações e para elementos de redes, como roteadores e chaves de roteamento. Provê um alto nível de QoS garantido, granulosidade de alocação de recursos e detalhe de *feedback* para aplicações QoS-enabled nos usuários. RSVP é um protocolo *soft-state*, e a reserva é iniciada pelo receptor, ao contrário de outros protocolos de reserva de recursos, como o ST-II e ST-II+. Estes últimos são *hard-state*, e a reserva é iniciada pelo emissor. A indústria tem fabricado produtos com RSVP, em vez de outros protocolos, para a reserva de recursos [Bra97].

Parâmetros Utilizados

Fluxos de dados para uma sessão RSVP são caracterizados pelos emissores no Tspec¹ contidos nas mensagens PATH² e espelhados na RSpec³ enviadas pelos receptores nas mensagens RESV⁴. Parâmetros como *token* e *peak rate* são parte do TSpec e RSpec [Wro97]. A seguir, serão descritos brevemente os parâmetros presentes nessas mensagens, segundo [She97b]:

- *Token-bucket rate* (r): taxa de transmissão média sustentável, durante o período de transmissão do fluxo. Este valor corresponde à taxa com que *tokens* são liberados. Para transmitir um pacote, é necessário existir um número de *tokens* igual ao tamanho do pacote. A unidade de medida desse parâmetro é bytes por segundo.
- *Token-bucket depth (size)* (b): através desse parâmetro, é estimada a quantidade de espaço a ser reservado nos *buffer* ao longo da caminho do fluxo de dados. Esse parâmetro define o tamanho máximo do "depósito" de *tokens*, dessa forma é possível limitar o tamanho máximo de uma rajada de dados. A unidade de medida desse parâmetro é bytes.

¹ Definição das parâmetros do tráfego desejado

² Mensagem enviada da fonte ao endereço unicast ou multicast destino para reservar recursos

³ Parâmetros do serviço integrado efetivamente reservado

⁴ Mensagem de confirmação de reserva do destino para a fonte (Tspec → Rspec)

- *Peak Rate (p)*: taxa máxima de envio, caso seja conhecida e controlada.. A unidade de medida desse parâmetro é bytes por segundo. Esse valor pode ser indicado como indeterminado, caso não seja possível obter uma precisão mínima para o caminho.
- *Minimum policed size (m)*: tamanho em bytes do menor pacote gerado pela aplicação emissora. Esse valor leva em conta os *headers* da camada 3 e superiores (IP, TCP, UDP, etc.), mas não contempla os *headers* relativos à camada de enlace de dados. Não é um número absoluto, mas nos casos onde a porcentagem de pequenos pacotes é pequena, esse número deve ser aumentado para reduzir o *overhead* estimado para esse fluxo. Esse parâmetro tem como objetivo propiciar uma estimativa de alocação de recursos por pacote para processar o fluxo de pacotes, pois, a partir dos valores de *b* e *m*, é possível estimar a taxa de transmissão máxima deles. Também é possível obter uma estimativa de *overhead* em termos de largura de banda atribuída aos *headers* dos protocolos de transmissão em relação aos pacotes de tamanho *m*. Os pacotes menores que *m* são tratados como de tamanho *m*.
- *Maximum packet size (M)*: maior pacote a ser transmitido, em bytes, desde que continue em conformidade com as especificações do tráfego. Deve ser considerado absoluto, e qualquer pacote maior é considerado fora de especificação e pode não receber serviço de QoS.

Os parâmetros (*r*) e (*b*) são parte do modelo de encaminhamento *token bucket*. Nesse modelo o pacote que chega é imediatamente encaminhado se existe *token* disponível - o parâmetro (*b*) representa a quantidade dessas fichas; - senão, o pacote é enfileirado até que existam *tokens* suficientes para enviá-lo - o parâmetro (*r*) define a taxa com que as fichas são disponibilizadas. Dessa forma, a taxa real de transmissão de pacotes é disciplinada segundo o modelo *token bucket*. Nas mensagens Rspec, além dos parâmetros Tspec, já mencionados para o serviço do tipo garantido, também estão presentes:

- *Bandwidth (R)*: largura de banda reservada para a comunicação de dados. *R* deve ser maior ou igual ao valor *r* (*token-bucket rate*), e sua medida é feita em bytes por segundo.
- *Slack Term (S)*: diferença em microssegundos entre o atraso desejado para o fluxo de dados e o atraso real obtido através da reserva *R* realizada. O parâmetro *S* pode ser usado pelo elemento de rede para diminuir suas exigências de reserva de recursos para um certo fluxo.

Nas mensagens ADSpec¹ estão sempre presentes os seguintes parâmetros:

- *Global Brak Bit*: bit sinalizador da existência de um ou mais roteadores que não implementa o protocolo RSVP. O valor será 0 caso o elemento da rede suporte RSVP, ou 1 caso contrário.
- *Number-of-IS-hops*: número de *hops* (roteadores) visitados e que oferecem suporte RSVP.
- *Path-BW*: mínima largura de banda, em bytes por segundo, a ser garantida para o fluxo. Localmente irá significar uma estimativa de largura de banda disponível para transporte de pacotes. Para o caminho todo a ser percorrido, fim-a-fim, terá significado da largura de banda disponível para qualquer pacote. A regra de composição é a utilização do mínimo valor existente, ou seja, o valor composto desse parâmetro irá indicar a mínima largura de banda disponível ao longo do caminho. Caso não existam condições de um certo elemento da rede indicar sua estimativa de largura de banda, esse elemento irá repassar o valor zero. Sendo assim, quando o valor zero chegar ao destino, esse *host* saberá que não existe uma estimativa confiável da largura de banda disponível para o caminho do fluxo de dados, como um todo.
- *MPL (Minimum Path Latency)*: somatório de atraso, em microssegundos, mínimo imposto pelos roteadores. Ele acontece devido ao atraso de propagação da velocidade da luz ou limitações no processamento dos pacotes. Esse parâmetro não engloba nenhum atraso relativo a filas nos roteadores. O valor local desse parâmetro pode levar em conta diferentes atrasos para diferentes encaminhamentos de saída. Outra possibilidade é a definição do valor zero para o atraso, caso uma mínima precisão não seja possível. A regra de composição para um certo caminho é o somatório dos mínimos atrasos locais, dando, assim, ao *host* final uma estimativa de atraso mínima para todo o caminho de transmissão de dados.
- *Path-MTU*: tamanho máximo de pacote, em bytes, suportado pelos roteadores no caminho. Localmente esse parâmetro define o tamanho máximo do pacote que pode ser transmitido pelo *host*, sem a necessidade de fragmentação do pacote. Esse valor leva em conta os *headers* da camada de rede e superiores, mas não contempla os *headers* relativos à camada de enlace de dados. A regra de composição é utilizar o mínimo dos possíveis valores obtidos ao longo do caminho. Com esse valor, é possível assegurar que todos os

¹ Objeto opcional das mensagens PATH. Possui características adicionais do tráfego desejado.

pacotes com tamanho igual ou menor passarão por todo o caminho sem fragmentação de qualquer um deles.

Nas mensagens ADSpec, para serviço garantido estão presentes vários parâmetros. Os parâmetros C e D são utilizados para caracterizar um desvio possível de acontecer em um fluxo de transmissão de um serviço garantido. Esses parâmetros são termos de erro, ou seja, ambicionam quantificar um desvio de um modelo de QoS desejado. São dois parâmetros basicamente:

- C: representa o atraso que um fluxo pode experimentar devido a parâmetros de taxa de transmissão acordados para a transmissão. Esse atraso é medido em bytes;
- D: os atrasos possíveis de serem obtidos independentemente da taxa de transmissão são computados nesse parâmetro. Como exemplo, podem-se citar atrasos devidos a tempos de configuração ou *boot* de elementos e serviços de rede. Outro exemplo são redes que trabalham com *slots* para transmissão de dados; nesse caso, o parâmetro D iria computar o tempo de espera de células que, estando prontas, precisam aguardar o próximo slot para transmissão. A unidade de medida desse parâmetro é microssegundos.

A partir desses dois parâmetros, anteriormente citados, outros quatro são definidos:

- Ctot: somatório dos valores de C obtidos ao longo do caminho de transmissão;
- Dtot: somatório dos valores de D obtidos ao longo do caminho de transmissão;
- Csum: somatório dos valores de C obtidos ao longo do caminho desde o último até o atual ponto de *reshaping* do fluxo¹;
- Dsum: somatório dos valores de D obtidos ao longo do caminho desde o último até o atual ponto de *reshaping* do fluxo.

Além desses parâmetros, também é definido uma *flag*:

- *Guaranteed Service Break Bit*: bit sinalizador da existência de um roteador que implementa o protocolo RSVP mas não suporta o serviço garantido.

Mensagens ADSpec para serviço de carga controlada também contêm uma *flag*:

- *Controlled-load Service Break Bit*: bit sinalizador da existência de um roteador que implementa o protocolo RSVP mas não suporta o serviço de carga controlada.

¹ Ponto de uma transmissão multicast onde um fluxo dividido é reagrupado para a continuidade do fluxo

4 Redes Ativas

4.1 Introdução

Entre 1994 e 1995, o DARPA (*Defense Advanced Research Projects Agency's*) iniciou uma discussão sobre como usar redes ativas para tentar resolver o problema da largura de banda existente. As redes ativas, entre outras possibilidades, fazem uso mais eficiente e inteligente da largura de banda.

Redes ativas oferecem suporte para que os nós tenham conhecimento do estado da rede e de outros nós, dos mais próximos, por exemplo. A rede ativa é um novo paradigma proposto em arquiteturas de redes [Ten97]. Uma rede ativa é uma rede inteligente que suporta modificação dinâmica de seu comportamento. Aplicações podem realizar computações personalizadas injetando programas embutidos em pacotes de rede, chamados pacotes ativos. Com essa capacidade, a rede de comunicação torna-se mais flexível.

O controle dinâmico fornecido pelas redes ativas permite a adaptação de serviços para as condições atuais da rede. Esses serviços têm o potencial de melhorar o desempenho fornecido às aplicações. Esforços para melhorar desempenho usando redes ativas se espalham por uma variedade de áreas, entre elas [Ten97]: gerenciamento de rede, qualidade de serviço, *caching e multicast*.

Como redes ativas podem mudar ou adaptar seu comportamento ou o estado dos seus nós, podem-se captar e processar os dados transportados entre o emissor e o receptor. É possível estar consciente do estado da comunicação entre máquinas. Formas de interação baseadas nesse panorama, buscando melhor a QoS da comunicação entre usuário e sistema, podem ser criadas.

As redes ativas são baseadas em métodos similares aos de programação orientada a objeto. Os usuários podem pré-programar certos nós da rede, como roteadores, com um código apropriado antes de rodar uma aplicação. Pode-se fazer isto programando pacotes de dados que levam o código aos nós, ao longo do caminho, até chegarem ao seu destino. O código configura a rede para satisfazer a necessidade da aplicação que gerou os pacotes. Esse código pode ainda dinamicamente adaptar os recursos da rede e rapidamente tornar mais eficiente o ajuste às necessidades de uma aplicação específica [Dar01].

Um pacote tradicional inclui apenas dados e endereço para onde esses dados estão indo. Um nó tradicional simplesmente encaminha o pacote para sua rota. As redes ativas incluem pacotes ativos com dados, endereço e um pequeno programa que diz para a rede o que fazer com um ou mais pacotes. O pequeno programa acessa funções dentro das bibliotecas de métodos dos nós para enviá-lo corretamente.

Na Figura 4.1 abaixo, o nó ativo (caminho de cima) está comprimindo, fundindo e transmitindo vários pacotes. Com isso, economiza-se largura de banda, e (caminho de baixo) o pacote é redirecionado para escapar de um possível congestionamento de rede [Ten97].

Figura 4.1 - Exemplo de funcionamento de Redes Ativas [Bec00]

Para atender às necessidades que surgem com novas aplicações, características adicionais precisam ser incorporadas às redes, como a facilidade de adição de novos serviços e de adaptação dos serviços já existentes. Exemplos recentes incluem a introdução de serviços integrados e diferenciados às redes IP, para fornecer um nível QoS mais satisfatório.

No paradigma de rede tradicional, a introdução de novos serviços é geralmente um processo manual e demorado. O objetivo das redes ativas é simplificar isso gerando redes que suportam explicitamente o processo de criação e implementação de novos serviços.

4.1.1 Duas Escolas de Pensamento

Na área de redes ativas, existem duas escolas de pensamento. Uma delas é endossada pela comunidade Opensig [Opc01]. Esta comunidade argumenta que, através da modelagem do hardware de comunicação, com o uso de uma série de interfaces de redes programáveis abertas, é possível fornecer um acesso aberto a roteadores e *switches*.

A comunidade Opensig argumenta que, através da abertura das *switches* da forma proposta, o desenvolvimento de novas arquiteturas poderá ser alcançado. O objetivo é a criação de uma rede programável. Existe uma grande ênfase na criação de serviços com garantia de QoS também.

Recentemente, o projeto IEEE P1520 [P1520] sobre interfaces de programação para redes adotou o enfoque do Opensig. Trata-se de uma tentativa de gerar padrões para interfaces programáveis em *switches* ATM, roteadores IP e redes de telecomunicações.

Nesse modelo, os equipamentos físicos da rede são abstraídos como objetos de computação distribuída, por exemplo, *switches* virtuais, *switchlets* e estações-base virtuais em redes *wireless*, com interfaces de programação abertas muito bem definidas. Tais interfaces permitem que os provedores de serviços possam manipular o estado da rede com o uso de ferramentas *middleware*, como CORBA. Isso possibilita a construção e o gerenciamento de novos serviços em redes.

A segunda escola de pensamento é endossada pela comunidade DARPA, chamada comunidade AN (*active networks*). Esta advoga a viabilização dinâmica de novos serviços de forma *ad-hoc* e, principalmente, confinados às redes IP já existentes. O nível de suporte dinâmico aos novos serviços vai muito além da proposta da comunidade Opensig. Isso é ressaltado quando se considera a possibilidade de o escalonamento, execução e encaminhamento de pacotes serem realizados dinamicamente através dessa tecnologia. Essa possibilidade toma por base o conceito de pacote ativo (*Active Packet* - AP), também conhecido como cápsula ou pacote "inteligente".

Em um caso extremo, o AP poderia conter programas executáveis. Nas ANs, a mobilidade de código representa o principal meio para o controle e construção de serviços. A granulosidade de controle fornecida pode variar desde o nível de pacote até o nível de fluxo. Nesse enfoque, o termo "granulosidade de controle" refere-se ao escopo do comportamento do roteador/*switch* que pode ser modificado por um pacote recebido por ele.

Por um lado, um simples AP poderia gerar a inicialização ou *boot* de um ambiente de software completo em um nó da rede, o qual passaria a ser enxergado por todos os novos pacotes que chegassem a esse nó. Em outro extremo, um AP poderia modificar o comportamento do nó enxergado somente por ele mesmo. A AN possibilita a configuração dos serviços de redes em nível de transporte de pacotes, em vez de fazê-lo através de um plano de controle programável.

As redes ativas oferecem uma flexibilidade maior para a criação de serviços, porém, a um custo correspondente à adição de uma maior complexidade no nível de programação. O enfoque AN é, no entanto, bem mais dinâmico que o enfoque adotado pelo Opensig para interfaces de programação de redes. Este último é considerado como sendo "quase estático". Ambas as comunidades compartilham um objetivo:

- Ir além dos enfoques e tecnologias existentes atualmente para a construção, implementação e gerenciamento de novos serviços em redes de telecomunicações e de computadores.

Ambos os movimentos incluem um longo espectro de projetos com diferentes enfoques arquiteturais. Por exemplo, nem todos os projetos AN consideram cada pacote como sendo um AP, e, de forma similar, poucos projetos do Opensig consideram as interfaces para as redes programáveis como completamente estáticas. O enfoque Opensig, no entanto, separa claramente o controle da rede do transporte da informação - seu foco está em *switches* programáveis que possam fornecer algum nível de suporte para a garantia de QoS. De forma contrária, projetos AN têm focado historicamente as redes IP, onde a transmissão de dados e de informações de controle se faz de forma combinada.

De forma geral, as redes ativas fornecem uma interface programável nos seus nós. Sendo assim, tornam visíveis os recursos, os mecanismos e as políticas que dão sustentação a uma crescente oferta de funcionalidades. Além disso, fornecem mecanismos capazes de construir e refinar novos serviços a partir daqueles já existentes. Logo, do ponto de vista dos usuários, as redes ativas são capazes de suportar modificações dinâmicas no comportamento da rede.

4.1.2 Conceitos Básicos

O propósito da rede é permitir o compartilhamento dos recursos de transmissão. Uma rede ativa é uma rede do tipo "armazena-e-encaminha", consistindo em um conjunto de nós interconectados por meio de *links* de transmissão. A unidade básica de multiplexação desses recursos de transmissão é o pacote.

Os nós recebem APs de usuários e de outros nós e executam uma computação baseada no estado interno e na informação contida no cabeçalho dos pacotes. Como resultado, esses nós encaminham um ou mais pacotes para outros nós ou para outros usuários na rede. Tendo esse panorama como motivação, alguns conceitos são essenciais para o entendimento desse novo paradigma. Eles são descritos a seguir.

4.1.2.1 Interfaces de Programação

Uma definição básica da maior importância é a interface de programação da aplicação (*Application Programming Interface* - API) da rede, que define todos os aspectos de comportamento visíveis para os usuários finais. Esses aspectos vão desde o comportamento do processamento de pacotes nó a nó até o código, através do qual os usuários podem controlar o envio de pacotes pela rede.

Uma rede ativa fornece uma API de rede programável. Uma grande variedade de enfoques para redes ativas pode ser caracterizada através dos seguintes aspectos:

- Poder de expressão da linguagem
- Estabelecimento de estados (*statefulness*)
- A granulosidade de controle

O primeiro aspecto é o poder de expressão da linguagem. Isso define o grau de "programabilidade" oferecida pela API de rede. Pode variar desde uma simples lista de parâmetros de tamanho fixo, que são selecionados a partir de conjuntos predefinidos, até uma linguagem Turing-completa, capaz de descrever qualquer computação efetiva.

A vantagem de uma linguagem menos poderosa está na existência de um limite dos possíveis comportamentos dos nós. Sendo assim, é simplificada a análise da correção do processo. Em uma linguagem mais poderosa é implementada alguma forma de restrição no seu poder de expressão. A motivação desse fato é garantir que o efeito de qualquer pacote enviado na rede seja limitado. Por exemplo, uma linguagem pode admitir somente programas com seqüências de instruções, sem laços ou desvios.

O segundo aspecto é o estabelecimento de estados (*statefulness*), isto é, a possibilidade de estabelecer estados no interior dos nós da rede e referenciar outros estados já estabelecidos através de outros pacotes. Nas APIs de rede onde esta capacidade faz-se presente, devem existir mecanismos de controle para proteger os estados dos usuários contra acessos não autorizados.

O último aspecto é a granulosidade de controle. Refere ao escopo de comportamento do nó, o qual pode ser modificado a partir de um pacote recebido. Uma das possibilidades a serem consideradas é o caso onde um único pacote possa modificar o comportamento do nó. Pode ocorrer que esse comportamento passe a ser enxergado por todos os pacotes que estiverem chegando a esse nó. Vale a pena ressaltar que esta modificação no comportamento do nó em questão irá persistir até que ela seja sobrescrita por um outro pacote que possa

modificar novamente o comportamento desse nó. Em outro extremo, existe o caso onde um único pacote modifica o comportamento enxergado somente por ele mesmo. Entre esses extremos, pode-se considerar a possibilidade de modificações se aplicarem a um fluxo de transmissão do nó.

4.1.2.2 Nós Ativos

Um nó ativo realiza os trabalhos de um roteador normal e mais um conjunto de tarefas específicas das redes ativas. Ele é capaz de carregar e executar programas dinamicamente. Esses programas são carregados dentro da área de dados de um AP [Cam99].

Os nós ativos necessitam de mais processamento comparados aos roteadores tradicionais, pois devem executar os programas acoplados nos pacotes ativos [Ten97]. Durante o processamento do pacote, o nó ativo é responsável pela integridade da rede e pelo tratamento de erros. Existe um mecanismo de propagação de códigos especiais, peculiar desse tipo de rede, para um conjunto de nós, caso isso seja necessário.

4.1.2.3 Pacote Ativo

Comporta-se como objeto de dado. O programa existente em um AP pode viabilizar aplicações como *multicast*, áudio e vídeo em tempo real e IP móvel com QoS [Dar01]. A função de um AP é chamar programas para serem executados em qualquer nó ativo em que passarem.

Para um AP é importante a existência de técnicas que permitam comprimir os programas neles contidos para não sobrecarregar a rede. Além disso, o AP é encapsulado para ser transportado pela rede. Nesse sentido foi desenvolvido o protocolo ANEP explicado em 4.1.2.6.

4.1.2.4 Redes Programáveis x Redes Ativas

As redes programáveis e as redes ativas estão relacionadas, porém são distintas. As primeiras propõem um controle aberto da rede através de interfaces de programação padronizadas que permitem às aplicações manipularem recursos de mais baixo nível na rede para a construção e o gerenciamento de serviços. Adicionalmente, o paradigma de redes ativas permite que, dinamicamente, um AP contendo códigos de programação seja executado em cada nó da rede. Sob esse ponto de vista, as redes ativas são mais flexíveis do que as redes programáveis, pois, em um caso extremo, permitem que os serviços de rede sejam otimizados por pacote em vez de serem otimizados por aplicação [Gon00].

4.1.2.5 Redes Ativas x Agentes Móveis

Um agente móvel é um programa que atua como procurador de um usuário ou outro programa. É capaz de se mover dentro da rede sob seu próprio controle. O agente decide quando e para onde migrar, podendo interromper a própria execução ou continuar em qualquer outro lugar da rede. O paradigma de agentes móveis trata a rede como um conjunto de ambientes amigáveis aos agentes, que, por sua vez, são tratados como entidades programadas para realizarem tarefas para os usuários. Contudo, a idéia das redes ativas é mais geral.

Nas redes ativas, a rede é vista como uma coleção de nós ativos capazes de realizar qualquer tipo de computação e como uma coleção de pacotes ativos que transportam programas. Sob esse ponto de vista, um agente móvel pode ser considerado um tipo específico de AP, assim como um nó compatível com a tecnologia de agentes móveis pode ser considerado um tipo específico de nó ativo. A diferença fundamental entre o conceito de redes ativas e agentes móveis está no fato de no primeiro, os APs serem executados na camada de rede, enquanto, no segundo, o código é executado como programa na camada aplicação. Sendo assim, a rede ativa, devido à sua natureza programável, oferece funcionalidades básicas à execução dos agentes móveis [Gon00].

4.1.2.6 ANEP - *Active Network Encapsulation Protocol*

Para implementar redes ativas em redes tradicionais, faz-se necessário um protocolo capaz de adaptar as tecnologias de transmissão já consagradas para suportarem o transporte de pacotes inteligentes. O ANEP [ANE97] é o protocolo mais usado para essa função e, para isso, trabalha com encapsulamento de pacotes.

O formato sugerido habilita o uso em qualquer infra-estrutura de rede existente (como IP [RFC791] ou IPv6 [RFC1883]) ou transmissão sobre a camada de enlace. Esse mecanismo habilita a coexistência de diferentes ambientes de execução e a multiplexação adequada do recebimento de pacotes. A Figura 4.2 mostra o cabeçalho:

Figura 4.2 - Cabeçalho ANEP [Ane97]

O campo *Version* indica o formato do cabeçalho em uso. A versão atual é 1. Esse campo só será mudado se o cabeçalho ANEP mudar. Se um nó ativo receber um pacote com número de versão que não reconhece, o pacote é rejeitado.

Flags indica o que o nó deve fazer se não reconhecer o *Type ID*. Se o valor for 0, o nó poderia tentar enviar o pacote usando o mecanismo da opção de distribuição, caso a informação necessária esteja disponível no campo *Options*. Se o valor for 1, o pacote é rejeitado. Recomenda-se que esse campo esteja ajustado para zero.

Type ID indica o ambiente da avaliação da mensagem. O nó ativo deve avaliar o pacote no ambiente apropriado. A autoridade apropriada para atribuir valores do *Type ID* é a ANANA [Ana02]. O valor 0 do *Type ID* é reservado para a camada de rede e as possíveis mensagens de erro. Se o valor contido nesse campo não for reconhecido, o nó deve verificar o valor do bit mais significativo do campo *Flags* e decidir o que fazer com o pacote.

O campo *ANEP Header Length* especifica o comprimento do cabeçalho em palavras de 32bits. Se nenhuma opção for incluída no pacote, seu valor deve ser 2. O *ANEP Packet Length* especifica o comprimento do pacote inteiro, incluindo o *payload*. Esse campo será usado para recuperar o pacote se ele for transmitido.

Options pode ser incluído no pacote, imediatamente depois do cabeçalho básico. Ele está dividido em três campos: 2bits para *Flags*, 14bits para *Option Type* e 16bits para *Option Length*.

4.1.3 Localização do Código

A AN pode ser dividida em três grandes grupos conforme onde o código está localizado. Nos tópicos seguintes, eles serão mais bem descritos.

4.1.3.1 Código apenas nos pacotes

Nessas redes, código executável e dados normais podem ser transportados em um mesmo pacote da rede. A maioria das primeiras implementações de redes ativas permite o transporte de programas nos pacotes de rede. Os nós desse tipo de rede também são considerados ativos por permitirem a execução dos programas; no entanto, não existe nenhum código armazenado nesses nós. Após a execução do código transportado pelo pacote, nos nós ativos, o estado do nó, o comportamento do nó ou mesmo os dados transportados pelo pacote podem sofrer alterações. Como exemplo desse grupo, podem-se citar os seguintes projetos:

- *Smart Packets*, criado pela BBN Technologies [Sma01] [Spr02] [Gon00] [Smi99].
- *Active IP Option*, criado pelo MIT [Ten96].
- *M0 Architecture*, criado pela Universidade da Califórnia em Berkley e Universidade de Zurich [Tsc97].

4.1.3.2 Código apenas nos nós ativos

O pacote transporta apenas identificadores dos códigos que desejam executar em seu nome. Existem esquemas de distribuição e *download* de código, de modo que um nó ativo não precise possuir todos os códigos executáveis existentes em uma dada topologia. Pacotes são considerados ativos quando decidem quais funções serão chamadas para execução nos nós. Além disso, os pacotes carregam os parâmetros necessários para a execução das funções armazenadas.

As funções existentes na rede são criadas pelos seus provedores. Essas funções irão definir o conjunto de serviços que serão suportados pela rede. A partir desse conjunto, os usuários poderão compor seus APs para trafegarem na rede. Esse tipo de AN possui muitas restrições quanto à flexibilidade, pois um usuário comum não pode implementar novas funções. O limite é dado pelo conjunto de opções oferecido pelos provedores. A vantagem dessas implementações está na maior facilidade em assegurar integridade e segurança de rede e dados. Como exemplo dessa arquitetura, podem-se citar os seguintes projetos:

- CANES, Arquitetura criada pelo instituto de tecnologia da Geórgia [Can01] [Bha97];
- DAN *Architecture*, criada pela Universidade de Washington e em ETH Zurich [Dec98];
- ANTS, criado pelo MIT e atualmente gerido e evoluído pela Universidade de Utah [Leg98] [Wet98] [Wet98b] [Wet99].

4.1.3.3 Código nos pacotes e nós da rede

Nesse caso, é possível transportar o código através de APs e também armazená-lo nos nós ativos. Visando otimizar o uso da rede, geralmente os programas com códigos mais elaborados e maiores ficam armazenados nos nós ativos, e os programas mais simples são transportados nos pacotes. Nessa rede, o usuário pode privilegiar uma arquitetura ou outra. O fator determinante dessa escolha é o conjunto de necessidades de cada aplicação. Diferentes aplicações podem utilizar diferentes formas de localização de código em uma mesma rede. Em detrimento disso, existem as complicações adicionadas quanto à segurança e integridade de dados e rede. Como exemplo dessa arquitetura, podem-se citar os seguintes projetos:

- *SwitchWare* criada pela Universidade da Pennsylvania [Aak98] [Ale98] [Swi02];
- *Netscript*, criada pela Universidade de Columbia [Yem96] [Sil98].

4.2 Mecanismos de Funcionamento

Grupos de pesquisa vêm desenvolvendo uma arquitetura que descreve os componentes funcionais e a interface desses componentes para as redes ativas. Os objetivos do desenvolvimento desta arquitetura incluem a minimização de padronizações globais, a escalabilidade global, o suporte ao gerenciamento da rede, o suporte ao envio de pacotes IPv4/IPv6 pelos nós ativos e o suporte a diferentes níveis de qualidades ou classes de serviço.

4.2.1 Conceitos Gerais

De modo a alcançar um serviço de rede programável, a abordagem empregada tem sido a especificação da arquitetura de um nó ativo "padrão". Para tal, devem ser definidas funcionalidades de base comum. Como exemplo, podem-se citar a forma como os pacotes são processados, quais recursos estarão disponíveis no nó e como ter acesso a esses recursos. Sendo assim, a arquitetura define a funcionalidade básica da interface de programação no nó ativo, apesar de ela não especificar nenhuma linguagem ou forma de codificação em particular para esta interface. Esse enfoque tem a vantagem de minimizar a quantidade de acordos globais e de padronizações exigidas para a implementação de uma rede ativa.

A funcionalidade de um nó em uma rede ativa é dividida entre ambientes de execução (AE) e o sistema operacional do nó (nodeOS). A organização geral desses componentes está ilustrada na Figura 4.3. O AE é responsável pela implementação da API de rede, enquanto o nodeOS gerencia o acesso aos recursos do nó local por meio dos AEs.

Figura 4.3 - Componentes da arquitetura para redes ativas [Tei00]

Cada AE é análogo a um programa "shell" em um sistema de computação de propósito geral, por exemplo, o Linux, fornecendo uma interface através da qual os serviços em redes fim-a-fim são fornecidos para os usuários. Assim, a arquitetura permite que múltiplos AEs estejam presentes em um único nó ativo. Todos os pedidos de acesso feitos por usuários aos recursos do nó (incluindo a banda passante de transmissão) são atendidos por meio de um AE.

O sistema operacional do nó (nodeOS) fornece as funções básicas a partir das quais os AEs constroem abstrações que formam a API de rede. Ele gerencia os recursos do nó ativo e escalona a demanda por esses recursos. Como exemplo, podem-se citar a transmissão, o processamento e o armazenamento de dados. O nodeOS isola os AEs dos detalhes de gerência de recursos e também da existência de outros AEs. O AE, por sua vez, esconde do nodeOS a maior parte dos detalhes da interação com o usuário final. Os usuários e as outras entidades da rede são representados por uma abstração chamada de "principal".

As políticas de segurança são definidas em termos de *principals*: o nodeOS é o responsável pela imposição de tais políticas. Quando um AE requisita um serviço ao nodeOS, o pedido é acompanhado por um identificador e, possivelmente, por uma credencial, dirigido ao *principal* em atenção ao pedido feito. Esse *principal* pode ser o próprio AE ou outra entidade, por exemplo, um usuário em nome de quem o AE estiver atuando. O nodeOS apresenta esta informação para um motor de segurança, que verifica a sua autenticidade. É consultado também o banco de dados de políticas de segurança para verificar se o *principal* está autorizado a receber o serviço ou a executar a operação solicitada. Os AEs podem implementar suas próprias políticas para incrementar aquelas já existentes em um nó, porém não podem sobrescrever as políticas do nodeOS.

O nodeOS implementa canais de comunicação, sobre os quais os AEs enviam ou recebem pacotes. Esses canais são *links* de transmissão físicos, por exemplo, Ethernet e ATM, com o processamento do protocolo associado com as camadas de mais alto nível, por

exemplo, TCP, UDP e IP. Quando um nó ativo recebe um pacote vindo de um *link* físico, o nodeOS classifica esse pacote baseando-se no conteúdo de seus cabeçalhos. A seguir, cada pacote ou é associado a um canal existente ou é descartado.

O mapeamento dos pacotes que chegam aos canais é controlado por meio de um padrão especificado pelo AE desde que ele cria o canal. No caso típico, um AE requisita a criação de um canal para aqueles pacotes que seguem um certo padrão de cabeçalho, por exemplo, uma combinação do protocolo IP com os números de porta TCP ou UDP. É de total responsabilidade do motor de segurança garantir que a um dado *principal* seja permitida a criação de um canal com um padrão particular.

Para fornecer QoS, o nodeOS possui mecanismos de escalonamento que controlam o acesso aos recursos de computação e de transmissão do nó. Esses mecanismos isolam o tráfego de usuários dos efeitos causados pelo tráfego gerado por outros usuários. Desse modo cada um parece ter a sua própria máquina e/ou *link* virtual.

Quando os canais são criados, o AE requisitante especifica para o escalonador qual é o tratamento desejado. Esse tratamento pode incluir a reserva de uma quantidade específica de banda passante para o tráfego no canal ou pode incluir o isolamento de outros tipos de tráfego, ou ainda, pode incluir o compartilhamento justo da banda passante disponível com outros canais. Os canais de entrada são escalonados somente para computação, enquanto os canais de saída devem ser escalonados tanto para computação quanto para transmissão.

4.2.2 Fluxo nos nós ativos

Todo pacote, ao chegar a um nó ativo, obedece a um certo fluxo até alcançar a porta de transmissão novamente, se for esse o caso. Uma representação do fluxo de pacotes em um nó ativo está representada na Figura 4.4. Ao chegar ao nó, o pacote é classificado com base nas informações contidas em seu cabeçalho, determinando-se os protocolos relacionados ao seu processamento e o respectivo AE. O protocolo ANEP [Ane97] provê o meio pelo qual os usuários controlam o roteamento dos pacotes a determinados AEs. A arquitetura do nó é projetada de modo a suportar várias APIs de rede simultaneamente. O IPv4 e o IPv6 podem ser exemplos de APIs.

Figura 4.4 - Fluxo de pacotes através de uma rede ativa [Gon00]

Observando-se a figura 4.4, é possível perceber vários ambientes de execução. Alguns ambientes podem processar grupos de pacotes, ou até mesmo, fluxos de pacotes. Outros, porém, processam apenas um único pacote por vez e, ao terminar a execução, o AE é destruído. É importante ressaltar um tratamento especial para um fluxo IPv4, no nó ativo, representado pela figura. Nesse nó, pacotes IPv4, de um determinado fluxo, são simplesmente encaminhados para uma determinada porta de saída. Esse pode ser um exemplo de implementação de Serviços Diferenciados em conjunto com redes ativas.

4.3 Implementações de Ambiente de Execução

Os principais objetivos das ANs são a adequação do serviço de rede aos requisitos das aplicações e às condições da rede e a facilidade de inserção de novos serviços. A partir disso, são projetados modelos dos ambientes e os recursos a serem disponibilizados. Conforme descrito na seção 4.2.1.1, cada implementação pode ser caracterizada por:

- expressividade da linguagem;
- capacidade de estabelecimento de estados;
- granulosidade do controle.

A seguir, serão descritas, em linhas gerais, algumas implementações de AEs. O ANTS é uma implementação de um AE, porém será descrito em particular no apêndice da dissertação por ter sido o AE utilizado para os testes com redes ativas neste trabalho.

4.3.1 SwitchWare

O projeto SwitchWare da universidade da Pennsylvania tem como objetivo atingir a melhor relação possível entre a flexibilidade de uma rede programável e os requisitos de segurança inerentes a uma infra-estrutura compartilhada [Aak98] [Alc98]. Com esse objetivo, a arquitetura definida nesse projeto é dividida em três níveis: pacotes ativos, extensões ativas e infra-estrutura do roteador ativo.

Os APs são programas que substituem os pacotes tradicionais e são constituídos de código e dados. O código de um pacote ativo desempenha a função de controle antes realizada pelo cabeçalho, mas de uma forma mais flexível, pois o processamento de um pacote não está mais restrito a uma simples consulta à tabela de rotas.

Para a programação dos Aps, foi criada uma linguagem denominada PLAN (*Programming Language for Active Networks*) [Pla02]. Linguagem simples que fornece um conjunto restrito de funcionalidades, evitando, em vários casos, processamentos extras para análise de código nos nós ativos [Hic98]. Um exemplo de funcionalidade não disponível na linguagem PLAN é a mudança do estado de um nó. Outra linguagem que está sendo experimentada para a programação dos pacotes é a Caml [Cam02]. A Caml é uma linguagem de uso geral e, sendo assim, não possui as restrições de funcionalidades existentes na PLAN, mas exige a verificação do código.

A limitação de funcionalidades imposta pela linguagem PLAN é compensada pelo uso de extensões ativas. Estas extensões são carregadas nos roteadores e fornecem serviços para vários APs, aumentando desta forma as possibilidades para o processamento. Assim, as extensões são consideradas funcionalidades básicas carregadas dinamicamente, ao invés de código móvel, necessitando de verificação do código somente ao serem carregadas.

O primeiro nível da arquitetura é a infra-estrutura do roteador ativo seguro, enquanto os dois níveis superiores focalizam mobilidade e alteração dinâmica de código. O primeiro é basicamente estático e tem como objetivo a formação de uma base segura sobre a qual são construídos os outros dois níveis.

4.3.2 Smartpackets

O projeto Smartpackets da BBN [Sma01] [Spr02] [Gon00] [Smi99] tem como principal enfoque o gerenciamento de redes. Cada datagrama IP carrega um procedimento a ser executado nos nós ativos em que passam. Os procedimentos não são armazenados nos nós ativos, eles existem somente durante o período de execução. Para a programação das rotinas,

utiliza-se uma linguagem chamada Sprocket [Spr02], que é compilada em uma linguagem de representação compacta e similar a uma linguagem *assembly* genérica denominada *Spanner*.

O AE para a *Spanner* inclui o suporte para o acesso de variáveis de gerenciamento de rede em bases de informações gerenciais (*Management Information Bases* - MIBs). Não existe acesso a outras chamadas de sistema ou memória fora do escopo de execução do datagrama. Um suporte adicional existe para a autenticação dos pacotes visando controle e monitoramento seguros da rede [Gon00] [Smi99].

4.3.3 Plataforma Joust e o Software Líquido

Pesquisadores da universidade do Arizona definem software líquido (*liquid software*) [Har99] como sendo um código móvel, orientado à comunicação e de baixo nível, o qual pode fluir facilmente entre os nós de uma rede. A meta é o transporte eficiente de dados, logo diferentes características são demandadas do SO e do AE que o suportam. Como exemplo, podem-se citar garantia de QoS e restrições de tempo real associado à comunicação. Com esse objetivo, foi utilizado um sistema operacional configurável através da composição de módulos, denominado Scout [Sco02] visando construir uma plataforma para AN chamada Joust [Jou02].

A plataforma Joust possui um módulo interpretador para a linguagem Java, utilizada para programar o serviço de rede. No entanto, Java é uma linguagem interpretada e, normalmente, os seus interpretadores restringem o acesso aos recursos do sistema. Para solucionar esse problema, foram inseridos módulos para a tradução do código interpretado para código objeto, em tempo de execução (compiladores *just-in-time*). Além disso, foram adicionados módulos para introduzir funcionalidades de comunicação na API da linguagem Java, ambos visando o aumento do desempenho global da arquitetura.

4.3.4 Netscript

O projeto Netscript [Yem96] da universidade de Columbia procura definir um conjunto mínimo de funcionalidades em um nó, a partir das quais é possível desenvolver uma grande variedade de serviços. Com essa finalidade, é utilizado o conceito de agente, o qual é definido como um programa que pode ser enviado para a execução em um nó qualquer da rede. Esse agente pode adicionar funções primitivas a esse nó para o processamento dos pacotes dos fluxos e para a alocação de recursos.

O principal objetivo dos agentes nesse projeto é a programação de funções de controle e de gerenciamento da rede. É possível a implementação de protocolos padronizados e não padronizados. Os pacotes, chegando aos nós, são processados pelos agentes apropriados para cumprir as funcionalidades estabelecidas pelos protocolos.

4.4 Segurança em Redes Ativas

Com as AN ganha-se em flexibilidade; no entanto, aumenta-se consideravelmente os cuidados necessários com segurança. É importante reduzir o risco de enganos e comportamentos não esperados. Além disso, deve-se garantir a integridade, privacidade e disponibilidade da rede frente a possíveis ataques. Como o código de um AP pode alterar o estado de um nó, várias exigências devem existir para permitir a execução desse código em um nó de uma AN [Pso99].

Em redes tradicionais, os únicos recursos necessários para o processamento de pacotes são memória e alguns ciclos de CPU. Em NA, são necessários esses recursos, em maior escala, e algumas vezes outros, como acesso a arquivos. Nesse contexto, ataques podem ocorrer caso não exista uma gerência de recursos adequada [Mic98]. Dentre os problemas possíveis de ocorrer, podem-se destacar os seguintes:

- *Damage*: Pacotes podem destruir ou modificar os recursos ou serviços de um nó ativo, reconfigurando, modificando ou apagando dados;
- *Denial of Service*: Pacotes podem sobrecarregar nós ativos consumindo demasiadamente recursos de rede ou mesmo de CPU;
- Roubo: Um pacote pode ter acesso a uma informação confidencial de um nó;
- *Compound Attack*: Um usuário pode enviar, de diversas fontes, pacotes ativos para um roteador, objetivando consumir todo o seu *link* disponível, por exemplo.

Proteger pacotes e nós, em um ambiente flexível, não é uma tarefa fácil. Dentre as técnicas para proteção dos nós ativos, destacam-se as seguintes:

- Autenticação de pacotes ativos: Todo pacote ativo deve possuir uma credencial de autenticação produzida por um ou mais algoritmos, tal como o algoritmo de assinatura de chave pública. Essa técnica não irá garantir a procedência de um pacote, apenas irá identificar o responsável por ele;
- Monitoramento e controle: Utilizado para restringir recursos, serviços e informações de sistema a que um pacote pode ter acesso. Políticas de segurança definem os direitos de cada pacote. Cada conjunto de direitos está associado a uma credencial de autenticação;

- Estabelecimento de limites: o principal objetivo dessa técnica é impedir que pacotes monopolizem recursos da rede. Existem três tipos clássicos:
 - Limite de tempo de execução de um pacote em um nó;
 - Limite de nós visitados por um pacote ativo;
 - Limite de duplicações de pacotes permitidos.
- *Proof Carrying Code* (PCC)[Geo97]: Parte do princípio que é mais fácil verificar uma resposta que produzi-la. Logo, quem conhece a correção de um programa é o criador do programa e não o nó ativo. Com isso, pode-se comparar o programa móvel de cada PA com uma prova de sua correção. O nó ativo compara e, só então, executa o programa ou não. O principal problema é gerar uma prova segura e adequada através do algoritmo de criação.

Para a proteção de pacotes ativos, duas técnicas principais destacam-se: tolerância a falhas e criptografia. A criptografia deve ser usada tanto para o código, como para os dados que trafegam na rede em um pacote. O programa pode ser executado em um nó sem a necessidade de uma decodificação. Esse é o conceito de criptografia móvel[Tom97]. Tolerância a falhas reúne três conceitos básicos:

- Replicação de pacotes em cada nó ativo;
- Persistência de pacotes armazenados por um certo tempo por contingência;
- Redirecionamento: pacotes com rotas alternativas em caso de problemas.

Réplica e persistência não são, normalmente, aceitas nas arquiteturas de rede, elas consomem memória e largura de banda. Somente pacotes de extrema importância, como pacotes responsáveis por uma mudança de um protocolo de rede, podem usufruir dessas técnicas. Redirecionamento e criptografia têm ampla possibilidade de aplicação em segurança de pacotes ativos, pois, normalmente, consomem apenas alguns ciclos de CPU. Combinando as técnicas sugeridas, um pacote, ao chegar a um nó ativo, pode realizar a sequência de ações:

- Verificar a autenticidade da credencial do pacote;
- Identificar o elemento remetente da rede;
- Identificar o usuário remetente;
- Autorizar o acesso aos recursos conforme as credenciais e identificações;
- Permitir a execução do programa, baseando-se em autorizações e políticas de segurança;
- Monitorar e controlar o acesso aos recursos utilizados durante a execução;
- Se necessário, criptografar o pacote para proteger código e dado em trânsito.

Quando alguma necessidade não é satisfeita, o programa pode ser processado em um ambiente de execução extremamente restrito, ou simplesmente, não é executado. Isso depende da política de segurança utilizada, e nenhuma das implementações existentes se utiliza de todos os recursos de segurança possíveis. Em muitos dos casos, o custo-benefício da adoção dessas técnicas não compensa.

4.4.1 SANE (Security Active Network Environment)

Os empreendedores desse projeto são pesquisadores da Universidade da Pennsylvania [Sco98] [Aak98] [Aaa02]. Segurança deve ser encarada em dois níveis, estática e dinâmica. Estática é aquela que é verificada esporadicamente, somente após um período de inatividade da rede, ao realizar um *bootstrapping*. Já a dinâmica deve ser verificada constantemente. Devem ser analisadas verificações estáticas e dinâmicas sempre sob o prisma do nível de segurança e processamento necessário. Políticas de segurança estáticas requerem pouco processamento, porém com nível de segurança baixo. Já políticas de segurança dinâmicas possuem um comportamento exatamente contrário. Em uma rede, onde os estados dos nós são constantemente alterados, espera-se um grande núcleo de políticas de segurança dinâmicas.

SANE é uma arquitetura de camadas. Nas camadas básicas da arquitetura, é assegurado que o sistema inicie em um determinado estado. Essa é uma verificação estática. Após isso, verificações dinâmicas são feitas por remetente ou por pacote. As camadas mais altas dessa arquitetura são responsáveis por essa última tarefa. A partir de então, o sistema mantém a segurança da seguinte maneira:

- Primeiro, executa autenticação remota nó a nó quando necessário;
- Segundo, provê um AE restrito para avaliação dos programas recebidos;
- Por último, utiliza um esquema de espaço de nomes de serviços existentes nos nós e disponíveis para os usuários da rede ativa.

Todo elemento ativo da AN possui um par de chaves pública e privada. Através dessas chaves, os serviços são autorizados ou não, internamente. SANE foi implementada no ambiente SwitchWare, já descrito anteriormente. Nesse AE, estudou-se o impacto no desempenho frente à utilização de políticas de segurança.

Para exemplificar, pode-se citar um caso de teste. Foi realizado um ping, em uma rede ativa, com taxa de transmissão de 100Mb/s Ethernet, para uma outra máquina. O pacote foi carregado, avaliado e enviado de volta. Foi carregado e avaliado novamente. Nesse caso, o tempo total foi de 8.052ms. Realizando-se a mesma ação, sem autenticação de pacotes, o

tempo gasto foi de 5.085ms. Foram realizados testes, nesse mesmo ambiente, relacionados ao impacto de *throughput*. Utilizando-se autenticação, o resultado foi 26% pior que sem autenticação, no recebimento de pacotes e 62% pior no envio de pacotes [Aaa02].

Essa rede utilizada em teste possuía apenas dois nós. Outros testes são necessários para avaliar melhor o impacto na utilização de tecnologias de segurança. Um aspecto fica claro: é possível obter uma rede ativa segura apesar de existir uma proporcionalidade direta quanto a flexibilidade oferecida e à degradação de desempenho obtida.

4.5 Experiências com uma Implementação de Redes Ativas

As experiências foram realizadas com a rede ativa ANTS 2.0.0. Esse pacote foi instalado em duas máquinas pertencentes à rede local do laboratório intermídia ICMC/USP. Após a instalação e configuração da rede ativa composta por essas duas máquinas foi utilizado o serviço ping, implementado, utilizando-se o ANTS, que já vem no pacote para realizar os testes. Porém, essa implementação apenas simula uma rede ativa em uma única máquina. Para entender o funcionamento da rede ativa ANTS, foram inseridas mensagens de alerta em diversos pontos do código do serviço ping. Isso, inicialmente, foi realizado em apenas uma máquina, em uma simulação da rede ativa.

Após ter compreendido o código e o funcionamento da rede ativa, foi construída a nova configuração dela, agora não mais simulando, mas realmente comunicando as duas máquinas. Na sequência, o serviço ativo ping foi alterado para executar nas duas máquinas. Com o sucesso desses testes, foi organizada uma terceira etapa. Esse serviço foi alterado para que, quando o pacote ativo ping alcançasse a outra máquina, uma variável inteira fosse incrementada em cinco unidades. Essa soma foi realizada na máquina destino do serviço e não na máquina origem. Além disso, ao retornar o pacote ativo para a máquina origem, seria exibido o novo valor da variável.

Todos esses testes foram executados com sucesso e propiciaram um bom entendimento do funcionamento global da rede ativa ANTS, bem como comprovaram as funcionalidades prometidas pelo paradigma de redes ativas.

5 Serviço Ativo para Provimento de Parâmetros de QoS

5.1 Introdução

Um serviço para provimento de parâmetros de QoS utilizando a infra-estrutura de redes ativas, ou seja, um serviço ativo, visa permitir que outros serviços ativos, como de reserva de recursos, o utilizem para a obtenção dos dados necessários à sua execução. Tal serviço deve ser um dos componentes do protótipo de ITV que irá garantir QoS aos usuários do sistema.

Esse serviço realiza o cálculo e a disponibilização de parâmetros de QoS de sistema, rede, aplicação e usuário. Estes dois últimos podem ser criados conforme o conjunto de aplicações disponíveis e a necessidade dos usuários da rede em um ambiente administrativo *web* dedicado ao administrador da rede. Todos os parâmetros são disponibilizados em arquivos e obedecem a um padrão de armazenamento dos dados previamente conhecido, o que possibilita a utilização de tais dados por sistemas ou outros serviços que conheçam esse padrão. Além disso, é oferecido ao administrador da rede, pelo ambiente de administração *web*, monitorar a QoS oferecida pela rede entre uma máquina fonte e outra destino.

A seção 5.2 descreve alguns conceitos relevantes quando se fala de parâmetros de QoS. A seção 5.3 lista os parâmetros estudados e planejados durante a construção desse serviço ativo. A seção 5.4 apresenta a especificação do serviço proposto. A seção 5.5 descreve o projeto do serviço proposto. A seção 5.6 discute as modificações necessárias ao pacote de redes ativas ANTS 2.0.0 para a construção desse serviço.

5.2 Parâmetros de QoS

Parâmetros de caracterização são usados para descobrir ou caracterizar o ambiente de gerenciamento de QoS ao longo de caminho de um fluxo de pacotes que requer controle ativo de QoS fim-a-fim [She97b].

A definição de cada parâmetro usado para caracterizar um caminho através da rede possui dois tipos de semântica: local e composto. O valor local caracteriza um único elemento da rede. Valores compostos refletem a composição corrente de valores locais ao longo de um caminho, segundo uma regra de composição.

Cada definição de parâmetro especifica sua regra de composição. Essa regra diz como combinar um valor composto (da porção já percorrida do caminho) com o valor local. A partir de então, gera-se um novo valor a ser passado ao próximo elemento da rede no caminho [She97b].

Uma regra de composição pode ser especificada tanto para *downstream* (em direção ao receptor) como para *upstream* (em direção ao emissor). O valor do parâmetro relacionado ao elemento local da rede deve possuir uma única definição, porém mais de uma regra pode ser definida para a composição de um parâmetro ao longo do caminho, dependendo da necessidade e da natureza do parâmetro.

Os parâmetros de caracterização possuem essencialmente o sentido de "*per-next-hop*". Dessa forma, elementos de rede com várias possibilidades de encaminhamento, roteadores de borda de grandes subredes, por exemplo, podem assumir diferentes valores de composição de um parâmetro, dependendo do endereço do próximo elemento no caminho.

Determinados parâmetros, em certos elementos de rede, para um dado momento, podem ou não ser calculados, com um mínimo de precisão no cálculo. Isso pode ser provocado por algum problema com o elemento ou mesmo com a rede. Quando isso acontece, é emitido um valor-padrão de incerteza para o parâmetro em questão e, dessa forma, o valor composto do parâmetro para o caminho é questionável [She97b].

A tabela a seguir mostra alguns exemplos de aplicações que necessitam de garantias quanto a certos parâmetros de QoS para oferecer um serviço de qualidade aceitável para o usuário final.

Tabela 5.1 - Parâmetros de QoS a considerar pelas aplicações [Arg98]

Aplicação	Parâmetros de QoS Necessários
Ftp	Largura de banda
Telnet	Atraso e Jitter
http	Largura de banda
Telefonia	Largura de banda, Atraso, Jitter e Taxa de perdas
Vídeo sob demanda	Largura de banda, Atraso, Jitter e Taxa de perdas

5.3 Parâmetros de QoS Estudados

A seguir, são listados os parâmetros de qualidade de serviço estudados. Esses parâmetros estão presentes em várias tecnologias descritas anteriormente, como ATM e RSVP.

Tabela 5.2 - Parâmetros de QoS estudados

Parâmetro	Descrição
PCR - (<i>Peak Cell Rate</i>)	Número máximo de células transmitidas por segundo
SCR - (<i>Sustained Cell Rate</i>)	Taxa de transmissão de células sustentável ($>$ média)
MBS (<i>Maximum Burst Size</i>)	Tamanho máximo da rajada de dados em uma conexão
BT (<i>Burst Tolerance</i>)	Intervalo entre rajadas consecutivas tolerados
MCR (<i>Minimum Cell Rate</i>)	Taxa mínima aceitável numa transmissão de dados
MFS (<i>Maximum Frame Size</i>)	Tamanho máximo do quadro de dados permitido
CDVT (<i>Cell Delay Variance Tolerance</i>)	Tolerância na variação do atraso das células (quantidade de "jitter" aceito pela rede)
CLR (<i>Cell Loss Ratio</i>)	Taxa de perda de células
Max-CTD (<i>Cell Transfer Delay</i>)	Atraso máximo de transferência (de origem até destino)
pp-CDV (<i>Cell Transfer Delay</i>)	Variação pico a pico do atraso (de origem até destino)
CER (<i>Cell Error Ratio</i>)	Taxa de células recebidas com erro
SECBR (<i>Severely - Errored Cell Block Ratio</i>)	Taxa de blocos de células (células transmitidas consecutivamente) recebidos com erro
CMR (<i>Cell Misinsertion Ratio</i>)	Taxa de células inseridas em conexões (VC) erradas
MCR - (<i>Minimum cell rate</i>)	Taxa de envio mínima garantida pela fonte de dados
ICR - (<i>Initial Cell Rate</i>)	Taxa de envio permitida após período de inatividade
RIF - (<i>Rate Increase Factor</i>)	Fator de aumento da taxa de envio ($PCR * RIF$)
RDF - (<i>Rate Decrease Factor</i>)	Fator de diminuição da taxa de envio
TBE - (<i>Transient Buffer Exposure</i>)	Número negociado de células que a fonte deve enviar durante períodos de <i>start-up</i>
FRTT - (<i>Fixed Round-Trip Time</i>)	Soma do atraso de propagação e fixo da fonte até o destino mais afastado de uma rede local e seu retorno
Nrm	Número máximo de células que uma fonte pode enviar em cada encaminhamento de célula RM realizado
Trm	Intervalo de tempo permitido entre encaminhamentos sucessivos de células RM por uma fonte ativa
ACR - (<i>Allowed Cell Rate</i>)	Taxa atual de envio de células que uma fonte está autorizada a transmitir.
CRM - (<i>Missing RM cell count</i>)	Limita o número de envios de células RM que podem ser enviadas quando não se recebem células de retorno RM.
CDF - (<i>Cutoff Decrease Factor</i>)	Controla a diminuição do ACR associado com o CRM
ADTF - (<i>ACR Decrease Time Factor</i>)	Tempo permitido entre envios de células RM antes de a taxa de envio de células ser diminuída para ICR
TCR - (<i>Tagged Cell Rate</i>) -	Limita a taxa pela qual uma fonte pode enviar células RM "out-of-rate" (fora das taxas acordadas)
r - <i>Token-bucket rate</i>	Taxa de transmissão média sustentável durante o período de transmissão do fluxo
b - <i>Token-bucket depth (size)</i>	Limitar o tamanho máximo de uma rajada de dados
p - <i>Peak Rate</i>	Taxa máxima de envio se conhecida e controlada
m - <i>Minimum policed size</i>	Tamanho do menor pacote gerado pela fonte emissora
M - <i>Maximum packet size</i>	Tamanho do maior pacote gerado pela fonte emissora
R - <i>Bandwidth</i>	Largura de banda reservada para transmissão de dados
S - <i>Slack Term</i>	Diferença entre o atraso desejado para o fluxo de dados e o atraso real obtido através da reserva R realizada
Path-BW	Mínima largura de banda a ser garantida pelo fluxo

MPL (<i>Minimum Path Latency</i>)	Somatório de atraso mínimo imposto pelos roteadores
Path-MTU	Tamanho máximo de pacote suportado pelos roteadores
C (Parâmetro ADSpec - RSVP)	Atraso devido a parâmetros de taxa de envio acordados
D (Parâmetro ADSpec - RSVP)	Atraso independentes de taxa de envio acordados
Vazão do tráfego (<i>throughput</i>)	Taxa de transmissão de dados disponível
Máximo de Perdas Consecutivas	Número máximo de perdas consecutivas aceitável
Intervalo de Medição de Perdas	Tempo para medição do número de perdas ocorridas
Disponibilidade da rede	Porcentagem do tempo em que a rede esteve disponível
Ociosidade da Memória Primária	Quantidade de memória primária não utilizada
Ociosidade de processamento	Quantidade de ciclos de CPU livres por segundo
Disponibilidade de disco	Espaço em disco disponível para utilização

Esses parâmetros servem como base para a definição do conjunto de parâmetros de QoS a serem disponibilizados pelo serviço ativo a ser implementado.

5.4 Especificação do Serviço

O serviço de provimento de parâmetros de QoS em questão é um serviço ativo, pois é projetado para ser implementado em uma rede ativa. A seguir, serão descritos os detalhes referentes à especificação desse serviço ativo.

5.4.1 Objetivos

Os principais objetivos possíveis de serem atingidos são:

- Projetar em uma infra-estrutura de rede ativa um serviço de provimento de parâmetros de qualidade de serviço;
- Prover parâmetros de QoS de uma rede. Através desses parâmetros, é possível obter uma estimativa das condições da rede;
- Oferecer condições para que outros serviços possam utilizar os parâmetros coletados, trabalhá-los e disponibilizar, por exemplo:
 - Caminho menos congestionado;
 - Reservar recursos;
 - Implementar políticas de controle de admissão;
- Disponibilizar uma interface de criação de novos parâmetros de QoS de aplicação a partir dos parâmetros de rede e de sistema existentes. Da mesma forma, permitir a criação de novos parâmetros de QoS de usuário a partir dos parâmetros aplicação;
- Delegar ao administrador do sistema a responsabilidade de definir as fórmulas de transformação de parâmetros de rede e de sistema em parâmetros aplicação.

Delegar também ao administrador a definição de fórmulas de transformação de parâmetros de aplicação em parâmetros de usuário;

- Definir uma interface flexível e amigável de administração dos parâmetros de qualidade de serviço de aplicação e de usuário;
- Permitir a expansão do conjunto dos parâmetros de QoS sem necessitar da interrupção dos serviços existentes;
- Oferecer uma interface para análise dos parâmetros de QoS de aplicação e de usuário disponibilizados entre quaisquer dois pontos da rede administrada.

5.4.2 Parâmetros Disponibilizados

Na Tabela 5.2, é mostrado um conjunto de parâmetros de qualidade de serviço utilizados por algumas tecnologias atualmente. A partir desse conjunto, foi selecionado um subgrupo de parâmetros de QoS a ser trabalhado pelo serviço ativo descrito neste trabalho. Para a escolha de quais parâmetros fariam parte desse serviço, foram utilizados os seguintes critérios de seleção:

- Caracterizar um descritor das condições da rede - o parâmetro deve ser relevante para ser utilizado a fim de definir o estado da rede em conjunto com outros parâmetros;
- Possibilidade de ser medido e disponibilizado em um ambiente de rede ativa - parâmetros específicos de certas tecnologias e que sejam inviáveis de serem obtidos sem a utilização dessa tecnologia foram descartados;
- Definição de um conjunto de parâmetros significativo - o conjunto dos parâmetros deve ser capaz de permitir a criação de novos parâmetros de aplicação e usuário relevantes à rede em questão.

A tabela a seguir lista os parâmetros de QoS disponibilizados.

Tabela 5.3 - Parâmetros de QoS Disponibilizados

Parâmetro	Descrição	Unidade
Atraso	Valor médio de atraso entre dois nós por um determinado período	Microssegundos
Variação de Atraso	Valor médio da variação dos atrasos ocorridos entre dois nós vizinhos em um certo período	Microssegundos
Perda	Porcentagem de perda de pacotes entre dois nós, (erro + não recebimento) por um período.	Porcentagem
Disponibilidade	Porcentagem de tempo em que uma conexão esteve apta a enviar dados em um certo período	Porcentagem
Vazão	Taxa de transmissão média alcançado entre dois nós	Bytes/Segundo

(Throughput)	vizinhos por um certo período	
MTU de Transmissão	Tamanho Máximo da Unidade de Transmissão aceita por um <i>link</i> sem divisão dessa unidade.	Bytes
Ociosidade de Memória	Porcentagem de memória primária livre média em um nó da rede por um certo tempo	Porcentagem
Ociosidade de processamento	Porcentagem de processamento (CPU) livre média em um nó da rede por um certo tempo	Porcentagem
Disponibilidade de disco	Espaço livre de armazenamento em disco livre em um nó da rede	Bytes
Período de cálculo das médias	Tempo para coleta dos valores dos parâmetros sem reinicializar o cálculo das médias	Segundos
Tempo entre coleta dos parâmetros	Tempo entre envio de pacotes ativos responsáveis pela coleta dos valores dos parâmetros por conexão	Segundos

Para cada nó ativo da rede será disponibilizada uma tabela de valores de cada um dos parâmetros acima mencionados. Esses valores podem ser de três tipos:

- Parâmetros de sistema - não dependem da rede, dependem apenas das condições locais da máquina. Os parâmetros desse tipos são:
 - Ociosidade de memória;
 - Ociosidade de processamento;
 - Disponibilidade de disco.
- Parâmetros de rede - esses valores, residentes em cada nó ativo, são obtidos através da análise das características e da condição do *link* de rede entre o nó em questão e seus vizinhos imediatos. Os parâmetros desse tipos são listados abaixo:
 - MTU de transmissão;
 - Atraso;
 - Variação de atraso;
 - Perda;
 - Disponibilidade;
 - Vazão.
- Parâmetros de ajuste das medidas - utilizados para realizar ajustes da forma como os parâmetros são medidos. Os parâmetros desse tipos são listados abaixo:
 - Período de cálculo das médias;
 - Tempo entre coleta dos parâmetros.

Os parâmetros de ajuste de medidas têm por objetivo ajustar a taxa de carga da rede utilizada especificamente para o cálculo dos parâmetros, bem como refinar a precisão das medidas. O primeiro é possível obter aumentando o tempo entre a coleta dos dados. Já o segundo necessita de um aumento do período de cálculo das médias juntamente com a

diminuição do período entre coleta dos dados. Os parâmetros desse último tipo possuem algumas características especiais:

- Podem ser geridos apenas pelo administrador do sistema;
- Somente através da interface de administração eles podem ser alterados;
- Não necessitam de regra de composição;
- Seu valor é global e vale para toda a rede ativa em questão.

5.4.2.1 Regras de Composição dos Parâmetros

Os parâmetros da Tabela 5.3, exceto os parâmetros de ajuste, possuem, além de um valor local, também um valor para cada caminho de transmissão entre fonte e destino. Dessa forma, a cada parâmetro existente deve ser associada uma regra de composição do valor. Essa regra define o que irá acontecer quando o serviço passar por cada nó ativo com o objetivo de calcular o valor dos parâmetros de QoS para um dado caminho. As regras de composição levam em conta todos os nós pertencentes ao caminho entre fonte e destino, incluindo-se esses dois últimos. Essas regras são listadas na Tabela 5.4, logo abaixo:

Tabela 5.4 - Regras de Composição dos Parâmetros de QoS Disponibilizados

Parâmetro	Regra de Composição
Atraso	Somatório dos atrasos medidos
Variação de Atraso	Somatório das variações dos atrasos medidos
Perda	Valor da maior porcentagem de perda obtida
Disponibilidade	Valor da menor porcentagem de disponibilidade obtida
Vazão (<i>Throughput</i>)	Valor da menor vazão obtida
MTU de Transmissão	Menor valor de MTU permitido obtido
Ociosidade de Memória	Menor valor de ociosidade de memória primária obtido
Ociosidade de Processamento	Menor valor de ociosidade de processamento obtido
Disponibilidade de Disco	Menor valor de disponibilidade de disco obtido

Sempre que houver a necessidade de conhecer a situação atual da rede entre dois pontos, fonte e destino, deverá ser requisitado o serviço ativo para a coleta dos parâmetros de QoS para o caminho entre os pontos em questão.

Esse serviço irá percorrer o caminho entre a fonte e o destino, e coletar os parâmetros atuais de cada nó ativo da rede. Para cada nó ativo visitado, seus parâmetros de QoS são lidos e computados, conforme a regra de composição de cada parâmetro, com os valores já lidos e computados de outros nós. Ao chegar ao destino e tendo sido calculados todos os valores de parâmetros necessários no caminho, os dados retornam até a origem para serem disponibilizados.

Caso seja necessário, esses parâmetros podem ser transformados em parâmetros de QoS de aplicação e, por sua vez, podem ser transformados em parâmetros de usuário. Essas transformações serão mais bem detalhadas no item 5.5.

5.4.3 A Interface Administrativa do Serviço

Faz parte do serviço ativo proposto uma interface administrativa para o gerenciamento do próprio serviço. Essa interface possui, basicamente, quatro módulos:

- Administração dos parâmetros de ajuste de medidas;
- Administração dos parâmetros de aplicação;
- Administração dos parâmetros de usuário;
- Análise da qualidade oferecida pela rede.

5.4.3.1 Administração de Parâmetros de Ajuste de Medida

Esse módulo administrativo tem como objetivo disponibilizar uma interface capaz de gerenciar os parâmetros responsáveis por ajustes de tempo do serviço ativo. São dois os parâmetros gerenciáveis nessa seção:

- Tempo entre coleta dos parâmetros - esse parâmetro define de quantos em quantos segundos serão coletados os parâmetros de sistema e de rede em cada nó ativo;
- Período de cálculo das médias - esse parâmetro define de quantos em quantos segundos os valores coletados dos parâmetros de QoS de sistema e de rede são inicializados para novos cálculos em cada nó ativo.

A interface administrativa deve oferecer as seguintes funcionalidades para o administrador da rede:

- Alteração dos valores dos parâmetros em questão a qualquer momento;
- Imediata disponibilização dos novos parâmetros para a utilização do serviço ativo, sem que isso signifique queda da disponibilidade do serviço.
- A interface deve ser amigável e de fácil acesso, independentemente da localização física do administrador.

5.4.3.2 Administração dos Parâmetros de Aplicação

O objetivo desse módulo administrativo é permitir ao administrador da rede criar parâmetros de QoS específicos para cada aplicação. As funcionalidades que essa seção administrativa deve oferecer são as seguintes:

- Criar, editar e remover aplicações pertencentes à rede ativa em questão;

- Criar, editar e remover parâmetros de QoS de Aplicação;
- Definir as funções de transformação de parâmetros de QoS para cada parâmetro de aplicação. Isso tendo como base os parâmetros de QoS de sistema e de rede existentes e disponibilizados pelo serviço;
- Associar os parâmetros de QoS de aplicação criados às aplicações também criadas.

5.4.3.3 Administração de Parâmetros de Usuário

Esse módulo administrativo tem como objetivo permitir ao administrador da rede criar parâmetros de QoS específicos para cada usuário de determinada aplicação. As funcionalidades dessa seção administrativa são as seguintes:

- Criar, editar e remover parâmetros de usuário de cada aplicação;
- Definir as funções de transformação de parâmetros de aplicação para parâmetros de usuário. Os parâmetros base são os parâmetros associados à aplicação escolhida;
- Permitir a definição de intervalos de valores na transformação de parâmetros de aplicação em parâmetros de usuário.

Os parâmetros de usuários pertencem a uma determinada aplicação. Sendo assim, para cada aplicação existente, podem existir quatro componentes associados:

- Parâmetros de QoS de aplicação;
- Funções de transformação de parâmetros de rede e parâmetros de sistema em parâmetros de aplicação;
- Parâmetros de QoS de usuário;
- Funções de transformação de parâmetros de aplicação em parâmetro de usuário.

Até a consecução dos parâmetros de usuário, é necessário que o serviço obtenha todos os tipos de parâmetros de QoS. Logo, é preciso obter os parâmetros de rede e os parâmetros de sistema, para então calcular os parâmetros de aplicação e, nesse momento, é possível calcular os parâmetros de usuário. Como exemplo, pode-se citar um caso fictício em que a rede ofereceu os seguintes parâmetros de rede e sistema:

- (d) Atraso = 10000 microssegundos;
- (lp) Perda = 5 % (0,05);
- (b)Vazão = 100.000 bytes por segundo;
- (p) Ociosidade de processamento = 90 % (0,9);
- MTU – 50 bytes.

Continuando no caso fictício, existe uma aplicação com os seus parâmetros, suas respectivas funções de transformação e, como consequência, seus "atuais" valores. A Tabela 5.5 abaixo mostra esses valores:

Tabela 5.5 - Parâmetros de Qos e Funções de Transformação Fictícios I

Parâmetro de Aplicação	Função de Transformação	Valor Obtido
Tamanho do Quadro (tq)	$tq = 2 * MTU$	100 bytes
Atraso Fim-a-fim (aff)	$aff = d * (1 + lp + (1-p))$	11500 microssegundos
Perda de Quadros (pq)	$pq = lp = 0,01$	0,06 (6%)

A partir desses parâmetros de aplicação, é possível obter os valores dos parâmetros de usuário utilizando-se as funções de transformação. A Tabela 5.6 abaixo finaliza o exemplo.

Tabela 5.6 - Parâmetros de Qos e Funções de Transformação fictícios II

Parâmetro de Usuário	Função de Transformação	Valor Obtido
Qualidade do Vídeo (Qv)	$Qv = \text{bom} - \text{se } aff < 12000 \text{ e } pq < 0,05$ $Qv = \text{médio} - \text{se } 0,51 < pq < 0,61$ $Qv = \text{ruim} - \text{caso contrário}$	Médio
Qualidade do Áudio (Qa)	$Qa = \text{bom} - \text{se } aff < 10000$ $Qa = \text{médio} - \text{se } 10001 < aff < 11000$ $Qa = \text{ruim} - \text{caso contrário}$	Médio

Através do exemplo, pode-se perceber como é possível criar os parâmetros de aplicação e de usuário bem como suas respectivas funções de transformação, até chegar à qualidade possível de ser oferecida a um usuário final de uma determinada aplicação.

5.4.3.4 Análise da QoS Oferecida pela Rede

Esse módulo administrativo visa permitir ao administrador da rede analisar os parâmetros de qualidade de serviço oferecido entre dois pontos quaisquer da rede. Através desse módulo, deve ser possível visualizar as seguintes informações entre determinadas fonte e destino:

- Parâmetros de sistema;
- Parâmetros de rede;
- Parâmetros de aplicação;
- Funções de transformação de parâmetro de sistema e de rede em parâmetros de aplicação;
- Parâmetros de usuário;
- Funções de transformação de parâmetro de aplicação em parâmetros de usuário.

Através desse módulo, o administrador pode confirmar ou verificar uma série de fatos. As seguintes situações, dentre possivelmente muitas outras, podem ser constatadas:

- Congestionamento em um determinado caminho;
- Indisponibilidade de um certo nó ativo da rede;
- Pontos de maior e menor carga da rede;
- Incoerências nos cálculos dos parâmetros de QoS;
- Necessidade de mudança das fórmulas de transformação de parâmetros;
- Necessidade de redimensionamento dos recursos da rede;
- Necessidade de redimensionamento dos recursos de um determinado nó ativo.

Para essa análise, levam-se em conta os parâmetros de QoS e as respectivas funções de transformação existentes na rede.

5.4.4 Aplicações

Através desse serviço, outras aplicações, ativas ou não, projetadas tendo como base o serviço em questão, poderão usufruir dos dados coletados para implementarem novos serviços e disponibilizarem na rede. Na seqüência, são listados alguns exemplos de implementações a partir desse serviço ativo:

- Procura do caminho menos congestionado;
- Reserva de recursos em nós ativos intermediários entre nós fonte e destino;
- Políticas de controle de admissão de novas conexões;
- Monitoramento de QoS assegurado às conexões;
- Negociação de QoS.

5.5 Projeto do serviço

Para o projeto completo do serviço, são necessárias algumas decisões quanto à tecnologia utilizada, ao ambiente, dentre outras. A seguir, são detalhados todos os tópicos relevantes à construção do projeto do serviço.

5.5.1 Escolha da Rede Ativa

Como já descrito anteriormente, a tecnologia de redes ativas possui duas escolas de objetivos distintos. A tecnologia ANTS enquadra-se na escola AN e não na frente de desenvolvimento da Opensig [Ope01].

Devido à necessidade de criação e sobretudo aquisição de novos equipamentos, os quais iriam viabilizar o projeto da escola Opensig, tecnologias nessa linha precisam de um investimento financeiro inicial bastante razoável quando comparado com tecnologias da linha

AN. Somada a esse argumento, encontra-se a facilidade de aprendizado de algumas tecnologias da escola AN. Como algumas dessas tecnologias trabalham com linguagens tradicionais e consagradas como JAVA, a opção por uma implementação de rede ativa da linha AN demonstrou-se mais apropriada.

Porém, essa é apenas parte da análise que deve se feita no trabalho de optar por uma implementação de rede ativa. Dentro da escola AN, existem atualmente várias implementações disponíveis para desenvolvimento de testes e soluções, porém a implementação escolhida foi a ANTS 2.0.0.

A opção pelo ANTS tem por base três itens:

- Existência de gerenciamento de recursos;
- Camada de segurança adaptável;
- Desenvolvimento inteiro em JAVA.

O gerenciamento de recursos é realizado por um NodeOS (*Node Operating Systems*), também desenvolvido em JAVA, denominado Janos Java NodeOS [Jan01]. Maiores detalhes podem ser obtidos no apêndice dessa dissertação. Com a utilização do software Janos, a partir da versão 2.0.0, além de se obter uma melhor gerência em termos de recursos, ganhos de performance também ocorreram na medida em que esta tarefa não é mais efetuada pelo ambiente de execução da rede ativa ANTS.

A camada de segurança adaptável, também, é uma nova funcionalidade propiciada pela adoção do software Janos. Na versão 2.0.0 do ANTS, é possível criar políticas de segurança específicas para determinados nós ativos. Essas políticas são compiladas em um arquivo, que é passado como parâmetro na execução desse nó. Ainda tratando-se da questão segurança em ANTS, é possível especificar por aplicação qual direito de acesso é requerido.

Somando-se esses dois pontos relativos à segurança, pode-se dizer que esse tipo de tratamento pode ser rígido ou simples o suficiente conforme a necessidade de um determinado nó ativo e as aplicações que nele forem executadas.

O terceiro e decisivo ponto a favor da implementação ANTS é ser totalmente desenvolvido em JAVA. Com isso, é possível elencar uma série de vantagens oriundas da utilização dessa linguagem, entre as quais, podem-se citar as seguintes:

- Facilidade de entendimento - como essa linguagem é muito utilizada atualmente, o entendimento dos conceitos ANTS e a produção de códigos e testes demandam um tempo menor quando comparada com uma linguagem proprietária, por exemplo;
- Portabilidade - com o JAVA os problemas de incompatibilidade entre máquinas de diferentes plataformas é inexistente. Basta para isso instalar os softwares necessários e configurar corretamente as máquinas que formam a arquitetura de rede ativa;
- Estabilidade - como o JAVA é amplamente usado, testado e evoluído, essa linguagem é mais estável quando comparada com uma linguagem recentemente criada especificamente para o uso em redes ativas;
- Adaptabilidade entre tecnologias - existem, hoje em dia, várias implementações prontas e disponíveis de certas tecnologias em JAVA. Como exemplo, pode-se citar o SNMP. Como essas implementações e o ANTS foram desenvolvidos em JAVA, uma integração de tecnologias é menos custosa e mais factível.

Esses foram os fatores preponderantes para a adoção da tecnologia ANTS 2.0.0 no serviço ativo para provimento de parâmetros de qualidade de serviço.

5.5.2 Parâmetros de ajuste

Os parâmetros de ajuste são definidos através da interface administrativa do serviço ativo, a qual será mais bem descrita no item 5.5.6. Nesse tópico, será discutida a forma de armazenamento desse tipo de parâmetro.

Os parâmetros de ajuste são armazenados em um arquivo chamado "param.ajuste", em um diretório específico visualizado por todos os nós ativos da rede ativa. Para tal, o sistema de arquivos distribuído é utilizado. Pode-se utilizar, por exemplo, o NFS. Isso é necessário devido ao fato de os parâmetros de ajuste serem os mesmos para toda a rede.

Cada parâmetro será armazenado em uma linha do arquivo, sendo que no início da linha deve aparecer o ID do parâmetro, na sequência o sinal de igual, e, logo depois, o valor do parâmetro. Como exemplo, pode-se citar uma linha com os seguintes caracteres:

- A=800

A interpretação desse valor é que o primeiro parâmetro de ajuste, o "Período de cálculo das médias" está com o número oitocentos associado no momento em que a leitura foi realizada. A mesma lógica deve ser seguida para o outro parâmetro de ajuste.

5.5.3 Descrição do Ambiente

O ambiente capaz de hospedar o serviço ativo proposto deve possuir as seguintes características:

- Uma rede local convencional é a base para a criação da rede ativa;
- Os roteadores o podem ser de fato ou, então, ser configurados para que ajam como tais para a execução dos testes;
- Todos os roteadores devem utilizar o sistema operacional Linux com a versão mais recente de preferência;
- Todos os roteadores existentes no ambiente devem possuir uma máquina virtual java instalada com versão JDK 1.1 ou superior;
- A rede ativa será composta por todos os roteadores que formam a rede local;
- A rede deve possuir um sistema de arquivo para redes, por exemplo NFS;
- Com a utilização do sistema de arquivos para redes, é possível instalar o ANTS apenas em uma máquina, inclusive as cápsulas, protocolos e aplicações, e por meio do sistema, compartilhar de todos os nós da rede. Além disso, certos diretórios de uma certa máquina, também, podem ser compartilhados pelo grupo de máquinas desejadas.

5.5.4 Coleta dos Parâmetros Locais

Em cada nó ativo da rede, será disponibilizada uma tabela com os valores dos parâmetros de QoS do sistema e da rede. Os parâmetros de sistema dependem apenas da condição do nó ativo e, sendo assim, para cada parâmetro existirá um valor. Os parâmetros de rede dependem da condição do *link* entre os nós ativos. Dessa forma, para cada parâmetro existirá um valor associado a cada *link* entre o nó ativo em questão e os seus vizinhos imediatos. A seguir, são listados os parâmetros de QoS de sistema e de rede.

Tabela 5.7 - Parâmetros de Sistema e de Rede Disponibilizados

Parâmetros de sistemas	Parâmetros de rede	
<ul style="list-style-type: none"> ▪ Ociosidade de memória ▪ Ociosidade de processamento ▪ Disponibilidade de disco 	<ul style="list-style-type: none"> ▪ Atraso ▪ Variação de atraso ▪ Perda 	<ul style="list-style-type: none"> ▪ Disponibilidade ▪ Vazão (<i>Throughput</i>) ▪ MTU de transmissão

5.5.4.1 Parâmetros Envolvidos

Os valores obtidos são médias de sucessivas medidas realizadas ao longo de um período. O número de medidas a serem realizadas é definido pela seguinte fórmula envolvendo os parâmetros de ajuste:

$$N = (\text{Período de cálculo das médias}) / (\text{Tempo entre coleta dos parâmetros}).$$

Esse valor N , um parâmetro auxiliar, deve ser o maior número inteiro menor ou igual ao resultado da divisão. N é armazenado em sistema de arquivo, juntamente com o parâmetro auxiliar x . Esse parâmetro x inicialmente possui o mesmo valor de N , porém a cada coleta de parâmetros realizada, x é decrementado. Quando x for -1 , após sucessivas subtrações, seu valor será novamente o valor de N , e os parâmetros de QoS são reinicializados com zero. Os parâmetros auxiliares serão armazenados em um arquivo de nome "param.aux" em um diretório previamente conhecido e será específico para cada nó ativo.

Os parâmetros de ajuste são globais, isto é, valem para toda a rede. Os valores dos parâmetros são atualizados segundo uma média aritmética, envolvendo a média já obtida dos valores dos parâmetros e o número de coletas realizadas.

A coleta dos parâmetros será através do envio de cápsulas ativas segundo o protocolo ativo ping. As cápsulas são enviadas pela aplicação ativa, no caso, a aplicação ping. Para cada vizinho é enviada uma cápsula ativa e, após visitar o vizinho imediato, a cápsula retorna normalmente, e com o seu retorno os parâmetros de rede são calculados. Essa cápsula será enviada de tempos em tempos conforme o valor do parâmetro de ajuste "Tempo entre coleta dos parâmetros".

A estratégia de implementação desse envio de cápsulas conforme o parâmetro de ajuste utiliza-se de um parâmetro auxiliar *wait* w . Esse parâmetro inicialmente possuirá o mesmo valor do parâmetro "Tempo entre coleta dos parâmetros". Além disso, existirá um *script* que será executado pelo crontab do servidor, a cada trinta segundos, responsável por ler o valor de w e subtrair 30. Obtendo-se zero ou um número negativo como resultado, a cápsula ativa será enviada; caso contrário, após a subtração, a execução do *script* será finalizado. Se o resultado for zero, o parâmetro auxiliar w irá receber o valor do parâmetro de ajuste envolvido. Caso o resultado seja um número negativo, w irá receber o resultado da subtração entre o parâmetro de ajuste envolvido e o módulo do número negativo obtido.

O tempo de trinta segundos escolhido para a execução do *script* anteriormente citado tem dois objetivos principais:

- Não propiciar um possível motivo de sobrecarregar o nó ativo com uma granularidade menor que trinta segundos;
- Permitir aos parâmetros de ajuste assumirem valores na escala de segundos caso o administrador da rede deseje.

Além dos objetivos descritos, também podem ser citadas duas conseqüências da escolha desse tempo de trinta segundos:

- O parâmetro de ajuste "Tempo entre coleta dos parâmetros" não poderá assumir valores menores que trinta;
- Será evitado o problema de execução do *script* envolvido sem o término da primeira execução do mesmo *script*. Isso acontece devido à própria lógica do serviço ativo. Como o tempo-limite de espera de uma cápsula é um segundo, caso ela não retorne nesse período, será contabilizada como perdida, e o *script* será encerrado com margem de segurança suficiente até a próxima execução do mesmo *script*.

5.5.4.2 Informações Básicas para Cálculo dos Parâmetros

Para o cálculo dos parâmetros, é necessário um grupo de informações básicas que serve como matéria-prima para a definição dos parâmetros de rede. As informações básica são as seguintes:

Tempo de ida e volta da cápsula ativa

Um *timestamp* é inserido na cápsula ativa para o cálculo do tempo de ida e volta estimado da cápsula ativa. Quando a cápsula retorna é calculado o *timestamp* nesse exato momento. Subtraindo-se o valor de tempo existente na cápsula desse último *timestamp* calculado, obtém-se o tempo de ida e volta estimado que se deseja. A partir desse valor, é possível calcular o tempo gasto para a cápsula sair do nó fonte e chegar ao nó destino. Dividindo-se o tempo resultante por dois, é obtido o tempo médio (t) de percurso.

Esse valor trata-se de uma estimativa; logo por uma questão de simplicidade, foi escolhida essa estratégia de cálculo. Em média, conforme os tempos de ajustes definidos pelo administrador do sistema (vide item 5.5.6), esse valor obtido pode aproximar-se do valor real existente.

É utilizado o *timestamp* na cápsula ativa por uma questão de praticidade. É possível realizar o cálculo do tempo de percurso da célula no nó destino e não no nó fonte, porém para isso é preciso calcular o *timestamp* no nó destino no momento de chegada da cápsula. Dessa

forma, é indispensável que ambas as máquinas estejam sincronizadas aumentando assim as exigências do serviço. Realizando-se o cálculo de tempo pela mesma máquina, o nó fonte, tal problema não existe.

Definição de uma cápsula perdida

A aplicação ativa responsável por enviar a cápsula ativa deve esperar um tempo-limite para o retorno da cápsula; caso contrário, corre-se o risco de esperar indefinidamente. Sendo assim, após 1 segundo de espera, caso a cápsula não retorne, ela é considerada perdida.

Dentro do período de cálculo dos parâmetros de QoS, definido pelo parâmetro de ajuste "Período de cálculo das médias", a cada célula perdida, um contador é incrementado. Esse contador (c) é armazenado no sistema de arquivos local do nó ativo. O arquivo "param.aux" irá armazenar esse contador por se tratar de mais um parâmetro auxiliar.

Taxa de transmissão do link

A taxa de transmissão do *link* (r) é igual ao valor máximo de transmissão de dados que pode ser enviado por segundo, conforme a tecnologia de enlace de dados que o *link* entre dois nós ativos possuem. Como exemplo, pode-se citar a tecnologia Ethernet, que pode suportar 10, 100, 1000 ou até 10000Mbps.

Para se obter o valor real de cada *link* de transmissão, é necessário realizar uma consulta local no nó ativo ao chegar a cápsula ativa. Tal consulta é feita através da execução do comando `ifconfig`, de qual um dos parâmetros de retorno é a tecnologia da camada enlace de dados. Com essa informação e um tabela de associação presente na própria cápsula, pode-se obter o valor da taxa de transmissão de certo *link*.

Tamanho da cápsula enviada

O tamanho da célula enviada será representado pelo símbolo (s). Esse valor é facilmente conhecido através da função `ANTS lenght()`, que informa o tamanho de uma cápsula. O objeto em questão é a própria cápsula. Esse tamanho é o calculado pela aplicação emissora das cápsulas, e sua unidade de medida é bytes.

5.5.4.3 Cálculo dos Parâmetros de QoS de Rede

Os parâmetros de rede são calculados para cada interface de rede existente no nó ativo. Estando em um determinado nó ativo, cada parâmetro de rede irá quantificar a QoS do *link* entre o nó em questão e seus vizinhos imediatos.

Atraso

O atraso experimentado por uma cápsula nesse serviço ativo é definido como a diferença de tempo, em microssegundos, que a cápsula levaria caso a velocidade de transmissão da camada de enlace realmente fosse obtida e o tempo real de transmissão alcançado. Os parâmetros necessários para esse cálculo são:

- Tempo de percurso entre fonte e destino (t);
- Tamanho da cápsula enviada (s);
- Taxa de transmissão da camada de enlace (r).

A fórmula de cálculo do atraso é a seguinte:

$$\text{Atraso} = (t - (s/r)) * FC * 10^6 - \text{resultado em microssegundos.}$$

Como a taxa prometida pela camada de enlace não é alcançável, graças a atrasos existentes em diversos momentos da transmissão, é aplicado um fator de correção (FC) ao valor obtido. Esse fator atualmente é fixo, e seu valor é 0,9. Dentre os fatores que contribuem para ocorrer atrasos, podem-se destacar os seguintes:

- Tempo de preenchimento dos pacotes;
- Tempo de propagação no meio físico em questão;
- Tempo de serialização;
- Tempo de armazenamento nas filas de espera;
- Tempo de processamento dos equipamentos da rede. Isso por sua vez depende de algumas variáveis como:
 - Capacidade de processamento;
 - Disponibilidade de memória;
 - Mecanismos de cache.

Variação de atraso

A variação de atraso pode ser definida de várias formas, porém pode-se destacar a seguinte:

- "Flutuação no atraso, ou variação estatística do atraso".

A partir dessa definição, adaptada para o serviço ativo, pode-se concluir que a variação de atraso pode ser definida como uma média aritmética. Essa média envolve as variações de atrasos sucessivas experimentadas durante um período de tempo definido pelo parâmetros

"Período de cálculo das médias". A fórmula para o cálculo desse parâmetro de rede necessitará dos seguintes valores:

- Variação do atraso anterior (V_{aa}) = tendo os dados para o cálculo da variação do atraso atual, é armazenado em memória o valor anterior;
- Número de coletas durante o "Período de cálculo das médias" - (N);
- Número de coletas que falta para reiniciar os parâmetros (x);
- Atraso anterior (A_a) - ao calcular o atraso atual é armazenado em memória o valor do atraso anterior para ser utilizado nesse momento;
- Atraso atual (A) - novo valor para o atraso é calculado.

A fórmula para o cálculo da Variação de atraso fica definida da seguinte forma:

$$\text{Variação de atraso atual} = \{ (V_{aa}) * [N - (x + 1)] + (|A_a - A|) \} / (N - x)$$

O resultado de $A_a - A$ deve ser em módulo, por isso aparece como $|A_a - A|$. Alguns fatores que influenciam a variação de atraso podem ser destacados:

- Diferenças de prioridade e tamanho nas filas dos elementos de rede;
- Multiplexação realizada nos elementos de rede;
- Pico da taxa de transmissão;
- Tamanho de *buffer* no percurso da transmissão.

Perda

Perda é um índice de qualidade fortemente dependente do mecanismo de armazenamento e reserva de recursos envolvido na transmissão de dados ao longo da rede [Par92].

Em redes de comutação de pacotes, os pacotes recebidos pelos nós intermediários são armazenados em *buffers* até serem transmitidos para o próximo nó. Como o tamanho dos *buffers* é limitado, pode ocorrer perda, por exemplo, devido a congestionamento.

Através dessa taxa, é possível estimar a qualidade de transmissão de dados de uma determinada rede. Esse campo pode também ser usado para, por exemplo, sugerir que a rede selecione a rota de transmissão com menor taxa de erro ou perda.

Para o cálculo das perdas existentes em um determinado *link*, o serviço ativo utiliza-se dos seguintes parâmetros:

- Número de coletas durante o "Período de cálculo das médias (N)";
- Número de coletas que faltam para reiniciar o "Período de cálculo das médias" - (x);

- Número de cápsulas perdidas (c) durante o "Período de cálculo das médias".

Sendo assim, a fórmula para o cálculo desse parâmetro ficou da seguinte forma:

Se $c = 0$, Perda = 0, caso contrário Perda = $c/(N - (x))$.

Alguns fatores que proporcionam perdas podem ser destacados:

- As taxas de erros sobre as linhas de transmissão (bits ou pacotes);
- Congestionamentos da rede (sobrecarga nos *buffers*);
- Rajadas de dados não controladas (sobrecarga nos *buffers*);
- Políticas de descarte de pacotes errôneas;
- Tamanho dos *buffers* no percurso dos dados inadequado.

Disponibilidade

A disponibilidade da rede (D) visa obter a porcentagem do tempo em que uma determinada rede esteve disponível para o transporte dos dados.

Esse parâmetro pode ser útil para se ter uma idéia da frequência com que os serviços da rede sofrem quedas, mesmo que parciais. Essa informação, em conjunto com o nó ativo ou *link* responsável pela indisponibilidade, permite ao administrador da rede gerar uma série de ações corretivas na fonte do problema e até preventivas em relação a toda a rede.

O cálculo desse parâmetro leva em conta os seguintes fatores:

- Número de cápsulas perdidas (c) durante o "Período de cálculo das médias";
- Tempo entre a coleta dos parâmetros "Tempo entre coleta dos parâmetros";
- Tempo de cálculo das médias dos parâmetros "Período de cálculo das médias".

A fórmula para o cálculo desse parâmetro assim ficou constituída:

$$D = c * (\text{Tempo entre coleta dos parâmetros}) / (\text{Período de cálculo das médias})$$

Dentre vários fatores que levam à degradação da disponibilidade da rede, podem-se destacar os seguintes:

- Equipamentos com tolerância a falhas;
- Redundância da topologia da rede;
- Disponibilidade dos equipamentos utilizados na rede.

Vazão

A banda disponível num certo momento define a vazão do tráfego. Esse índice está relacionado com a necessidade de caracterizar como o tráfego do fluxo será injetado na rede.

Expressa a quantidade de dados que será enviada através da rede por unidade de tempo [Par92].

Uma possível definição para vazão pode ser a "quantidade máxima de informação que pode trafegar através de um canal de transmissão". Para o cálculo da vazão de uma determinada interface de rede, levam-se em conta os seguintes fatores:

- Tempo de percurso entre fonte e destino (t);
- Tamanho da cápsula enviada (s).

Tendo em vista esta definição, a fórmula de cálculo ficou a seguinte:

$$\text{Vazão} = s/t.$$

Dentre vários fatores que influenciam a vazão, podem-se destacar:

- Largura de banda de cada *link* (interface de rede);
- Tamanho dos *buffers* disponíveis nos elementos de rede;
- Número de conexões simultâneas em uma máquina.

MTU

Possui basicamente a função de limitar a quantidade de tempo em que um fluxo pode transmitir, ininterruptamente, num meio compartilhado. Com esse valor, pretende-se minimizar a necessidade de divisão de pacotes, ao longo de um caminho de transmissão, devido à existência de MTUs de diferentes tamanhos.

A definição desse parâmetro é "Comprimento máximo em número de bytes do maior pacote passível de ser transmitido em um fluxo" [Par92]. Para a obtenção do valor de MTU de cada vizinho imediato, basta realizar uma consulta ao nó ativo local.

Para o caso de MTU, é preciso obter o IP real de cada vizinho ativo imediato. Isso é possível obter através do arquivo de configuração de cada aplicação ativa, arquivos ".config". Tendo esse dado com o comando "ifconfig", é possível obter o respectivo MTU. Através do objeto ANTS "*Neighbortable*", é possível obter os vizinhos ativos imediatos.

Caso exista um nó não ativo entre dois nós ativos vizinhos, o MTU que será utilizado para cálculo é o MTU da interface de rede padrão de roteamento, por exemplo eth0. A interface-padrão de roteamento pode ser obtida por meio da consulta à tabela de roteamento com o comando "route". Essa decisão é importante em nós ativos que possuem mais de uma interface de rede.

Esse parâmetro não é obtido através da transmissão de uma cápsula ativa como os demais parâmetros de rede. É a aplicação ativa que o obtém, pois esse parâmetro não necessita do tráfego da cápsula, mas sim de trabalhar a saída da execução do comando `ifconfig` e, se necessário, `route` do próprio nó ativo.

5.5.4.4 Cálculo dos Parâmetros QoS de Sistema

Os parâmetros de sistema são obtidos através de consultas a arquivos do sistema operacional ou através de processamentos realizados a partir do retorno de determinados comandos linux. Esse trabalho é realizado pela aplicação ativa, no próprio nó ativo. Todos os parâmetros de QoS são armazenados em sistema de arquivos em um local predeterminado e, portanto, conhecido pelo serviço. Cada um dos parâmetros é descrito a seguir.

Ociosidade de Memória

A ociosidade de memória mede a porcentagem de utilização da memória, informando quanto, em média, de memória o nó ativo não está utilizando. O principal objetivo desse parâmetro é oferecer uma medida do quanto está sobrecarregado um determinado nó ativo.

O diretório `/proc` do sistema operacional linux contém as informações sobre os processos em execução, assim como informações de sistema, como IRQs e portas de entrada e saída utilizadas [Con01]. Esse diretório será acessado para a obtenção de alguns parâmetros necessários a esse serviço ativo.

Para a obtenção desse parâmetro, basta ler o arquivo `meminfo` do diretório `/proc` de qualquer sistema operacional linux. Esse arquivo reflete a situação da memória em cada momento. Dentre as informações desse arquivo, uma é o total de memória do sistema e o total de memória livre. Por esses dois dados, é possível calcular a porcentagem de ociosidade de memória.

Ociosidade de Processamento

A ociosidade de processamento mede a porcentagem de processamento não utilizada, em média, num determinado nó ativo. Juntamente com o parâmetro ociosidade de memória, esse parâmetro visa oferecer um indicativo do quanto o nó ativo em questão está sobrecarregado.

Para conseguir esse dado, basta trabalhar com o comando linux `"top"`. Com o retorno desse comando, pode-se obter o total de processamento da máquina e quanto não está sendo

utilizado. Com esses dois valores, é possível calcular a porcentagem de ociosidade de processamento da máquina em questão.

Disponibilidade de Disco

O parâmetro disponibilidade de disco mede a porcentagem de disco que ainda não foi utilizada pelo nó ativo em questão. Essa medida, juntamente com os outros parâmetros de sistema, visa oferecer uma estimativa do quanto que uma determinada máquina não está sendo utilizado, em relação a todo o seu potencial. Para o caso específico de disco, esse parâmetro pode ser útil para determinadas aplicações que demandem espaço de armazenamento. Um exemplo dessas aplicações pode ser uma biblioteca distribuída de vídeo, onde os vídeos são armazenados de forma a equilibrar o quanto de espaço cada servidor disponibiliza para a aplicação e, como consequência, distribuir a carga de transmissão.

Para o cálculo desse parâmetro, basta trabalhar com o retorno do comando linux "df". Esse comando retorna o total possível de armazenamento em cada unidade (por exemplo hda e possíveis partições hda1, hda2, etc) e o quanto está livre. Para a obtenção do valor final do parâmetro, basta somar o total de disco, em bytes, não utilizado em cada unidade.

5.5.4.5 Armazenamento dos Dados Locais

Existem três tipos de parâmetros que devem ser armazenados localmente por poderem assumir valores diferentes de nó ativo para nó ativo. São eles:

- Parâmetros de rede, calculados levando-se em consideração cada um dos vizinhos ativos imediatos;
- Parâmetros de sistema, calculados a partir das condições do próprio nó ativo;
- Parâmetros auxiliares, calculados a partir de outros parâmetros. N, x e c, já discutidos anteriormente, são os parâmetros auxiliares do serviço ativo.

Esses três tipos de parâmetros são armazenados em um diretório específico, fora dos diretórios de ação do sistema de arquivos distribuídos. Porém, para cada tipo de arquivo, existe um ou mais arquivos responsáveis pelo armazenamento dos dados.

Os parâmetros de rede possuirão um arquivo para cada vizinho imediato existente. O nome desse arquivo pode ser, por exemplo, o IP ANTS do vizinho imediato ao nó ativo local. Os parâmetros de sistema serão todos armazenados em um único arquivo, que se pode chamar "param.sys". Os parâmetros auxiliares também serão armazenados em um arquivo chamado com o nome "param.aux".

Todos os arquivos desse diretório respeitarão a mesma estratégia de armazenamento interna. Cada parâmetro será armazenado em uma linha do arquivo, sendo que no início da linha deve aparecer o ID do parâmetro, na sequência o sinal de igual, e, logo depois, o valor do parâmetro.

5.5.5 Coleta dos Parâmetros de um Caminho

Um caminho de rede é composto por todos os nós ativos existentes entre a máquina origem e o destino. Além disso, também fazem parte do caminho, os *links* de transmissão que ligam os nós integrantes do caminho.

Para o cálculo dos parâmetros de QoS de um caminho completo, é necessária uma outra cápsula ativa, além da cápsula incumbida de realizar o percurso entre um determinado nó e seu respectivo vizinho imediato. Essa segunda cápsula ativa tem por responsabilidade visitar cada um dos nós ativos integrantes do caminho entre o nó fonte e o destino.

5.5.5.1 A Cápsula Ativa

Duas informações são indispensáveis para o correto funcionamento da cápsula que realizará o serviço de cálculo dos parâmetros de um determinado caminho:

- Lógica de construção do percurso entre os nós fonte e destino;
- Regra de composição de cada um dos parâmetros de rede e de sistema.

A primeira informação é obtida pela própria natureza da cápsula ativa. A mesma lógica do protocolo ping será usada nessa cápsula. A única diferença com relação à primeira cápsula ativa, utilizada para o cálculo dos parâmetros locais, está no fato de esta última, sabidamente, realizar o protocolo ping entre seus vizinhos imediatos. Para o cálculo dos parâmetros de um caminho, é executado o mesmo protocolo, porém os nós fonte e destino não são necessariamente vizinhos; podem estar a qualquer distância, desde que contidos na rede ativa local. A entidade responsável por indicar qual é o próximo nó ativo a ser visitado, tendo como objetivo atingir o nó destino, é a própria rede ativa através dos dados de como ela está estruturada. Esses dados estão contidos no arquivo ".routes", criado pela própria rede ativa, quando o administrador estrutura a configuração de nós ativos pertencentes à rede ativa segundo a implementação ANTS.

A segunda informação, as regras de composição de cada parâmetro, seja ele de rede, seja de sistema, está contida na própria cápsula ativa de maneira fixa. Para alterar a regra de composição de um parâmetro, é necessário gerar uma nova cápsula.

5.5.5.2 Estratégia de Funcionamento

O valor de cada parâmetro, para o caminho em questão, é calculado à medida que a cápsula visita um novo nó ativo. A cápsula ativa, ao chegar em um determinado nó ativo, busca os valores dos parâmetros de sistema do nó visitado. Esses valores são encontrados no arquivo "param.sys" em um local previamente conhecido pelo serviço. Após lidos os valores do arquivo, para cada parâmetro é executada a regra de composição entre o valor local coletado e o valor já calculado para o caminho até então, e já presente na cápsula ativa. A seguir, pode-se observar um exemplo para o parâmetro ociosidade de memória com os seguintes dados:

- Nó ativo possui valor igual a 0,8 (oitenta por cento de ociosidade);
- Valor existente na cápsula ativa igual a 1 (valor inicial do parâmetro, logo acabou de sair do nó fonte).

Como a regra de composição diz que o menor valor deverá ser o resultado da operação, a cápsula ativa irá seguir adiante com valor 0,8 para o parâmetro ociosidade de memória e não mais 1, que era o valor anterior. Essa mesma lógica vale para cada um dos parâmetros segundo suas próprias regras de composição.

Após coletados e calculados os parâmetros de sistema, na sequência são calculados os valores para os parâmetros de rede, isso no mesmo nó ativo. Nesse momento, é levantada a informação, da própria rede ativa, sobre qual será o próximo nó ativo a ser visitado, caso o nó em questão não seja o nó destino. Tendo essa informação, é mapeado o arquivo no sistema de arquivos local que contém os parâmetros de QoS de rede entre o nó local e o próximo a ser visitado. Coletados os valores dos parâmetros, é aplicada a regra de composição para cada parâmetro de rede, conforme mencionado para os parâmetros de sistema. De posse desses novos valores para o caminho desejado, a cápsula ativa é encaminhada para o próximo nó ativo conforme a lógica do protocolo ping.

A cápsula deve retornar ao nó fonte após atingir o nó destino. Nesse momento, o valor de cada parâmetro de QoS de todo o caminho já está computado.

5.5.5.3 Armazenamento dos Dados de um Caminho

Os valores calculados para o caminho específico são armazenados em sistema de arquivos, em um certo diretório previamente estabelecido; que é compartilhado por todos os nós ativos da rede. Cada caminho possuirá um arquivo nesse diretório, com os valores de seus respectivos parâmetros. O nome desse arquivo deve representar os nós fonte e destino

envolvidos. Uma possibilidade é a concatenação do número IP ANTS fonte com o número IP ANTS destino separados por hífen e com a extensão "*value*".

Ao chegar a cápsula ao nó fonte, com todos os valores dos parâmetros de rede e de sistema, também são atualizados todos os parâmetros de QoS de aplicação e de usuário para o caminho em questão.

Para tal cálculo, a aplicação ativa, responsável por enviar e receber a cápsula, irá acessar o diretório específico do serviço, o qual contém arquivos com as fórmulas de transformação de cada parâmetro de aplicação e de usuário. Todos os arquivos sem extensão, ou seja, arquivos que contêm os dados desejados de aplicação, são lidos. Serão mantidos em memória o ID de identificação da aplicação e o de cada parâmetro da aplicação com sua respectiva fórmula de transformação. Após isso, todos os arquivos com extensão "*user*", ou seja, arquivos que contêm os dados desejados de usuário, também são lidos, e mantidos em memória o ID de identificação e sua respectiva fórmula de transformação.

Aplicação ativa já possui os novos valores dos parâmetros de sistema e de rede, e as fórmulas de transformação para a obtenção de todos os parâmetros de aplicação e de usuário. Nesse momento, o arquivo que contém os valores dos parâmetros para um caminho deve ser atualizado. Para isso, as fórmulas de transformação são aplicadas e, assim, são obtidos os parâmetros de aplicação e de usuário. Os parâmetros de aplicação do caminho podem ser armazenados no arquivo, o qual armazenará todos os valores dos parâmetros do caminho, na sequência dos parâmetros de sistema e rede. Para a separação entre parâmetros de sistema e rede, pode ser utilizada uma linha divisória com um grupo de cinco caracteres "\$" (dolar). Para armazenar os valores dos parâmetros de aplicação e usuário, pode-se usar a mesma técnica de linha divisória, porém com outros caracteres. Nesse arquivo, cada linha armazena um único parâmetro com seus respectivos ID e valor com um separador, que pode ser, por exemplo, o sinal de igual "=".

O cálculo dessas duas últimas classes de parâmetros é realizado a partir das fórmulas de transformação de parâmetros criadas pelo administrador do sistema. As funções do administrador do sistema serão mais bem descritas no item 5.5.6.

Dessa forma, o arquivo de nome igual ao IP ANTS fonte e IP ANTS destino separados por hífen e com a extensão "*value*" pode ser organizado da seguinte forma:

1. Parâmetros de sistema - um parâmetro em cada linha;
2. Uma linha divisória com os caracteres "\$\$\$\$":
3. Parâmetros de rede - um parâmetro em cada linha.

Para cada aplicação existente na rede ativa pode existir um bloco com os seguintes valores:

1. Uma linha divisória com os caracteres "AAAAA-ID-AAAAA", indicando que, a seguir, serão listados os parâmetros de aplicação. O ID será o ID da própria aplicação, o qual é igual ao nome do arquivo que contém os parâmetros da aplicação em questão;
2. Parâmetros de aplicação - um parâmetro em cada linha;
3. Uma linha divisória com os caracteres "UUUUU", indicando que, a seguir, serão listados os parâmetros de usuário;
4. Parâmetros de usuário - um parâmetro em cada linha.

Para melhor entendimento de tal necessidade de organização, é importante ler o item 5.5.6.

5.5.5.4 Forma de Utilização dos Parâmetros

Graças ao fato de todos os parâmetros de QoS serem armazenados em sistema de arquivos, qualquer aplicação que conheça o serviço ativo pode utilizar-se dele.

Para isso, deve-se primeiro requisitar o cálculo dos parâmetros para um determinado caminho, a fim de, então, aguardar a atualização do arquivo que conterá os parâmetros de QoS. A aplicação requisitante de parâmetros de QoS pode, por exemplo, aceitar parâmetros de um determinado caminho, desde que respeitado um tempo máximo de desatualização. Caso os parâmetros não existam ou o tempo máximo de atualização seja extrapolado, é requisitada uma atualização dos parâmetros desse caminho. Essa análise de extrapolação de tempo é feita através da análise da data do arquivo que armazena os parâmetros de QoS do caminho. Caso a última atualização dos dados do caminho seja satisfatória para a aplicação requisitante, ela pode apenas ler os dados do arquivo e trabalhá-los conforme desejado.

5.5.6 Interface Administrativa

A interface administrativa visa oferecer flexibilidade e monitoramento da rede ao administrador do serviço ativo. A flexibilidade é obtida através da administração dos parâmetros de ajuste, de aplicação e de usuário. Já o monitoramento da rede é viabilizado pela seção de análise da qualidade de serviço oferecida pela rede. Essa seção administrativa é uma interface web, e, por isso, acessível de qualquer ponto, desde que devidamente conectado e autorizado.

Ao entrar na seção administrativa, será exibida uma tela com as opções que o administrador tem disponível para gerenciamento. As opções aparecerão em forma de *link* e serão as seguintes:

- Administração dos parâmetros de ajuste;
- Administração dos parâmetros de aplicação e de usuário;
- Análise da QoS oferecida pela rede.

A Figura 5.1 ilustra como foi projetada a tela inicial de administração:

Figura 5.1 - Projeto da Tela inicial da Interface Administrativa

Cada um dos *links* dessa seção administrativa será detalhado nos tópicos seguintes.

5.5.6.1 Administração dos Parâmetros de Ajuste

Tendo escolhido essa seção, o administrador da rede visualizará uma tela conforme o projeto ilustrado na Figura 5.2.

A imagem mostra a interface de administração dos parâmetros de ajuste. No topo, há um título centralizado: "Administração do Serviço Ativo de Provedimento de Parâmetros de QoS". Abaixo do título, há um subtítulo centralizado: "Administração dos parâmetros de ajuste". Abaixo do subtítulo, há dois campos de entrada de texto. O primeiro campo contém o valor "800" e é seguido pelo texto "Período de cálculo das médias". O segundo campo contém o valor "80" e é seguido pelo texto "Tempo entre coleta dos parâmetros". Na base da interface, há dois botões: "Voltar" e "Confirmar".

Figura 5.2 - Projeto da Interface de Administração dos Parâmetros de Ajuste

No modelo exibido na Figura 5.2, é possível observar dois campos de um formulário. Cada um desses campos corresponde a um parâmetro de ajuste conforme mostra a figura. A cada visualização dessa tela, os valores correntes desses parâmetros são carregados e preenchidos nos formulários. Acionando-se o *link* "Voltar", a tela administrativa inicial é carregada. Caso o *link* "Confirmar" seja acionado, é testado se houve alguma alteração de valores. Em caso negativo, nada ocorre com o serviço ativo, e a tela administrativa inicial é carregada novamente.

Ao ocorrer alteração em algum dos parâmetros, seus valores são armazenados adequadamente no arquivo "param.ajuste". Após isso, o parâmetro auxiliar N é recalculado, segundo os novos valores dos parâmetros de ajuste; e caso o novo valor de N seja diferente do atual, o novo valor de N é armazenado em uma cápsula, a cápsula de inicialização. Suas características serão detalhadas no próximo tópico.

Cápsula de Inicialização

Esta cápsula tem como função avisar cada nó da rede ativa de que os parâmetros de ajuste foram alterados e levar o novo valor do parâmetro auxiliar N. Ao chegar essa cápsula a cada nó ativo, os outros parâmetros auxiliares, de rede e de sistema, são atualizados com o valor de inicialização do serviço, recebendo, portanto, o valor zero exceto o parâmetro auxiliar x, que receberá o mesmo valor de N.

A lógica de funcionamento dessa cápsula, ao chegar a um nó ativo, obedece aos seguintes passos a partir do envio da primeira cápsula pelo próprio sistema administrativo:

1. Comparação entre o valor de N local com o valor recebido;
2. Caso sejam iguais nada é realizado;
3. Caso sejam diferentes:
 - 3.1. Todos os valores dos parâmetros locais (auxiliares, rede e sistema) são inicializados;
 - 3.2. Uma cópia da cápsula é enviada a cada vizinho imediato, exceto para aquele do qual recebeu a cápsula.

5.5.6.2 Administração dos Parâmetros de QoS de Aplicação e Usuário

Nessa seção administrativa, serão gerenciados os parâmetros de aplicação e de usuário. Esses parâmetros estão juntos pelo motivos de ambos estarem ligados diretamente a uma determinada aplicação. Uma aplicação, nesse serviço ativo, é uma entidade lógica que une um grupo de parâmetros de aplicação e outro de usuário. Isso ocorre devido ao fato de determinados parâmetros serem semanticamente significativos somente quando associados a uma aplicação, originando assim os parâmetros de aplicação. Da mesma forma, pode-se pensar em relação aos parâmetros de usuários, os quais só farão sentido quando forem apresentados aos usuários de determinada aplicação. Conforme a lógica descrita, após a escolha dessa seção na tela inicial, o administrador da rede visualizará uma tela conforme a Figura 5.3 ilustra.

**Administração do Serviço Ativo de Provisão
de Parâmetros de QoS**
Administração de parâmetros de Aplicação e Usuário

Aplicações existentes:

Selecione:

Videoconferência
 Vídeo sob demanda
 Telerrobótica

Gerenciar Parâmetros de:

<u>Aplicação</u>	<u>Usuário</u>	<u>Remover</u>
<p style="text-align: center;">Nova aplicação:</p> <p>Nome: <input style="width: 150px;" type="text"/></p> <p>ID: <input style="width: 100px;" type="text"/></p> <p style="text-align: center;"><u>Inserir</u></p>		

Voltar

Figura 5.3 - Projeto da Interface de Administração dos Parâmetros de Aplicação e Usuário

Através dessa interface, é possível realizar basicamente quatro operações:

- Voltar para a tela inicial;
- Inserir uma nova aplicação na rede ativa;
- Remover uma aplicação;
- Gerenciar as aplicações já existentes.

Ao acessar o *link* "Voltar", última palavra da parte inferior da tela, o administrador irá voltar para a tela inicial da seção de administração.

Para inserir uma nova aplicação, basta utilizar os objetos *"text box"* do quadrante logo acima do *link* "Voltar". Nesses objetos devem ser digitados o nome da aplicação e seu ID de identificação, o qual deve ser único. Após isso, basta efetuar um clique no *link* "Inserir".

Após essa ação, será analisado um diretório específico do serviço ativo responsável por armazenar todos os dados referentes às aplicações. Esse diretório está na área de atuação do sistema de arquivos distribuídos, ou seja, todos os nós ativos vêem e podem atualizar esse diretório. Essa análise consiste em verificar se existe algum arquivo com o nome igual ao ID digitado pelo administrador. Caso exista, isso significa que já existe uma aplicação com o ID em questão. Sendo assim, é exibida uma mensagem com essa informação ao administrador na própria tela onde foram inseridos os dados. Caso não exista, um arquivo com o nome igual ao

ID digitado é criado e, em sua primeira linha, é armazenado o nome da aplicação. Após isso, retorna-se à mesma tela onde foram inseridos os dados, porém agora com a nova aplicação já disponível no "*select box*" das aplicações existentes.

Caso o desejo seja de remover uma certa aplicação, deve-se selecionar a aplicação desejada, no "*select box*" e acessar o *link* "Remover". Nesse momento, serão apagados os dois arquivos, um com os parâmetros de aplicação e o outro de usuário relacionados à aplicação que foi removida. Em seguida, serão acessados todos os arquivos que contêm dados dos parâmetros de um caminho. Esses arquivos estão em um diretório conhecido e compartilhado por todos os nós ativos. Em cada um dos arquivos desse diretório, será procurada uma linha com uma seqüência de caracteres igual a "AAAAA-ID-AAAAA", sendo que o ID é igual ao ID da aplicação que será removida. Achada essa linha, ela será removida do arquivo juntamente com todas as linhas na seqüência, até ser encontrado o fim do arquivo ou outra linha que comece e termine com cinco caracteres "A". Achada essa linha ou o fim do arquivo, a remoção dentro do arquivo em questão é finalizada.

Para gerenciar os parâmetros de aplicação ou de usuário, o administrador deve selecionar a aplicação que deseja trabalhar. Em seguida, deve-se acessar o *link* "Aplicação" caso a intenção seja gerenciar parâmetros de aplicação; caso contrário, deve-se acessar o *link* "Usuário" para gerenciar parâmetros de usuário.

Gerenciamento de Parâmetros de Aplicação

Tendo escolhido a aplicação e acessado o *link* "Aplicação", a próxima tela a aparecer é ilustrada pela Figura 5.4.

**Administração do Serviço Ativo de Provisão de
Parâmetros de QoS**

Administração de Parâmetros de Aplicação - Aplicação: Videoconferência

Parâmetros existentes:

ID	Nome	Transformação	Remover
TQ	<input type="text" value="Tamanho do Quadro"/>	<input type="text" value="2*MTU"/>
AQ	<input type="text" value="Atraso do Quadro"/>	<input type="text" value="lp+lp*0,01"/>

[Atualizar](#)

Inserir Novos Parâmetros:

Nome:

ID:

Transformação:

☐ [Lembrar IDs de parâmetros
de Rede e Sistema](#)

[Inserir](#)

[Voltar](#)

Figura 5.4 - Projeto da Interface de Administração dos Parâmetros de Aplicação

No projeto de interface acima ilustrado, é possível realizar três ações basicamente:

- Voltar para a tela anterior - para isso basta acessar o *link* "Voltar";
- Inserir um novo parâmetro de aplicação;
- Gerenciar os parâmetros de aplicação já existentes.

Caso seja necessário inserir um novo parâmetro de aplicação, basta trabalhar no quadrilátero logo acima do *link* "Voltar". Para realizar tal inserção, é suficiente digitar nos campos adequados do formulário o nome, o ID de identificação e a fórmula de transformação. Como a fórmula de transformação utiliza-se de parâmetros de rede e de sistema, existe um *link* no mesmo quadrilátero, que, ao ser acessado, irá exibir em um *pop up* a lista de tais parâmetros existentes com os IDs respectivos. Após digitar os dados, basta acessar o *link* "Inserir" na parte inferior do quadrilátero. Nesse momento, é verificado se todos os campos do formulário foram preenchidos; caso algum esteja faltando, uma mensagem de erro é exibida. Sendo preenchido todos os dados, será acessado o arquivo de nome igual ao ID da aplicação em questão. No exemplo ilustrado pela Figura 5.4, é o ID da aplicação videoconferência. Esse arquivo localiza-se em um diretório acessível por qualquer nó ativo da rede através da utilização do sistema de arquivo de rede NFS, já comentado anteriormente. Nesse arquivo, é verificado se existe algum parâmetro dessa aplicação com o mesmo ID. Caso exista, uma mensagem de erro é exibida no próprio formulário com os dados preenchidos objetivando a

mudança do ID. Caso não exista ID igual, o parâmetro é inserido no arquivo e volta-se para a tela ilustrada pela Figura 5.4 com o novo parâmetro já inserido e disponível no quadrilátero superior da tela.

No primeiro quadrilátero com título "Parâmetros Existentes", todos os parâmetros de aplicação são carregados - um por linha - sendo que o nome e a fórmula de transformação são carregados em *"text box"*, podendo assim ser alterados. Nesse quadrilátero, duas tarefas podem ser realizadas:

- Editar parâmetro;
- Remover o parâmetro.

Para a edição, basta alterar os valores desses campos e acessar o *link* "Atualizar". Nesse momento, será verificado se alguma fórmula de transformação foi alterada. Caso nenhuma tenha sido alterada, os nomes dos parâmetros são atualizados no respectivo arquivo, e a tela ilustrada pela Figura 5.4 é exibida com uma mensagem de operação realizada com sucesso. Caso alguma fórmula de transformação tenha sido alterada, as alterações também são gravadas adequadamente, porém outra ação se faz necessária.

A remoção de um parâmetro pode ser realizada através do acesso ao botão da coluna remover, um para cada parâmetro. A cada acesso a esse botão, o parâmetro em questão é removido da lista dos parâmetros da aplicação, e a tela é recarregada já sem o parâmetro. Após a remoção, outra ação se faz necessária.

Após tais operações, serão acessados todos os arquivos que contêm dados dos parâmetros de um caminho. Esses arquivos estão em um diretório conhecido e compartilhado por todos os nós ativos. Em cada um desses arquivos, será procurada uma linha com uma sequência de caracteres igual a "AAAAA-ID-AAAAA", sendo que o ID é igual ao ID da aplicação que teve seu parâmetro alterado ou removido. Achando-se essa linha, será procurada uma outra que comece com o ID do parâmetro removido seguido do sinal de igual. Sendo achada, ela será removida. Essa operação é repetida para cada parâmetro alterado ou removido.

Gerenciamento de Parâmetros de Usuário

Tendo escolhido a aplicação e acessado o *link* "Usuário", a próxima tela a aparecer é ilustrada pela Figura 5.5.

**Administração do Serviço Ativo de Provimento de
Parâmetros de QoS**
Administração de Parâmetros de Usuário - Aplicação: Videoconferência

Parâmetros Existentes:			
ID	Nome	Transformação	Remover
QV	Qualidade do Video	If AQ<1111 then	☒
QA	Qualidade do Audio	If AQ<999 then	☒

[Atualizar](#)

Inserir Novos Parâmetros:

Nome:

ID: Lembrar IDs de parâmetros
de Aplicação

Transformação:

[Inserir](#)

[Voltar](#)

Figura 5.5 - Projeto da Interface de Administração dos Parâmetros de Usuário

Como é possível observar, as Figuras 5.4 e 5.5 são bastante parecidas e, como consequência, oferecem funcionalidades bastante semelhantes. Sendo assim, as funcionalidades básicas também são três:

- Voltar para a tela anterior - para isso basta acessar o *link* "Voltar";
- Inserir um novo parâmetro de usuário;
- Gerenciar os parâmetros de aplicação já existentes.

Para inserir um novo parâmetro de usuário, deve-se utilizar o quadrilátero logo acima do *link* "Voltar". Para tal, é suficiente digitar nos campos adequados do formulário os respectivos dados do parâmetro de usuário. A fórmula de transformação em questão utiliza parâmetros de aplicação como base; logo é disponibilizado um *link* que, ao ser acessado, irá exibir em um *pop up* a lista dos parâmetros da aplicação vigente. Após digitar os dados, basta acessar o *link* "Inserir" na parte inferior. Nesse momento, é verificado se todos os campos do formulário foram preenchidos. Caso algum esteja faltando, uma mensagem de erro é exibida. Sendo preenchidos todos os dados, será acessado o arquivo de nome igual ao ID da aplicação em questão e com extensão "user". Esse arquivo localiza-se, como outros já mencionados, em um diretório acessível por qualquer nó ativo da rede. Nesse arquivo, é verificado se existe algum parâmetro de usuário com mesmo ID. Caso exista, uma mensagem de erro é exibida no próprio formulário com os dados preenchidos para a correção. Caso não exista ID igual, o

parâmetro é inserido no arquivo e volta-se para a tela ilustrada pela Figura 5.5 com o novo parâmetro já inserido.

No quadrilátero com título "Parâmetros existentes", todos os parâmetros de usuários da aplicação são carregados com possibilidade de alteração do nome e fórmula de transformação. Para isso, só é necessário acessar o *link* "Atualizar". Nesse momento, é verificado se alguma fórmula de transformação foi alterada. Caso nenhuma tenha sido alterada, os nomes dos parâmetros são atualizados, e a tela ilustrada pela Figura 5.5 é exibida com uma mensagem de operação realizada com sucesso. Se alguma fórmula de transformação foi modificada, as alterações também são gravadas, porém em uma outra ação, semelhante ao ocorrido com os parâmetros de aplicação é necessária.

Outra funcionalidade disponível é a remoção de um parâmetros. Para isso, basta acessar o botão da coluna remover de cada parâmetro. A cada acesso realizado, é removido o parâmetro relacionado, e a tela é recarregada já sem o parâmetro. Após cada remoção, outra ação se faz necessária, a qual consiste em acessar todos os arquivos que contêm dados dos parâmetros de um caminho. Em cada um desses arquivos, será procurada uma linha com uma seqüência de caracteres igual a "AAAAA-ID-AAAA", sendo que o ID é igual ao ID da aplicação que teve seu parâmetro de usuários alterado ou removido. Sendo achada, será procurada uma linha que consiste de cinco caracteres "U". Se for encontrada, será procurada outra linha que comece com o ID do parâmetro de usuário removido seguido do sinal de igual. Se for achada, ela será removida. Essa operação é repetida para cada parâmetro de usuário alterado ou removido.

Uma funcionalidade existente apenas para os parâmetros de usuário é a possibilidade de construção de intervalos na definição das fórmulas de transformação. Para tal deve-se utilizar o comando "*if then else*" com os parâmetros de aplicação e usuário desejados. Essa característica visa resolver a situação exemplificada na Tabela 5.6. A seguir, é visto um exemplo para o caso do parâmetro Qa com a seguinte característica:

- Qa = bom - se $aff \leq 10000$
- Qa = médio - se $10000 < aff \leq 11000$
- Qa = ruim - caso contrário

A fórmula de transformação ficaria da seguinte forma:

```
If (aff <= 10000)
    then Qa = 'bom'
else if ((aff > 10000) AND (aff <= 11000))
    then Qa = 'médio'
else Qa = 'ruim'
```

5.5.6.3 Análise da QoS oferecida pela rede

Tendo escolhido essa opção, conforme ilustrado pela Figura 5.1, o administrador da rede visualizará uma tela conforme ilustra a Figura 5.6.

Análise da QoS oferecida pela rede
Digite os IPs das máquinas fonte e destino do
caminho a ser analisado:

IP ANTS Fonte: 18.31.12.1

IP ANTS Destino: 18.31.12.4

Voltar Analisar

Figura 5.6 - Projeto da Interface de Análise da QoS Oferecida pela Rede

As opções oferecidas ao administrador são basicamente duas:

- Voltar para a tela anterior - volta-se para a tela ilustrada pela Figura 5.1.
- Analisar o caminho.

Para realizar a análise de um caminho, o qual pode passar por dois ou mais nós ativos, basta digitar os IPs ANTS fonte e destino nos campos adequados do formulário e acessar o *link* "Analisar". Nesse momento, é verificado se os dois campos do formulário foram preenchidos. Caso algum esteja em branco, uma mensagem de erro é exibida. Sendo todos preenchidos, será montada a cápsula ativa de análise. Essa cápsula é endereçada ao IP ANTS fonte e carrega consigo os IPs ANTS fonte e destino digitados pelo administrador. O funcionamento dessa cápsula será mais bem detalhado no próximo tópico.

Com o retorno da cápsula de análise, ela já trará consigo os valores de todos os parâmetros existentes no serviço. São eles:

- Rede;
- Sistema;
- Todos os parâmetros de aplicação existentes no serviço;
- Todos os parâmetros de usuários de cada aplicação existentes no serviço.

Tendo esses dados, o serviço exibe ao administrador um *pop up* com os dados de todos os parâmetros medidos entre o nó ativo fonte e o nó ativo destino, digitados no formulário conforme exemplifica a Figura 5.6. Além desses dados, também são exibidas as fórmulas de transformação de parâmetros de rede e sistema em aplicação, e desse último em parâmetro de usuário.

Cápsula de Análise

Esse cápsula irá realizar basicamente o protocolo ping entre a máquina onde está instalada a interface administrativa do serviço ativo e o IP fonte digitado no formulário da Figura 5.6. Essa cápsula, ao chegar ao nó fonte do caminho a ser analisado, irá executar as seguintes ações:

- Verificar se existe um arquivo de nome igual a IP ANTS fonte e IP ANTS destino separados por hífen e com a extensão "*value*". Esse arquivo será atualizado assim que a coleta dos parâmetros do caminho estiver finalizada. O local onde esse arquivo é armazenado é previamente conhecido e está presente na cápsula de análise. Caso o arquivo exista, sua data de alteração é lida pela cápsula ativa, e o status de atualização é setado. Caso contrário, o status de criação é definido.
- Executar a aplicação ativa incumbida de enviar a cápsula ativa responsável por calcular os parâmetros de um caminho.

Após executar tais ações, a cápsula irá aguardar o acontecimento do primeiro dentre dois possíveis eventos:

- Caso o status da cápsula de análise seja criação, de segundo em segundo a cápsula de análise verifica se o arquivo com os parâmetros do caminho foi criado. Caso o status seja atualização, de segundo em segundo é verificado se a data de alteração do arquivo foi atualizada;
- Término de tempo limite que pode ser, por exemplo, dez segundos. Após esse tempo limite, caso o status seja atualização, o arquivo com os parâmetros do caminho, porém não atualizado, é lido e a cápsula ativa retorna com o status igual a desatualizado. Caso o status seja criação, a cápsula retornará com status *timeout*.

Após o acontecimento de um desses parâmetros, dar-se-á o retorno da cápsula de análise. O status e possivelmente os dados coleados são exibidos para o administrador.

5.6 Modificações Necessárias ao ANTS

Para a implementação do serviço ativo, conforme projetado, a rede ativa ANTS 2.0.0, da forma como é disponibilizada, não é capaz de realizar. Como o software que implementa o ANTS 2.0.0 é totalmente aberto, para fins de utilização acadêmica tais alterações são totalmente factíveis. Existem basicamente duas características que precisam ser modificadas:

- *Timeout* de execução de uma cápsula ativa em um nó ativo;
- Permissão para as cápsulas ativas lerem, criarem e alterarem arquivos nos nós ativos.

O *timeout* de uma cápsula ativa ANTS é bem menor que o valor sugerido no projeto do serviço ativo para a cápsula de análise, nesse caso dez segundos. Dessa forma, se faz necessário alterar as classes que definem esse valor.

Permitir que as cápsulas ativas possam trabalhar, sem nenhuma restrição, com todos os arquivos de cada nó ativo em que ela passar, pode trazer sérios problemas de segurança para toda a rede. Sendo assim, deve-se alterar o ANTS para permitir que as cápsulas trabalhem com liberdade de leitura e escrita com os arquivos de somente alguns diretórios específicos. Esses diretórios são aqueles descritos no projeto do serviço e, além disso, os seus arquivos nunca devem ter o direito de execução apenas de leitura e escrita. Para os arquivos específicos do sistema operacional do nó ativo, como os arquivos do diretório /proc, a cápsula ativa deve ter o direito apenas de leitura.

6 Conclusões

6.1 Considerações Iniciais

Esse projeto teve como base o estudo de algumas tecnologias que visam agregar QoS às implementações que a utilizam. Dentre essas tecnologias, destacam-se ATM [Cri99] [Atm02] [Rod99] [Atm93] [Rob94] e RSVP [Bra97] [Wro97] [Wro97b] [She97] [She97b] [She97c]. As experiências com rede ativa foram realizadas com o pacote de software que implementa a rede ativa ANTS 2.0.0. Tal trabalho, em conjunto com as tecnologias ATM e RSVP, contribuiu para a concepção e projeto do serviço proposto.

6.2 Resultados Alcançados

Como resultados alcançados neste trabalho, destacam-se:

1. Instalação, testes e análise de funcionamento de uma implementação de rede ativa;
2. Projeto de um serviço de rede ativa capaz de gerenciar parâmetros de QoS;
3. Projeto de interface Web e lógica de funcionamento para a administração e análise da rede ativa como um todo e seus elementos;
4. Definição de uma arquitetura capaz de suportar a adição de novos serviços e aplicações sem interromper o funcionamento dos serviços disponíveis;
5. Flexibilidade de criação, remoção e alteração de parâmetros de QoS de aplicação e usuário;
6. Flexibilidade de ajuste das variáveis que definem os parâmetros de QoS de sistema e de rede.

6.3 Contribuições

As principais contribuições deste trabalho são:

1. Identificação e estudos das principais tecnologias que implementam QoS, juntamente com as principais implementações existentes de redes ativas;
2. Documentação do procedimento necessário para a instalação do pacote ANTS 2.0.0, bem como dos passos necessários para configurar a rede ativa;
3. Disponibilização de um grupo de parâmetros de qualidade de serviço para a utilização em outras aplicações presentes na rede;

4. Flexibilidade de criação de parâmetros de aplicação e usuário pertinentes a cada rede que utiliza o serviço ativo proposto;
5. Possibilidade de o administrador da rede monitorar as condições de QoS oferecidas pela rede, sobretudo em períodos de utilização em massa dos seus recursos;
6. Proposta de uma interface amigável para a administração da rede ativa utilizando-se o ambiente Web.

6.4 Limitações Observadas

O serviço proposto traz as seguintes limitações:

- O desempenho da rede ativa ANTS 2.0.0 não é satisfatório o suficiente para a utilização em redes de grande escala no que se refere a número e distância de roteadores;
- A quantidade de parâmetros de sistema e de rede são limitados. Para a implementação de novos parâmetros desse tipo, se faz necessária a reprogramação das cápsulas ativas e suas aplicações responsáveis pelo cálculo de tais parâmetros;
- O trabalho com arquivos para armazenamento dos valores dos parâmetros sobretudo de rede e sistema, contribui consideravelmente para o previsível desempenho não satisfatório do serviço;
- As estimativas dos parâmetros de rede e sistema num dado momento podem não corresponder aos valores de QoS necessários para uma transmissão no momento dessa transmissão. Isso ocorre devido ao fato de a condição de carga da rede poder ser totalmente variável. Num dado instante, a condição da rede pode estar muito favorável, porém no instante seguinte, por uma coincidência, por exemplo, muitas transmissões podem começar ao mesmo tempo. O objetivo dos valores atribuídos a cada parâmetro é obter uma estimativa e não o valor exato de quando será realizada a transmissão.

6.5 Trabalhos Futuros

Como trabalhos futuros pode-se propor:

1. Implementação do serviço proposto, utilizando-se a tecnologia ANTS visando à comprovação da eficácia do serviço ou possíveis ajustes necessários para alcançar o objetivo inicial do trabalho;
2. Estudo e implementação do serviço utilizando-se outras implementações de redes ativas. Exemplo de possíveis implementações interessantes de serem analisadas são Netscript, Smartpackets entre outras;

3. Nova implementação do serviço proposto, porém analisando-se a viabilidade de manter os parâmetros, sobretudo de rede e sistema em uma memória compartilhada, o que poderia contribuir para diminuir problemas de desempenho;
4. Implementações de um serviço de política de reserva de recursos, utilizando-se a tecnologia de redes ativas e tomando-se como base os parâmetros de QoS obtidos por esse trabalho;
5. Estudo da viabilidade da implementação de mecanismos que permitam obter flexibilidade de criação de novos parâmetros de rede e de sistema aos moldes da flexibilidade alcançada por esse trabalho no que se refere a parâmetros de aplicação e usuário;
6. Implementação de uma aplicação, por exemplo distribuição de áudio e vídeo, que utilize os parâmetros de QoS de usuário e de aplicação, permita ao usuário especificar o nível de QoS desejado e obtenha uma resposta da viabilidade ou não do pedido. Essa implementação é interessante, pois todos os parâmetros são abordados bem como o mapeamento entre eles.

6.6 Considerações Finais

O serviço proposto tem como objetivo ilustrar uma possível implementação de administração de parâmetros de QoS por meio da utilização da tecnologia de redes ativas, mais especificamente, utilizando-se como base o pacote ANTS 2.0.0. Tal serviço está preparado para ser implementado e complementado com outros serviços ativos. Dessa forma, é possível obter um ambiente de redes ativas em constante evolução e permitindo, assim, as análises necessárias desse novo paradigma de arquitetura de rede, que ainda se encontra em fase de estudo e evolução.

Referências Bibliográficas

- [Aaa02] Alexander, A. S., Anagnostakis, K. G., Arbaugh, W. A., Keromytis, A. D. and Smith, J. M.. *Sane: The Price of Safety in an Active Network*. http://www.cis.upenn.edu/~switchware/sane_os/ [20/06/02].
- [Aak98] Alexander, D. S., W. A., Keromytis, Keromytis, A. D. and Smith, J. M. *A Secure Active Network Environment Architecture: Realization in Switch-Ware*. IEEE Network, (p. 37-45), junho, 1998.
- [Agg96a] Aggarwal, J.L., Wolf, Yu. *On Optimal Piggyback Merging Policies for Video on demand Systems*. Technical Report RC 20337 (90078), IBM Research Division, fevereiro, 1996.
- [Agg96b] Aggarwal, J.L., Wolf, Yu. *Adaptive Piggybacking Schemes for Video on demand Systems*. Technical Report RC 20635 (91350), IBM Research Division, novembro, 1996.
- [Ale98] Alexander, D. S. *The SwitchWare Active Network Architecture*. IEEE Network, junho, 1998, (p. 29-36).
- [All01] Allan Jeffrey and Ian Wakeman, *A Survey on Semantic Techniques for Active Networks* – [On Line] – <http://www.cogs.susx.ac.uk/projects/safetynet> - [12/07/2001].
- [Ana02] ANANA - *Active Networks Assigned Number Authority* – [On Line] – <http://www.isi.edu/~braden/anana/> - [15/02/2002].
- [And91] Anderson, D.P., Herrtwich R.G., and C. Schaefer. *SRP: A Resource Reservation Protocol for Guaranteed Performance Communication in the Internet*. Internal Report , University of California at Berkeley, 1991.
- [Ane97] Alexander, S., Gunter, C., Keromytis, A., Minden, G., Wetherall, D., Braden, B. and Jackson, A. *The Active Network Encapsulation Protocol (ANEP)*, Active Networks Group Draft, julho, 1997.
- [ANT01] *ANTS* – [On Line] – <http://www.cs.washington.edu/research/networking/ants/> [13/12/2001].
- [ANT02] *ANTS* – [On Line] – <http://anet.it.rit.edu/ANTS> [14/01/2002].
- [Arg98] Argon, E., Arrowpoint, R. and Rajagopalan, B. *A Framework for QoS-based Routing in the Internet*. Relatório Técnico. NEC, H. Sandick, Bay Networks, agosto, 1998.
- [Atm02] ATM Forum/af-uni-0010.002. *ATM User -Network Interface Specification 3.1*. Aprovado em 1994. – [On Line] – <ftp://ftp.atmforum.com/pub/approved-specs/af-uni-0010.002.ps> [10/04/2002].
- [Atm93] ATM Forum. *ATM User-Network Interface Specification Version 2.4*. Agosto, 1993.

[Atv00] *Advanced Television Enhancement Forum - Enhanced Content Specification*, 2000 – [On Line] – http://www.atvtf.com/library/spec1_1a.html - [11/02/2001].

[Bec00] Becker, M. S. *Redes Ativas*. Relatório técnico. Unisinos, julho, 2000.

[Bha97] Bhattacharjee, S., Calvert, K. L. and Zegura, E. W., *An Architecture for Active Networking*, Proc. IEEE INFOCOM '97, abril, 1997.

[Bla00] Blackketter, D. J. – *The Local Identifier lid URI Scheme* – [On Line] – <http://ietf.org/internet-drafts/draft-blackketter-lid-00.txt> - [21/12/2000].

[Bra94] Braden, R., Clark, D. D. and Shenker, S. *Integrated Services in the Internet Architecture: an Overview*. Internet RFC 1633, junho, 1994.

[Bra97] Braden, B., Estrin, D., Herzog, S. and S. Jamin. *Resource ReSerVation Protocol, RSVP - Version 1 Functional Specification*. Internet RFC 2205, setembro, 1997.

[Bus96] Busse, I. D. B. and Schulzrinne, H. *Dynamic QoS Control of Multimedia Applications based on RTP*. Computer Communications, [s.l.], vol 19, (p. 49-58), janeiro, 1996.

[Cal98] Calvert, K., Zegura, E. W., Bhattacharjee, S., and Sterbenz, J. *Directions in Active Networks*. IEEE Commun. Magazine 36, (p. 72-78), outubro, 1998.

[Cam02] *Linguagem Caml* – [On Line] – <http://caml.inria.fr/> – [On Line] – [17/02/2002].

[Cam94] Campbell, A., Coulson, G., and D. Hutchison. *Flow Management. Internal Report No. MPG-94-09*, Department of Computing, Lancaster University, Lancaster LA1 4YR.

[Cam99] Campbell, A.T. and Meer, H. G. and Kounavis, M. E. and Miki, K. and Vicente, J. B. and Villela, D. *A Survey of Programmable Networks*. Computer Communications Review, Vol. 29 No. 2, abril 1999.

[Can01] *CANEs: Composable Active Network Elements* – [On Line] – <http://www.cc.gatech.edu/projects/canes/> [17/07/2001].

[Cer74] Cerf, V. and R. Kahn. *A Protocol for Packet Network Intercommunication*, IEEE Trans. on Comm., Vol. Com-22, No. 5, maio, 1974.

[Cha00] Chakravorty, R., Kar, S., Farjami, P. *End-to-End Internet Quality of Service: An Overview of Issues, Architectures and Frameworks*. Proceeding ICIT 2000, dezembro, 2000.

[Cla88] Clark, D. *The Design Philosophy of the DARPA Internet Protocols*, ACM SIGCOMM '88, agosto, 1988.

[Con01] Conectiva®. *Apostila do curso: "Treinamento avançado - Administração do Linux I"*. 16/04/2001.

[Cos01] Costa, L. H. M. K. and Duarte, O. C. M. B. *Roteamento Baseado em Requisitos de Qualidade de Serviço* – [On Line] – www.gta.ufrj.br [23/02/2001].

[Cri99] Cristiano, D. S. – *Experiências com Sincronização de Streams de Vídeo Usando Transporte ATM*. Mini-dissertação de Mestrado ICMC/USP, 1999.

[Css02] Cascading Style Sheets. Level 1.0 - <http://www.w3.org/TR/REC-CSS1> – [On Line] – [07/05/2002].

[Dar01] DARPA Active Network - <http://www.darpa.mil/ito/research/anets> - [22/01/2001].

[Dec98] Decasper, D. and Plattner, B., DAN: *Distributed Code Caching for Active Networks*, Proc. IEEE INFOCOM '98, San Francisco, CA, 29 de março-2 de abril, 1998.

[Del00] Delfino, G. M., Santos F^o, J. V., Roman, R. C., Leão, J. L. S. and Pedroza, A. C. P. *An Integrated Toolset for the Design of Protocols with QoS Requirements (ProtCAD/RT)*. Protocols for Multimedia Systems - PROMS'2000 Conference, Krakow, Poland, outubro, 2000.

[Dom02] W3C Working Draft - *Document Object Model (DOM) Level 1 Specification (Second Edition) Versão 1.0* – [On Line] – <http://www.w3.org/TR/2000/WD-DOM-Level-1-20000929> – [07/05/2002].

[Dua97] Duarte J. E. P., Mansfield, G., N. T. and NOGUCHI, S. *Non-Broadcast Network Fault Monitoring Based on System-Level Diagnosis*, Proc. IEEE/IFIP IM'97, (p. 597-609), San Diego, maio, 1997.

[ECM02] *ECMAScript Language Specification* – [On Line] – <http://www.ecma.ch/ecma1/STAND/standard.htm> [07/05/2002].

[Far00] Faria G. B. *Tendências na Implementação de Televisão Interativa. Qualificação de Mestrado*. ICMC/USP, 2000.

[Fin00] Finseth, C., Thomas, G.– *Guide to TV Broadcast URLs* – [On Line] – <http://ietf.org/internet-drafts/draft-finseth-guide-01.txt> - [21/04/2002].

[Fon97] Fonseca, N.L.S., Franco, C.M.R., Schaffa, F. *Projeto de Redes para o Suporte a Aplicações "Distributed Home Theatre"*. Anais do 15º SBRC, maio, 1997.

[Fuk98] Fukuada, K., Wakamiya, N., Murata, M. e Miyahara, H. *Effective Algorithms for Multicast Video Transport to Meet Various QoS Requirements*. IEEE Trans. Commun., vol. E81-B, no. 8, (p. 1599-1607), agosto, 1998.

[Geo97] George C. Necula, *Proof-Carrying Code*, Proc. 24th Annual ACM SIGPLAN-SIGACT Symp. on Principles of Programming Languages, ACM Press, 1997.

[Gom99] Gomes, A. T. A. *Um Framework para Provisão de Qos em Ambientes Genéricos de Processamento e Comunicação*. Dissertação de Mestrado. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de informática, 1999.

[Gon00] Gonçalves, P. *Um serviço ativo de distribuição de vídeo multiponto. Tese de mestrado*, UFRJ, março, 2000.

[Har99] Hartman, J. J. *Joust: A Platform for Liquid Software*. IEEE Computer 32, (p.50-56), abril, 1999.

[Hic98] Hicks, M. *PLAN: A Packet Language for Active Networks*. In International Conference on Functional Programming, (p. 86-93), 1998.

[Hua97] Huard, J.-f., Lazar, A. A. *On End-to-End QoS Mapping*. Proc. IFIP Fifth International Workshop on Quality of Service (IWQoS'97), New York, maio, 1997.

[Hut94] Hutchison, D., Coulson G., Campbell, A., and G. Blair. *Quality of Service Management in Distributed Systems*. Internal Report No. MPG-94-02 Department of Computing, Lancaster University, Lancaster LA1 4YR, 1994.

[Htm02] *HTML - 4.0* – [On Line] – <http://www.w3.org/TR/html4/> [07/05/2002].

[Int01] Qualidade de Serviço na Internet - [on line] - <http://www.lcmi.ufsc.br/redes/redes99/renato/SemI/> - [29/12/2001].

[Itu94] ITU-T Recommendation E-800. *Quality of service and dependability vocabulary*. Geneva, Suíça, novembro, 1994.

[Jac99] Jacobson, V., Nichols, K. and Poduri, K. *An Expedited Forwarding PHB*. Internet RFC 2598, junho, 1999.

[Jam98] James D. M. *Practical Computer Network Analysis and Design*, Morgan Kaufmann Series in Networking, 1998.

[Jan01] *Janos Java NodeOS* – [On Line] – <http://www.cs.utah.edu/flux/janos/> - [11/01/2002].

[Jan99] *Designing a Programming Language for Active Networks*, – [On Line] – <http://www.cogs.susx.ac.uk/projects/safetynet>, 1999 - [21/12/2000].

[Job99] Joberto Martins. *Redes Corporativas Multiserviço - Caracterização das aplicações e Parâmetros Básicos de Operação*, em – [On Line] – <http://www.jsmnet.com/> - [12/11/2000].

[Jou02] Joust – [On Line] – <http://www.cs.arizona.edu/scout/joust.html> [16/02/2002].

[Jun97] Junqueira, F. P., Costa, L. H. M. K., and Duarte, O. C. M. B. *Uma Arquitetura de Roteamento com Qualidade de Serviço Baseada em Redes Ativas*, II Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos, SFBSID'97, (p. 257-268), Fortaleza, novembro, 1997.

[Jun99] Junqueira, F., P. *Comunicação multiponto confiável para estações móveis em redes ativas*. Tese de mestrado, UFRJ, março, 2000.

[Kuo98] Kuo, F. F., Effelsberg, W. and Garcia-Luna-Aceves, J.J.. *Multimedia Communications - Protocols and Applications*. New Jersey: Prentice Hall. 1998.

[Leg98] Legedza, U., Wetherall, D. J., and Guttag, J. V. *Improving the Performance of Distributed Applications Using Active Networks*. In IEEE INFOCOM'98, (p. 590-599), São Francisco CA, EUA, abril, 1998.

[LIA97] Liao, W., Li, V. O. K. *Distributed Multimedia Systems*. Proceedings of IEEE. New York, vol. 85, nº 7, (p. 1061-1108), julho, 1997.

[Mag97] Magalhães, M., Pinto, A. de S., Figueiredo, A. *Serviços Integrados e RSVP sobre ATM*, Departamento de Engenharia da Computação e Automação Industrial, Unicamp, dezembro, 1997.

[Mar99] Martins, J. *Qualidade de Serviço em Redes IP - Princípios básicos. Parâmetros e Mecanismos*. ITTELCON - JSMNet Networking Reviews - Vol. 1 - Nº 1, setembro, 1999.

[Met99] Metz, C. *IP QoS: Traveling in First Class on the Internet*. IEEE Internet Computing, (p. 95-99), maio/junho 1999.

[Mic98] Michael S. Greenberg, Jennifer C. Byington, and David G. Harper, *Mobile Agents and Security*. IEEE Commun. Mag., vol. 36, no. 7, julho, 1998.

[Nah94] Nahrstedt, K., Smith, J. A *Service Kernel for Multimedia Endstations*. Proceedings IWACA '94, Heidelberg, Germany, Setembro, 1994.

[Nah95] Nahrstedt, K., Steinmetz, R. *Resource Management in Networked Multimedia Systems*. IEEE Computer, (p. 52-63), maio, 1995.

[Ope01] *Open Signalling Working Group*, - [On Line] - <http://www.comet.columbia.edu/opensig/> [01/07/2001].

[P1520] Biswas, J. *The IEEE P1520 Standards Initiative for Programmable Network Interfaces*. IEEE Commun. Magazine Special Issue on Programmable Network, outubro, 1998.

[Pla02] *Plan* - [On Line] - <http://www.cis.upenn.edu/~switchware/PLAN/> [19/02/2002].

[Pso99] Psounis, K. *Active networks: Applications, Security, Safety and architecture*. IEEE Commun. Surveys, março, 1999.

[Rob94] Robin, P., Campbell, A., Coulson, G., Blair, G., and M. Papathomas. *Implementing a QoS Controlled ATM Based Communication System in Chorus*. Internal Report No. MPG-94-05 Department of Computing, Lancaster University, Lancaster LA1 4YR, março, 1994.

[Roc00] Rochol, J., Bastos, E. L., Mouriac, H. and Liane, C. *Discriminação de Fluxos de Tráfego para Obtenção de parâmetros de UPC*. Relatório Técnico Parcial - Projeto Metropoa, Universidade Federal do Rio Grande do Sul, junho, 2000.

[Rod99] Rodrigues, C. A., *Distribuição de Dados IP sobre ATM com Reserva de Qualidade de Serviço*, Dissertação de Mestrado - COPPE/UFRJ, Rio de Janeiro, 1999.

[San99] Santos, A. P. S. *Qualidade de Serviço na Internet*, 1999, - [On Line] - <http://www.rnp.br/newsgen/9911/qos.shtml#p0>. [26/03/2001].

[Sch01] Schulzrinne, H. *RSVP home page* - [On Line] <http://www.cs.columbia.edu/~hgd/internet/rsvp/rsvp.html> [22/07/2001].

[Sco02] *Scout* - [On Line] - <http://www.cs.arizona.edu/scout/> [10/02/2002].

[Sco98] Scott, D. *A Secure Active Network Environment Architecture*, IEEE Network, special issue on Active and Programmable Networks, maio/junho 1998, vol. 12, no. 3.

[She97] Shenker, S., Partridge, C., and R Guerin. *Specification of Guaranteed Quality of Service*. RFC 2212, setembro, 1997.

[She97b] Shenker, S., and J. Wroclawski. *General Characterisation Parameters for Integrated Service Network Elements*. RFC 2215, setembro, 1997.

[She97c] Shenker, S., and J. Wroclawski. *Network Element QoS Control Service Specification Template*. RFC 2216, September 1997.

[Sil98] Da silva, S., Florissi, D. and Yemini, Y. *Netscript: A Language-Based Approach to Active Networks*. Technical Report, Computer Science Dept., Columbia University, janeiro, 1998.

[Slu91] Sluman, C. *Quality of Service in Distributed Systems*. BSI/IST21/-/1/5:33, British Standards Institution, UK, outubro, 1991.

[Sma01] *SmartPackets* - [On Line] - <http://www.ir.bbn.com/projects/spkts/smpkts-index.html> [10/07/2001].

[Smi99] Smith, J. *Activating Networks: A Progress Report*. IEEE Computer 32, (p. 32-41), abril, 1999.

[Spr02] *Sprocket* - [On Line] - <http://www.ir.bbn.com/projects/spkts/smpkts-index.html> [09/02/2002].

[Swi02] *Plan* - [On Line] - <http://www.cis.upenn.edu/~switchware/> [19/02/2002].

[Szt00] Sztajnberg, A. and Loques, O. G. *Bringing QoS Specifications to the Architectural Level*. ECOOP'2000 Workshop on QoS for Distributed Object Systems, France, junho, 2000.

[Tan96] Tanenbaum, A. S. *Computer Networks*, 3rd Edition. Upper Saddle River, New Jersey, USA - 1996.

[Taw93] Tawbi, W., Horn, F., Horlait, E. and Stefani, J. B. *Video Compression Standards and Quality of Service*. The Computer Journal, vol. 36, n. 1, (p. 43-54), 1993.

[Tei00] Teixeira, J. H., Moraes, L. F. M. and Teixeira, S. R. *Uma Proposta Para o Emprego da Tecnologia de Redes Ativas no Gerenciamento de Redes*. - relatório técnico - COPPE/UFRJ, Rio de Janeiro, 2000.

[Ten96] Tennenhouse, D. and Wetherall, D. L. *The ACTIVE IP Option*. SIGOPS European Workshop, 1996.

[Ten97] Tennenhouse, D. *A Survey of Active Network Research*. IEEE Commun. Magazine, January 1997.

[Tsc97] J. Vitek and Chr. Tschudin. *The Messenger Environment M0 - A Condensed Description*. In: *Mobile Object Systems - Towards the Programmable Internet*, (p. 149-156), Springer LNCS 1222, 1997.

[Tva00a] *TV Anytime Forum. Requirements Series:R1 : The TV-Anytime Environment (Informative)* - (Agosto de 2000) - [On Line] - <http://www.tv-anytime.org/> - [11/03/2001].

[Tva00b] *TV Anytime Forum. - Requirements Series:R2 : The System Description (Informative)* - (Abril de 2000) - [On Line] - <http://www.tv-anytime.org/> - [11/03/2001].

[Tva00c] *TV Anytime Forum. - Requirements Series:R3 : Metadata Requirements (Normative)* - (Abril de 2000) - [On Line] - <http://www.tv-anytime.org/> - [11/03/2001].

[Tva00d] *TV Anytime Forum. - Requirements Series:R4 : Content Referencing (Normative)* - (Abril de 2000) - [On Line] - <http://www.tv-anytime.org/> - [11/03/2001].

[Tva00e] *TV Anytime Forum. - Requirements Series:R5 : Rights Management (Normative)* - (Abril de 2000) - [On Line] - <http://www.tv-anytime.org/> - [12/03/2001].

[Tva00f] *TV Anytime Forum. - Specification Series:S4 : Content Referencing (Normative)* - (Agosto de 2000) - [On Line] - <http://www.tv-anytime.org/> - [12/03/2001].

[Uri02] *Naming and Addressing: URIs, URLs, ...* - [On Line] - <http://www.w3.org/TR/uri-clarification/> [07/05/2002].

[Ver02] Verdi, F. L. and Madeira, E. R. M. *MS2VAN - Um Sistema para Gerência de Serviços em Redes Ativas Virtuais*. XX Simpósio Brasileiro de Redes de Computadores (SBRC2002), Búzios, RJ, maio 2002.

[Vic00] Vickres, M., Zigmond, D. *Uniform Resource Identifiers for Television Broadcasts* - [On Line] - <http://ftp.ics.uci.edu/pub/ictf/uri/draft-zigmond-tv-uri-03.txt> - [21/04/2002].

[Vid01] Vidal, M. T. L. and Duarte, O. C. M. B. *Qualidade de Serviço até o Nível de Aplicação* - [On Line] - www.gta.ufrj.br [15/02/2001].

[Wet98] Wetherall, D., Legedza, U., and Guttag, J. *Introducing New Internet Services: Why and How*. IEEE Network, junho, 1998.

[Wet98b] Wetherall, D. J., Guttag, J. V., and Tennenhouse, D. L. *ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols*. IEEE OPENARCII'98, São Francisco, CA, abril, 1998.

[Wet99] Wetherall, D. J., Guttag, J., and Tennenhouse, D. *ANTS: Network Services Without the Red Tape*. IEEE Computer 32, (p. 42-48), abril, 1999.

[Wro97] Wroclawski, J. *The Use of RSVP with Integrated Services*. RFC 2210, setembro, 1997.

[Wro97b] Wroclawski, J. *Specification of the Controlled Load Quality of Service*. RFC 2211, setembro, 1997.

[Xia99] Xiao, X. and Ni, L. M. *Internet QoS: the Big Picture*, Department of Computer Science, Michigan State University IEEE Network, vol 13, no. 2, (p. 1-13), Março/Abril de 1999.

[Yem96] Yemini, Y., and da Silva, S. *Towards Programmable Networks*. In IFIP/IEEE International Workshop on Distributed Systems: Operations and Management. L'Aquila, Itália, outubro, 1996.

[Yam99] Yamazaki, T., Matsuda J. *On QoS Mapping in Adaptive QoS Management for Distributed Applications*. ATR Adaptive Communications Research Laboratories, Kyoto Japão, 1999.

[Zha93] Zhang, L., Deering, S., Estin, D., Shenker S. and D. Zappala. *A New Resource ReSerVation Protocol* – [On Line] – parcftp.xerox.com/transient/rsvp.ps, agosto, 1993.

[Wro97] Wroclawski, J. *Specification of the Controlled-Load Network Element Service*, Internet RFC 2211, setembro, 1997.

Apêndice: ANTS - Active Network Transport System

A.1 Introdução

O ANTS (*Active Node Transfer System*) [Leg98] [Wet98] [Wet98b] [Wet99] é um *toolkit* desenvolvido no MIT utilizando a linguagem java para construção de uma rede ativa e suas aplicações. A versão utilizada como base dos trabalhos é a 2.0.0. Os pacotes convencionais que transportam dados do usuário são substituídos por cápsulas contendo, além dos dados, uma descrição do processamento a ser executado na rede. Esta descrição é utilizada a cada nó ativo para identificar o procedimento a ser executado e que caracteriza a cápsula.

Nesse sistema, um protocolo é definido por um conjunto de cápsulas. Um protocolo e todo o código associado é transportado entre nós através de solicitações de envio de protocolo. Para isso, existe um mecanismo de distribuição de código. Quando uma cápsula chega a um nó ativo, o identificador do procedimento da cápsula é lido e o procedimento buscado em um cache. Se o conjunto de procedimentos do protocolo não for encontrado, um pedido é feito ao nó anterior. Desta forma, os procedimentos (código fonte) de um protocolo são difundidos pelo caminho onde são utilizados.

Para a programação dos procedimentos, são disponibilizadas primitivas que fornecem acesso aos recursos e às facilidades disponíveis nos nós. Entre essas primitivas, encontram-se as utilizadas para a manipulação do armazenamento não-transiente (*soft-state*) e para o controle de cápsulas. O *soft-state* consiste em um espaço de memória não-transiente, isto é, ele persiste à execução de uma cápsula. Após a execução de uma cápsula são guardadas informações de um fluxo por um período de tempo determinado, chamado de tempo de expiração. Assim, as cápsulas de um mesmo fluxo podem colocar e alterar informações necessárias para a execução dos procedimentos. Tais informações podem ser úteis, por exemplo, para decidir o próximo nó para o qual uma cápsula deve ser enviada, ou ainda se uma cápsula deve ser descartada ou replicada para várias interfaces.

A.2 Instalação e Configuração

O *toolkit* ANTS possui compatibilidade com as seguintes tecnologias:

- JDK 1.1.x ou mais novo.
- FreeBSD, Linux ou Solaris.

Para a execução dos trabalhos foram instalados em duas máquinas o Java (JDK 1.3.1) com SO linux. Em uma delas foi utilizada a distribuição Red Hat 6.2 e em outra a distribuição Mandrake 5.1. A existência da linguagem Java já instalada é pré-requisito para a instalação do ANTS. Após descompactar o arquivo ANTS deve-se ir até a raiz da estrutura de diretórios, no caso o diretório ANTS-2.0.0. Nesse diretório deve-se executar os seguintes comandos[Ant01]:

1. *./configure*
2. *make all*
3. *make check*

Esses passos instalam e configuram o ANTS em uma máquina. Terminado tais passos deve-se rodar um *script* de análise, o *sanitycheck*. Esse *script* é responsável por verificar se todas as variáveis de ambiente e a JVM (*Java Virtual Machine*) estão funcionando corretamente. Esse *script* encontra-se dentro do diretório *script*, que por sua vez é um subdiretório de ANTS-2.0.0. Caso a resposta seja positiva, o ANTS está instalado.

É possível realizar *debugging* durante a execução de uma aplicação ANTS. Basta executar o JVM com um parâmetro para exibir na tela mensagens de *debug*. Os parâmetros possíveis podem ser encontrados em <ants-2.0.0/src/ants/core/Trace.java>. Pode-se utilizar o parâmetro "ALL", o qual define que todos os parâmetros são desejados. Para tal, um parâmetro deve estar presente ao executar o comando java para iniciar a JVM. Esse parâmetro é o -Dants.debugFlags=ALL.

Para começar a desenvolver as próprias aplicações, ou até modificar o ANTS, é conveniente criar uma instância (cópia) da versão original. Para isso existe um *script* dentro do diretório ants-2.0.0, logo deve-se executar os seguintes comandos [Ant02]:

1. *cd ants-2.0.0*
2. *mkdir developAnts*
3. *cd developAnts*
4. *../configure*
5. *make*

Após esses passos, dentro do diretório developAnts existirá uma cópia dos arquivos necessária para a execução do ANTS.

A.3 Estrutura de Diretórios

Com a instalação é criada uma estrutura de diretórios com os vários componentes que compõem o ANTS. Na Figura A.1 é possível visualizar essa estrutura.

Figura A.1 - Estrutura de Diretórios - ANTS [ANT02]

O diretório ants-2.0.0 é a raiz de toda instalação. A seguir pode-se observar a lista dos subdiretórios existentes e suas principais funções:

- autoconf - contém arquivos usados pelo *script* de configuração "*configure*" durante o processo de instalação e configuração;
- doc - possui toda a documentação, incluindo o javadoc da API (*reference*) e alguns *papers* (*papers*) importantes para o entendimento da solução ANTS;
- lib - armazena todos os arquivos *.zip* e *.jar* necessários para emular o Janos Java NodeOS sobre o SO linux. Sobre essa plataforma roda-se o ANTS;
- scripts - contém vários *shell scripts* para executar e configurar os exemplos de aplicações vindos juntos no pacote de instalação;
- src - contém o código fonte do ANTS, aplicações exemplo e alguns testes.

A.4 A Arquitetura das Aplicações ANTS

Faz parte da arquitetura ANTS, uma emulação de um NodeOS (*Node Operating Systems*) chamado *Janos Java NodeOS* [Jan01]. O Janos foi criado pela faculdade de Utah nos Estados Unidos e inserido na tecnologia ANTS a partir da versão 2.0.0. Essa emulação é obtida através da inserção de pacotes java (*.jar* e *.zip*), vindos no pacote ANTS 2.0.0, na variável de ambiente CLASSPATH. Essa inserção proporcionou um alívio de carga para o ambiente de execução ANTS (AE ANTS), a medida que todas as rotinas referentes aos nós de uma rede ativa são tratados pelo Janos. Com a utilização dessa tecnologia foi possível obter ganhos em duas principais questões:

- Gerenciamento de alocação de recursos.
- Segurança - pois, possibilita um tratamento especial nessa questão caso haja necessidade para uma determinada aplicação.

Para melhor compreender como funciona todo o mecanismo de uma aplicação ANTS, será utilizado um exemplo de uma aplicação implementada nessa arquitetura. A aplicação escolhida é o Ping. Sua função é muito parecida com a aplicação já consagrada e muito utilizada, visitar uma determinada máquina e retornar a máquina origem, porém todo o mecanismo é implementado utilizando a tecnologia ANTS. Nesse primeiro exemplo, apenas é simulado uma rede de computadores em uma mesma máquina. Para isso são utilizadas algumas portas UDP. Cada porta faz o papel de uma determinada máquina nessa simulação. A configuração do ambiente virtual pode ser observada na Figura A.2.

Figura A.2 - Topologia da Rede Virtual para Aplicação Ping - [ANT02]

Nessa topologia, os endereços IP são endereços virtuais. Toda a comunicação existente é realizada via UDP através do *localhost* (127.0.0.1), distinguindo apenas pelas portas que são 8004, 8005 e 8006. Em cada uma destas portas estará executando uma *thread* diferente, a qual é a responsável por receber o fluxo de dados que estaria sendo recebido por um nó ativo real. Essas configurações são controladas pela diretiva *channel* existente no arquivo de configuração de cada aplicação ativa. Tal arquivo será melhor explicado posteriormente.

Para executar a aplicação ping é preciso rodar o *script* `ping.start` do diretório *scripts*. A partir de então, esse *script* irá executar outros processos e carregar alguns arquivos de configuração que forem necessários. Todo esse processo é descrito pela Figura A.3.

Figura A.3 - Processo de Funcionamento da Aplicação Ping - [ANT02]

Primeiramente será descrito o processo como um todo para obter um entendimento global desse funcionamento. Após isso será descrito em mais detalhes cada uma das partes que compõem todo esse processo. As setas pontilhadas representam leituras de arquivos de configuração, já as setas em negrito representam chamadas de processos ou execução de *scripts*. De uma maneira global o funcionamento da aplicação ping acontece obedecendo os seguintes passos [Ant02]:

1. Ao digitar `/ping.start`, esse *script* é executado. Nesse momento são configuradas algumas variáveis de ambiente e em seguida executado o *script* `nodesetup.sh`.
2. O *script* `nodesetup.sh` cria até 10 nós ativos virtuais, dependendo das variáveis do *script* `ping.start`. Para a topologia descrita na Figura A.2 são 3 nós ativos. Para cada um dos nós ativos é criado um objeto `ConfigurationManager`.
3. O `ConfigurationManager` lê as informações relativas a topologia e roteamento provenientes dos arquivos `ping.config` e `ping.routes`. Todos os nós ativos, virtuais ou reais, lêem os mesmos arquivos `.config` e `.routes`. Cada nó retira desses arquivos as informações que lhes são importantes através do `ConfigurationManager`.

Cada aplicação, como o ping, pode gerar o arquivo `.routes` a partir do `.config`. Para isso existe o *script* `makeroutes`, que será melhor explicado na seção A.4.3.2. A seguir serão descritos com mais detalhes cada arquivo de configuração de uma aplicação

A.4.1 Ping.start

A função desse *script* é atribuir valores para algumas variáveis de ambiente antes de executar o *script* nodesetup.sh. A seguir é comentado linha a linha o que significa os comandos presentes nesse *script* [Ant02].

```
#!/bin/sh
# Ping
# Copyright (c) 2001 University of Utah and the Flux Group.
# All rights reserved.
# Permission to use, copy, modify, and distribute this file
# for any purpose with or without restriction is hereby
# granted.
```

Comentário inicial do arquivo com autores e restrições de utilização

```
CONFIG_FILE=ping.config
```

Indicação do nome de arquivo de configuração usado pela aplicação

```
NODE1_IP=18.31.12.1
NODE1_LABEL="Ping Source"
```

Endereço IP do nó 1 e nome a ser utilizado no terminal do nó

```
NODE2_IP=18.31.12.2
NODE2_LABEL="Middle Router"
```

Endereço IP do nó 2 e nome a ser utilizado no terminal do nó

```
NODE3_IP=18.31.12.3
NODE3_LABEL="Target Router"
```

Endereço IP do nó 3 e nome a ser utilizado no terminal do nó

```
NODECT=3
```

Quantidades de nós existentes na simulação

```
./nodesetup.sh
```

Execução do *script* nodesetup.sh. Vale ressaltar que todas as variáveis anteriormente inicializadas estarão disponíveis para o nodesetup.sh.. Isso graças a forma como esse *script* é executado. Usando a forma ". filename" o *script* é executado no *shell* atual mantendo todas as variáveis e seus valores.

```
#eof
```

A.4.2 Nodesetup.sh

Esse *script* é essencial para a simulação de vários terminais em uma mesma máquina. O nodesetup.sh é o responsável por criar um terminal para cada nó ativo, exceto o primeiro, pois, parte-se do princípio que a execução da aplicação é feita também em um terminal, logo esse terminal irá hospedar o primeiro nó ativo.

Como esse *script* é mais extenso que o ping.start, serão destacados do texto trechos de código para as explicações. Nenhum trecho será omitido, apenas não estarão fisicamente juntos como na explicação do código anterior.

```
if test "$NODECT" -lt 2 -o "$NODECT" -gt 10; then
    echo ".nodesetup.sh Requires between 2 and 10 nodes be configured."
    exit 1
fi
```

As primeiras linhas são responsáveis por algumas verificações de integridade. Primeiramente é analisado se existe o arquivo de configuração indicado no *script* ping.start. Também é verificado se existem pelo menos dois nós indicados no arquivo ping.start. Caso uma das condições não sejam satisfeitas, a execução é interrompida, pois sem existir ao menos dois nós, não faz sentido executar a aplicação ping. Caso não exista arquivo de configuração, não será possível continuar a execução [Ant02].

```
# Default to zero nodes
NODECT=${NODECT:-0}
```

} Caso a variável NODECT não estiver inicializada ela receberá o valor 0.

```
if test "$NODECT" -lt 2 -o "$NODECT" -gt 10; then
    echo "/nodesetup.sh Requires between 2 and 10 nodes be configured."
    exit 1
fi
```

Verifica-se se a variável NODECT tem valores entre 2 e 10. Para um valor menor que 2, a aplicação não faz sentido. O limite de 10 é arbitrário e dado pelo próprio *script* nodesetup.sh. Valor maior que 10 tende a gerar problemas de performance na máquina.

```
# Source defaultEnv scripts if it exists
if test -f ./defaultEnv; then
    ./defaultEnv
fi
```

} Caso exista o *script* defaultEnv, este será executado. Este *script* define as variáveis de ambiente JVM_DEFAULT, XTERM_DEFAULT, PS_DEFAULT e KILL_DEFAULT. No próximo tópico será melhor explicado.

```
# Define vars from defaultEnv values, or real defaults.
JVM=${JVM:-${JVM_DEFAULT:-java}}
XTERM=${XTERM:-${XTERM_DEFAULT:-"xterm -sb -si 4000"}}
SH=${SH:-"sh"}
PS=${PS:-${PS_DEFAULT:-"ps xwww"}}
KILL=${KILL:-${KILL_DEFAULT:-"/bin/kill"}}
```

Nesse momento são atribuídos valores as variáveis de ambientes JVM, XTERM, SH, PS, e KILL, a partir dos valores das variáveis vindas do defaultEnv. Estas variáveis serão comandos a serem executados em cada aplicação teste [Ant02].

```
## Default JVMFLAGS to "
#JVMFLAGS=
```

} As flags do comando JVM podem ser atribuídas neste momento. Originalmente, nada é atribuído.

```
## Double-check that the JVM works.
echo "Trying $JVM"
$JVM -version > /dev/null 2>&1
if test $? -ne 0; then
    echo "ERROR: The JVM '$JVM' either isn't in your path or doesn't work. Please fix it."
    echo "(Tried '$JVM -version')."
    exit 12
fi
```

Esse trecho verifica se é possível executar o comando JVM com os valores já atribuídos anteriormente. Caso não seja possível é exibida na tela uma mensagem de erro.

```
# convenience
ANTS_CONFIGMGR="edu.utah.janos.nodeos.Main ants.core.ConfigurationManager"
```

Essa variável irá armazenar o comando necessário para colocar em execução um nó ativo. Como o próprio comentário sugere, esta variável existe por uma questão de conveniência. Com essa variável, evita-se de repetir todo o comando a cada nó ativo colocado em execução. O comando ConfigurationManager será mais bem explicado posteriormente.

```
# Node 10
if test $NODECT -ge 10; then
    echo "Starting node 10 ($NODE10_IP): $NODE10_LABEL"
    $XTERM -T "($NODE10_IP) $NODE10_LABEL" -e $SSH -c "$JVM $JVMFLAGS
$ANTS_CONFIGMGR $CONFIG_FILE $NODE10_IP && cat || cat" &
    J10=$!
    sleep 1
fi
```

} Uma única linha

Os nós são inicializados em ordem reversa (10, 9, ...). O trecho de código acima refere-se a inicialização do nó ativo 10. Inicialmente é verificado o valor da variável NODECT. Para o exemplo do ping, como o valor de NODECT é 3, somente os nós 3 e 2 serão inicializados por esse *script*. Os nós de 4 até 10 não serão inicializados [Ant02].

Primeiramente é exibida uma mensagem no terminal referente a cada inicialização de um nó ativo. Após isso são iniciadas as sessões xterm representando um nó ativo. O nome da janela será o valor que estiver entre as aspas duplas após a opção -T do comando XTERM. O comando que será executado é tudo o que vier após a opção -e do comando XTERM. Esse comando irá executar um shell (\$SSH) e dentro dele irá, por sua vez, executar o comando que vier após a opção -c. Esse último comando, irá executar a máquina virtual JAVA com as possíveis *flags* e com duas classes já instanciadas, a Main e a ConfigurationManager. O restante dos argumentos são os parâmetros do ConfigurationManager. Esses parâmetros são o arquivo de configuração da aplicação e o endereço IP do nó. Até a inicialização do nó 3, nada de diferente existe. São cópias do bloco de comandos comentado anteriormente. Para a inicialização do nó 2, existe uma pequena diferença, não é realizado a verificação relativa ao comando if. Isso devido ao teste inicial, onde pelo menos 2 nós ativos são requeridos para iniciar a aplicação.

```
if test "${J10}-${J9}-${J8}-${J7}-${J6}-${J5}-${J4}-${J3}-${J2}" = "-----"; then
    echo "No PIDs for backgrounded JVMs. Something is wrong."
    exit 1
fi
```

Nesse momento é testado o PID de cada nó ativo inicializado. Esse valor é obtido ao salvar o valor de \$!. Esse simbolo corresponde ao ID do processo inicializado. Caso todas as variáveis fiquem vazias, algo errado ocorreu e a aplicação é cancelada sem iniciar o nó ativo 1 [Ant02].

```
## Cleanup code to run on trap or at the end of the script:
CLEANUP_CODE="$KILL -9 ${J10} ${J9} ${J8} ${J7} ${J6} ${J5} ${J4} ${J3} ${J2}" } Uma única
>/dev/null 2>&1; sleep 1; \} linha
```

```
JVMPIDS=`$PS | grep "\$ANTS_CONFIGMGR $CONFIG_FILE" | awk '{print \$1}'`; \
test -n "\$JVMPIDS" && (echo "Killing all $CONFIG_FILE-related JVMs..."; \
$KILL \$JVMPIDS >/dev/null 2>&1)"
```

```
# Kill the other two and exit if anything goes wrong.
trap "echo; echo 'Abort. Cleaning up...'; $CLEANUP_CODE; exit 1" 1 2 3 13 15
```

O código acima realiza a liberação dos recursos alocados para a execução da aplicação caso ela termine ou algum problema ocorra durante a execução (CLEANUP_CODE).

```
echo "Sleeping for 5 seconds to let nodes get set up..."
sleep 5 || exit 1

# Node 1
echo "Starting node 1 ($NODE1_IP): $NODE1_LABEL"
$JVM $JVMFLAGS $ANTS_CONFIGMGR $CONFIG_FILE
$NODE1_IP
```

Caso nenhum problema tenha acontecido, o sistema aguarda 5 segundos para dar tempo dos processos dos outros nós virtuais iniciarem. Esse tempo é totalmente empírico, se em alguma aplicação o tempo demonstre-se muito ou pouco basta alterar nesse ponto do código. Após isso é inicializado o nó ativo 1 e com ele a aplicação que irá rodar na arquitetura de nós ativos implementada. Nesse caso, uma simulação em uma única máquina.

```
# NOTE: uncomment the following line to prevent the code cleanup
# from destroying the xterms before you are ready.
#cat

echo 'Cleaning up...'
eval $CLEANUP_CODE
exit 0

#eof
```

Tendo finalizado a execução da aplicação, o código de limpeza é executado para terminar e liberar os recursos utilizados em cada terminal que simulou um nó ativo.

A.4.2.1 DefaultEnv

Esse *script* é incluído no *script* nodesetup.sh. Sua função é definir algumas variáveis que são utilizadas durante a execução desse último *script*. A seguir é comentado cada parte de seu código.

```
## XTERM
#XTERM_DEFAULT="xterm -sb -sl 4000 -geometry 130x20"
XTERM_DEFAULT="xterm -sb -sl 4000"

## JVM
JVM_DEFAULT="antsvm"
```

Esta variável contém o comando para criar um terminal default

Variável que armazena a máquina virtual java que irá rodar o ANTS e suas aplicações. Este *script* será descrito com maiores detalhes no próximo tópico

```
## PS, default to 'ps -fA -o 'pid args'' on Solaris (need to have PID in first column)
case `uname` in
  *SunOS*) PS_DEFAULT="ps -f -A -o 'pid args'"; ;
  *) PS_DEFAULT="ps xwww"; ;
esac
```

Composto do comando ps com as opções certas, conforme a plataforma onde está rodando a aplicação ANTS

```
## KILL. Should be the executable, and not the shell built-in
KILL_DEFAULT="/bin/kill"
```

Variável que armazena o comando para finalizar os processos

A.4.2.2 Antsvm

O *script* antsvm tem como função inicializar a máquina virtual java e tudo que for necessário para iniciar o funcionamento do ANTS. Para isso, associa valores a determinadas variáveis, as quais são importantes nesse processo. A seguir é comentado cada trecho de código desse *script*.

Topbuilddir=/redes_ativas/ants/ants-2.0.0 → Diretório onde o ants foi instalado

jar_ext=jar → Valor da extensão dos arquivos jar

jnodeos_lib=../lib/jnodeos-pj-debug.zip → Caminho do arquivo .ZIP da API do NodeOS

jvmulation_lib=../lib/emujanosVM.jar → { Caminho do arquivo jar de emulação do
Janos Virtual Machine

ants_lib="\${topbuilddir}/src/ants.\${jar_ext}" → Caminho de arquivos jar da API do ANTS

ants_apps_lib="\${topbuilddir}/src/apps.\${jar_ext}" → { Caminho de arquivos jar com
exemplos de aplicações ANTS

ants_tests_lib="\${topbuilddir}/src/tests.\${jar_ext}" → { Caminho de arquivos jar com
conjunto de testes ANTS

```
## Default JAVA to 'java'
if test -z "$JAVA"; then
    JAVA=java
fi
```

{ Verificação do comando para iniciar o ANTS. Poderia ser
qualquer comando, não havendo nenhum, o java é usado

```
## Append ANTS classes to classpath
ants_classes="${jnodeos_lib}:${jvmulation_lib}:${ants_lib}:${ants_apps_lib}:${ants_tests_lib}"
if test -n "$CLASSPATH"; then
    CLASSPATH=${CLASSPATH}:${ants_classes}
else
    CLASSPATH=${ants_classes};.
fi
```

{ O CLASSPATH é configurado para
incluir todas as bibliotecas
necessárias

export CLASSPATH → O CLASSPATH é definido para todo o sistema

exec \$JAVA "\$@" → O java é executado

A.4.3 ConfigurationManager

O ConfigurationManager é a classe inicial do ANTS 2.0.0. Essa classe recebe como parâmetro o nome do arquivo .config de uma determinada aplicação e também o endereço IP do nó ativo correspondente. Esses parâmetros são recebidos através da linha de comando na execução dessa classe. A função dessa classe é iniciar o ANTS e uma aplicação, colocando-os para rodar em um determinado nó ativo. O *script* nodesetup.sh é responsável por executar o ConfigurationManager várias vezes, uma para cada nó pertencente a rede virtual de nós ativos.

Ao ler o arquivo de configuração, essa classe irá levar em conta apenas as linhas cujo endereço IP ANTS (será explicado melhor posteriormente) são iguais ao endereço IP recebido como parâmetro [Ant02]. Com isso é utilizado o mesmo arquivo de configuração para os nós ativos, pois cada um lerá somente aquilo que lhe é pertinente.

O arquivo de roteamento (.routes) é passado de nó ativo para nó ativo quando o ConfigurationManager lê o comando node do arquivo de configuração (.config). As diretivas connect são ignoradas nesse momento. Essas diretivas só farão sentido quando utilizado o *script* makeroutes, esse *script* será melhor explicado no tópico Ping.routes.

A.4.3.1 Ping.config

Esse arquivo tem a função de controlar a execução e a topologia de uma aplicação em uma determinada rede, seja ela virtual ou real. Cada diretiva existente nesse arquivo irá definir um nó, um endereço NIC (*Network Identifier Channel*), uma aplicação a ser executada ou uma conexão entre nós ativos. A seguir será comentada cada linha que compõe um arquivo de configuração. Tomando como exemplo o arquivo ping.config.

```
node 18.31.12.1 -routes ping.routes -log 255 -consoleport 2050
```

A diretiva node define um nó ativo com seu endereço IP virtual, um arquivo para definir suas rotas de comunicação entre nós ativos, um nível de *logging* e uma porta para ser utilizada em uma conexão, via telnet, com o console do nó ativo correspondente [Ant02].

```
channel 18.31.12.1 localhost:8004 -log 255
```

A diretiva channel cria um NIC. Para isso são especificados o endereço IP virtual, o IP real ou localhost e uma porta. Na linha de comando acima será mapeado um tráfego de dados do endereço IP virtual 18.31.12.1 para a máquina localhost através da porta 8004.

O comando application determina ao ANTS para iniciar uma determinada aplicação em

```
application 18.31.12.1 apps.ping.PingApplication  
-target 18.31.12.3 -iter 20 -interv 300 -lossage 1 -prime 1000
```

um determinado nó. Uma aplicação é o código que não é enviado através da rede automaticamente, mas sim aqueles que são executados em um único nó. Para o caso do ping a aplicação é responsável por enviar as cápsulas ativas pela rede.

Os argumentos "-target 18.31.12.3 -iter 20 -interv 300 -lossage 1 -prime 1000" são específicos para cada aplicação. Para o caso do ping, indicam respectivamente:

- endereço destino do ping;
- número de pings a serem realizados;
- quantos milissegundos esperar entre cada envio de uma cápsula;
- qual a taxa de perda de cápsulas esperada e;
- qual o intervalo entre a primeira cápsula enviada (prime) e a segunda a qual, na verdade, é a primeira relativa a aplicação.


```
node 18.31.12.2 -routes ping.routes -log 255 -consoleport 2051
channel 18.31.12.2 localhost:8005 -log 255
```

O bloco de comandos acima define um outro nó ativo, no caso do ping denominado *Middle*, com seus parâmetros já descrito anteriormente.

```
node 18.31.12.3 -routes ping.routes -log 255 -consoleport 2051
channel 18.31.12.3 localhost:8005 -log 255
```

Estas linhas de comando define um terceiro nó ativo que irá participar da aplicação.

```
connect 18.31.12.1 18.31.12.2
connect 18.31.12.2 18.31.12.3
```

O comando connect define a topologia da rede virtual ativa. No exemplo acima será criado uma conexão direta entre o nó ativo 18.31.12.1 e o nó ativo 18.31.12.2. Esse por sua vez terá uma conexão direta com o nó ativo 18.31.12.3 [Ant02].

A.4.3.2 Ping.Routes

Esse arquivo contém informações de uma tabela de roteamento estática. Estas informações são obtidas do arquivo de configuração (.config), porém também pode ser criado. Caso não seja criado, esse arquivo é gerado pelo *script* makeroutes, o qual utiliza como fonte o arquivo de configuração da aplicação. A sintaxe para utilização do *script* makeroutes no caso da aplicação ping é: - makeroutes -route <ping.config> ping.routes

A seguir é mostrado um exemplo de um arquivo de roteamento estático (.routes).

# source	destination	next	addr
18.31.12.1	18.31.12.2	18.31.12.2	localhost:8005
18.31.12.1	18.31.12.3	18.31.12.2	localhost:8005
18.31.12.2	18.31.12.1	18.31.12.1	localhost:8004
18.31.12.2	18.31.12.3	18.31.12.3	localhost:8006
18.31.12.3	18.31.12.1	18.31.12.2	localhost:8005
18.31.12.3	18.31.12.2	18.31.12.2	localhost:8005

Cada linha corresponde a um endereço IP virtual fonte, um endereço IP virtual destino, o próximo endereço IP virtual e a interface de acesso. Essa interface é composta por um endereço IP virtual e a porta onde o nó pode acessar o próximo nó ativo descrito na tabela.

A.4.4 Exemplos de Aplicações

O ANTS traz um conjunto de aplicações exemplos juntamente com o código fonte dessa tecnologia. Todos os exemplos simulam vários nós ativos formando uma rede ativa

virtual em uma mesma máquina, conforme comentado anteriormente. Para executar uma dessas aplicações, deve-se estar no diretório *scripts* e executar o arquivo *.start*. Os arquivos que devem estar no diretório onde será executado a aplicação (*.start*) são os seguintes:

- *<nome_da_aplicação>.(config, route e start)*
- *nodesetup.sh*
- *defaultEnv*

O diretório onde a aplicação estiver rodando deve estar no path da máquina. Dessa forma, o seguinte comando executado, estando no diretório desejado, satisfaz tal exigência:

```
export PATH="$PATH:."
```

Tendo concluído os passos anteriores, basta executar o *script* *<nome_da_aplicação>.start* para rodar a aplicação desejada.

A.5 Executando ANTS em Múltiplas Máquinas

Para execução de uma aplicação em uma única máquina simulando três nós ativos, as diretivas *channel* existentes no arquivo de configuração poderiam ser as seguintes:

```
channel 18.31.12.1 localhost:8004 -log 255  
channel 18.31.12.2 localhost:8005 -log 255  
channel 18.31.12.3 localhost:8006 -log 255
```

Com esses dados, uma rede ativa virtual, igual a representada na Figura A.2 seria criada. Para a execução de uma aplicação de maneira distribuída, em três computadores, é necessário mudar a diretiva *channel* para expressar corretamente o IP real, de cada máquina. Feito as devidas mudanças as diretivas poderiam ficar da seguinte forma [Ant02]:

```
channel 18.31.12.1 10.1.1.1:8004 -log 255  
channel 18.31.12.2 10.1.1.2:8004 -log 255  
channel 18.31.12.3 10.1.1.3:8004 -log 255
```

Para cada máquina real é associado, da mesma forma feita com máquinas virtuais, um endereço ANTS. Esse endereço pode ser o mesmo de quando a distribuição era simulada. Com as novas diretivas *channel*, a rede de nós ativos ficaria da seguinte forma:

Figura A.4 - Topologia da Rede Ativa Virtual para Três Máquinas [ANTS 01]

O próximo passo é regerar o *script* .routes da aplicação, para que essa rede de nós ativos esteja espelhada nesse arquivo. Para realizar essa operação basta utilizar o *script* makeroutes, O comando a ser executado em um terminal, caso a aplicação seja o ping é [Ant02]:

```
sh makeroutes -route <ping.config >ping.routes
```

Tendo gerado o arquivo .routes, ou mesmo atualizado manualmente, é necessário espelhar a arquitetura de diretórios ANTS em todas as máquinas da rede. Para isso existem duas maneiras:

1. Instalando o pacote ANTS em cada máquina, ou
2. Utilizando um sistema de arquivo de redes como o NFS (*Network File System*).

Para o primeiro caso é gerado um trabalho extra para cada atualização de um arquivo ou mesmo a criação de novas aplicações ativas. Os arquivos .routes e .png devem ser exatamente os mesmos em todas as máquinas, além, é claro, de todo o código fonte da tecnologia ANTS. Utilizando o NFS, a sincronização dos arquivos nas máquinas da rede pode ocorrer de maneira automática. Apenas um arquivo por aplicação será específico para cada máquina, o arquivo .start [Ant02]. Para esclarecer melhor será tomado a arquitetura exemplificada na Figura A.4 e assumido o ping como aplicação a ser executada. A Tabela A.1 mostra 3 arquivos ping.start para cada IP real:

Tabela A.1 - Exemplo do *script* .start em 3 máquinas

IP Real	10.1.1.1	10.1.1.2	10.1.1.3
Script ping. Start	CONFIG_FILE=ping.config NODE1_IP=18.31.12.1 NODE1_LABEL="Rot 1" NODECT=1 ./nodesetup.sh #eof	CONFIG_FILE=ping.config NODE1_IP=18.31.12.2 NODE1_LABEL="Rot 2" NODECT=1 ./nodesetup.sh #eof	CONFIG_FILE=ping.config NODE1_IP=18.31.12.3 NODE1_LABEL="Rot 3" NODECT=1 ./nodesetup.sh #eof

A.6 Segurança e ANTS 2.0.0

O modelo de segurança adotado é simples e pode ser estendido dependendo da necessidade de cada rede ativa. É implementado um mecanismo de controle de acesso simples, o qual trabalha com uma matriz que descreve os direitos existentes e as permissões necessárias para obter tal direito [Ant01]. Para implementar tal mecanismo, foi criado um pacote, o qual contém todas as classes que lidam com segurança do ANTS. Esse pacote é o ants.core.security. Os direitos são representados na classe "*Principal*" e as permissões

necessárias são definidas na classe "*Permission*". As verificações de controle de acesso são realizadas pela classe estática "*ReferenceMonitor*". Essa classe recebe como parâmetros o direito requerido e a permissão possuída e retorna o direito de acesso ou não. Esse método é chamado da seguinte forma:

```
ReferenceMonitor.checkPermission(Principal who, Permission what);
```

Uma classe de acesso abstrata define a matriz que diz qual permissão é necessária para a obter cada direito, essa classe é a "*policy*". Existem duas formas de utilizar uma política de segurança.

1. Utilizar a política de segurança padrão definida na classe "*SecurityDefault*".
2. Criando uma política de segurança específica.

Para a primeira alternativa nada precisa ser feito. Existe um conjunto de direitos, chamados de *principals* e de permissões padrão. Os direitos são os seguintes:

- Admin - Para o *PrimordialNode* e o *shell*.
- Bootstrap - Para as aplicações durante a inicialização.
- LocalUsers - Para protocolos e aplicações locais.
- RemoteUsers - Para protocolos e aplicações remotas.
- DanteUser - Utilizado para o servidor e o cliente Dante.
- RouteUser - Utilizado pelo protocolo de roteamento.

Além dos direitos padrão, mencionados acima, existem algumas permissões padrão:

- MODIFY_ROUTES: modificação de tabela de roteamento.
- READ_NEIGHBORS: leitura da tabela dos vizinhos imediatos.
- MODIFY_NEIGHBORS: modificação da tabela dos vizinhos imediatos. SHUTDOWN: Finalização de um nó.
- LOG: Escrita de arquivos de log.
- SET_ADDRESS: Definir endereços dos nós.
- READ_PHYSICAL_PROTOCOL_SPEC: Leitura da especificação do protocolo física usada por um nó.
- READ_PHYSICAL_ADDRESS_SPEC: Leitura da especificação de endereço física usada por um nó.
- LOCAL_SEND: envio de cápsulas.
- START_THREAD: Inicializar uma *thread*.
- ADMIN_PERMISSION: Todas as permissões existentes.

Para a criar uma política de segurança específica é necessário acrescentar na diretiva node a opção "-policy", como exemplificado abaixo:

```
node 18.31.12.1 -policy policy.ser
```

O parâmetro *policy.ser* deve ser um arquivo que contém um objeto do tipo *Policy* serializado. O direito de uma determinada aplicação pode ser definida em seu arquivo de configuração através da opção "-principal" [Ant01]. A seguir é mostrado um exemplo:

```
application 18.31.12.255 ants.dante.DanteServer -principal DanteUser
```

A.7 Componentes de um Serviço ANTS

Um serviço ANTS é composto basicamente de três componentes:

- Cápsula - código que será transportado para cada nó ativo pertencente ao caminho entre fonte e destino de uma rede ativa. O método *evaluate* é o trecho de código que será executado em cada nó ativo visitado.
- Protocolo - Composto por um ou mais grupos. Cada grupo pode reunir uma ou mais cápsulas que compõem um serviço ativo. O protocolo é responsável por delimitar o escopo de ação do serviço a medida que define quais cápsulas pertencem ao serviço.
- Aplicação - Possui os métodos responsáveis por enviar e receber as cápsulas, dessa forma deve ser executado na máquina que deseja iniciar o serviço ativo. Nessa classe pode ser dado o tratamento necessário tanto para o envio, quanto na recepção de uma cápsula que circulou pela rede ativa.

De maneira geral um serviço ativo contém os seguintes componentes:

- Arquivo de configuração dos parâmetros do serviço ativo (.conf).
- Arquivo que define a topologia da rede ativa que o serviço pertence (.route).
- Arquivo para iniciar a execução do serviço, esse arquivo é opcional (.start).
- Um ou mais arquivos dependendo da quantidade de cápsulas do serviço.
- Arquivo com o código da aplicação em si.
- Arquivo com a definição do protocolo ativo, ou seja, as cápsulas que forma o serviço.