OSCAR THYAGO JOSÉ DUARTE DANTAS LISBÔA MOTA

UMA ARQUITETURA ADAPTÁVEL PARA PROVISÃO DE QOS NA INTERNET

DISSERTAÇÃO DE MESTRADO

Departamento de Informática

Rio de Janeiro, 28 de maio de 2001

OSCAR THYAGO JOSÉ DUARTE DANTAS LISBÔA MOTA

UMA ARQUITETURA ADAPTÁVEL PARA PROVISÃO DE QOS NA INTERNET

Dissertação de Mestrado apresentada ao Departamento de Informática da PUC-Rio, como parte dos requisitos para obtenção do título de Mestre em Informática: Ciência da Computação.
Orientador: Prof. Luiz Fernando Gomes Soares.

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro

Rio de Janeiro, 28 de maio de 2001

Este trabalho é dedicado

Aos meus pais José Lisbôa Mota e Maria do Carmo Zenah Duarte Dantas Lisbôa Mota, pelo amor e carinho que sempre me dedicaram em todos os momentos de minha vida.

Agradecimentos

Inicialmente, gostaria de agradecer ao Professor Luiz Fernando Gomes Soares pela oportunidade de ter sido seu aluno e orientando nesses dois anos de trabalho. Com o seu estilo informal e bem-humorado, sempre soube transmitir, nos momentos mais difíceis, palavras de incentivo que foram fundamentais para que este trabalho pudesse ser concluído.

Durante o desenvolvimento desta dissertação de mestrado, tive o privilégio de trabalhar com pessoas de excepcional interesse e dedicação à pesquisa nas áreas de redes, sistemas distribuídos e multimídia no Laboratório TeleMídia da PUC-Rio. Dentre essas pessoas, gostaria de agradecer a Débora Muchaluat Saade, Rogério Rodrigues, Sérgio Colcher e Antônio Tadeu Gomes por estarem sempre dispostos a ajudar com a sua experiência e conhecimento. Reservo um agradecimento especial a Sérgio Colcher e Antônio Tadeu Gomes, que me ajudaram a definir a linha de pesquisa seguida pelo presente trabalho.

São os amigos que, sempre dispostos a nos ouvir e aconselhar, nos fazem acreditar que podemos superar todas as dificuldades. Gostaria de agradecer especialmente a Milene Selbach Silveira e Antônio Sampaio Junior pela amizade e apoio nesses dois anos de convívio. Também gostaria de agradecer aos demais amigos do já institucionalizado *momento de descontração MM*: Claudia, Patrícia, Marcel, Jorge, Clarissa, Beatriz e Antônio Carlos. Por último, mas não menos importante, agradeço as palavras de apoio e incentivo que recebi dos colegas da Justiça Eleitoral durante esses dois anos de afastamento.

Finalmente, gostaria de reconhecer o apoio do Tribunal Regional Eleitoral do Ceará à elaboração deste trabalho. Em especial, agradeço ao Desembargador Stênio Leite Linhares, pela concessão do meu afastamento, e aos Doutores José Danilo Correia Mota e José Bezerra de Morais, pelo empenho para que a referida concessão se concretizasse. Agradeço também ao CNPq, pelo suporte financeiro, e a todos os professores e funcionários do Departamento de Informática da PUC-Rio, que, com sua dedicação diária, conseguem fazer deste departamento um dos centros brasileiros de excelência acadêmica e reconhecimento internacional.

RESUMO

A diversidade de configurações possíveis dos modelos *intserv* e *diffserv* e das várias modalidades de provisão de QoS no nível das sub-redes torna difícil a compreensão de que tipo de modelo e tecnologia de sub-rede deve ser utilizado para se realizar um determinado serviço. Adicionalmente, a contínua evolução tecnológica sugere o desenvolvimento de arquiteturas flexíveis o suficiente para acomodar, em tempo de operação, adaptações que hoje só são possíveis por meio de procedimentos menos dinâmicos como atualização de *hardware* ou *firmware*. Este trabalho propõe uma arquitetura adaptável para provisão de QoS na Internet que é independente do modelo de serviço e dos mecanismos de provisão, incluindo a tecnologia de sub-rede empregada pelo provedor de serviços. Mostra-se como, a partir da definição de frameworks genéricos, pode-se especializar pontos de flexibilização para implementar esses dois modelos utilizados para prover serviços com QoS na Internet. Em seguida, é proposta uma arquitetura geral que permite que as funções de provisão de QoS em estações e roteadores passem a ser representadas independente dos modelos de serviços e sub-redes existentes.

Palavras-chave: Qualidade de Serviço, intserv, diffserv, framework

ABSTRACT

The variety of *intserv* and *diffserv* possible configurations as well as the various QoS provision techniques used in subnetworks makes it difficult to identify which model and subnetwork technology should be used to accomplish a specific service. Additionally, the fast technological evolution suggests the development of flexible architectures that accommodate, in execution time, adaptations nowadays only possible by less dynamic procedures such as hardware or firmware updates. This paper proposes an adaptable architecture for Internet QoS provision that is independent of service models and provision mechanisms, besides the used provider subnetwork technology. Through a generic framework definition the paper shows how hot-spots can be specialized to implement the two mentioned models used for Internet QoS provision. Following this, the paper proposes a general architecture that allows the QoS provision representation, in hosts and routers, independent from service models and existing subnetworks.

Keywords: Quality of Service, intserv, diffserv, framework

Índice Analítico

CAPÍTULO 1 - INTRODUÇÃO 10

1.1 OBJETIVOS DA DISSERTAÇÃO 12
1.2 ESTRUTURA DA DISSERTAÇÃO 13

CAPÍTULO 2 - QUALIDADE DE SERVIÇO NA INTERNET 15

2.1 FASES DE PROVISÃO DE QOS 16
2.1.1 Iniciação do Provedor de Serviços 16
2.1.2 Solicitação de Serviços 17
2.1.3 Estabelecimento de Contratos de Serviço 17
2.1.4 Manutenção de Contratos de Serviço 20
2.1.5 Término de Contratos de Serviço 21
2.2 MODELO DE SERVIÇOS INTEGRADOS 22
2.2.1 Solicitação de Serviços 23
2.2.2 Estabelecimento de Contratos de Serviço 24
2.2.3 Manutenção de Contratos de Serviço 27
2.3 MODELO DE SERVIÇOS DIFERENCIADOS 28
2.3.1 Solicitação de Serviços 30
2.3.2 Estabelecimento de Contratos de Serviços 33
2.3.3 Manutenção de Contratos de Serviços 38
2.4 Serviços Integrados Sobre Diferenciados 38
2.5 MODELOS DE SERVIÇOS E TECNOLOGIAS DE SUB-REDE 41
2.5.1 MPLS 43
2.5.2 Serviços Diferenciados sobre MPLS 46
2.6 SUMÁRIO 49

CAPÍTULO 3 - DESCRIÇÃO DA ARQUITETURA 50

3.1 PARAMETRIZAÇÃO DE SERVIÇOS 51
3.1.1 Componentes do Framework para Parametrização de Serviços 54
3.1.2 Exemplo de Aplicação do Framework para Parametrização 55
3.2 COMPARTILHAMENTO DE RECURSOS 58
3.2.1 Componentes dos Frameworks para Compartilhamento de Recursos 60
3.2.2 Exemplo de Aplicação dos Frameworks para Compartilhamento 62
3.3 ORQUESTRAÇÃO DE RECURSOS 64
3.3.1 Componentes do Framework Para Negociação de QoS 66
3.3.2 Exemplos de Aplicação do Framework para Negociação de QoS 70
3.3.3 Componentes do Framework para Sintonização de QoS 73
3.3.4 Exemplo de Aplicação do Framework para Sintonização de QoS 74
3.4 SUMÁRIO 75

CAPÍTULO 4 - EXEMPLO DE APLICAÇÃO DA ARQUITETURA 76

4.1 DESCRIÇÃO DO CENÁRIO 78
4.2 INICIAÇÃO DO SISTEMA 80
4.3 PARAMETRIZAÇÃO DE SERVIÇOS 81
4.4 IDENTIFICAÇÃO DOS FLUXOS 82
4.5 ANÚNCIO DE TRÁFEGO 83
4.6 SOLICITAÇÃO DE SERVIÇOS 88
4.7 ESTABELECIMENTO DE CONTRATOS DE SERVIÇO 90
4.8 MANUTENÇÃO DE CONTRATOS DE SERVIÇO 97
4.9 SUMÁRIO 98

CAPÍTULO 5 - CONCLUSÕES 99

5.1 CONTRIBUIÇÕES DA DISSERTAÇÃO 100
5.2 TRABALHOS FUTUROS 101

BIBLIOGRAFIA 103

LISTA DE ACRÔNIMOS 110

Índice de Figuras

Figura 1: Interfaces presentes em um BB. 19
Figura 2: *Intserv* sobre *diffserv - cenário 1*. 40
Figura 3: *Intserv* sobre *diffserv* - cenário 2. 41
Figura 4: Framework para Parametrização de Serviços. 55
Figura 5: Exemplo de aplicação do Framework para Parametrização. 56
Figura 6: Árvore de recursos virtuais em um roteador *intserv*. 59
Figura 7: Visão conjunta dos Frameworks para Compartilhamento de Recursos. 61
Figura 8: Exemplo de aplicação dos Frameworks para Compartilhamento. 63
Figura 9: Framework para Negociação de QoS. 69
Figura 10: Aplicação do Framework para Negociação de QoS em um provedor *intserv*. . 71
Figura 11: Aplicação do Framework para Negociação de QoS em um provedor *diffserv*. 72
Figura 12: Framework para Sintonização de QoS. 74
Figura 13: Aplicação do Framework para Sintonização de QoS. 75
Figura 14: Cenário de provisão de serviços *intserv* sobre ATM. 79
Figura 15: Ligação entre roteadores IP vizinhos. 80
Figura 16: Parametrização de serviços. 81
Figura 17: Identificação dos fluxos de tráfego do usuário. 83
Figura 18: Eventos de QoS. 85
Figura 19: Sessão de QoS. 86
Figura 20: Anúncio de tráfego. 87
Figura 21: Negociação de QoS. 92
Figura 22: Classe FlowState. 94
Figura 23: Estratégia de mapeamento. 96

Índice de Tabelas

Tabela 1: Principais primitivas da API de solicitação de serviços.....89

Capítulo 1

Introdução

Nos últimos anos, tem-se observado um crescente interesse pela utilização da Internet por aplicações distribuídas como conferência multimídia, telefonia, vídeo sob demanda, telemedicina, entre outras. Essas aplicações caracterizam-se, principalmente, pelo emprego de diversos tipos de mídia que impõem requisitos distintos de Qualidade de Serviço ou QoS (*Quality of Service*) ao sistema de comunicação, como retardo máximo, variação máxima do retardo, taxas de transmissão e de perda de informações, disponibilidade, etc.

Desde a sua concepção, a Internet oferecia apenas o serviço de melhor-esforço, caracterizado pela ausência de garantias de QoS às aplicações. Ao fornecer esse serviço, o sistema de comunicação compromete-se a realizar o transporte das informações com a maior celeridade possível, objetivando fornecer um serviço da mais alta qualidade. Porém, é incapaz de tomar medidas preventivas (controle de admissão, policiamento) ou corretivas (redistribuição de tráfego, renegociação) que mantenham a qualidade fornecida em um patamar satisfatório ao conjunto de aplicações atendidas. Nas situações de congestionamento, que representam a escassez dos recursos disponíveis, podem ocorrer atrasos indeterminados na entrega dos pacotes de informação nos destinatários. Nos casos mais extremos, o serviço de transporte oferecido pelo sistema de comunicação pode cessar por completo.

A inadequação do serviço de melhor-esforço aos requisitos das aplicações distribuídas supracitadas motivou, em um período recente, o desenvolvimento de propostas para um novo modelo de serviços que agregasse o conceito de QoS na Internet. Dentre essas propostas destacam-se os modelos de serviços integrados (*intserv*) [BRADEN94] e diferenciados (*diffserv*) [BLAKE98].

A Internet constitui um sistema de comunicação formado pela interligação de um grande número de subsistemas menores, chamados, comumente, de provedores de serviço ou ISPs (*Internet Service Providers*). ISPs possuem escopos que podem variar desde uma comunidade local (provedores de acesso, redes corporativas) a subsistemas de alcance global (*backbones* internacionais). A escolha do modelo de serviços a ser adotado por cada ISP depende, dentre outros fatores, da quantidade de fluxos de tráfego a serem recebidos e processados no mesmo. Devido à diversidade de provedores existentes na Internet, espera-se que diferentes modelos possam ser combinados em uma provisão de serviços fim-a-fim. Vários trabalhos têm sido publicados, nos últimos anos, abordando os mecanismos normalmente empregados em cenários que combinam modelos de serviços específicos [BAKER00b, CHARNY01].

Os modelos de serviços *intserv* e *diffserv* representam arquiteturas de nível 3 que podem empregar, no nível das suas sub-redes[1], diferentes tecnologias de provisão de QoS, como a 802.1p[2] [IEEE98], a ATM [ITU99] ou a MPLS [CALLON01]. Cabe ao ISP realizar a adequação dos modelos de serviço utilizados a cada uma das tecnologias de sub-rede presentes no seu domínio de atuação. Vários artigos têm procurado versar sobre as alternativas possíveis na realização dessas adequações em casos particulares [BAKER00a, CRAWLEY00, GHANWANI00, BERGER98a, BERGER98b, BORDEN98, BAKER98a, CHEVAL01].

[1] Parte do nível de rede dependente de tecnologia de transmissão [ISO87].
[2] O padrão 802.1p é definido em conjunto com o 802.1q, que especifica como determinados bits do cabeçalho dos quadros 802 podem ser utilizados na configuração de VLANS (*Virtual LANs*). O padrão 802.1p, especificamente, emprega bits de prioridade de tráfego na definição de classes de serviço ou CoS (*Class Of Service*) no nível de enlace.

O uso combinado dos modelos de serviços *intserv* e *diffserv* e das diversas abordagens de provisão de QoS ao nível das sub-redes abre um leque de possibilidades de configuração, como, por exemplo, o uso do *intserv* sobre *diffserv*, do *intserv* ou *diffserv* sobre MPLS, ou ainda sobre ATM. Apesar de um grande número de trabalhos relacionados ao tema *QoS na Internet* ter sido publicado nos últimos anos, o estudo sistemático das combinações possíveis de serviços e tecnologias de provisão de QoS, com a identificação dos conceitos e técnicas comuns empregados, ainda não foi devidamente abordado. Diante da vasta quantidade de opções existentes, torna-se difícil para o projetista de um ISP entender e decidir que tipo de modelo e tecnologia deve ser utilizado para se realizar um determinado serviço. Uma vez escolhida uma configuração, a falta de um modelo de estruturação dos serviços e funções de provisão de QoS torna custosas futuras adaptações na referida configuração, como a introdução de novos serviços ou a adoção de novas tecnologias, por exemplo. No que se refere a esse aspecto, uma outra questão a ser considerada é a dificuldade de reutilização das funcionalidades comuns identificadas nos vários subsistemas. O emprego de um modelo de provisão de QoS baseado em uma arquitetura genérica seria de grande benefício para o desenvolvimento de estratégias de orquestração de recursos fim-a-fim, uma vez que a organização interna dos vários subsistemas envolvidos estaria definida de forma estruturada, com mecanismos de acesso às suas funcionalidades conhecidos.

1.1 Objetivos da Dissertação

O presente trabalho propõe uma arquitetura genérica capaz de modelar as funções de provisão de QoS, presentes nas estações e roteadores da Internet, de forma independente dos modelos de serviço e tecnologias de sub-rede empregadas pelos ISPs. A arquitetura proposta representa uma aplicação dos *frameworks para provisão de QoS em ambientes genéricos de processamento e comunicação* apresentados em [GOMES99]. Neste último trabalho, as funções de

provisão de QoS recorrentes nos vários subsistemas foram identificadas, bem como o papel que cada uma dessas funções desempenha na orquestração dos recursos em um ambiente de fornecimento de serviços verdadeiramente fim-a-fim. Essas funções genéricas foram estruturadas sob a forma de frameworks objetivando facilitar a identificação de pontos de flexibilização (*hot-spots*) que possam ser preenchidos no contexto de um ambiente específico. O presente trabalho mostra como os referidos pontos de flexibilização podem ser preenchidos na modelagem de cenários de provisão de QoS baseados na tecnologia Internet.

De forma a permitir que a arquitetura proposta possa ser adaptável a diferentes modelos de serviço ou tecnologias de sub-redes, incluindo aqueles que venham a surgir no futuro, alguns dos pontos de flexibilização propostos em [GOMES99] foram deixados propositadamente incompletos. Esses pontos podem estar associados às funções de provisão de QoS que devem ser programadas para dar suporte a diferentes requisitos de QoS. Isto permite, por exemplo, através do preenchimento de um desses pontos, que um novo algoritmo de escalonamento de pacotes possa ser instalado, em tempo de operação, nas estações e roteadores de um provedor *diffserv* implementado sobre uma plataforma de comunicação programável, viabilizando a introdução de um novo serviço. Todos esses pontos de flexibilização são de fundamental importância para a arquitetura proposta, uma vez que permitem a sua adaptação à introdução de novos serviços, independente do modelo adotado internamente pelo sistema de comunicação.

1.2 Estrutura da Dissertação

O presente trabalho encontra-se estruturado da forma descrita a seguir. No Capítulo 2, serão apresentados os principais modelos e tecnologias de provisão de QoS disponíveis atualmente na Internet. O enfoque dessa apresentação consistirá na identificação dos aspectos comuns a essas diversas abordagens. Esses aspectos serão determinantes na escolha dos pontos de

flexibilização da arquitetura proposta no presente estudo. Em seguida, no Capítulo 3, a arquitetura proposta será descrita, sendo apresentado como os frameworks propostos em [GOMES99] podem ser especializados na definição de um modelo adaptável aos vários cenários de provisão de QoS na Internet. A apresentação dos frameworks que constituem a arquitetura será pontuada por vários exemplos que visam ilustrar como as estruturas definidas podem ser empregadas na modelagem de contextos específicos. Um exemplo mais completo, no entanto, será reservado ao Capítulo 4, onde será mostrada uma aplicação da arquitetura proposta em um cenário real de provisão de QoS na Internet. Finalmente, no Capítulo 5, são feitos comentários finais sobre o trabalho, salientadas as contribuições e identificados possíveis estudos futuros no contexto da presente pesquisa.

Capítulo 2

Qualidade de Serviço na Internet

O objetivo deste capítulo é caracterizar os serviços e identificar as principais funções e estruturas de provisão de QoS presentes nos modelos *intserv* e *diffserv*. A combinação desses modelos, bem como a adequação dos mesmos aos mecanismos de provisão de QoS próprios do nível das sub-redes, também é discutido. A nomenclatura utilizada na identificação das referidas funções e estruturas segue a definida inicialmente em [GOMES99]. A idéia contida por trás dessa abordagem é determinar os aspectos comuns aos diversos cenários de provisão de QoS que podem ser vislumbrados atualmente na Internet. Desta forma, espera-se facilitar a compreensão dos frameworks constituintes da arquitetura proposta e, de certa forma, justificar os pontos de flexibilização definidos objetivando tornar a solução final adaptável aos diversos cenários identificados.

Inicialmente, são descritas as fases de provisão de QoS e como elas se sucedem no contexto do fornecimento de serviços com qualidade na Internet. Na descrição de cada uma dessas fases, são identificadas as funções e estruturas normalmente empregadas nos diversos modelos de serviços. A particularização dessas funções e estruturas, destacadas em *itálico* para uma maior ênfase, nos modelos *intserv* e *diffserv* é realizada em seguida. Ao final, são discutidos os mecanismos particularmente relevantes na combinação entre modelos de serviços e tecnologias de provisão de QoS em um ambiente Internet, como a negociação

entre domínios distintos, com a conseqüente adequação de parâmetros de QoS entre serviços compatíveis.

2.1 Fases de Provisão de QoS

A provisão de QoS, de maneira geral, implica em uma série de funções que devem ser acrescentadas ao cliente e provedor de serviços. As funções de provisão de QoS atuam em diferentes fases do ciclo de vida de um serviço [GOMES01]:

- Iniciação do Provedor de Serviços;
- Solicitação de Serviços;
- Estabelecimento de Contratos de Serviço;
- Manutenção de Contratos de Serviço;
- Término de Contratos de Serviço.

2.1.1 Iniciação do Provedor de Serviços

Na fase de iniciação, os recursos disponíveis ao fornecimento dos serviços são identificados, sendo definido o estado interno inicial do provedor. A efetiva associação desses recursos aos serviços oferecidos pode ser realizada, nessa fase, por intermédio de um dimensionamento apropriado do provedor. Esta é a configuração mais empregada atualmente na Internet, embora esse quadro deva ser alterado nos próximos anos, cedendo espaço a estratégias mais dinâmicas, que permitam a modificação do estado interno do provedor em tempo de operação.

2.1.2 Solicitação de Serviços

Concluída a fase de iniciação, o provedor encontra-se apto a receber as solicitações de serviço dos clientes, podendo ser empregados, para tanto, mecanismos estáticos (uma solicitação verbal ou escrita ao administrador do provedor, por exemplo) ou procedimentos mais dinâmicos, como aqueles que fazem uso de um protocolo de solicitação de serviços apropriado. Em ambos os casos, o estado interno do provedor é alterado após a sua iniciação. A solicitação estática, no entanto, impõe uma carga administrativa na maioria das vezes proibitiva. Assim como mencionado na fase de iniciação, a tendência atual é o emprego de mecanismos mais dinâmicos, sendo assumido, neste trabalho, o uso de protocolos de sinalização específicos na solicitação de quaisquer serviços com requisitos de QoS. Uma solicitação de serviços normalmente contém estruturas que permitem a *caracterização do tráfego* correspondente aos fluxos[3] de dados gerados pelo usuário e a *especificação da QoS* que se espera que seja aplicada aos mesmos.

2.1.3 Estabelecimento de Contratos de Serviço

Uma solicitação de serviço resulta na configuração de um contrato implícito entre o cliente, normalmente representado por uma aplicação distribuída, e o provedor. Por esse contrato, o cliente compromete-se a gerar fluxos de dados conforme com o que foi especificado na solicitação de serviço. O provedor, por sua vez, assegura o transporte dos referidos fluxos com a QoS solicitada, desde que o tráfego introduzido esteja de acordo com o prometido. Este contrato de serviço é, normalmente, denominado de SLA (*Service Level Aggrement*).

[3] No contexto do presente trabalho, um fluxo representa um conjunto de pacotes transmitidos por uma aplicação origem, em particular, para uma ou mais aplicações destino.

O estabelecimento de um SLA implica na execução de uma série de funções de provisão de QoS pelo ISP (*Internet Service Provider*). Inicialmente, um mecanismo de *roteamento* é empregado na identificação dos subsistemas[4] que efetivamente participarão da provisão do serviço solicitado. Cabe à função de *negociação* realizar, entre outras tarefas, a *orquestração* dos recursos, isto é, determinar os recursos necessários em cada um dos subsistemas escolhidos. Essa função depende do *mapeamento* dos parâmetros especificados pela aplicação quando da solicitação do serviço em valores que permitam quantificar os recursos que devem ser reservados nesses subsistemas. Será com base nesses valores que o mecanismo de *controle de admissão* poderá determinar a possibilidade de atendimento ou não ao serviço solicitado. Havendo recursos suficientes, os módulos de encaminhamento presentes nas estações e roteadores, compreendendo o *classificador* e o *escalonador* de pacotes, devem ser apropriadamente configurados. Para que a orquestração de recursos seja efetiva, os mecanismos específicos de *reserva de recursos* das sub-redes envolvidas também devem ser acionados.

Caso os recursos do provedor sejam reservados e configurados por meio de uma intervenção manual do administrador, diz-se que o SLA é estático. Por outro lado, caso essa configuração seja realizada pelo emprego de um protocolo de sinalização específico, classifica-se o SLA de dinâmico. Nessa última modalidade de provisão de serviços, distinguem-se duas abordagens: a centralizada e a distribuída.

Na abordagem de provisão de serviços centralizada, um *negociador de recursos* ou BB (*Bandwidth Broker*) [JACOBSON99b] é responsável pelas decisões de *reserva de recursos* e *configuração* do provedor, tendo total conhecimento dos recursos disponíveis, bem como das modalidades de serviço suportadas. O BB disponibiliza uma interface de solicitação de serviços, que pode ser acessada diretamente, no caso de uma solicitação manual, efetuada pelo operador do provedor, ou indiretamente, quando são empregados protocolos de

[4] Roteadores ou outros domínios de roteamento, administrados ou não pelo mesmo ISP.

solicitação de serviços específicos. As decisões de *reserva de recursos* e *configuração* do provedor são comunicadas pelo BB aos roteadores do domínio de serviços pela interface *intradomínio*, presente no negociador. Vários protocolos de sinalização têm sido adaptados, nos últimos anos, objetivando a comunicação dessas decisões, como o COPS [DURHAM00] e o Diameter [CALHOUN01], para citarmos os mais discutidos [CHIMENTO01]. Finalmente, também é função do BB de um domínio *negociar*, com os provedores de serviço adjacentes, qualquer acréscimo no tráfego de saída externo que acarrete alterações nos SLAs acordados previamente. Essa negociação é realizada por intermédio da interface *interdomínio* do BB, sendo utilizados, para tanto, protocolos específicos, como o SIBBS (*Simple Interdomain Bandwidth Broker Signalling*), por exemplo, proposto em [CHIMENTO01]. A Figura 1 ilustra as interfaces normalmente presentes em um BB.

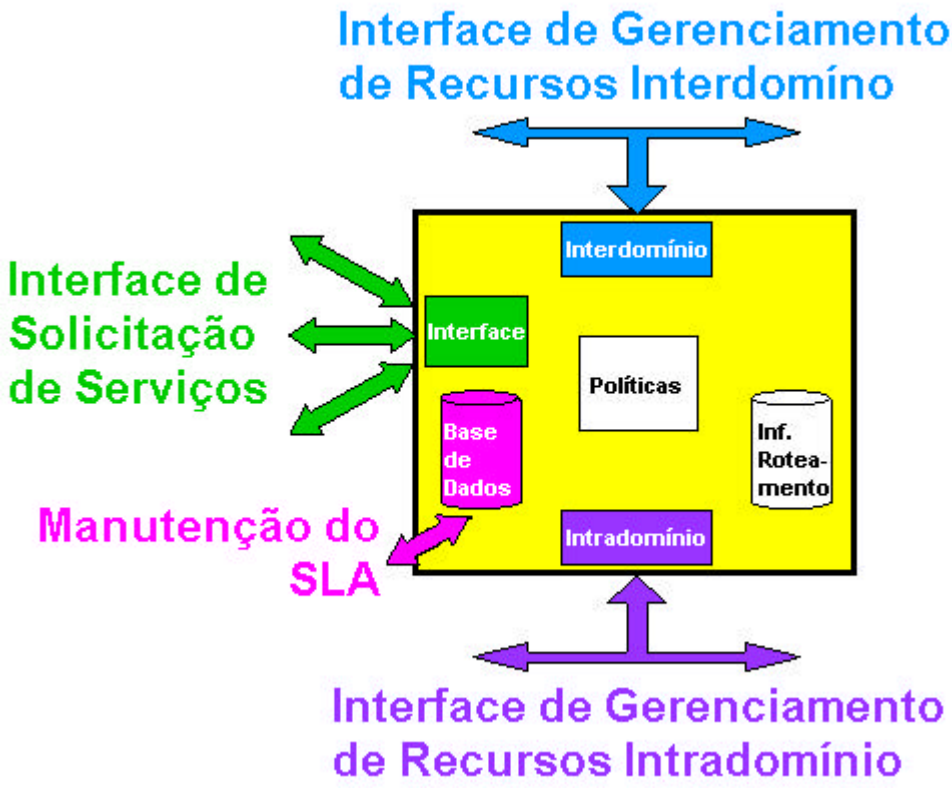

Figura 1: Interfaces presentes em um BB.

A abordagem de provisão de serviços centralizada, apresentada nesta seção, é baseada em uma arquitetura de dois níveis discutida em vários trabalhos [JACOBSON99b, CHIMENTO01, TERZIS99]. O primeiro e o segundo nível dessa arquitetura correspondem à gerência de recursos *intra* e *interdomínio*, respectivamente. O enfoque maior desses trabalhos está na padronização da interface *interdomínio* de forma a permitir o fornecimento de serviços fim-a-fim independente das tecnologias de provisão de QoS adotadas internamente por cada provedor de serviço.

Na abordagem distribuída, cada estação e roteador presente no provedor é dotado de um *negociador* próprio, responsável pelas decisões locais de *reserva de recursos*. Esses *negociadores* trocam informações de controle (*orquestração*) entre si, por meio de um protocolo de sinalização específico, de forma a determinar a melhor *reserva de recursos* possível no contexto do provedor, e que seja suficiente para o atendimento aos serviços solicitados. O protocolo mais utilizado na negociação baseada na abordagem distribuída é o RSVP (*Resource reSerVation Protocol*) [BRADEN97].

2.1.4 Manutenção de Contratos de Serviço

Após o estabelecimento do SLA, o provedor deve empregar mecanismos apropriados de forma a manter o nível da QoS solicitada pelo usuário. Caso o módulo de *escalonamento* tenha sido configurado de forma apropriada, cada fluxo do usuário terá uma parcela de utilização dos recursos disponíveis suficiente ao adequado fornecimento do serviço requisitado. Porém, toda essa configuração depende da manutenção, por parte dos usuários, do volume de tráfego acordado no contrato de serviço. De forma a garantir que os usuários não ultrapassem a sua cota de utilização de recursos, o provedor pode fazer uso de mecanismos específicos, como o de *policiamento*, por exemplo, que determina se o tráfego entrante no provedor corresponde ao caracterizado pelos usuários na fase de solicitação de serviço. O *policiamento* procura evitar que o transporte de pacotes que não estejam de acordo com o perfil de tráfego

informado comprometa aqueles fluxos bem comportados e que se atém aos limites contratados. O provedor de serviços pode utilizar-se das mais variadas *ações de policiamento* visando penalizar aqueles fluxos considerados não-conformes, como descartar pacotes ou encerrar abruptamente o SLA contratado, por exemplo.

Em paralelo à monitorização dos usuários, o provedor deve se auto-fiscalizar, de forma a ser capaz de responder prontamente a degradações ocasionais na QoS fornecida. Essas degradações podem ser motivadas por um dimensionamento pouco conservador dos recursos necessários, normalmente baseado em estatísticas de uso passadas, ou por falhas em algum componente do provedor (um enlace apresentando problemas, por exemplo). Funções de *sintonização* são acionadas nos casos em que a qualidade provida diverge daquela contratada pelos usuários, quando então uma nova *orquestração* de recursos se torna necessária. O provedor pode valer-se de inúmeras *ações de sintonização*, como, por exemplo, a escolha de subsistemas alternativos (por meio da função de *roteamento*) ou a *renegociação* do SLA com o usuário de forma a acordar um contrato que possa ser efetivamente mantido.

2.1.5 Término de Contratos de Serviço

Tendo cessado as condições que motivaram a solicitação do serviço previamente, o cliente deve informar ao provedor a sua intenção em finalizar o contrato estabelecido. Ao receber um pedido de término de contrato de serviço, que pode ser realizado, a exemplo da solicitação, por mecanismos estáticos ou dinâmicos, o provedor inicia o procedimento de *liberação dos recursos* que haviam sido reservados nos vários subsistemas envolvidos com o fornecimento do serviço correspondente. Nos casos em que o próprio contrato especifica o período em que o serviço será fornecido, os procedimentos de finalização poderão ser automatizados.

2.2 Modelo de Serviços Integrados

A especificação do modelo de serviços integrados, ou simplesmente *intserv* [BRADEN94], descreve as principais funções de provisão de QoS que devem estar presentes em um ambiente de fornecimento de serviços. O modelo apresenta, como uma das suas características mais particulares, a alta granularidade na reserva de recursos, realizada por fluxos individuais. Uma ênfase especial é reservada no modelo à especificação de serviços, que deve sempre ocorrer por intermédio de classes ou categorias padronizadas. Até o momento da escrita deste trabalho, as categorias de serviço *intserv* resumiam-se à *garantida* [SHENKER97] e à de *carga controlada* [WROCLAWSKI97].

A categoria de serviço *garantida*, ou simplesmente serviço *garantido*, como o próprio nome indica, fornece aos seus usuários uma garantia de retardo máximo fim-a-fim, sem perda de pacotes, sendo indicada às aplicações em tempo-real intolerantes [CLARK92]. Esse serviço representa o outro extremo em relação ao controle de retardo dos pacotes em comparação com o serviço de melhor-esforço. Os recursos necessários ao fornecimento do serviço *garantido* devem ser reservados exclusivamente, o que torna esse o serviço de maior custo quando comparado com os demais.

Uma generalização da categoria de serviços *garantidos*, não plenamente suportada pelo modelo *intserv*, é a dos serviços probabilísticos, onde a garantia é fornecida em termos estatísticos. Um exemplo possível seria uma garantia de retardo a 90% dos pacotes transportados. Os 10% restantes podem sofrer atrasos maiores do que o solicitado, dependendo das condições de uso momentâneas do sistema de comunicação. Como vantagem, a categoria de serviços probabilísticos, também chamada de serviços previsíveis ou ainda preditivos [CLARK92], permite o compartilhamento de um percentual dos recursos, o que diminui os custos de implementação. Por outro lado, as aplicações que

fazem uso desse serviço terão que ser mais tolerantes a variações ocasionais na qualidade fornecida pela infra-estrutura de comunicação.

As dificuldades de utilização do serviço de melhor-esforço no suporte às aplicações multimídia reside na degradação da QoS fornecida pelo provedor em situações de congestionamento. Em condições normais, esse serviço, mesmo sem fornecer garantias de QoS, apresenta uma qualidade considerada satisfatória pela grande maioria das aplicações multimídia que utilizam a Internet atualmente. A partir dessas constatações, foi proposta a categoria de serviço de *carga controlada*, ou simplesmente serviço de *carga controlada*, que fornece um retardo baixo, próximo ao oferecido por um serviço de melhor-esforço em uma infra-estrutura de rede controlada (sem congestionamento). A definição do que vem a ser um retardo baixo vai depender da forma com que os recursos necessários ao fornecimento do referido serviço foram reservados por cada provedor. Cabe a este controlar o número de usuários atendidos simultaneamente de forma a manter o nível de utilização da rede em um patamar relativamente baixo. O serviço de *carga controlada* é indicado às aplicações em tempo-real tolerantes, em especial às tolerante-adaptativas [CLARK92]. O serviço de *carga controlada* também é considerado um serviço probabilístico, onde o percentual de garantia é determinado pelo provedor, ou seja, sem controle por parte da aplicação.

2.2.1 Solicitação de Serviços

A solicitação de serviços no modelo *intserv* normalmente emprega procedimentos dinâmicos e que dependem de protocolos de sinalização específicos, sendo o RSVP o mais utilizado, atualmente, com esse propósito. A especificação do modelo descreve as estruturas de dados que devem ser empregadas na solicitação de um serviço. As informações de *caracterização do tráfego* correspondentes aos fluxos de dados a serem gerados pela aplicação transmissora são descritas na estrutura *tspec* (*traffic specification*). Uma outra estrutura de dados, denominada *rspec* (*reservation specification*), contém os *parâmetros de QoS* relativos ao serviço que está sendo solicitado. As estruturas

tspec e *rspec* compõem a *flowspec* (*flow specification*) de um serviço. A categoria de serviço de *carga controlada* não exige que a aplicação especifique *parâmetros de QoS*, uma vez que estes são definidos unilateralmente por cada provedor que disponibilize esse serviço. Dessa forma, para a solicitação de um serviço de *carga controlada*, a *flowspec* é constituída apenas pelas informações de *caracterização de tráfego* (*tspec*).

Através de interfaces apropriadas, a aplicação informa a categoria de serviço desejada (*garantida* ou *carga controlada*) juntamente com a estrutura *flowspec* devidamente preenchida. Em [BRADEN98] encontra-se definida uma API[5] que permite que essas informações possam ser mapeadas diretamente em estruturas específicas do protocolo RSVP.

2.2.2 Estabelecimento de Contratos de Serviço

O modelo *intserv* não determina a modalidade de SLA que deve ser empregada pelo provedor de serviços. Porém, dentre as implementações do modelo, aquela que tem recebido a maior atenção utiliza o protocolo RSVP em uma provisão de QoS dinâmica e distribuída. Para tornar mais objetiva a discussão que se segue sobre os mecanismos de provisão de QoS em um provedor *intserv*, será assumida a utilização do RSVP no processo de negociação, embora tudo que for discutido possa ser perfeitamente adaptado a outras modalidades de estabelecimento de SLAs, possivelmente empregando protocolos distintos.

O processo de configuração de um SLA *intserv* quando se utiliza o protocolo RSVP, devido a sua natureza dinâmica, ocorre em duas etapas (*two-pass reservation model*). Na primeira, os transmissores em uma *sessão de QoS*[6] anunciam as *características do tráfeg*o relativo aos fluxos que se encontram aptos

[5] *Application Program Interface*: uma interface constituída por primitivas que permitem às aplicações o acesso a funcionalidades específicas de um protocolo ou outra facilidade.
[6] Uma *sessão de QoS* identifica um ou mais fluxos de dados que tenham em comum os mesmos destinatários. Uma *sessão RSVP*, por exemplo, representa um tipo particular de *sessão de QoS*, caracterizada pela tupla endereço de destino, tipo e identificação da porta do protocolo de transporte utilizada na recepção dos fluxos transmitidos.

a gerar. Essas informações são formatadas em mensagens RSVP específicas, denominadas PATH, sendo encaminhadas, pelo provedor de serviços, aos destinatários da sessão. Cada *roteador RSVP*[7], no caminho entre os transmissores e os destinatários da sessão, deve armazenar, para uso posterior, a identificação das interfaces de rede por onde as mensagems PATH foram recebidas. Este procedimento visa permitir que as mensagens de solicitação de serviços, com a conseqüente reserva de recursos, sejam encaminhadas no sentido inverso por onde os fluxos anunciados serão efetivamente transmitidos. Adicionalmente, informações relativas aos recursos disponíveis no caminho escolhido para o transporte dos fluxos anunciados são acrescentadas às mensagens PATH pelos *roteadores RSVP* intermediários, em uma estrutura dados *intserv* denominada *adspec* (*advertisement specification*). Essas informações podem ser utilizadas pelos destinatários da *sessão de QoS* como uma estimativa das condições de utilização momentânea dos recursos do provedor, servindo de base para a escolha de serviços e *parâmetros de QoS* com maior probabilidade de serem fornecidos pelo sistema de comunicação.

A etapa seguinte ao processo de estabelecimento de um SLA compreende a efetiva solicitação do serviço, quando a *flowspec* correspondente é formatada, sendo, em seguida, transportada em mensagens RSVP do tipo RESV aos transmissores dos fluxos selecionados pela aplicação receptora. As mensagens RESV podem conter, além das informações relativas ao serviço solicitado, dados necessários à autenticação do usuário no provedor. Essas informações são repassadas a um módulo especial de *controle de políticas* do provedor, normalmente presente nos *roteadores RSVP*. A especificação do modelo *intserv* deixa totalmente a cargo do projetista a implementação desse módulo, tornando livre a estruturação da base de direitos de acesso ou se a mesma será centralizada ou distribuída.

[7] Roteador capaz de capturar e processar as mensagens de sinalização RSVP de forma a participar efetivamente do processo de negociação *intserv*.

Uma vez tendo sido verificadas as políticas de acesso do provedor, o *roteador RSVP* deve iniciar o processo de *controle de admissão*, responsável por determinar se a realização do serviço solicitado não resultará na degradação da QoS dos fluxos previamente aceitos e transportados no momento pelo roteador em questão. Com esse objetivo, o *roteador RSVP*, a partir dos parâmetros de tráfego e de QoS contidos na estrutura *flowspec*, quantifica os recursos necessários à realização do serviço solicitado. Os mecanismos de *mapeamento* são responsáveis por traduzir os parâmetros contidos na *flowspec* em outros específicos ao ambiente de provisão de QoS do roteador, como páginas de memória a serem alocadas, tempo estimado de uso do processador, percentual da banda disponível a ser reservado, tamanho do *buffer* a ser alocado na fila do escalonador de pacotes, etc. Estes parâmetros específicos dependem tanto do *sistema operacional* do roteador como da *tecnologia de sub-rede* empregada. A especificação *intserv* não descreve os mecanismos de *controle de admissão* ou as estratégias de *mapeamento* que devem ser empregadas, ficando a cargo do projetista a efetiva implementação dessas funções. Na arquitetura proposta no presente trabalho, como se verá, as estratégias de *controle de admissão* e de *mapeamento* são deixadas, propositadamente, em aberto, objetivando permitir a adaptação da mesma a várias configurações possíveis resultantes do uso combinado de diferentes modelos de serviço, sistemas operacionais ou tecnologias de sub-rede.

Seguindo-se à etapa de *controle de admissão*, tem-se a efetiva *reserva de recursos* e *configuração* dos mecanismos de encaminhamento do roteador (*classificador* e *escalonador*). A reserva permite que uma porção dos recursos disponíveis no roteador possa ser associada a um ou mais fluxos de forma a ser possível a efetiva provisão da QoS solicitada pelos usuários. Da mesma forma que o *controle de admissão*, a *reserva de recursos* depende diretamente das características particulares do ambiente de provisão de QoS (sistema operacional e tecnologia de sub-rede), sendo reservado ao projetista a determinação de como será realizada a configuração apropriada do referido ambiente.

O mecanismo de *classificação* objetiva identificar o fluxo de dados ao qual cada pacote pertence. No modelo *intserv*, um fluxo de dados é determinado, caso seja utilizada a versão 4 do IP (IPv4), pela concatenação dos seguintes campos, presentes no cabeçalho dos pacotes: endereço de origem, tipo e porta de protocolo de transporte utilizados pelo transmissor. Acrescenta-se a esses campos o identificador da *sessão de QoS*. A versão 6 do protocolo IP (IPv6) introduziu o campo *flow label* objetivando permitir uma identificação dos fluxos de forma mais eficiente, pela simples leitura do endereço do transmissor juntamente com o campo de *flow label*. A determinação do fluxo de dados ao qual cada pacote pertence permite ao roteador fornecer um serviço de transporte apropriado, condizente com os parâmetros de QoS previamente assumidos nos SLAs contratados com os usuários.

Cabe ao módulo de *escalonamento* determinar o instante correto de encaminhamento dos pacotes de forma a ser mantida a QoS associada aos fluxos correspondentes. Para alcançar esse objetivo, o escalonador pode utilizar-se das mais variadas estratégias, como a mudança da classe de serviço ou mesmo o descarte direto daqueles pacotes identificados como fora do perfil de tráfego pelo módulo de *policiamento*, caso as filas de entrada estejam em condições críticas de utilização, por exemplo. Outra estratégia pode ser o atraso dos pacotes de melhor-esforço, de forma a ser priorizado o tráfego em tempo-real. O modelo *intserv* não especifica os algoritmos de *escalonamento* ou as estratégias que devem ser empregados pelos roteadores, contanto que sejam suficientes para o fornecimento do nível de QoS prometido. Esses algoritmos e estratégias constituem, na arquitetura proposta, em novos pontos de flexibilização do framework desenvolvido, como será posteriormente visto.

2.2.3 Manutenção de Contratos de Serviço

Um dos mecanismos empregados na manutenção de SLAs é o de *policiamento*, que determina se os fluxos de tráfego dos usuários se comportam conforme o especificado quando da solicitação do serviço. No modelo *intserv*,

tanto os *mecanismos* quanto as *ações de policiamento* fazem parte da definição dos serviços. Dessa forma, a especificação dos serviços *garantido* e de *carga controlada* determinam que o mesmo mecanismo utilizado na caracterização dos fluxos (o filtro *token bucket*, especificamente) também deve ser empregado no *policiamento* dos mesmos, devendo aqueles pacotes fora do perfil de tráfego ser tratados como de melhor-esforço.

Uma característica do protocolo RSVP particularmente útil na manutenção de SLAs é a troca periódica de mensagens de sinalização, devido ao emprego de estados de reserva temporários nos roteadores do provedor. Essa característica torna possível, por exemplo, a implementação de mecanismos automáticos de renegociação na presença de falhas ou de degradações da QoS acordada com o provedor de serviços.

2.3 Modelo de Serviços Diferenciados

O crescimento do número de fluxos a serem processados e transmitidos, com o conseqüente aumento da carga administrativa ou de sinalização, tornou inviável a aplicação do modelo *intserv* em provedores de serviço de maior porte, devido aos custos altos envolvidos. A necessidade de manutenção de uma grande quantidade de estados associados a fluxos que compartilhavam enlaces de alta velocidade, típicos em provedores de *backbone*, requereria a fabricação de roteadores cada vez mais eficientes e a preços competitivos, o que representa, ainda hoje, um grande desafio tecnológico.

Diante das dificuldades em se adequar o modelo *intserv* a provedores que processavam uma maior quantidade de fluxos, novas alternativas foram propostas [BLAKE97, KILKKI97, CLARK97, HEINANEN97, JACOBSON99b]. Muitas das idéias contidas nesses trabalhos acabaram influenciando a especificação do modelo de serviços diferenciados, ou simplesmente *diffserv* [BLAKE98]. Este modelo propõe a agregação de fluxos em *classes de serviço* como uma forma de reduzir o número de estados que devem

ser mantidos nos roteadores do provedor. A agregação de fluxos é realizada através da identificação da *classe de serviço* em um campo específico do cabeçalho dos pacotes IP, chamado de DS (*Differentiated Services*) [BAKER98b]. Este campo corresponde ao *tipo de serviço* ou *classe de tráfego* presentes nos cabeçalhos das versões 4 e 6 do protocolo IP, respectivamente.

Deve-se enfatizar que a agregação de fluxos proposta no *diffserv* corresponde a uma solução de compromisso objetivando viabilizar a aplicação de um modelo de serviços em provedores de maior porte. Em muitos casos, o mecanismo de agregação pode implicar na associação de um fluxo a uma classe que forneça um serviço cuja qualidade é melhor do que a requerida, o que reduz, obviamente, a eficiência da reserva de recursos no provedor.

No modelo *diffserv*, o mecanismo de *classificação* determina a *classe de serviço* que deve ser associada a cada fluxo, podendo ser baseado em vários campos do cabeçalho dos pacotes (endereços de origem e destino, tipo e porta do protocolo de transporte, etc), na chamada classificação MF (*Multi Field*), ou, simplesmente, no campo DS, sendo, neste caso, denominada de classificação BA (*Behavior Aggregate*). A *marcação* da *classe de serviço* no campo DS é, normalmente, realizada pelo roteador de entrada do domínio de serviços *diffserv*[8], o chamado ER (*Edge Router*). Porém, quando a leitura de alguns dos campos do cabeçalho dos pacotes não pode ser realizada pelos roteadores do domínio, o que ocorre quando são utilizados mecanismos de criptografia [ATKINSON98], por exemplo, cabe ao próprio usuário efetuar a *marcação*[9]. Outras situações também podem requerer que a *marcação* seja realizada pelo usuário, como quando são utilizadas portas de transporte ou endereços IP voláteis[10], por exemplo. Em ambos os casos, o valor da *classe de serviço* a ser *marcado* no campo DS deve ser informado pelo provedor *diffserv*, uma vez que o usuário desconhece essa informação *a priori*. A exceção ocorre quando são utilizados valores universais para o referido campo, os chamados GWSIDs (*Global Well-known Services IDs*),

[8] *Router-marking*.
[9] *Host-marking*.
[10] Uso do DHCP (*Dynamic Host Configuration Protocol*) [DROMS93].

normalmente oriundos de alguma órgão de padronização (IETF[11], por exemplo). Os pacotes, uma vez identificados pela *marcação* da *classe de serviço* no campo DS, receberão um tratamento diferenciado nos roteadores do provedor de serviços *diffserv*.

A divisão de funcionalidade entre os roteadores do provedor de serviços é uma outra característica particular do modelo *diffserv*. Cabe aos ERs as tarefas mais complexas de *classificação*, *marcação* e *policiamento* do tráfego entrante no domínio de serviços *diffserv*, também referenciado por *domínio DS*. Opcionalmente, funções de *moldagem*[12] do tráfego de saída do domínio também podem ser acrescentadas ao ER. A inclusão dessas funções de processamento mais intenso nos roteadores de borda não introduz problemas de escalabilidade ao modelo, uma vez que é esperado, nessa região, um menor volume de tráfego. Por outro lado, no núcleo do *domínio DS*, onde as velocidades e a quantidade de fluxos são proporcionalmente maiores, o processamento realizado pelos roteadores internos ou TRs (*Transient Routers*) é simplificado ao máximo, sendo eliminada a função de *policiamento* e *moldagem* dos fluxos, além de reduzida a complexidade da função de *classificação*, que identificará as classes de serviço pela leitura direta do campo DS previamente marcado (classificação BA).

2.3.1 Solicitação de Serviços

Os contratos de serviços *diffserv*, quando comparados com aqueles do modelo *intserv*, são mais abrangentes e, normalmente, mais estáveis, apresentando um tempo de vida maior. Conforme visto na Seção 2.2, um SLA *intserv* limita-se aos aspectos técnicos da caracterização de um serviço, como o nível de garantia desejado, a descrição do tráfego correspondente aos fluxos e os parâmetros de QoS. Um SLA *diffserv*, no entanto, considera relevante outras características de um serviço, como a forma de cobrança, as penalidades

[11] *Internet Engineering Task Force*: órgão internacional que congrega representantes da indústria e comunidade acadêmica na busca da solução de problemas, elaboração de recomendações de protocolos, dentre outros assuntos relacionados à tecnologia TCP/IP.
[12] A moldagem formata um fluxo de dados de acordo com um perfil de tráfego especificado.

contratuais nos casos de interrupção, etc. Porém, no presente trabalho, será levada em consideração apenas a porção técnica do SLA *diffserv*, denominada SLS (*Service Level Specification*), que pode especificar, dentre outros requisitos:

- a disponibilidade do serviço, descrevendo ações a serem realizadas em situações de descontinuidade do mesmo;
- os mecanismos de autenticação e criptografia empregados;
- as restrições de rotas a serem utilizadas no encaminhamento do tráfego agregado;
- as formas de monitorização e auditoria da qualidade do serviço fornecida.

Além destes parâmetros, o SLS contém ainda o TCS (*Traffic Conditioning Specification*), responsável por descrever:

- os parâmetros de *caracterização do tráfego*;
- as *ações de policiamento* aplicadas aos pacotes não conformes com o perfil de tráfego prometido;
- os *parâmetros específicos de QoS*, como vazão esperada, probabilidade de descarte, retardo máximo, etc;
- o escopo do serviço, através de restrições de entrada e de saída por onde os fluxos serão transportados.

De acordo com o que foi descrito na Seção 2.2.1, o modelo *intserv* emprega categorias de serviço padronizadas, estando especificadas, atualmente, as categorias *garantida* e *carga controlada*. O uso de categorias padronizadas limita os serviços que podem ser fornecidos, dando pouco espaço à criatividade e competição entre os provedores. O modelo *diffserv* não padroniza serviços, especificando apenas comportamentos de encaminhamento ou PHBs (*Per Hop Behaviors*). Um PHB descreve como será realizado o encaminhamento dos pacotes pertencentes a uma mesma classe em cada roteador, sendo as unidades

básicas utilizadas na definição dos serviços oferecidos pelo provedor. A combinação de PHBs, juntamente com os demais parâmetros de caracterização do SLA, é que define o serviço a ser efetivamente fornecido pelo provedor *diffserv*. Devido à ausência de padronização de serviços, o modelo *diffserv* permite que diferentes provedores concorram entre si no fornecimento de funcionalidades que agreguem vantagens aos usuários, como disponibilidade, facilidades de restauração automática na presença de falhas, dentre outras. Cabe ao usuário a escolha do provedor que forneça o conjunto de serviços e funcionalidades que melhor atenda às necessidades específicas das suas aplicações.

Até o momento da escrita deste trabalho, dois PHBs encontravam-se padronizados pelo IETF: o encaminhamento expedido ou EF (*Expedited Forwarding*) [JACOBSON99a], e o assegurado ou AF (*Assured Forwarding*) [BAKER99]. O PHB EF é recomendável ao fornecimento de serviços com baixo retardo e variação, taxa de erros controlada e largura de banda assegurada. Serviços com estas características são normalmente referenciados, na literatura, como serviços de linha virtual alugada ou VLL (V*irtual Leased Line*) [BERNET99] ou serviços *premium* [JACOBSON99a]. A implementação deste PHB procurará eliminar as filas nos roteadores, o que é obtido através do uso de uma taxa de saída mínima maior ou igual do que a taxa de entrada agregada máxima. A disciplina de escalonamento usada deve garantir a proteção de fluxo, ou seja, o tráfego de uma classe de serviço não deve interferir no de outra. Como exemplo de aplicações que façam uso deste serviço, podemos citar: telefonia na Internet, redes privadas virtuais ou VPNs (*Virtual Private Networks*), aplicações em tempo-real intolerantes [CLARK92], dentre outras.

O PHB AF define, na realidade, um grupo formado por várias modalidades independentes de encaminhamento. O grupo AF permite a implementação de serviços com diferentes níveis de garantia de QoS, como os serviços probabilísticos, por exemplo. No âmbito de cada classe AF, pacotes são marcados, adicionalmente, por um valor de precedência de descarte. A probabilidade de entrega de um pacote servido por um PHB AF depende, portanto, de três fatores:

- classe AF ao qual pertence: quanto maior o valor, maior a probabilidade;
- tráfego agregado atual da referida classe;
- valor de precedência de descarte do pacote: quanto menor a precedência maior a probabilidade.

Cada pacote IP é identificado, no campo DS, por um valor AF_{ij}, onde i define a classe AF e j o valor de precedência de descarte do pacote dentro da referida classe. Até momento de escrita deste trabalho, quatro classes AF encontravam-se definidas, cada qual podendo ter até três níveis de precedência. Os parâmetros de qualidade a serem aplicados a cada uma dessas classes são definidos de forma independente por cada provedor. A única restrição da especificação descrita em [BAKER99] refere-se à preservação da ordem de garantia e de descarte no encaminhamento dos pacotes pertencentes ao grupo de serviços AF fornecidos pelo provedor.

Uma grande variedade de serviços diferenciados podem ser fornecidos usando-se o grupo de encaminhamento assegurado, como, por exemplo, os chamados *serviços olímpicos*, com as classes *ouro*, *prata* e *bronze* [JACOBSON99a]. Obviamente, de forma a garantir que o serviço da classe *ouro* apresente um desempenho de encaminhamento melhor do que o oferecido pelos serviços das demais classes, um dimensionamento apropriado dos recursos deve ser feito baseado nas demandas de tráfego esperadas para cada uma das classes que o provedor deseja suportar.

2.3.2 Estabelecimento de Contratos de Serviços

O estabelecimento de um SLA *diffserv*, a exemplo dos contratos *intserv*, também pode ser realizado a partir de uma abordagem centralizada ou distribuída. Um dos poucos trabalhos que segue a abordagem distribuída [BLAKE97] procurar estender o uso do protocolo RSVP no processo de negociação. Pela criação de *sessões de agregação*, mensagens PATH podem ser

enviadas informando as *características do tráfego* correspondente a um conjunto de fluxos. A partir dessas informações e dos requisitos próprios dos receptores, mensagens RESV podem ser formatadas. O valor a ser escrito no campo DS pode ser informado ao roteador de borda na entrada do domínio *diffserv*, caso seja responsável pela marcação, por intermédio do objeto DCLASS [BERNET00], inserido nas mensagens RESV.

Na abordagem distribuída, todos os roteadores do domínio *diffserv* participam do processo de negociação de QoS, sendo responsáveis por interpretar as mensagens do protocolo de negociação e realizar as configurações apropriadas. Por outro lado, na abordagem centralizada, os roteadores do domínio podem ser especializados unicamente nas tarefas de reconfiguração e encaminhamento de pacotes, o que permite a fabricação de equipamentos mais eficientes e a um menor custo. Talvez esse seja o principal fator responsável pela maior quantidade de experimentos práticos de provisão de serviços *diffserv* baseados na abordagem centralizada, sendo esta a configuração adotada pelo grupo de trabalho QBone da Internet2 [QBONE01]. Os esforços desse grupo estão voltados à definição de um protocolo de solicitação de serviços *interdomínios*, baseado em uma arquitetura de provisão de QoS centralizada, denominado, inicialmente, de SIBBS (*Simple Interdomain Bandwidth Broker Signalling*) [CHIMENTO01]. De forma a tornar mais objetiva as discussões que se seguem sobre as funções e mecanismos empregados em uma provisão de serviços *diffserv*, será assumida a configuração centralizada, baseada no protocolo SIBBS, embora outras configurações também possam ser utilizadas.

De acordo com a nomenclatura SIBBS, uma solicitação de serviço *diffserv* é chamada de RAR (*Resource Allocation Request*). Objetivando simplificar o processo de negociação, os serviços especificados em uma mensagem RAR devem ser, inicialmente, padronizados, sendo empregados, para tanto, valores universalmente aceitos (GWSIDs) no campo DS. Dependendo do serviço solicitado, diferentes SPOs (*Service Parametrization Objects*) devem ser apropriadamente formatados nas mensagens RAR.

Após o recebimento de uma mensagem RAR, o BB do provedor deve ser capaz de autenticar o usuário[13] que originou a solicitação de serviço. O protocolo SIBBS reservou os campos *Sender ID* e *Sender Signature* na mensagem RAR de forma a tornar possível a verificação das credencias do usuário. A seguir são descritas outras estruturas previstas pelo SIBBS e que permitem a caracterização do SLA solicitado:

- *Source Prefix*: identificação dos transmissores do agregado de fluxos;
- *Destination Prefix*: identificação dos receptores do agregado de fluxos;
- *Start Time*: início de operação do serviço;
- *Stop Time*: término do serviço;
- *Flags*: permitem a customização do SLA, como a solicitação de serviços pagos pelo receptor (*collect call*), se o provedor deve sugerir outros parâmetros de QoS caso não possua recursos suficientes para o atendimento ao serviço solicitado, etc.
- GWSID: identificação do código do serviço padrão desejado;
- SPO (*Service Parametrization Objects*): parâmetros específicos que permitem a caracterização do serviço (*descrição do tráfego* e dos *parâmetros de QoS*)

Os campos *source* e *destination prefix* em conjunto definem o escopo do serviço, identificando os fluxos associados. O período delimitado pelos campos *start* e *stop time* torna possível o agendamento de serviços, o que permite ao provedor prever, com um maior grau de precisão, a disponibilidade dos seus recursos no tempo. A partir dessas informações, pode-se implementar uma estratégia de negociação que informa ao usuário a partir de que instante o serviço

[13] Neste contexto, o termo usuário é utilizado no seu sentido mais amplo, podendo representar um outro provedor de serviços (*upstream domain*).

solicitado poderá ser fornecido, ao invés do simples retorno de uma mensagem de indeferimento padrão, por exemplo.

Após a autenticação do usuário, o BB deve iniciar o procedimento de *controle de admissão* do serviço solicitado com base nos parâmetros contidos na mensagem RAR. Na abordagem centralizada, a admissão de um serviço é, normalmente, dividida em duas etapas: a *intra* e a *interdomínio*. No *controle de admissão intradomínio*, o BB deve determinar, com base nas informações de disponibilidade de recursos, e auxiliado pela função de *roteamento*, os roteadores do domínio que serão empregados no encaminhamento dos fluxos relacionados com o serviço solicitado. Esse processo de admissão, a exemplo do que foi discutido no modelo *intserv*, depende das funções de *mapeamento*, que traduzem os parâmetros de caracterização do serviço, contidos nos objetos SPO da mensagem RAR, em valores que possam ser interpretados pelos roteadores do domínio, como percentual da banda a ser utilizado, quantidade de memória a ser reservada, etc.

O fornecimento de serviços verdadeiramente fim-a-fim implica na interação entre os vários provedores que participam no encaminhamento dos fluxos sobre os quais a QoS solicitada será aplicada. Desta forma, após a realização do *controle de admissão intradomínio*, o provedor *diffserv* deve determinar, com base nas informações de escopo do serviço, e auxiliado novamente pela função de *roteamento*, com quais domínios *diffserv* adjacentes deverá interagir visando a configuração apropriada do serviço fim-a-fim solicitado. É função do BB repassar a mensagem RAR recebida aos domínios *diffserv* adjacentes (também chamados de *downstream domains*) identificados como diretamente relacionados com o fornecimento do serviço.

Os procedimentos de *autenticação*, *controle de admissão intradomínio* e encaminhamento das mensagens RAR repetem-se em cada provedor *diffserv*, até que a solicitação de serviço seja recebida por um domínio identificado como terminal, de acordo com o especificado no escopo do serviço. Neste instante, caso o *controle de admissão intradomínio* tenha retornado uma

resposta positiva, o BB inicia o processo de *reserva de recursos* e configuração dos roteadores do domínio. Devido à divisão de funcionalidade característica do *diffserv*, a configuração dos roteadores será diferenciada entre os de borda (ERs) e aqueles localizados no núcleo (TRs) do provedor. Independente da localização do roteador, no entanto, os módulos de *classificação* e *escalonamento* devem ser, obrigatoriamente, configurados. É precisamente na configuração desses módulos que o valor do campo DS a ser associado ao serviço será informado pelo BB aos roteadores do domínio. Os módulos de *marcação*, *policiamento* e, possivelmente, de *moldagem de tráfego*, devem ser configurados nos roteadores de borda do provedor *diffserv* com base nos critérios especificados no SLA. Todas as informações de configuração dos roteadores do provedor são comunicadas pelo BB a partir de protocolos de gerência de recursos apropriados, como o COPS e o Diameter, para citarmos os mais discutidos atualmente no contexto dos serviços *diffserv*.

Devem ser previstos os casos em que um dos domínios *diffserv* não possua recursos suficientes para a realização do serviço solicitado. Essas situações são reportadas ao domínio que repassou a mensagem de solicitação, o chamado *upstream domain*, através de uma mensagem denominada RAA (*Resource Allocation Answer*). Cabe ao *upstream domain* escolher um caminho alternativo, envolvendo, possivelmente, outros provedores *diffserv*, objetivando o fornecimento do serviço.

Nas situações de disponibilidade de recursos, uma resposta positiva, também formatada em uma mensagem RAA, deve ser retornada ao *upstream domain*. O processo de configuração de cada domínio *diffserv* e encaminhamento das mensagens RAA repete-se na seqüência inversa a percorrida pelas mensagens de solicitação de serviços. Finalmente, em um determinado instante, a mensagem de confirmação deve ser recebida pelo domínio que originou a solicitação de serviço. Este, após a configuração apropriada dos roteadores, sinaliza o estabelecimento do SLA ao usuário.

2.3.3 Manutenção de Contratos de Serviços

Após o estabelecimento do SLA, o provedor *diffserv* deve iniciar o processo de monitorização dos agregados de fluxos do usuário de forma a permitir que medidas preventivas possam ser executadas caso o mesmo ultrapasse a cota de tráfego especificada no contrato, podendo, desta forma, prejudicar o desempenho de outros SLAs em operação. Conforme mencionado na Seção 2.3.1, os mecanismos de *policiamento* no *diffserv* podem ser definidos no momento da solicitação de serviços, juntamente com outras customizações, como as ações de restauração na presença de falhas, por exemplo.

2.4 Serviços Integrados Sobre Diferenciados

A diversidade de provedores de serviço na Internet abre a possibilidade de modelos diferentes para provisão de QoS terem que ser combinados na realização de serviços fim-a-fim. Essas combinações devem causar o menor impacto possível no ambiente de solicitação de serviços do usuário final, não devendo ser percebidas mudanças significativas no nível da QoS contratada pelo mesmo. Para tanto, as estratégias de negociação dos diferentes modelos devem ter pontos de integração que permitam a interpretação e mapeamento corretos dos parâmetros de QoS associados aos serviços solicitados. A solução final escolhida deve ainda resultar em um uso eficiente dos recursos do sistema de comunicação, sem introduzir, ao mesmo tempo, limitações de aplicabilidade que possam restringir o seu escopo de utilização.

A configuração conhecida como *serviços integrados sobre diferenciados*, discutida em inúmeros trabalhos relacionados à QoS na Internet [MAMELI00, CHARNY01], emprega o *intserv* nas redes de acesso e o *diffserv* nos provedores de *backbone*. A utilização do *intserv* nas redes de acesso é justificada pela melhor razão *utilização dos recursos* / *custo* obtida com a reserva por fluxo característica desse modelo. Outro fator de relevância para a adoção do referido modelo nas redes de acesso é o maior número atual de aplicações adaptadas a

algum mecanismo de solicitação de serviços *intserv*, normalmente empregando uma das diversas APIs existentes no mercado [SOLSTICE01, PCRSVP01, GQoS01]. Por outro lado, conforme previamente abordado na Seção 2.3, a aplicação do modelo *intserv* mostrou-se inviável nos grandes provedores de *backbone*, onde devem ser procuradas soluções mais escaláveis, como a *diffserv*.

Uma outra vantagem obtida com a escolha de uma solução *intserv* sobre *diffserv* é a possibilidade de se implementar mecanismos de *policiamento* por fluxo e autenticação de usuários individuais. Devido à agregação dos fluxos transmitidos pelos usuários em classes de serviço, o *policiamento diffserv* é realizado, normalmente, tendo em vista a monitorização de um conjunto de fluxos como um todo. Dessa forma, uma única fonte transmissora mal-comportada, gerando pacotes fora do perfil de tráfego esperado, pode comprometer o serviço fornecido a todo o agregado de fluxos ao qual pertence, uma vez que o *policiamento diffserv* pode inserir atrasos ou mesmo descartar quaisquer pacotes da classe de serviço, de forma indiscriminada. O modelo *intserv* fornece um mecanismo que permite às aplicações especificarem o tráfego associado aos seus fluxos, podendo essas informações ser empregadas na implementação de uma função de *policiamento* mais granular. Mecanismos semelhantes podem ser utilizados na determinação das credenciais de um usuário, podendo ser implementado um controle de acesso mais fino e individual. Em um provedor *diffserv*, a autenticação de um usuário refere-se, normalmente, à verificação dos direitos de todo um domínio (*upstream domain*) em solicitar determinados serviços ao provedor.

Dois cenários podem ser vislumbrados na configuração *intserv* sobre *diffserv*. No primeiro, ilustrado na Figura 2, o provedor *intserv*, representado por uma rede corporativa do usuário, se adequa ao *diffserv*, sendo responsável por implementar os mecanismos de *mapeamento* de parâmetros de serviço e *negociação* com o domínio *diffserv*. Nessa configuração, é recomendável que o provedor *intserv* realize o *policiamento* dos seus fluxos antes de encaminhá-los ao domínio *diffserv*. Adicionalmente, caso um SLA estático tenha sido configurado entre os domínios, é função do provedor *intserv* realizar a admissão dos serviços

solicitados a partir do controle dos percentuais de utilização do contrato acordado. Apesar de não ser requerido, nessa configuração, que o provedor de serviços diferenciados interprete as mensagens de *negociação intserv*, espera-se que o mesmo as encaminhe corretamente de forma a não permitir que ocorram interferências no processo de *negociação* com outros domínios. A Figura 2 ilustra, por meio de setas, o intercâmbio de mensagens de *negociação intserv* entre as redes corporativas. Finalmente, o domínio *intserv* deve ser capaz de estimar alguns parâmetros de desempenho (retardo máximo, largura de banda mínima) do provedor de serviços diferenciados de forma a permitir a atualização da estrutura *adspec* empregada na *negociação intserv* e informada aos receptores dos fluxos de tráfego. Esse talvez seja um dos pontos de maior dificuldade de implementação dessa configuração, uma vez que as informações necessárias a uma estimativa adequada dos referidos parâmetros de desempenho não se encontram, normalmente, disponíveis ao provedor de serviços integrados.

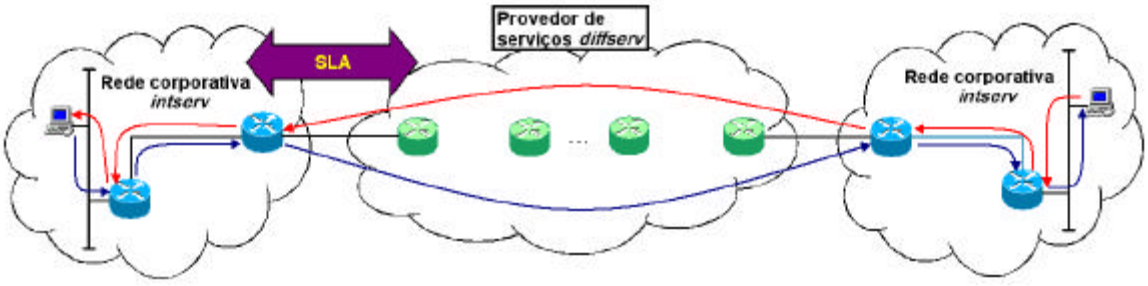

Figura 2: *Intserv* sobre *diffserv* - cenário 1.

Em um outro cenário, ilustrado na Figura 3, é o provedor *diffserv* que se adequa ao *intserv*, implementando os mecanismos apropriados de *negociação* e *mapeamento* de parâmetros de serviço. Nessa configuração, o domínio *diffserv* deve ser capaz de simular o funcionamento de um roteador *intserv*, inclusive no que se refere à atualização dos parâmetros de desempenho do provedor (estrutura *adspec*), manutenção de estados por fluxo, *policiamento*, *autenticação de usuários* e encaminhamento das mensagens de *negociação*. Como pode ser observado na Figura 3, o provedor *diffserv* participa ativamente em todo o

processo de negociação *intserv*, interpretando as mensagens do protocolo empregado para tanto.

Figura 3: *Intserv* sobre *diffserv* - cenário 2.

Um aspecto crítico em uma configuração que envolva mais de uma modalidade de provisão de QoS é o *mapeamento* entre parâmetros de serviço. Em [CHARNY01] são sugeridos esquemas de *mapeamento* entre serviços *intserv* e *diffserv*. Na arquitetura proposta, estratégias de *mapeamento* entre serviços são configuradas pelo projetista do sistema, responsável pelo preenchimento deste que constitui um outro ponto de flexibilização do framework desenvolvido, como será posteriormente visto.

2.5 Modelos de Serviços e Tecnologias de Sub-rede

O sucesso da implementação de um modelo de serviços com QoS depende da capacidade das estações e roteadores em reservar e configurar apropriadamente recursos específicos, como páginas de memória, tempo de utilização do processador, *buffers* de entrada e saída no escalonador, percentual da largura de banda, etc, nos seus subsistemas de processamento (sistema operacional) e de comunicação (sub-rede). Esses recursos são configurados por meio de APIs apropriadas, como:

- a API TC (*Traffic Control*) [GQoS01], que permite a configuração dos mecanismos de controle de tráfego

(*classificação*, *policiamento*, *escalonamento*) em estações baseadas no sistema operacional Windows 2000;

- as novas chamadas ao sistema operacional Linux definidas em [CHANDRA00], que possibilitam um maior grau de controle sobre o escalonamento de processos realizado pela CPU, bem como o acesso a disco nesses ambientes;
- a recomendação ITU I.150 [ITU99], que define primitivas para a abertura e manutenção de VCCs (*Virtual Channel Connections*) ATM a partir de parâmetros de QoS;
- as interfaces que permitem a configuração de LSPs (*Label Switched Paths*) MPLS sujeitos a restrições de QoS, como as disponibilizadas pelos protocolos RSVP-TE [AWDUCHE01a] e CR-LDP [JAMOUSSI01], por exemplo.

Para a utilização dessas APIs, os parâmetros de caracterização do serviço solicitado pelo usuário devem ser mapeados em valores condizentes com o nível de abstração dos referidos subsistemas. Favorecendo a esse aspecto em particular, a arquitetura proposta neste trabalho é recursiva, no sentido que as mesmas funções genéricas de provisão de QoS, identificadas nos frameworks desenvolvidos, como o *mapeamento*, por exemplo, podem ser aplicadas em vários níveis de abstração. Essa característica facilita o trabalho do projetista, que pode implementar as funções específicas de cada nível a partir de uma mesma base conceitual, facilitando o reuso de estrutura e código. Em uma implementação de um roteador MPLS por *software* em um ambiente Linux, por exemplo, o projetista pode se valer dos frameworks da arquitetura proposta na definição de estruturas e mecanismos mais genéricos, que possam, futuramente, ser traduzidos em chamadas a APIs específicas, de acordo com o ambiente, de forma a controlar a reserva de recursos tanto a nível de sistema operacional quanto a nível de sub-rede. Desta forma, a implementação resultante poderia ser adaptada mais facilmente a outras combinações de sistema operacional / sub-rede, bastando que sejam alteradas as chamadas às APIs específicas.

Sob a coordenação do grupo ISSLL (*Integrated Services over Specific Link Layers*) [ISSLL01] do IETF, vários trabalhos de pesquisa têm procurado abordar os aspectos particulares de *mapeamento* e *gerenciamento de recursos* quando são empregados modelos de serviço com QoS em redes IP sobre tecnologias de sub-rede específicas, como: Ethernet [BAKER00a, CRAWLEY00, GHANWANI00], PPP [BORMANN99], ATM [BERGER98a, BERGER98b, BORDEN98, BAKER98a] ou MPLS [CHEVAL01]. No que concerne às configurações mais adequadas a grandes provedores de serviços na Internet, a tecnologia MPLS (*MultiProtocol Label Switching*) [CALLON01] tem recebido, nos últimos anos, um destaque especial. Por sua natural integração com o IP, aliado a um encaminhamento eficiente proporcionado pela comutação por rótulos, sem os conhecidos problemas das soluções baseadas na tecnologia ATM[14], o MPLS tem sido apontado como o mecanismo mais apropriado ao transporte de fluxos com requisitos especiais de QoS na Internet, em especial nos grande provedores. Por essa e outras razões, o MPLS foi escolhido, no presente trabalho, para ilustrar os aspectos de interoperação entre modelos de QoS e tecnologias de encaminhamento no nível das sub-redes.

2.5.1 MPLS

O modelo tradicional de *roteamento IP*[15], baseado na determinação do próximo roteador para onde cada pacote deve ser enviado a partir do endereço de destino, mostrou-se inadequado nos grandes provedores de rede da Internet, onde o maior volume de fluxos transportados exige mecanismos de encaminhamento mais eficientes. Como uma alternativa a esse modelo, o *roteamento explícito* sugere a determinação *a priori* da seqüência completa dos roteadores que compõem uma rota. Calculada pelo roteador localizado na borda de entrada do provedor, a referida seqüência, chamada de *rota explícita*, é

[14] As operações de segmentação e remontagem de células ATM em enlaces de altíssima velocidade (acima de OC-48, por exemplo) tornaram-se um obstáculo tecnológico. Adicionalmente, o desperdício de banda devido a alta carga de sinalização ATM (*cell-tax overhead*) mostrou-se considerável nesse enlaces, não podendo mais ser ignorado.
[15] Normalmente chamado de *hop-by-hop routing*.

inserida no cabeçalho dos pacotes transportados pelo provedor. Dessa forma, a determinação do caminho a ser seguido por cada pacote é realizada pela simples leitura da *rota explícita*. Porém, a inclusão de uma seqüência de tamanho variável em cada pacote transportado representa um custo adicional de processamento e largura de banda. Para minimizá-lo, substituiu-se a referida seqüência por um rótulo pequeno e de tamanho fixo, no mecanismo de encaminhamento conhecido como *comutação por rótulos*, empregado, há bastante tempo, nas redes frame relay e ATM.

A tecnologia MPLS representa um esforço de padronização, organizado pelo IETF[16], das diversas técnicas de *comutação por rótulos* em redes IP[17], implementadas, nos últimos anos , através de soluções proprietárias de diversos fabricantes como a *IP Switching* (Ipsilon), a ARIS (IBM) e a *Tag Switching* (Cisco), para citarmos algumas. Seu funcionamento é baseado na identificação dos pacotes que são transportados por uma mesma rota ou, em outras palavras, que pertençam à mesma classe de encaminhamento ou FEC (*Forward Equivalence Class*). A identificação da FEC ao qual um pacote pertence é realizada pelo roteador de borda de entrada do *domínio MPLS*, responsável por acrescentar o rótulo ao mesmo. Os roteadores do domínio[18], chamados de LSRs (*Label Switching Routers*), na nomenclatura MPLS, encaminham os pacotes a partir da leitura e análise do rótulo identificado. Cabe aos LSRs de borda de saída do *domínio MPLS* a retirada do referido rótulo e a entrega dos pacotes originais ao destino.

É importante ressaltar que o rótulo MPLS pode assumir as mais variadas formas, dependendo do mecanismo de encaminhamento empregado na sua implementação, podendo ser representado por um par VPI / VCI ATM, um DLCI Frame Relay ou mesmo um novo cabeçalho (*shim-header*, como é chamado) acrescentado aos pacotes IP, caso sejam utilizadas diretamente

[16] O grupo [MPLS01] é responsável por reunir as propostas relacionadas a essa tecnologia objetivando-se chegar a um consenso que origine recomendações aos fabricantes.
[17] Apesar da ênfase maior ser dada às redes IP, o MPLS é multiprotocolo.
[18] Conjunto de roteadores sobre uma mesma administração que implementam o MPLS.

tecnologias de nível 2 como a Ethernet ou a PPP (*Point-to-Point Protocol*), por exemplo. A implementação de LSRs a partir de comutadores ATM, realizada pela atualização do *plano de controle*[19] dos referidos comutadores, tem como objetivo principal preservar o grande investimento inicial feito nessa tecnologia. Sendo essa uma solução temporária, a tendência natural, nos próximos anos, é a substituição gradativa desses equipamentos mistos por LSRs MPLS *puros*, com encaminhamento baseado em alguma tecnologia de transporte de quadros em altíssima velocidade, como PPP sobre SONET [MALIS99], por exemplo.

Independente do seu real formato, um rótulo MPLS sempre tem significado local em cada roteador, devendo a associação dos mesmos a FECs ser acordada entre LSRs vizinhos, o que pode ser feito estática ou dinamicamente. Na associação estática, os mapeamentos são configurados manualmente pelo administrador do domínio, em contraste com a associação dinâmica, onde um protocolo de sinalização é empregado com o mesmo propósito. O LDP (*Label Distribution Protocol*) [ANDERESSON01] é o protocolo recomendado pelo IETF na troca de associações de rótulos a FECs em provedores MPLS. Nos últimos anos, no entanto, protocolos originalmente concebidos para outros propósitos têm sido estendidos de forma a permitir a troca de associação de rótulos MPLS, como o BGP [REKHTER01] e o RSVP [AWDUCHE01a]. Quando as associações de rótulos a FECs possibilitam a comutação de pacotes entre dois LSRs, pode-se dizer que uma ou mais rotas, chamadas de LSPs (*Label Switched Paths*), encontram-se configuradas entre os mesmos.

Além de permitir um encaminhamento mais eficiente, baseado na comutação por rótulos, o MPLS apresenta outros atrativos que têm justificado o crescente interesse por essa tecnologia. O *roteamento baseado em restrições*

[19] O *plano de controle* é responsável, dentre outras funções, pelas decisões de roteamento. Em um comutador ATM, o *plano de controle* é baseado em protocolos de roteamento como o PNNI [ATMFORUM96], por exemplo. Para que esses comutadores possam desempenhar as funções de um LSR MPLS, o seu *plano de controle* deve ser capaz de implementar o protocolo de roteamento empregado pelo nível 3 (roteamento IP, por exemplo).

(*contraint-based routing*)[20], por exemplo, normalmente implementado através do mecanismo de *roteamento explícito*, é uma das facilidades disponibilizada pelo MPLS. Adicionalmente, rotas de contingência (*backup LSPs*) podem ser configuradas previamente em um domínio MPLS, fornecendo um mecanismo de reroteamento rápido (*fast re-route*) [ANDERSSON99] e uma maior confiabilidade às aplicações de missão-crítica, cada vez mais comuns na Internet. O *balanceamento de carga*, que permite a distribuição de fluxos de tráfego por diferentes rotas de forma a serem evitados pontos de congestionamento, é um dos mecanismos de *engenharia de tráfego* que também podem ser implementados pela utilização da tecnologia MPLS. Finalmente, a possibilidade de configuração de LSPs sujeitos a restrições de QoS tornou possível o emprego do MPLS no suporte às novas aplicações multimídia, com a vantagem adicional da facilidade de integração dessa tecnologia com o IP e os modelos *intserv* e *diffserv*.

2.5.2 Serviços Diferenciados sobre MPLS

De forma a permitir a configuração de LSPs sujeitos a restrições, o protocolo LDP foi estendido e renomeado para CR-LDP (*Contraint-based Routed* LDP) [JAMOUSSI01]. A partir da inserção de parâmetros de especificação de tráfego e QoS nas mensagens LDP de requisição de rótulos, o CR-LDP tornou possível a reserva de recursos em LSPs MPLS. Esses parâmetros permitem que sejam requisitados valores específicos de taxa de pico ou PDR (*Peak Data Rate*), taxa média ou CDR (*Commited Data Rate*), variação máxima do retardo (*frequency*), dentre outros. Processo inverso, no entanto, ocorreu com o protocolo RSVP. Na sua proposição inicial, esse protocolo era utilizado unicamente na configuração de rotas sujeitas a restrições de QoS. Objetivando-se a sua

[20] Uma generalização do *roteamento com QoS* (*QoS routing*) [CRAWLEY98], esse mecanismo de roteamento deve levar em consideração, além das tradicionais informações de alcancabilidade da rede, algum conhecimento sobre a disponibilidade de recursos, requisitos de QoS do fluxo ao qual o pacote que está sendo roteado pertence, além de outras considerações administrativas, normalmente restrições contidas no SLA acordado com o usuário [AWDUCHE01b]. O cálculo de rotas sujeitas a restrições é de processamento intenso, devendo ser realizado uma única vez, normalmente pelo roteador de entrada que recebe os fluxos de tráfego do usuário, informando a seqüência de roteadores (ou algum identificador equivalente) que compõem a rota calculada aos demais (*roteamento explícito*).

adequação ao MPLS, o RSVP foi estendido de forma a permitir a troca de rótulos entre LSRs, passando a ser referenciado por RSVP-TE (RSVP *Traffic Engineering*) [AWDUCHE01a].

Ao serem combinados modelos de provisão de QoS *intserv* ou *diffserv* com o MPLS, um ou mais fluxos de tráfego do usuário são mapeados em LSPs apropriadamente configurados. Nessas combinações, dois cenários possíveis podem ser imaginados, dependendo do número de classes de serviço transportadas por um LSP. No caso mais simples, o agregado de fluxos pertencentes a mesma FEC e classe de serviço, também chamado de *tronco de tráfego* (*traffic trunk*) [LI98], é transportado por um LSP exclusivo[21]. Uma outra possibilidade é o transporte de mais de um *tronco de tráfego* em um mesmo LSP[22]. Nesse caso, os LSRs que compõem a rota devem ser capazes de distingüir a classe de serviço ao qual os pacotes pertencem de forma a permitir que os seus mecanismos de escalonamento possam encaminhar os *troncos de tráfego* de forma apropriada. Como as informações necessárias à identificação da classe de serviço de um pacote IP não são normalmente visíveis aos comutadores MPLS, o próprio rótulo deve ser capaz de fornecer essas informações. Atualmente está sendo sugerido a utilização dos três bits EXP[23] do rótulo MPLS na identificação da classe de serviço associada a um *tronco de tráfego*. A associação do valor do campo EXP a uma determinada classe de serviço é, normalmente, informada pelo mesmo protocolo utilizado na criação de rotas MPLS.

Dependo se as classes de serviço reconhecidas pelos LSRs forem *intserv* ou *diffserv*, denomina-se a configuração resultante de serviços integrados ou diferenciados sobre MPLS, respectivamente. Essa última configuração, devido ao seu potencial de utilização em cenários mais abrangentes, como em grandes provedores de serviço na Internet, foi escolhida para as discussões que se seguem.

---

[21] Também chamado de L-LSP (*Label-only inferred* LSP).
[22] Também chamado de E-LSP (EXP *inferred* LSP).
[23] Nas recomendações iniciais da arquitetura MPLS, esses bits eram reservados à experimentação, sem uso definido.

Na configuração *diffserv* sobre MPLS [LI98, CHEVAL01], a solicitação de um serviço diferenciado pode resultar na configuração de um ou mais LSPs no provedor. A partir das características dos fluxos de tráfego a serem gerados e dos requisitos de QoS, o mecanismo de *roteamento* presente no LSR-*diffserv* deve escolher aquelas rotas capazes de atender às necessidades do usuário. Outras restrições, além das mencionadas, podem ser explicitamente especificadas no SLA acordado com o usuário, como a utilização de caminhos que forneçam um maior grau de confiabilidade, por exemplo. Após a escolha das rotas que melhor se adequem às restrições contidas na solicitação de serviço, LSPs devem ser configurados no domínio MPLS. Nesse ponto, conforme mencionado anteriormente nesta seção, dois cenários são possíveis: LSPs atendendo uma única classe de serviço, ou LSPs que podem transportar um agregado de classes. A configuração de um LSP envolve o *mapeamento* dos parâmetros de tráfego e de QoS contidos no solicitação de serviços *diffserv* em valores correspondentes aos empregados pelo mecanismo de sinalização MPLS. Caso o LSP configurado possa conter mais de uma classe, a associação entre os valores do campo DS e o padrão de bits EXP a ser empregado também são informadas no momento de configuração do LSP. Os protocolos CR-LDP e RSVP-TE foram novamente estendidos de forma a suportar estruturas de dados que permitam que essa associação seja informada. Dessa forma, cada LSR configura suas tabelas de encaminhamento, com a associação entre rótulos e FECs, e os seus mecanismos de *classificação* e *escalonamento*. O mecanismo de *classificação* objetiva determinar a classe de serviço *diffserv* a ser fornecida em cada *tronco de tráfego* transportado. É com base nessa *classificação*, que o escalonador determinará o instante correto em que os pacotes recebidos serão encaminhados de forma a ser preservada a QoS solicitada pelos usuários. Opcionalmente, caso seja especificado no SLA, um LSP de contingência pode ser configurado.

2.6 Sumário

Este capítulo apresentou as principais funções e estruturas identificadas na provisão de QoS em provedores baseados nos modelos *intserv* e *diffserv*, como as formas de solicitação de serviços (estática ou dinâmica) e de estabelecimento de contratos (estática, dinâmica centralizada ou distribuída), as estruturas empregadas na caracterização do tráfego e parametrização de QoS, os mecanismos de negociação, roteamento, mapeamento, controle de admissão e de políticas, classificação, escalonamento, reserva de recursos, policiamento, dentre outros. Esse estudo preliminar serviu de base para a definição dos elementos constituintes da arquitetura proposta no presente trabalho.

Também foram abordadas as formas possíveis de combinação desses modelos, bem como as adaptações necessárias quando diferentes níveis de abstração devem interagir, como ocorre quando são empregadas tecnologias de provisão de QoS no nível das sub-redes em conjunto com os modelos supracitados. Nesses cenários particulares, os mecanismos de mapeamento são fundamentais no processo de adaptação, sendo responsáveis por definir a linguagem utilizada na interação entre esses níveis.

Capítulo 3

Descrição da Arquitetura

Este capítulo descreve como os frameworks definidos em [GOMES99] podem ser especializados na modelagem das funções e estruturas identificadas no Capítulo 2, resultando na proposta de uma arquitetura adaptável (novos frameworks) aos vários cenários possíveis de provisão de QoS na Internet. Como uma ferramenta de auxílio aos projetistas na sua difícil tarefa de escolha e implementação de um modelo de QoS apropriado às suas necessidades, a referida arquitetura, dado o seu caráter recursivo, também pode ser empregada na estruturação dos mecanismos de provisão típicos do nível das sub-redes, também discutidos no Capítulo 2.

A descrição da arquitetura de provisão de QoS proposta neste trabalho segue o paradigma orientado a objetos, sendo dividida em três partes principais, que refletem a própria divisão adotada pelos frameworks para provisão de QoS apresentados em [GOMES99]:

- Parametrização de Serviços;
- Compartilhamento de Recursos;
- Orquestração de Recursos.

A utilização de frameworks [PREE95] como ferramenta de modelagem permite uma identificação clara dos pontos de flexibilização (*hot-spots*) que, no contexto da arquitetura proposta, devem ser completados pelo projetista na adaptação do modelo de provisão de QoS escolhido em ambientes específicos. A notação UML (*Unified Modeling Language*) [UML97] é empregada na representação dos diagramas de classes que descrevem os frameworks da arquitetura. Foram adotadas cores distintas na diferenciação entre as classes-base (cinzas) e as que representam possíveis instanciações dos pontos de flexibilização desses elementos (brancas). Durante a descrição textual dessas partes, foi utilizada também uma grafia diferenciada para denominar classes abstratas (em *itálico*) e concretas.

3.1 Parametrização de Serviços

Conforme mencionado no Capítulo 2, espera-se que, em uma solicitação de serviços, o usuário forneça a caracterização do tráfego correspondente aos fluxos que irá introduzir no sistema, e os parâmetros de QoS a serem aplicados pelo provedor sobre os referidos fluxos. As estruturas de dados utilizadas na descrição desses valores dependem, primeiramente, do nível de abstração considerado. No nível em que os modelos de QoS (*intserv* e *diffserv*) encontram-se definidos, por exemplo, o tráfego correspondente a um fluxo do usuário pode ser caracterizado por uma estrutura *tspec* que modela um filtro do tipo *token bucket*. A mesma caracterização, na configuração de um LSP MPLS, portanto ao nível das sub-redes, vai demandar outras estruturas, normalmente definidas pelo protocolo de sinalização empregado na criação das rotas e reserva de recursos. Situação semelhante ocorre na definição dos parâmetros de QoS.

Mesmo quando se restringe o enfoque a um único nível de abstração, deve-se esperar diferentes estruturas na caracterização desse valores. Normalmente, a definição das estruturas empregadas faz parte da especificação do modelo ou tecnologia de provisão de QoS. Porém, há situações onde a

especificação é, propositadamente, imprecisa ou omissa. É o caso do modelo *diffserv*, que especifica apenas que os parâmetros de caracterização de tráfego e de QoS fazem parte de uma estrutura denominada TCS (*Traffic Conditioning Specification*), sem, no entanto, entrar em maiores detalhes sobre como a referida estrutura encontra-se definida. Nessas situações, cabe ao protocolo específico empregado na solicitação de serviços o preenchimento dessa lacuna. O protocolo SIBBS (*Simple Interdomain Bandwidth Broker Signalling*), por exemplo, define, na sua mensagem própria de solicitação de serviços, a estrutura SPO (*Service Parametrization Object*) na modelagem de um TCS *diffserv*, conforme visto na Seção 2.3.1.

Deve-se enfatizar que a caracterização de tráfego e os parâmetros de QoS são as estruturas mínimas normalmente presentes em uma solicitação de serviços. Conforme mencionado no Capítulo 2, as solicitações de serviço *diffserv* são comumente mais abrangentes que as *intserv*, sendo constituídas por parâmetros que podem descrever outros requisitos do usuário, como a confiabilidade, as regras de policiamento, restrições de caminhos que podem ser escolhidos, dentre outros. Estes parâmetros, de acordo com a especificação *diffserv*, constituem a estrutura SLS (*Service Level Specification*). No presente trabalho, no entanto, serão referenciados, de maneira geral, por *parâmetros de caracterização de serviços*.

A escolha dos *parâmetros de caracterização de serviços* é de responsabilidade do provedor, que define as categorias ou classes de serviço disponíveis aos usuários. Desta forma, um provedor *intserv*, ao tornar disponível categorias de serviço garantida ou de carga controlada, está definindo, implicitamente, as estruturas de caracterização que devem ser empregadas na solicitação desses serviços (*tspec* e *rspec*).

Além dos parâmetros de caracterização, uma categoria de serviço pode definir também o *nível de garantia* desejado pelo usuário (100%, negociada,

fornecida pelo provedor) e o *estilo de compartilhamento*[24]. O conceito de *estilo de compartilhamento* permite que um mesmo serviço seja aplicado a fluxos distintos mas que preservem algum parâmetro em comum (endereço do receptor, por exemplo). Esta característica permite que o usuário alterne entre os fluxos sobre os quais os parâmetros de QoS devem ser aplicados, preservando as reservas de recursos realizadas pelo provedor. Um exemplo de aplicação que poderia fazer uso dessa facilidade seria uma conferência internacional, onde cada participante poderia escolher em qual idioma deseja receber os fluxos transmitidos, sendo facultado ao usuário realizar mudanças na sua escolha a qualquer instante, durante o tempo em que durar a conferência.

Em adição aos *parâmetros de caracterização de serviços*, o provedor emprega outras estruturas responsáveis por armazenar informações relativas ao seu desempenho. Na Seção 2.2.2, foi mencionada a estrutura *adspec*, utilizada como uma estimativa das capacidades de um provedor *intserv* disponíveis aos usuários. Estruturas semelhantes são utilizadas em uma configuração *diffserv* com negociação de QoS centralizada. De forma a ser possível a provisão *intradomínio*, discutida na Seção 2.3.2, o agente de negociação de QoS centralizada, costumeiramente chamado de BB (*Bandwidth Broker*), deve receber, periodicamente, informações relacionadas à capacidade disponível nos roteadores ativos em seu domínio de atuação. Mecanismos de monitorização da QoS fornecida pelo provedor também empregam parâmetros de desempenho. Será por meio desses valores que o provedor poderá tomar medidas preventivas, como o redirecionamento de tráfego, por exemplo, visando a manutenção dos contratos de serviço acordados com os usuários.

Na arquitetura proposta, os *parâmetros de caracterização de serviços* e de *desempenho do provedor* são estruturados a partir do **Framework para Parametrização de Serviços** definido em [GOMES99]. O propósito desse framework é fornecer um esquema que permita a estruturação desses parâmetros de forma independente dos serviços que possam vir a ser oferecidos por um

[24] Este conceito foi introduzido inicialmente pela especificação do protocolo RSVP, nos chamados estilos de reserva.

determinado provedor, o que torna possível que uma mesma definição de parâmetro possa ser reaproveitada na caracterização de vários serviços. Pelo emprego de *parâmetros abstratos*, hierarquias de derivação podem ser construídas. Dessa forma, serviços ou estratégias de negociação de QoS podem ser modelados em termos de *parâmetros abstratos*, adiando-se a definição concreta para o momento de escolha do ambiente específico em que os referidos serviços e estratégias serão implementados.

De forma semelhante aos *parâmetros de caracterização de serviços* e de *desempenho do provedor*, as categorias de serviço também são estruturadas, pelo framework discutido, em uma hierarquia de derivação. Essa característica permite que informações e mecanismos comuns a várias categorias possam ser definidos em níveis superiores da hierarquia, facilitando o reuso de estrutura e reaproveitamento de código. A definição de categorias abstratas pode ser utilizada também no desenvolvimento de interfaces de solicitação de serviços independentes do modelo de provisão de QoS empregado pelo provedor, conforme será visto com mais detalhes no Capítulo 4.

3.1.1 Componentes do Framework para Parametrização de Serviços

A Figura 4 ilustra o Framework para Parametrização de Serviços proposto em [GOMES99] que deve ser especializado pela arquitetura descrita neste trabalho. A classe abstrata *ServiceCategory* é utilizada na estruturação das categorias de serviço tornadas disponíveis pelo provedor. Cada objeto dessa classe tem associado um ou mais *parâmetros de caracterização de serviços*, todos definidos como especializações da classe abstrata *Parameter*. O *pattern* estrutural *bridge* [GAMMA95] é utilizado para representar as categorias de serviço como conjunto de parâmetros (relacionamento *parameterList*), possibilitando o desacoplamento entre as hierarquias de categorias de serviço e de derivação de parâmetros, tornado-as independentemente extensíveis. Como pode ser visto pela Figura 4, a classe *ServiceCategory* disponibiliza os métodos *addParameter()* e

getParameter() de forma a ser possível a adição e consulta de *parâmetros de caracterização de serviços*, respectivamente. O estilo de compartilhamento é definido pelo atributo *style*, acessado por intermédio dos métodos *setStyle()* e *getStyle()* definidos na classe *ServiceCategory*. Normalmente, um estilo pode representar um serviço compartilhado (*shared*) ou não (*fixed*) entre um agregado de fluxos especificado pelo usuário. Tanto as categorias de serviço como os parâmetros definidos são identificados por descritores, representados pelos atributos *categoryDescriptor* e *paramDescriptor*, respectivamente.

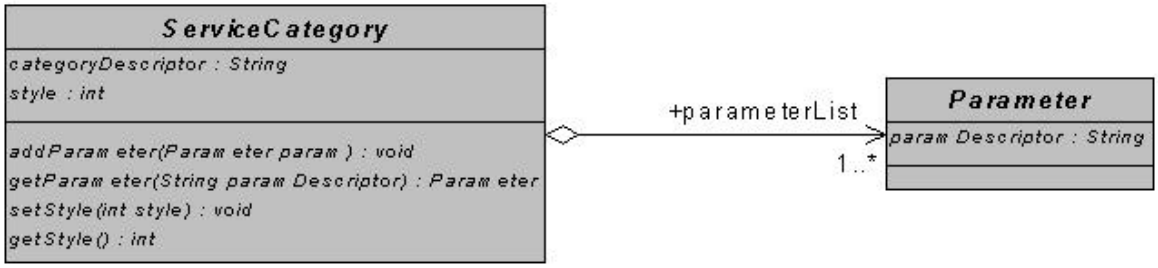

Figura 4: Framework para Parametrização de Serviços.

3.1.2 Exemplo de Aplicação do Framework para Parametrização

A Figura 5 apresenta um exemplo de aplicação do Framework para Parametrização de Serviços em um provedor que implementa os modelos *intserv* e *diffserv*. Tipicamente, esse provedor estaria configurado em um ambiente de provisão de serviços integrados sobre diferenciados, discutido na Seção 2.4. Na hierarquia de derivação de categorias de serviço ilustrada no exemplo da Figura 5, a classe *ServiceCategory* foi especializada nas classes *IntservServiceCategory* e *DiffservServiceCategory*. As categorias de serviço definidas atualmente no modelo *intserv* são representadas, no mesmo exemplo, pelas classes concretas GuarServiceCategory (serviço garantido) e CLServiceCategory (serviço de carga controlada). Também são ilustradas possíveis instanciações de categorias de serviço *diffserv*, representadas pelas classes GoldServiceCategory, SilverServiceCategory e BronzeServiceCategory.

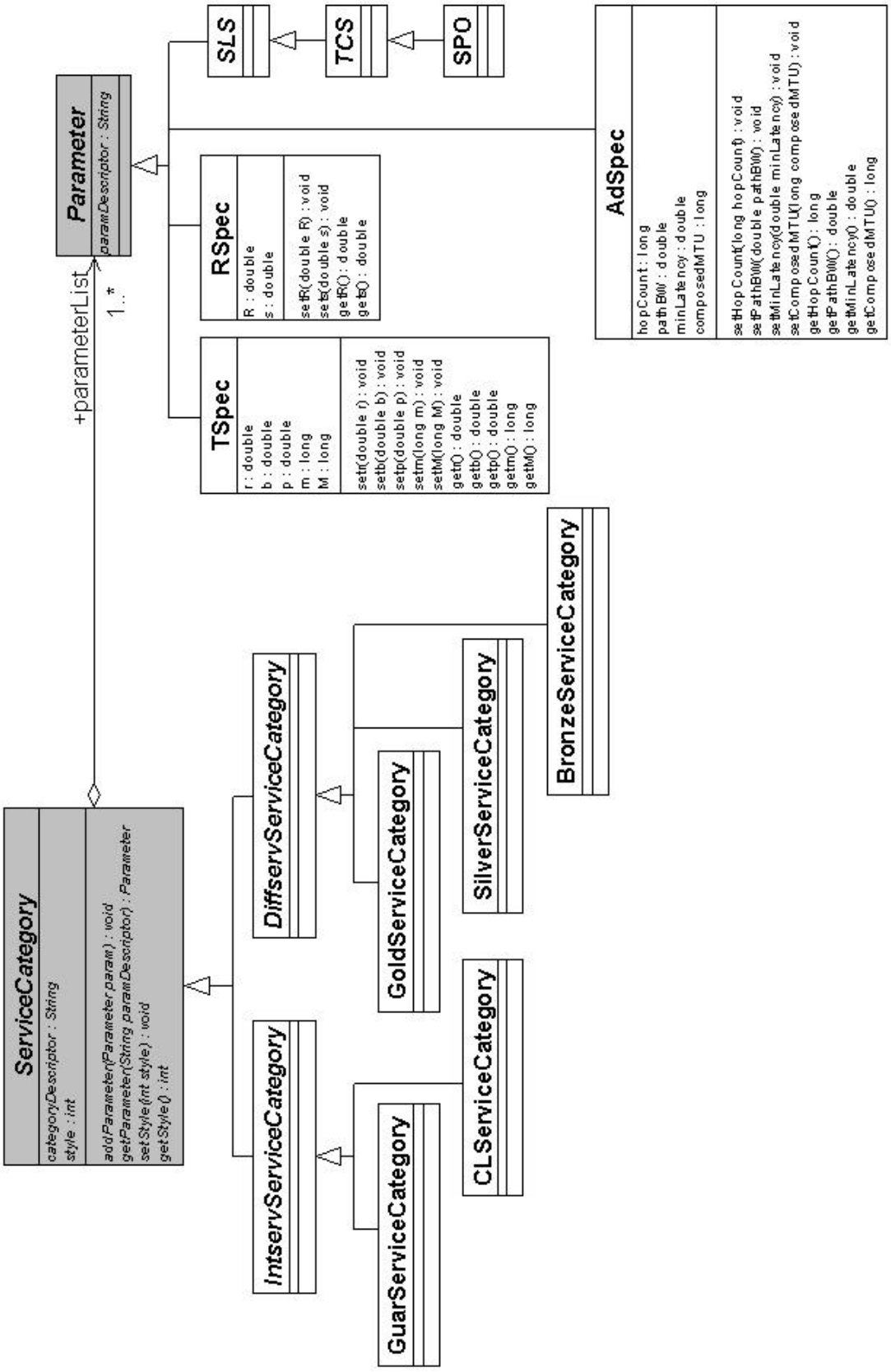

Figura 5: Exemplo de aplicação do Framework para Parametrização.

Na hierarquia de parametrização de serviços, as estruturas *tspec* e *rspec* são representadas, na modelagem ilustrada na Figura 5, por meio de classes concretas de mesmo nome, especializadas de *Parameter*. Esses parâmetros são adicionados a objetos do tipo GuarServiceCategory na configuração de serviços garantidos. A especificação de serviços de carga controlada, conforme visto na Seção 2.2.1, exige apenas que sejam informados os valores de caracterização dos fluxos de tráfego do usuário, representados por parâmetros do tipo TSpec, adicionados a objetos da classe CLServiceCategory na solicitação desses serviços. No mesmo exemplo, um parâmetro de desempenho típico de provedores de serviço *intserv*, denominado *adspec*, também é representado por uma classe apropriada, com atributos definidos de acordo com a especificação do modelo. Finalmente, os parâmetros de caracterização relativos a serviços *diffserv*, baseados nas estruturas descritas na Seção 2.3, são definidos em uma hierarquia própria, encabeçada pelas estruturas mais gerais *SLS* e *TCS*, representadas por intermédio de classes abstratas, e finalizada pela classe concreta SPO.

Embora o exemplo da figura tenha se restringido a categorias e parâmetros de serviço típicos do nível de abstração correspondente aos modelos *intserv* e *diffserv*, é importante enfatizar que o framework para parametrização pode ser aplicado, de forma semelhante, em configurações no nível das sub-redes. De forma análoga, as categorias de serviço ATM (CBR, VBR, etc), por exemplo, podem ser modeladas por intermédio de classes apropriadas, derivadas de *ServiceCategory*, sendo os seus parâmetros de caracterização (PCR, SCR, CDVT, etc) estruturados como subclasses de *Parameter*. No Capítulo 4, será descrita uma aplicação da arquitetura proposta que modela serviços *intserv* sobre ATM, onde as estruturas próprias dessa tecnologia de sub-rede serão discutidas com maiores detalhes.

3.2 Compartilhamento de Recursos

No atendimento aos serviços solicitados, o provedor deve ser dotado de mecanismos de escalonamento responsáveis por dividir a utilização dos seus recursos (processador, memória, largura de banda) em parcelas de tempo associadas a um ou mais fluxos de tráfego do usuário. Na arquitetura proposta, essas parcelas são denominadas *recursos virtuais*.

A escolha, dentre os *recursos virtuais* controlados por um escalonador, daquele que terá acesso a sua parcela de utilização do recurso real em um determinado instante faz parte da *estratégia de escalonamento*. Em muitas situações, é interessante ser possível a definição de diferentes *estratégias de escalonamento* de acordo com a categoria de serviço associada aos fluxos atendidos pelo provedor. Dessa forma, uma parcela do recurso real pode ser fornecida a cada categoria de serviço, sendo re-escalonada entre os fluxos atendidos pela referida categoria. Esse processo pode ser repetido em vários níveis ou subcategorias, dando origem a uma *árvore de recursos virtuais*, cujas folhas representam os *recursos virtuais* associados a um ou mais fluxos do usuário; os nós internos são constituídos por escalonadores que controlam o acesso a um determinado *recurso virtual*, sendo chamados, dessa forma, de *escalonadores de recursos virtuais*; e a raiz é representada por um *escalonador de recurso real*.

A Figura 6 ilustra a estruturação, em um provedor *intserv* hipotético, da *árvore de recursos virtuais* de um roteador. Na figura, são mostradas as categorias de serviço associadas aos *escalonadores de recursos virtuais*. A divisão do recurso real, que no exemplo corresponde à largura de banda, é realizada por meio de percentuais configurados por gerência. Cada *escalonador de recurso virtual* ilustrado na figura pode empregar diferentes *estratégias de escalonamento* no compartilhamento do percentual de recurso real que tem acesso. Os fluxos aceitos pelo provedor são associados a *recursos virtuais*

controlados por um *escalonador de recurso virtual* específico, dependendo da categoria de serviço solicitada pelo usuário.

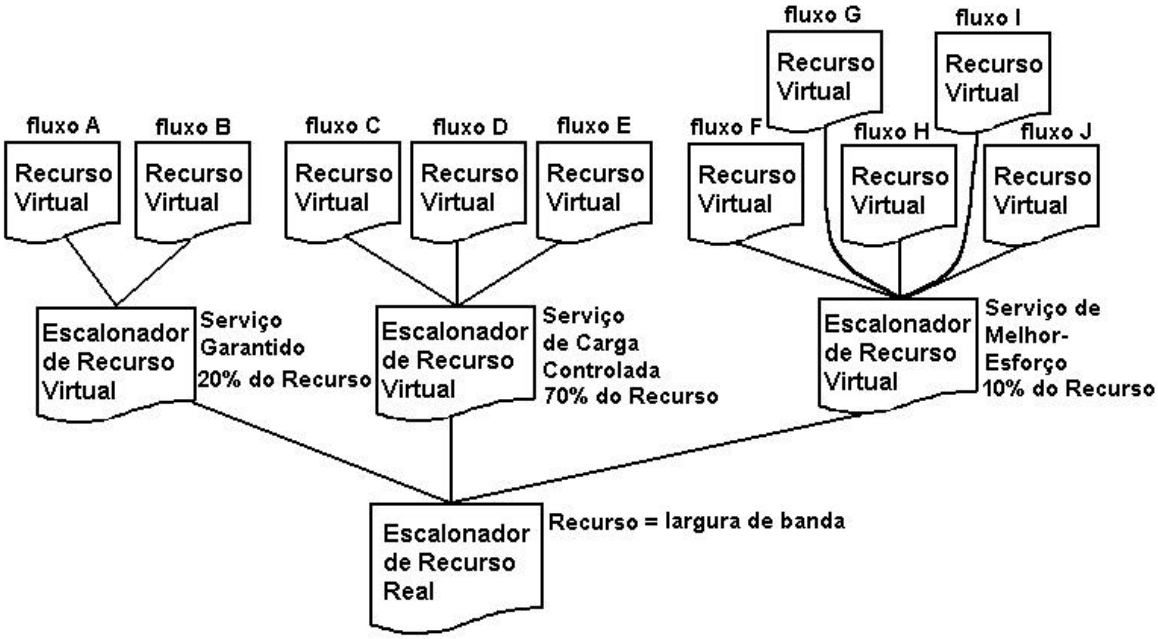

Figura 6: Árvore de recursos virtuais em um roteador *intserv*.

Cada escalonador (real ou virtual) presente na *árvore de recursos virtuais* tem associado um componente de criação de um determinado tipo de *recurso virtual*. Esse componente realiza também a adição do *recurso virtual* criado à lista de *recursos virtuais* controlados pelo referido escalonador. Após essa adição, o componente de criação configura os módulos de *classificação* e de *policiamento* de forma a ser possível a associação dos fluxos do usuário aos *recursos virtuais* criados e a monitorização dos referidos fluxos, respectivamente.

Em [GOMES99], os mecanismos de criação de *recursos virtuais* são modelados por intermédio do **Framework para Reserva de Recursos**, deixando-se as funções e estratégias empregadas no controle de tráfego (*classificação*, *policiamento* e *escalonamento*) para serem abordadas pelo **Framework para Escalonamento de Recursos**. Os referidos frameworks atuam durante as fases de estabelecimento e manutenção de contratos de serviço, respectivamente, e, por modelarem funções bastante relacionadas (criação, configuração e

escalonamento de recursos) são referenciados, em conjunto, por **Frameworks para Compartilhamento de Recursos**.

3.2.1 Componentes dos Frameworks para Compartilhamento de Recursos

Após a identificação dos escalonadores relacionados ao fornecimento de um serviço solicitado, o provedor deve iniciar o processo de criação dos *recursos virtuais* necessários, tendo como base os requisitos do referido serviço, devidamente mapeados em parâmetros condizentes com o nível de abstração considerado. No Framework para Reserva de Recursos, o componente responsável pela criação de *recursos virtuais* em um determinado escalonador é chamado de *VirtualResourceCreator*. Conforme mencionado no início da Seção 3.2, o processo de criação é, normalmente, seguido pela adição do novo *recurso virtual* à lista de *recursos virtuais* controlados pelo escalonador correspondente e pela configuração dos módulos de *classificação* e *policiamento*.

O *módulo de classificação* é configurado pela adição de filtros normalmente aplicados sobre campos específicos do cabeçalho dos pacotes (como o campo DS, por exemplo). Um filtro, ao ser aplicado, permite a identificação positiva de um determinado padrão de cabeçalho, retornando, dessa forma, a classe de serviço que deve ser associada ao pacote. O *agente de classificação* é modelado no Framework para Escalonamento de Recursos pela classe *Classifier*, que reserva o atributo *filterList* para armazenar as associações entre filtros e categorias de serviço, realizadas pela chamada ao método *addFilter()*, também definido na referida classe. A aplicação de um filtro é o resultado da invocação ao método *classify()*, disponibilizado pelo *agente de classificação*. Esse método recebe como parâmetros, normalmente, campos específicos do cabeçalho dos pacotes, retornando a classe de serviço associada ao mesmo com base nas associações informadas previamente.

O *mecanismo de policiamento*, modelado no Framework para Escalonamento de Recursos pela classe *PoliceAgent*, tem, como atributo principal, a lista *trafficProfileList*, que armazena associações entre perfis de tráfego do usuário e *ações de policiamento*, sendo a referida lista configurada pela chamada ao método *addTrafficProfile()*. O método *police()*, também definido na classe *PoliceAgent*, possibilita determinar se um pacote em particular encontra-se conforme ou não ao perfil de tráfego associado ao fluxo ao qual pertence. Caso o pacote não esteja de acordo com o perfil configurado, a *ação de policiamento* correspondente, contida na associação *trafficProfileList*, será executada. A Figura 7 ilustra uma visão conjunta das classes e relacionamentos que compõem os Frameworks para Compartilhamento de Recursos.

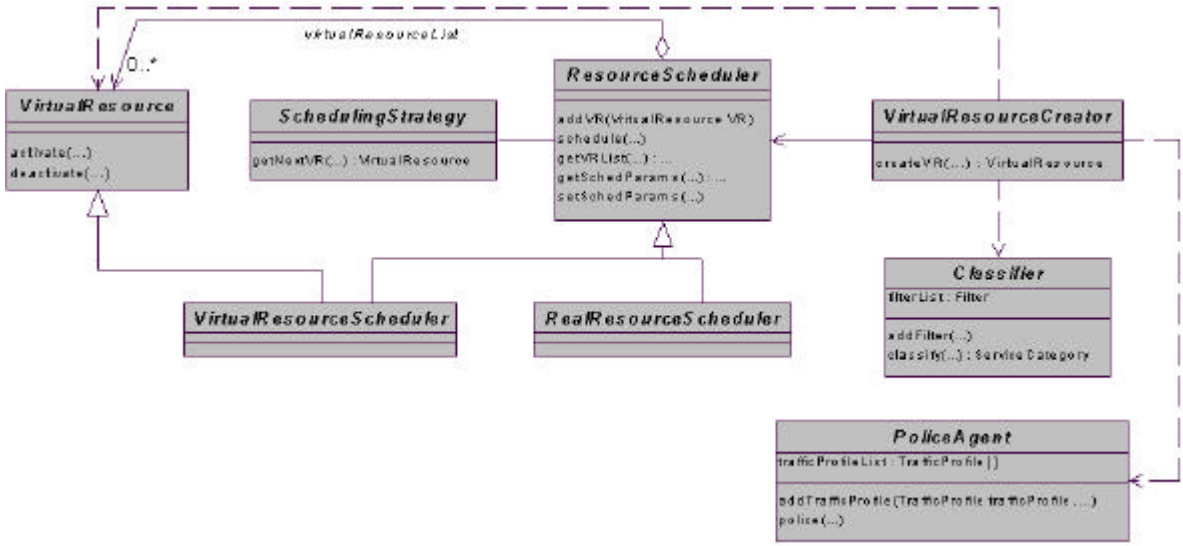

Figura 7: Visão conjunta dos Frameworks para Compartilhamento de Recursos.

Um *escalonador de recursos*, representado pela classe *ResourceScheduler*, deve escolher uma *estratégia de escalonamento*, dentre as disponibilizadas em *SchedulingStrategy*. A *estratégia de escalonamento*, conforme mencionado no início da Seção 3.2, é responsável por determinar o próximo *recurso virtual* (pela chamada ao método *getNextVR*), dentre aqueles controlados pelo escalonador, a ser ativado ou, em outras palavras, a ter o direito de uso sobre o recurso que tem o seu uso compartilhado. Em última instância, a estratégia

determinará, depedendo do *recurso virtual* ativado, qual fluxo do usuário será encaminhado. Normalmente, a escolha das estratégias adotadas ocorre na fase de iniciação do sistema, quando são configuradas as categorias de serviço que o provedor irá fornecer aos usuários. Conforme mencionado na Seção 3.2.1, um escalonador, dependendo do tipo de recurso que compartilha, pode ser real (classe *RealResourceScheduler*) ou virtual (classe *VirtualResourceScheduler*). Os *recursos virtuais* são representados pela classe *VirtualResource*, que disponibiliza métodos para a sua ativação (*activate*) ou desativação (*deactivate*).

3.2.2 Exemplo de Aplicação dos Frameworks para Compartilhamento

Dependendo do recurso a ser controlado, vários cenários podem ser modelados pelos frameworks para compartilhamento. Por ser o recurso normalmente mais escasso nos roteadores da Internet no escalonamento de pacotes, o compartilhamento de banda passante foi escolhido, nesta seção, como exemplo de aplicação dos referidos frameworks.

Conforme ilustrado na Figura 8, o controle do acesso à banda passante é realizado por um objeto da classe RealPacketScheduler, uma especialização de *RealResourceScheduler*. Escalonadores virtuais podem ser definidos para cada classe de serviço fornecida pelo provedor, sendo representados por objetos da classe VirtualPacketScheduler, especializada da classe *VirtualResourceScheduler*. Cada um desses escalonadores virtuais pode adotar diferentes *estratégias de escalonamento*. Caso os serviços oferecidos pelo provedor sejam *intserv*, por exemplo, pode-se utilizar o algoritmo WFQ (*Weighted Fair Queueing*) [PAREKH92], implementado pela classe WFQSchedStrategy, como a estratégia do escalonador virtual responsável pelo controle dos fluxos de serviço garantido. Outras estratégias mais simples, como a FIFO (*First In First Out*), representada pela classe FIFOSchedStategy, por exemplo, podem ser apropriadas a outras classes de serviço.

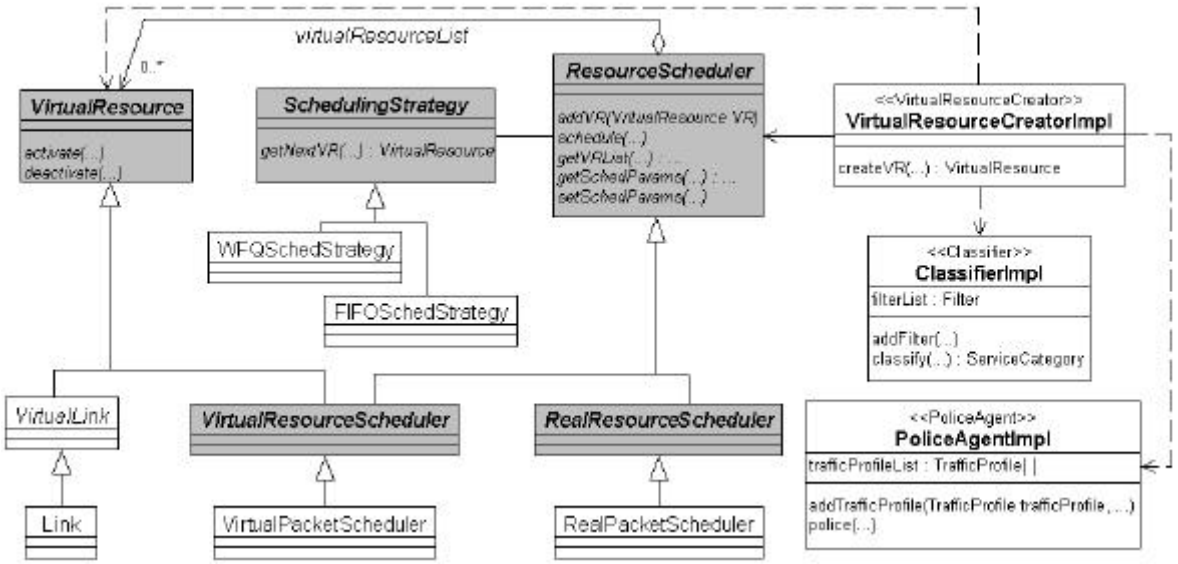

Figura 8: Exemplo de aplicação dos Frameworks para Compartilhamento.

O componente de criação de recursos virtuais é representado pela classe concreta VirtualResourceCreatorImpl. Na Figura 8, foi adotada uma notação especial que permite indicar, pela utilização dos sinais "<<" e ">>", que a classe VirtualResourceCreatorImpl é uma implementação de *VirtualResourceCreator*. Notação similiar foi utilizada para ilustrar as implementações ClassifierImpl e PoliceAgentImpl.

Voltando-se ao exemplo de configuração de um provedor *intserv*, discutido no início da Seção 3.2, após a admissão de um serviço solicitado, o provedor deve traduzir os parâmetros de caracterização e de QoS em valores que possam ser interpretados pelo componente de criação VirtualResourceCreatorImpl. Com base nesses valores e na categoria de serviço solicitada, um *recurso virtual* será criado, sendo em seguida associado ao fluxo de tráfego informado e adicionado à lista de *recursos virtuais* controlados pelo escalonador responsável pela referida categoria. Os *recursos virtuais* são representados, no exemplo da Figura 8, pela classe concreta Link, especializada da classe abstrata *VirtualLink*, que modela percentuais de acesso à banda passante. Em seguida, o componente de criação deve adicionar uma nova associação de filtro à categoria de serviço pela chamada ao método *addFilter* definido em ClassifierImpl. Finalmente, o perfil de tráfego informado pelo usuário,

juntamente com a *ação de policiamento* a ser executada nos casos em que for violado, devem ser informados como parâmetros ao *mecanismo de policiamento*, representado pela classe concreta PoliceAgentImpl, por intermédio do método *addTrafficDesc()*, concluindo, dessa forma, a fase de estabelecimento de contrato de serviço.

Durante a fase de manutenção de contratos de serviço, o escalonador RealPacketScheduler, do exemplo, fornece parcelas de utilização, configuradas por gerência, a cada uma das categorias oferecidas pelo provedor. O fornecimento dessas parcelas é representado por chamadas periódicas ao método *activate()* definido em objetos do tipo VirtualPacketScheduler. Esse escalonador, por sua vez, re-escalona a sua parcela de utilização do recurso real entre os *recursos virtuais* que controla, tendo como base uma estratégia configurada também por gerência. A escolha e ativação de um *recurso virtual* por parte desses escalonadores trará, como conseqüência imediata, o encaminhamento de pacotes do fluxo associado ao referido recurso.

3.3 Orquestração de Recursos

Todo os mecanismos envolvidos no estabelecimento de contratos de serviço devem ser coordenados entre os vários subsistemas participantes de forma integrada. Um *agente de negociação* é responsável por toda essa coordenação, estando normalmente presente nas estações e roteadores em um ambiente de provisão de QoS verdadeiramente fim-a-fim.

Conforme discutido na Seção 2.1.3, o estabelecimento de um SLA inicia-se quando um usuário solicita um serviço a um provedor qualquer. O serviço solicitado é encaminhado ao *agente de negociação* mais próximo, podendo ser empregadas duas modalidades de provisão de QoS. Na abordagem distribuída, os *agentes de negociação* presentes nas estações e roteadores cooperam entre si na escolha da melhor forma de se fornecer a QoS requisitada. Por outro lado, na abordagem centralizada, o *agente de negociação* que recebe a solicitação

simplesmente a repassa a um agente central, efetivamente responsável pelas decisões relacionadas ao fornecimento do serviço.

Independente da modalidade de provisão de QoS empregada, o *agente de negociação* é encarregado, inicialmente, de determinar os subsistemas envolvidos com o fornecimento do serviço, com base nas restrições contidas no SLA contratado pelo usuário. Essas restrições devem ser analisadas sob a luz das *políticas* configuradas no provedor, que reduzem ainda mais o universo dos subsistemas que podem ser efetivamente empregados. As credenciais dos usuários contidas na solicitação de um serviço, por exemplo, devem ser comparadas com as *políticas* de acesso configuradas no provedor, que podem reservar determinadas classes de recursos ou de serviços a usuários especiais. No Capítulo 2, foram vistos como essas *políticas* podem ser utilizadas em cenários de *negociação intserv* e *diffserv*.

Após a determinação dos subsistemas supracitados, que, normalmente representam outros provedores de serviço, roteadores adjacentes ou ainda escalonadores de recursos em um mesmo roteador, os parâmetros de caracterização do serviço devem ser mapeados em valores condizentes com o nível de abstração considerado. O *controle de admissão* nos referidos subsistemas é realizado com base nesses valores. Caso o serviço possa ser fornecido, o *agente de negociação* deve coordenar o processo de *reserva de recursos*, interagindo com criadores de recursos virtuais apropriados, e de *configuração* dos mecanismos de encaminhamento nos subsistemas escolhidos, ambos discutidos na Seção 3.2. Ao final, o *agente de negociação* deve sinalizar ao usuário o estabelecimento do SLA contratado.

Durante a manutenção de um contrato de serviço, pequenos ajustes podem ser necessários de forma a ser garantido o nível de QoS acordado com os usuários. Vários eventos podem disparar esses ajustes, como a falha de um enlace, ou a admissão de serviços com base em *matrizes de utilização* de

recursos que não refletem mais o tráfego atual[25]. De forma a ser possível a manutenção do serviço contratado, *ações de sintonização* devem ser empregadas, envolvendo a mudança de parâmetros em determinados escalonadores, de forma a ser priorizado um fluxo de tráfego em relação a outro, ou a escolha de caminhos alternativos, que permitam o fornecimento do mesmo nível de QoS inicialmente contratado.

As funções de **negociação** e **sintonização de QoS** supracitadas são modeladas em [GOMES99] por frameworks de mesmo nome, atuando nas fases de estabelecimento e manutenção de contratos de serviço, respectivamente.

3.3.1 Componentes do Framework Para Negociação de QoS

O *agente de negociação*, componente responsável pelo controle de todo o processo de estabelecimento de contratos de serviço, é modelado, no Framework para Negociação de Qos, pela classe *QoSNegotiator*. Os parâmetros relacionados ao fornecimento do serviço, como a descrição dos fluxos do usuário, a categoria de serviço desejada, a própria identificação do usuário, dentre outros, são informados ao referido agente por intermédio do método *request()*, disponibilizado pela classe *QoSNegotiator*. Será com base nesses parâmetros que o *agente de negociação* poderá verificar se as restrições impostas pelas *políticas* configuradas no provedor irão permitir ou não o fornecimento do serviço. No Framework para Negociação de QoS, o controle dessas *políticas* é função de um outro agente especializado, representado por objetos da classe *PolicyAgent*. Conforme mencionado anteriormente, no início da Seção 3.3, as referidas *políticas* podem conter, por exemplo, uma lista dos subsistemas ou categorias de serviço disponíveis a determinados usuários do provedor. O método *addPolicy()* disponibilizado pela classe *PolicyAgent* possibilita a adição de novas *políticas*, em uma lista denominada *policyStatementList*. O *agente de negociação* pode

[25] Uma *matriz de utilização* corresponde à distribuição do tráfego em um domínio medida em um determinado período, sendo utilizada no dimensionamento de recursos de um provedor e como base para muitos algoritmos de controle de admissão [DANZIG95].

consultar as *políticas* configuradas no provedor pela chamada ao método *verifyPolicy()*, também definido em *PolicyAgent*.

Respeitando as *políticas* configuradas no provedor, e levando em consideração os requisitos especificados pelo usuário, o *agente de negociação* determina os subsistemas mais indicados à realização do serviço solicitado. Normalmente, a escolha desses subsistemas é realizada com o auxílio de *mecanismos de roteamento com QoS*, discutidos na Seção 2.5.1 e modelados, no presente framework, pela classe *RoutingSupportAgent*. O método *resolve*, definido na referida classe, retorna uma lista de subsistemas (endereços de roteadores adjacentes, por exemplo) que satisfazem às restrições contidas na solicitação de serviço realizada pelo usuário. O *agente de negociação* deve proceder, em seguida, com o descarte daqueles subsistemas contidos na lista retornada que não atendem às restrições próprias do provedor, descritas em suas *políticas* de acesso internas.

Como parte do processo de estabelecimento de um contrato de serviço, o *agente de negociação* deve determinar a parcela de contribuição de cada subsistema escolhido no fornecimento do serviço solicitado como um todo. Para tanto, deve empregar mecanismos que mapeiem os parâmetros de caracterização do serviço solicitado em valores que correspondam às referidas parcelas e que sejam compatíveis com o nível de abstração ou modalidade de provisão de QoS dos subsistemas escolhidos. Nos casos em que um dos referidos subsistemas for constituído por um outro provedor de serviço, e que empregue um modelo de QoS distinto, como o que ocorre em configurações *intserv* sobre *diffserv*, por exemplo, um SLA compatível, resultado do *mapeamento* entre parâmetros de serviço *intserv* e *diffserv*, deve poder ser acordado entre *agentes de negociação* presentes em cada um dos respectivos provedores. Outra situação em que *mecanismos de mapeamento* devem ser empregados, ocorre quando o processo de negociação envolve a interação entre agentes localizados em níveis de abstração diferentes, como nas configurações *diffserv* sobre MPLS, por exemplo, ou quando são modelados *agentes de negociação* que controlam o gerenciamento de recursos locais. Neste último caso, há a necessidade dos

referidos agentes serem capazes de traduzir os parâmetros de requisição de serviços em valores que possam ser interpretados pelos escalonadores de recursos locais. Os *mecanismos de mapeamento* são modelados, no Framework para Negociação de QoS, pela classe *QoSMapper*, que pode empregar uma ou mais estratégias, descritas por objetos do tipo *MappingStrategy*. O método *translate()*, definido em *QoSMapper*, recebe os parâmetros de um serviço a ser mapeado na forma de um objeto do tipo *ServiceCategory*, retornando, em seguida, uma lista de categorias de serviço compatíveis, correspondentes a outros níveis de abstração ou modelos de provisão de QoS.

Será com base nos valores retornados pelo *mecanismo de mapeamento* que o *agente de negociação* responsável pelo gerenciamento dos recursos locais, modelado por uma especialização de *QoSNegotiator* denominada *QoSNResourceManager*, interagindo com o *mecanismo de controle de admissão*, determina se há ou não recursos suficientes para o atendimento ao serviço solicitado. O Framework para Negociação de QoS modela o *mecanismo de controle de admissão* por intermédio da classe *StreamAdmissionController*. Ao invocar o método *admit()*, definido nessa classe, o *agente de negociação* poderá determinar se o serviço solicitado pode ser atendido ou não pelo subsistema em questão. A exemplo dos *mecanismos de mapeamento*, várias *estratégias de admissão* podem ser empregadas, sendo representadas por instâncias da classe *AdmissionStrategy*. A Figura 9 ilustra as classes e relacionamentos que compõem o Framework para Negociação de QoS.

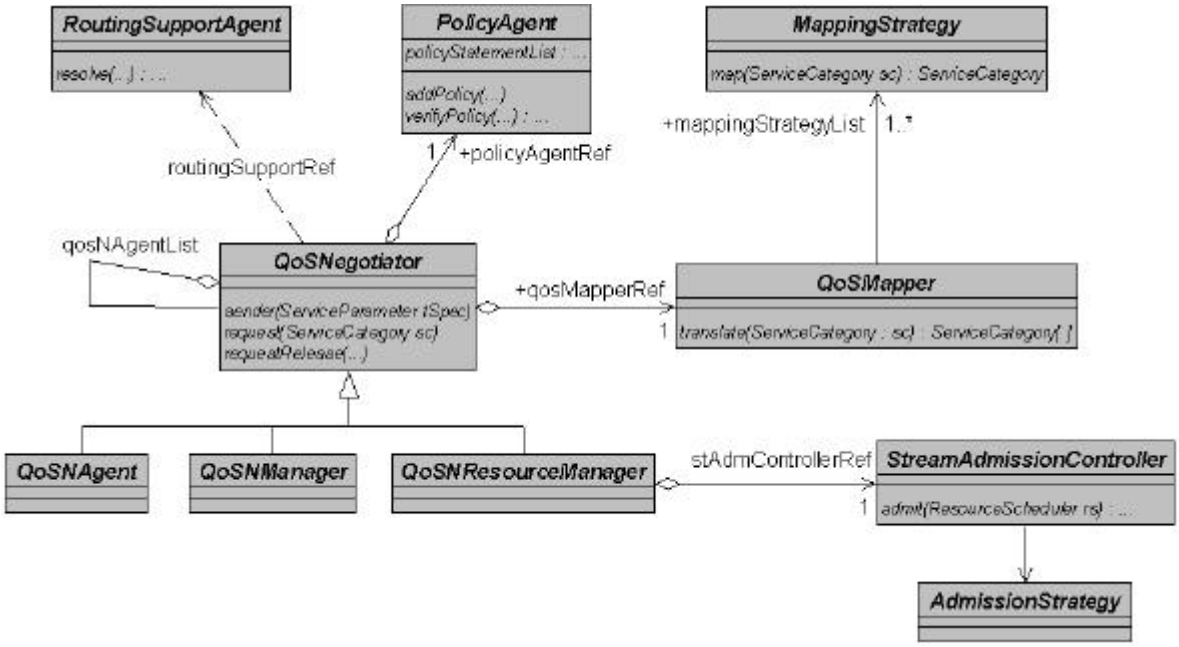

Figura 9: Framework para Negociação de QoS.

Uma vez que o serviço tenha sido admitido, o *agente de negociação* deve iniciar o processo de *reserva de recursos*, interagindo com objetos do tipo *VirtualResourceCreator* nos subsistemas envolvidos, e configuração dos mecanismos de controle de tráfego (*classificação*, *policiamento* e *escalonamento*), discutidos em detalhes na Seção 3.2.

No caso de uma negociação centralizada, o agente central, modelado no Framework para Negociação de QoS por uma especialização de *QoSNegotiator*, denominada *QoSNManager*, deve poder contactar todos os agentes secundários presentes no seu domínio de atuação. O auto-relacionamento *qosNAgentList* visa modelar essa possibilidade de *negociação*. Esses *agentes de negociação* secundários, modelados pela classe *QoSNAgent*, além de repassarem as solicitações de serviço recebidas, também são responsáveis por interagir com os agentes que controlam o gerenciamento local de recursos, que serão reservados com base nos valores informados pelo agente central.

3.3.2 Exemplos de Aplicação do Framework para Negociação de QoS

Na Seção 2.2.2, foram discutidos os *mecanismos de negociação* normalmente empregados no estabelecimento de contratos de serviço *intserv*, com sinalização RSVP. A Figura 10 ilustra como o Framework para Negociação de QoS pode ser utilizado na modelagem desse cenário. No exemplo da figura, os *mecanismos de negociação intserv* que não dependem do protocolo de sinalização são implementados na classe *IntservQoSNegotiator*. A especialização RSVPIntservQoSNegotiator será responsável por fornecer os mecanismos necessários ao transporte das informações de *negociação intserv* por intermédio de estruturas RSVP apropriadas.

A classe concreta QoSMapperImpl, uma implementação de *QoSMapper*, converte os parâmetros descritos nas categorias de serviço *intserv* solicitadas em valores que possam ser interpretados pelos escalonadores de recursos locais, como o VirtualPacketScheduler descrito na Seção 3.2.2, por exemplo. O agente RSVPIntservQoSNegotiator também é responsável pela gerência local dos recursos, como pode ser visto no exemplo da Figura 10, sendo, portanto, implementado por um objeto da classe *QoSNResourceManager*.

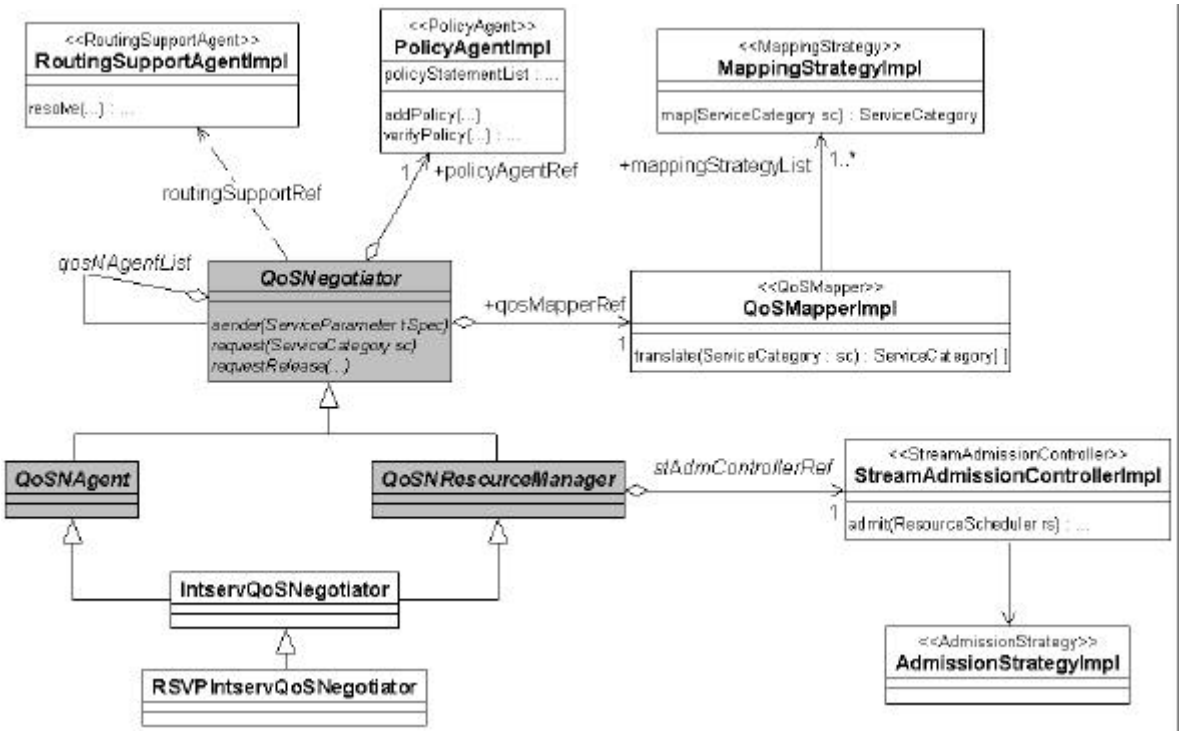

Figura 10: Aplicação do Framework para Negociação de QoS em um provedor *intserv*.

O Framework para Negociação de QoS, ao tornar disponível pontos de flexibilização, permite a implementação de vários cenários de provisão de QoS na Internet. A Figura 11 ilustra a modelagem de um provedor de serviços *diffserv* baseado na configuração centralizada discutida na Seção 2.3.2. Percebe-se, no exemplo, o emprego de um *gerente de negociação* de QoS (classe *QoSNManager*), que representa o *agente central de negociação* no provedor *diffserv*, tipicamente implementado em um BB (*Bandwidth Broker*). O agente central comunica-se com os demais agentes, modelados pela classe *QoSNAgent* e presentes nos roteadores do provedor, por intermédio do protocolo COPS. As classes *QoSNAgent* e *QoSNManager* foram especializadas de forma a modelarem a utilização específica desse protocolo. Os agentes locais também são responsáveis pela *gerência de recursos*, sendo, dessa forma, representados por objetos do tipo *QoSNResourceManager*, como pode ser visto no exemplo da Figura 11.

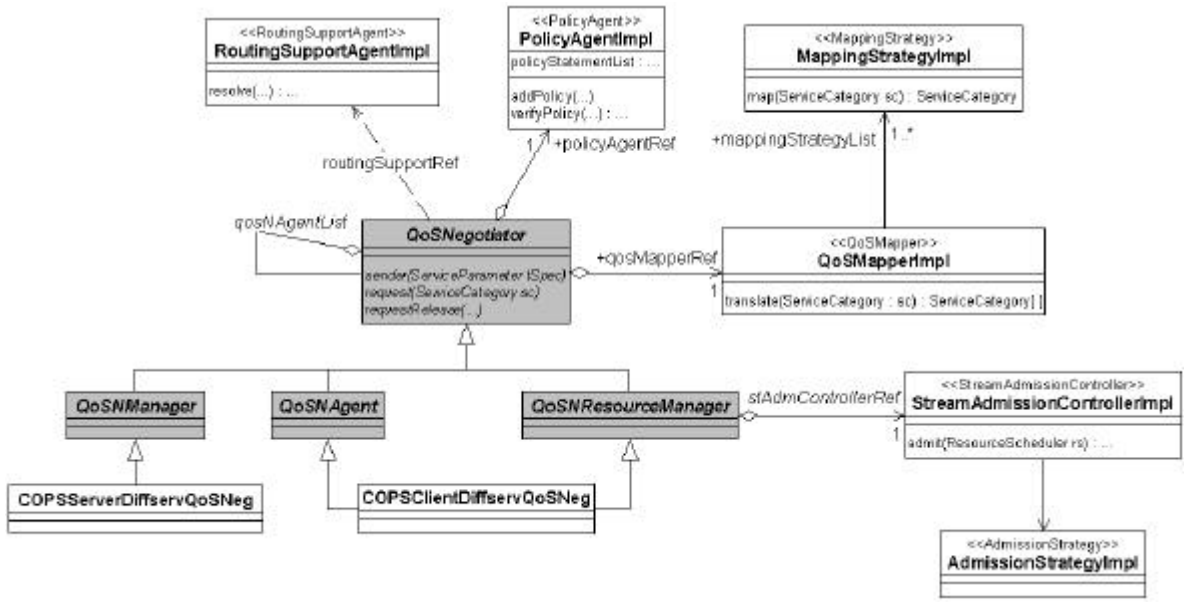

Figura 11: Aplicação do Framework para Negociação de QoS em um provedor *diffserv*.

Os exemplos ilustrados nas figuras 10 e 11 podem ser combinados em um cenário de configuração *intserv* sobre *diffserv*, bastando que o mapeador QoSMapperImpl implemente mecanismos de conversão entre serviços *intserv* e *diffserv*. Um roteador especial, normalmente localizado na borda de entrada do provedor *diffserv*, conforme configuração discutida na Seção 2.4, deve implementar agentes de negociação do tipo RSVPIntservQoSNegotiator, de forma a receber as solicitações de serviço *intserv*, e COPSClientDiffservQoSNeg, a fim de que as referidas solicitações, já devidamente convertidas em serviços *diffserv*, possam ser apropriadamente encaminhadas ao agente central de negociação do tipo COPSServerDiffservQoSNeg.

No Capítulo 4, um cenário de negociação *intserv* sobre ATM será discutido, onde será exemplificada a conversão de parâmetros de serviços integrados em valores apropriados ao nível de abstração da referida tecnologia de sub-rede.

3.3.3 Componentes do Framework para Sintonização de QoS

Após a admissão de um serviço, *mecanismos de monitorização* de fluxos devem iniciar um processo de medição periódica da real QoS que está sendo fornecida pelo provedor, obtida a partir de parâmetros de desempenho específicos, como os discutidos na Seção 3.1. Quando há indícios suficientes de que o contrato de prestação de serviço encontra-se na iminência de ser rompido, o *mecanismo de monitorização* envia alertas a um *agente de sintonização*, responsável por implementar ações preventivas que evitem a degradação do nível de QoS acordado. Como exemplo dessas ações, podem ser citadas a reconfiguração de um ou mais escalonadores envolvidos com o fornecimento do serviço, a escolha de escalonadores alternativos e, em último caso, o envio de um alerta ao agente de negociação, para que o mesmo procure caminhos alternativos ou renegocie com o usuário um novo contrato de serviço.

O *mecanismo de monitorização* de fluxos é modelado, no Framework para Sintonização de QoS ilustrado na Figura 12, pela classe *StreamMonitor*, que define o método *getStatistics()*, responsável por obter as medições relativas à real QoS que está sendo fornecida aos usuários. Caso os valores medidos indiquem que o contrato de serviço acordado encontra-se próximo a ser rompido, alertas são enviados ao *agente de sintonização* correspondente, modelado pela classe *QoSTuner*, que disponibiliza o método *tune()* com essa finalidade. A chamada ao método *tune()*, implica na efetivação de ações preventivas que objetivam manter o nível de QoS acordado. Cada *agente de sintonização* é responsável pelo controle de um ou mais objetos de *monitorização*, como pode ser visto, na Figura 12, pelo relacionamento de agregação *stMonitorList*.

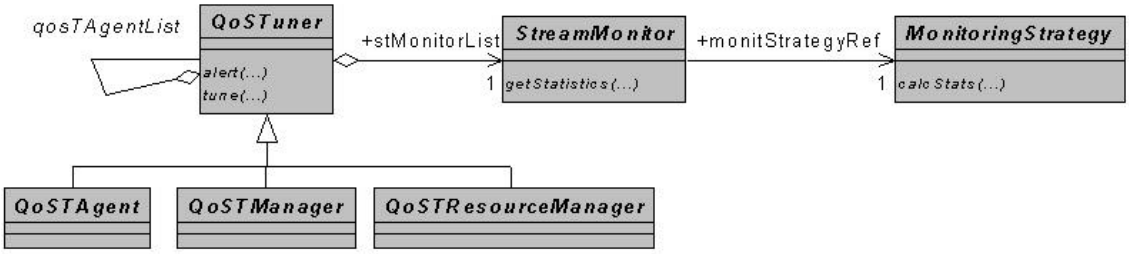

Figura 12: Framework para Sintonização de QoS.

A exemplo do processo de *negociação*, a *sintonização* também pode assumir as formas centralizada ou distribuída. Na abordagem centralizada, um agente do tipo *QoSTManager* é responsável por todas as ações de *sintonização* em um domínio. Os *agentes de sintonização* secundários, distribuídos nos roteadores do provedor e representados pela classe *QoSTAgent*, repassam os alertas recebidos pelos *mecanismos de monitorização* local ao agente central. Na abordagem distribuída, todos os agentes trocam informações de *sintonização* entre si, procurando realizar esse processo de forma cooperativa. Nos casos em que as *ações de sintonização* envolvam mudanças nos parâmetros do escalonador ou a redistribuição dos fluxos do usuário nos mesmos, o *agente de sintonização* deve ser capaz de gerenciar os recursos do provedor, sendo, nesses casos, objetos do tipo *QoSTResourceManager*.

Caso o *agente de sintonização* não seja capaz de manter o nível de QoS, deve contactar o *agente de negociação*, ou, em última instância, o próprio usuário, pelo envio de alertas (método *alert*). Nesses casos, um novo contrato de serviço deve ser renegociado com o usuário, com a escolha de caminhos alternativos, por exemplo.

3.3.4 Exemplo de Aplicação do Framework para Sintonização de QoS

A Figura 13 ilustra uma aplicação do Framework para Sintonização de QoS na modelagem de um provedor de serviços integrados com o uso do protocolo RSVP. Cada fluxo do usuário é monitorado por meio de objetos da classe StreamMonitorImpl. Para tanto, uma *estratégia de monitorização* especial

foi empregada, modelada por objetos da classe IntservFlowMonStrategy. Ao serem detectados fluxos que tenham o nível de QoS contratado sob o risco de ser rompido, o *mecanismo de monitorização* gera alertas ao *agente de sintonização*, pela chamada ao método *tune()*. O agente modelado no exemplo, representado pela classe *IntservQoSTuner*, derivada das classes *QoSTAgent* e *QoSTResourceManager*, procura, inicialmente, reconfigurar os mecanismos de escalonamento. Caso não obtenha os resultados desejados, o agente repassa o alerta ao *agente de negociação*, que se encarrega de escolher um caminho alternativo por onde serão encaminhadas as mensagens RSVP do tipo PATH. Como conseqüência do redirecionamento dessas mensagens, uma nova *negociação* ocorrerá entre os roteadores que compõem o caminho alternativo escolhido.

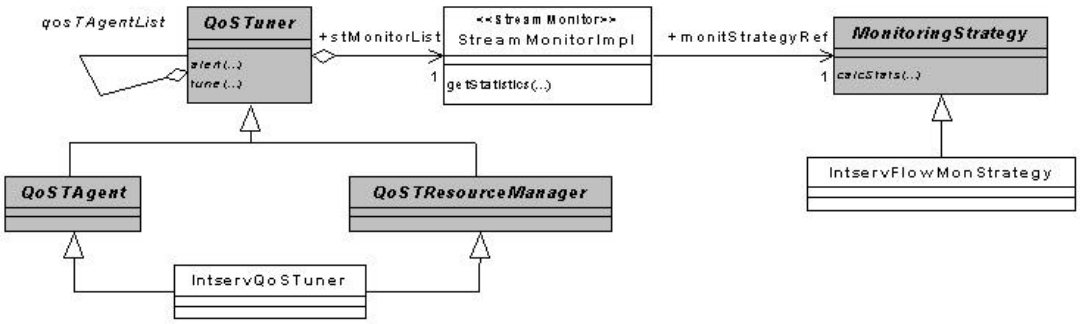

Figura 13: Aplicação do Framework para Sintonização de QoS.

3.4 Sumário

O conjunto dos frameworks descritos no presente capítulo definem uma arquitetura adaptável à provisão de QoS na Internet. Ao modelar as estruturas e funções comuns aos vários modelos e tecnologias de provisão de QoS, a referida arquitetura pode servir de base à construção de sistemas bastante flexíveis. Através de vários exemplos, foi mostrado como os frameworks discutidos podem ser empregados na modelagem de cenários reais de provisão de QoS na Internet, tendo como resultado a definição de sistemas mais facilmente adaptáveis à introdução de novos serviços e funções.

Capítulo 4

Exemplo de Aplicação da Arquitetura

Para fins de demonstração de como os frameworks da arquitetura proposta podem ser aplicados na modelagem de um cenário real de provisão de QoS na Internet, o presente capítulo descreve a implementação de uma configuração de serviços integrados sobre ATM, com negociação dinâmica e distribuída, empregando o protocolo RSVP. Vários motivos influenciaram a escolha dessa configuração.

Tendo sido formalizado como proposta em 1994, ano em que sua especificação foi publicada na forma de RFC (*Request For Comments*) do IETF [BRADEN94], o modelo *intserv* obteve, em pouco tempo, uma repercussão bastante positiva, tanto no ambiente acadêmico quanto no comercial. Atualmente, diversos fabricantes de roteadores IP disponibilizam módulos de controle *intserv* / RSVP como itens opcionais em seus produtos. Prevendo o crescimento no suporte a essa tecnologia pelos provedores de serviço na Internet, várias empresas e centros de pesquisa passaram a desenvolver APIs *intserv* / RSVP que possibilitassem às aplicações distribuídas especificar os seus requisitos particulares de QoS. A facilidade em se obter versões dessas APIs para os mais diversos ambientes de desenvolvimento foi um dos fatores que mais fortemente influenciou a escolha do modelo *intserv* na composição do cenário implementado.

Uma relação dos principais roteadores e APIs *intserv* / RSVP disponíveis pode ser obtida em [GAINES98].

Somando-se à facilidade supracitada, o modelo *intserv*, aliado a outras tecnologias de sub-rede (SBM e 802.1p [BAKER00a]), tem sido apontado como o mais indiciado à provisão de QoS nas redes de acesso do usuário. A Seção 2.4 salientou algumas das principais vantagens obtidas com a utilização desse modelo, como a reserva eficiente dos recursos, o controle mais fino do tráfego do usuário e a possibilidade de implementação de políticas de controle de acesso mais granulares.

O interesse em se demonstrar os mecanismos de mapeamento e a recursividade dos frameworks constituintes da arquitetura proposta tornava preponderante, na composição do cenário-exemplo, o emprego de uma tecnologia de provisão de QoS no nível de sub-rede, em conjunto com o *intserv*. A tecnologia ATM (*Assynchronous Transfer Mode*), disponível atualmente no laboratório *TeleMídia*, onde a implementação foi realizada, permite o fornecimento de serviços com requisitos rígidos de QoS. Embora a maior carga de sinalização (devido ao cabeçalho das células) possa desencorajar o seu uso em enlaces de altíssima velocidade, além das dificuldades técnicas em se desenvolver comutadores a essas velocidades, conforme discutido na Seção 2.5, a tecnologia ATM ainda continuará a ser empregada nos provedores de serviço da Internet por muitos anos. A razão para isso deve-se, em grande parte, ao ainda significativo número de implementações atuais baseadas nessa solução, sendo, portanto, perfeitamente adequada aos propósitos de demonstração da arquitetura proposta.

O restante do capítulo encontra-se estruturado como se segue. Inicialmente o cenário implementado é descrito, com a identificação dos subsistemas que o compõe e dos serviços que podem ser solicitados pelas aplicações usuárias desse ambiente de provisão de QoS. Em seguida, são citadas as tarefas que foram realizadas durante a preparação do referido cenário, como a configuração de agentes de negociação *intserv* nas estações e roteadores, além da definição dos recursos disponíveis aos usuários e oferecidos pelo provedor de

serviços. A estruturação dos serviços e dos parâmetros de caracterização e desempenho empregados pelos subsistemas que compõem o ambiente de provisão de QoS é discutida na seqüência, quando uma aplicação dos Frameworks para Parametrização de Serviços é descrita. Dando prosseguimento, os aspectos particulares do cenário escolhido são discutidos, como a forma de identificação dos fluxos transmitidos e das sessões de trabalho, os mecanismos empregados na notificação de eventos e no anúncio de tráfego, dentre outros. O restante do capítulo é dedicado a implementação dos principais mecanismos empregados nas fases de solicitação, estabelecimento e manutenção de contratos de serviço.

4.1 Descrição do Cenário

A Figura 14 esquematiza o cenário-exemplo, modelado, no presente capítulo, com o auxílio dos frameworks constituintes da arquitetura proposta. Com a implementação desse cenário, aplicações iniciadas em quaisquer das redes de acesso ilustradas na figura poderão negociar, junto ao provedor de serviços *intserv*, destacado no centro, parâmetros de QoS que satisfaçam aos seus requisitos especiais[26].

[26] No cenário desenvolvido, assume-se que a provisão de QoS nas redes de acesso é realizada por super-dimensionamento. Em configurações mais próximas da realidade, a utilização de protocolos de gerência local de recursos, como o SBM (*Subnet Bandwidth Manager*) [BAKER00a], em conjunto com o modelo *intserv* tem-se mostrado mais apropriado à provisão de QoS em redes como a do exemplo em questão.

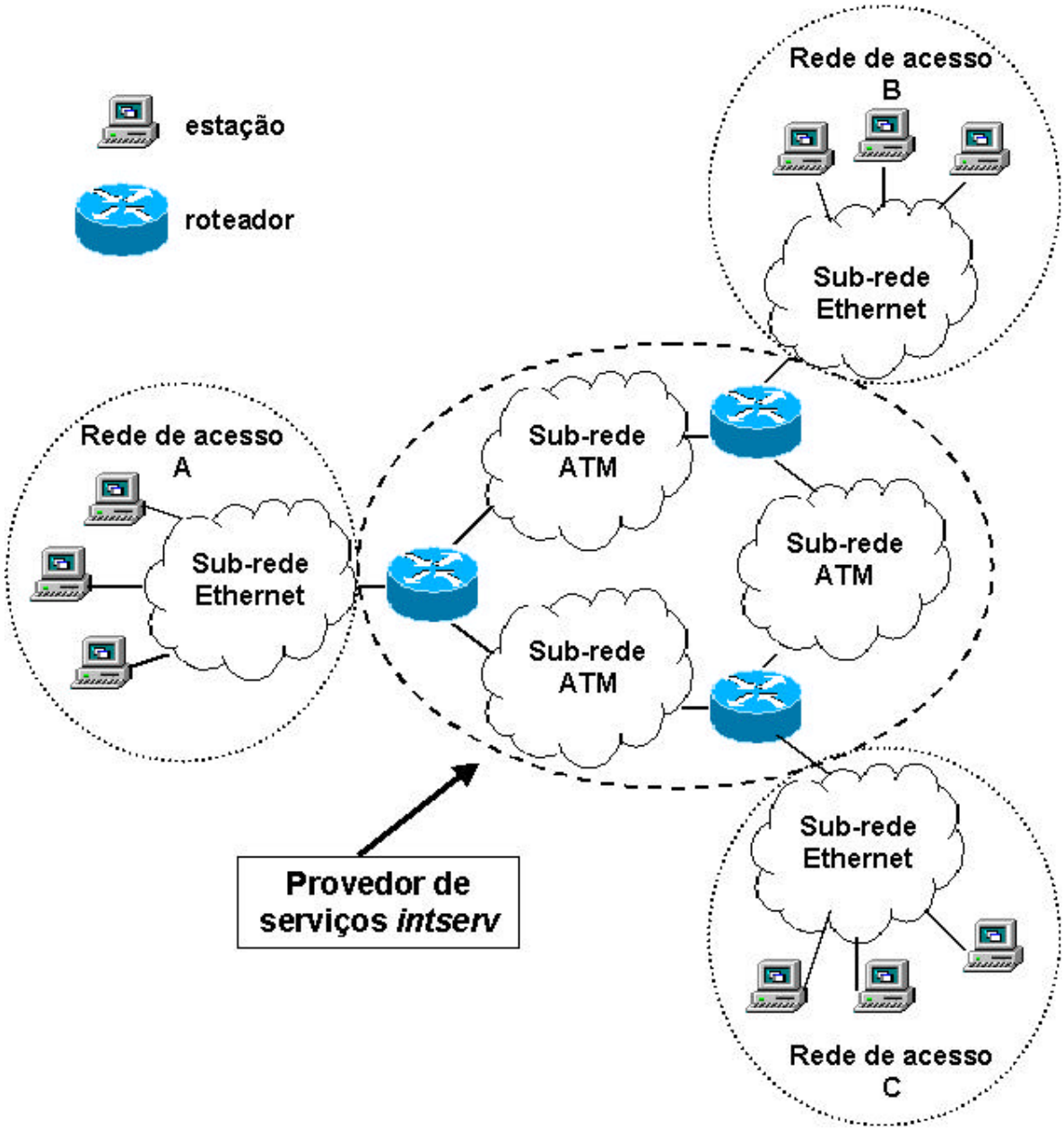

Figura 14: Cenário de provisão de serviços *intserv* sobre ATM.

Como pode ser visto pela Figura 14, o serviço *intserv* fornecido pelo provedor emprega, internamente, a tecnologia ATM na ligação entre os seus roteadores. A partir de estimativas sobre o tráfego que flui entre cada par de roteadores vizinhos, tráfego esse classificado de acordo com a categoria de serviço *intserv* associada, circuitos virtuais permanentes ou PVCs (*Permanent Virtual Circuits*) ATM foram configurados previamente. No processo de estabelecimento de um serviço *intserv* solicitado, caberá aos roteadores do

provedor realizar o controle de admissão nos referidos PVCs. Para tanto, um esquema de mapeamento entre parâmetros de serviços *intserv* e ATM foi definido.

4.2 Iniciação do Sistema

O sistema de comunicação modelado neste capítulo é formado por estações e roteadores, além de comutadores ATM. Em cada estação, foi configurado um agente de negociação *intserv* / RSVP e o endereço do roteador IP mais próximo, para onde as mensagens RSVP devem ser encaminhadas. Um outro agente de negociação *intserv* / RSVP foi configurado nos roteadores IP. Esse agente tem, como principal função, coordenar o processo de controle de admissão nas sub-redes ATM utilizadas na comunicação com os roteadores vizinhos, levando-se em consideração a categoria de serviço e os parâmetros de QoS informados pelas aplicações. Será na fase de iniciação do sistema que os PVCs mencionados na Seção 4.1 serão criados nas sub-redes ATM, dois para cada categoria de serviço *intserv* (garantida e carga controlada) e um terceiro adicional para o transporte do tráfego de melhor-esforço, dimensionados com base nas estimativas de tráfego das categorias de serviço correspondentes. A Figura 15 ilustra a ligação entre roteadores IP vizinhos por intermédio dos PVCs ATM configurados.

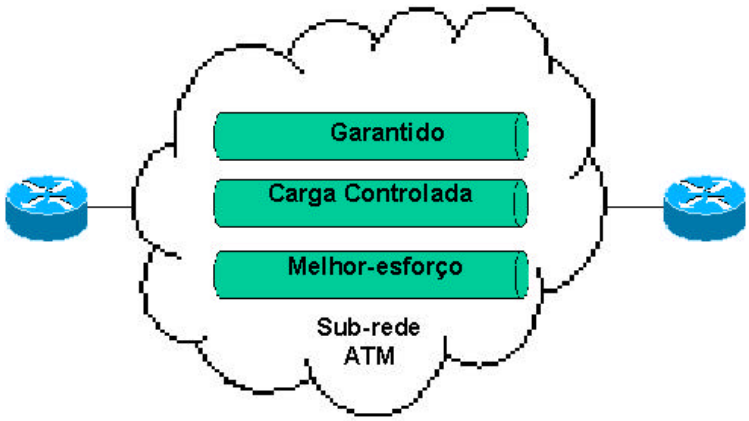

Figura 15: Ligação entre roteadores IP vizinhos.

4.3 Parametrização de Serviços

A Figura 16 ilustra como foi aplicado o Framework para Parametrização de Serviços na modelagem dos serviços empregados no cenário de provisão de QoS descrito nesse capítulo. De forma a simplificar o mecanismo de mapeamento, a ser discutido com maiores detalhes na Seção 4.7, foi definido um serviço ATM do tipo CBR (*Constant Bit Rate*), modelado pela classe CBRATMServiceCategory. Essa categoria possui um único parâmetro, correspondente à taxa de pico e representado por um objeto do tipo PCR (*Peak Cell Rate*).

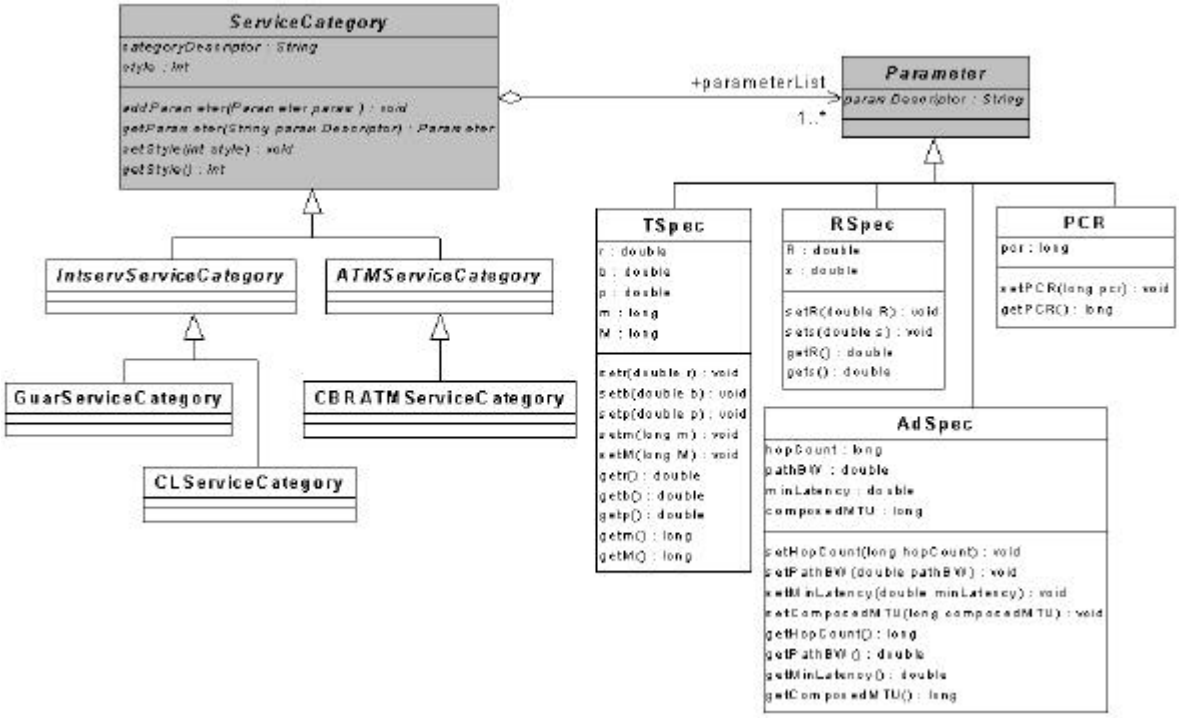

Figura 16: Parametrização de serviços.

4.4 Identificação dos Fluxos

Na solicitação de qualquer serviço com qualidade, os usuários devem identificar os fluxos de tráfego sobre os quais serão aplicados os parâmetros de QoS requisitados. Essa identificação servirá de base para que filtros apropriados possam ser configurados nos agentes de classificação, de acordo com procedimento descrito na Seção 3.2.1. No modelo *intserv*, a identificação de um fluxo depende da versão do protocolo IP. Na versão 4, por exemplo, um fluxo de tráfego é identificado pelo endereço IP do transmissor concatenado ao número da porta do protocolo de transporte utilizado. A versão 6, por sua vez, introduziu o campo *flowlabel*, de forma a serem evitados possíveis problemas decorrentes da leitura de portas de transporte, como a criptografia de nível 3 [ATKINSON98] ou fragmentação de pacotes[27].

A Figura 17 mostra a modelagem da identificação dos fluxos de tráfego do usuário utilizada na implementação do cenário-exemplo de provisão de serviços *intserv*. A classe IntservFilterSpec, derivada da classe abstrata *FilterSpec*, permite a identificação de fluxos apenas pelo endereço IP do transmissor. As especializações IPv4FilterSpec e IPv6FilterSpec, por sua vez, tornam possível uma identificação mais granular, baseado na concatenação do referido endereço com a porta do protocolo de transporte e o campo *flowlabel*, respectivamente.

[27] Nesta seção, está sendo considerada apenas a identificação de fluxos que pertençam a uma mesma *sessão de QoS*. O conceito de *sessão de QoS*, como será visto na Seção 4.5, define o grupo de receptores dos referidos fluxos, completando, dessa forma, o processo de identificação dos mesmos.

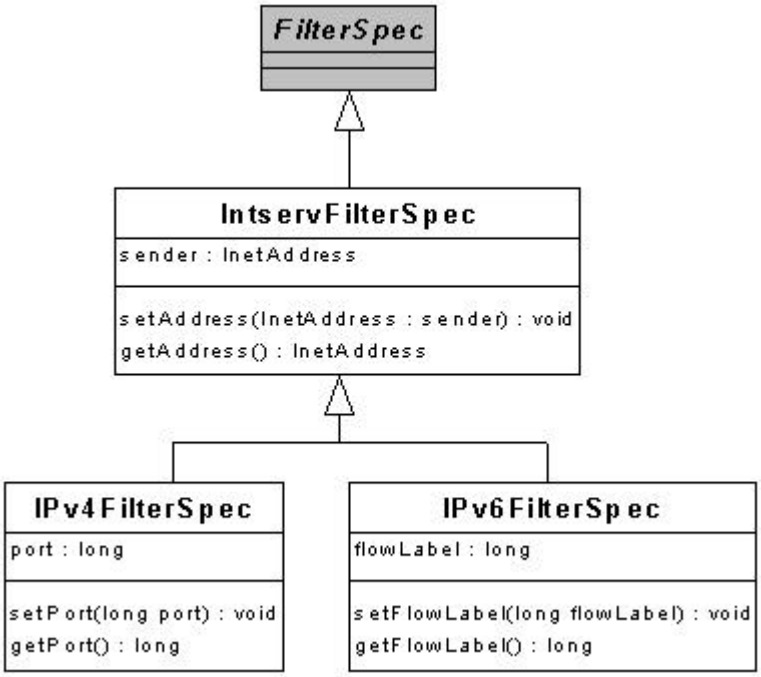

Figura 17: Identificação dos fluxos de tráfego do usuário.

Como estruturas auxiliares na solicitação de serviços e no conseqüente procedimento de configuração do agente de classificação, a modelagem da Figura 17 insere-se tanto no contexto do Framework para Negociação de QoS quanto no dos Frameworks para Compartilhamento de Recursos, descritos nas Seções 3.3.1 e 3.2.1, respectivamente.

4.5 Anúncio de Tráfego

Em uma cenário de provisão *intserv* / RSVP, a solicitação de serviços é, normalmente, precedida por um ou mais anúncios de tráfego por parte dos transmissores. Conforme mencionado na Seção 2.2.2, por intermédio desse mecanismo os transmissores podem informar as características do tráfego relativo aos fluxos que se encontram aptos a gerar. Essas informações são encaminhadas aos receptores pertencentes a uma mesma sessão de trabalho, denominada *sessão de QoS*. Em um cenário de provisão *intserv* / RSVP, a *sessão de QoS* é

chamada de *sessão RSVP*, sendo composta pela tupla endereço de destino, tipo e número de identificação da porta do protocolo de transporte utilizado.

Além de permitir que os destinatários de um fluxo sejam identificados, uma *sessão de QoS* possibilitará, no contexto específico da implementação descrita neste capítulo, que *eventos de QoS* sejam notificados à aplicação. Desta forma, ao se inscrever em uma *sessão de QoS*, a aplicação deve informar, por meio de uma referência (normalmente um ponteiro), qual objeto será responsável por receber e tratar os *eventos de QoS* ocorridos na sessão. Um *evento de QoS* pode ter vários significados, como a chegada de uma mensagem de anúncio de tráfego, a solicitação, admissão ou rejeição de um serviço, a ruptura do contrato estabelecido, enfim, qualquer ocorrência relacionada ao processo de negociação entre a estação e o sistema de comunicação. A Figura 18 ilustra os *eventos de QoS* utilizados na implementação do cenário descrito neste trabalho.

Uma aplicação inscreve-se em uma *sessão de QoS* pela criação de uma instância de *QoSSession*, quando é informado qual objeto irá receber e tratar os *eventos de QoS* ocorridos na referida sessão. Esse objeto tratador de eventos deve implementar, obrigatoriamente, a interface *QoSEventListener*, responsável por definir a assinatura do método eventDispatched(), chamado sempre que ocorrerem eventos na sessão supracitada. A Figura 19 mostra a hierarquia decorrente da especialização da classe *QoSSession* e o seu relacionamento com a interface *QoSEventListener*[28].

[28] Outros mecanismos de notificação de eventos poderiam ter sido empregados, em lugar do esquema sugerido, dependendo do ambiente de desenvolvimento. Por essa razão, optou-se por deixar a escolha do mecanismo mais apropriado totalmente a cargo do projetista do sistema, ao invés de inserir um determinado esquema de notificação como um dos componentes da arquitetura proposta.

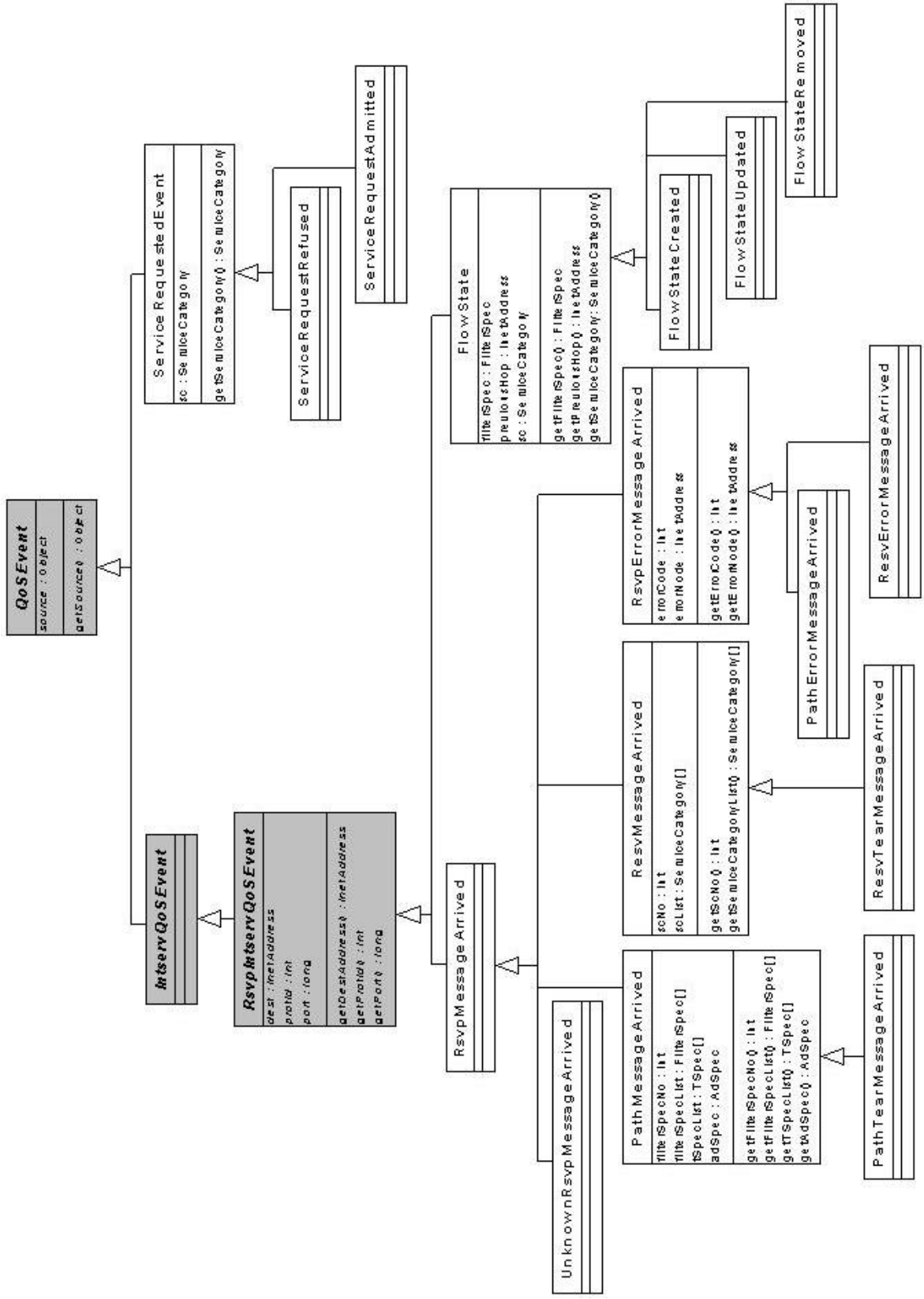

Figura 18: Eventos de QoS.

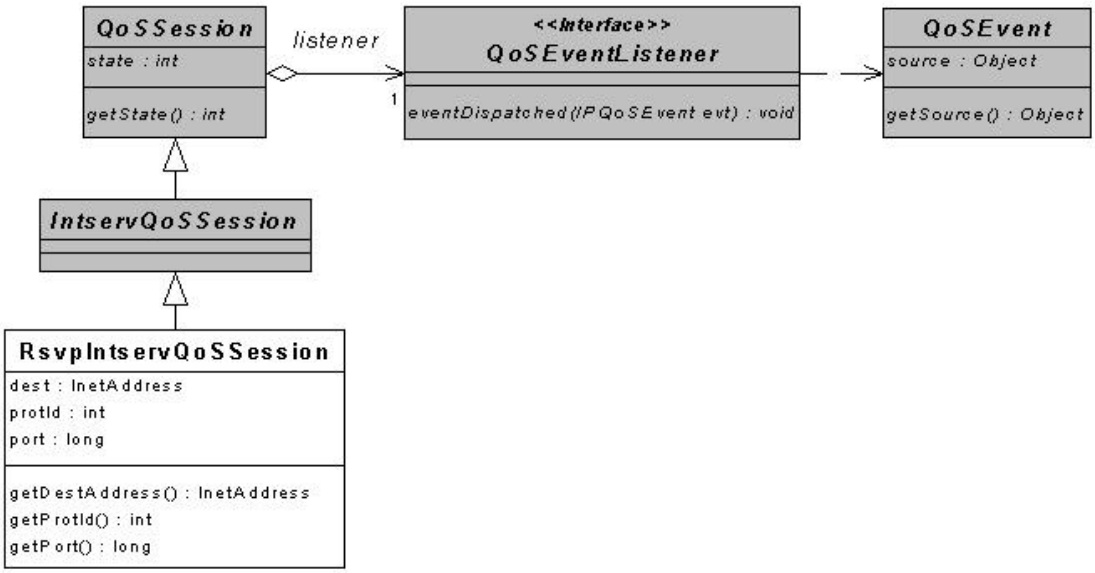

Figura 19: Sessão de QoS.

Após a sua inscrição na *sessão de QoS* correspondente ao grupo de possíveis receptores dos seus fluxos, a aplicação transmissora pode iniciar o processo de anúncio de tráfego. Na implementação do cenário utilizado como exemplo neste capítulo, esse processo é realizado pela chamada ao método *sender()*, definido no agente de negociação *intserv* / RSVP local. Esse método possui três parâmetros, que descrevem:

- a *sessão de QoS* onde o tráfego deve ser anunciado;
- a identificação dos fluxos anunciados (objeto FilterSpec);
- os parâmetros de caracterização do tráfego correspondentes aos fluxos anunciados.

No caso específico do cenário implementado, o agente de negociação local, ao receber a solicitação de anúncio de tráfego, formatará uma mensagem RSVP do tipo PATH e a enviará aos receptores da *sessão de QoS*, de acordo com o procedimento descrito na seção 2.2.2. Os agentes de negociação, localizados nas estações receptoras inscritas na mesma *sessão RSVP*, traduzirão a mensagem PATH recebida na forma de um *evento de QoS* do tipo

PathMessageArrived, notificando a(s) aplicação(ões)[29] pela chamada ao método eventDispatched(). A Figura 20 ilustra, com o auxílio de dois diagramas de seqüência, o processo de anúncio de tráfego no transmissor e no receptor.

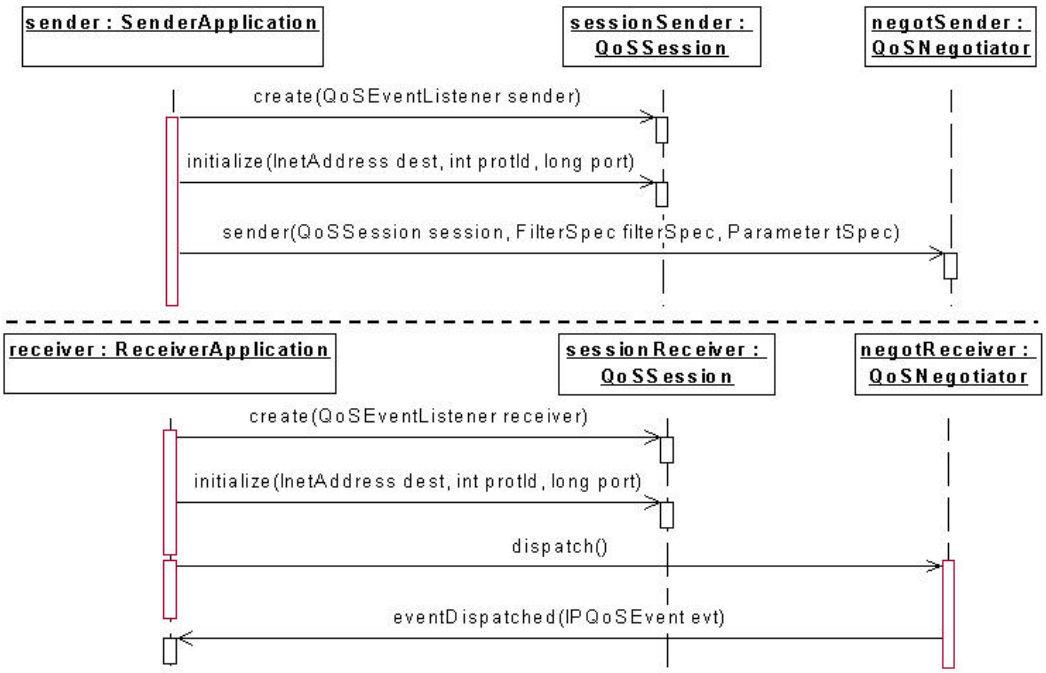

Figura 20: Anúncio de tráfego.

Conforme pode ser visto na Figura 20, o objeto sender, que representa a aplicação transmissora, inscreve-se na *sessão RSVP* pela criação de uma instância de *QoSSession*. Na ocasião, foi informado que o próprio objeto sender será responsável por receber e tratar os eventos ocorridos na sessão criada. Para tanto, a classe SenderApplication deve implementar, obrigatoriamente, a interface *QoSEventListener*. Em seguida, o método initialize() é invocado, onde são informados os demais parâmetros necessários à caracterização de uma *sessão RSVP*: endereço de destino, tipo e porta do protocolo de transporte utilizada na recepção dos fluxos. O anúncio de tráfego é realizado na seqüência, com a chamada ao método sender(). A aplicação receptora, por sua vez, representada no diagrama pelo objeto receiver, após ter se inscrito na mesma *sessão RSVP*, também como tratador de eventos de QoS

[29] Nada impede que, em uma mesma estação, haja mais de uma aplicação inscrita na mesma sessão.

(*listener*), aguarda a ocorrência de eventos, como a chegada de mensagens de anúncio de tráfego (PathMessageArrived). Para tanto, deve verificar, constantemente, junto ao agente de negociação local, se eventos ocorreram na sessão em que está inscrito. Essa verificação é realizada pela chamada ao método dispatch(), também disponibilizado pelo agente de negociação. Caso afirmativo, o referido agente notifica o evento ocorrido pela chamada ao método eventDispatched(), definido na aplicação receptora.

4.6 Solicitação de Serviços

Normalmente, após terem sido notificadas da ocorrência de eventos de anúncio de tráfego e estando de posse das informações relacionadas às características dos fluxos disponíveis na sessão de QoS ao qual estão inscritos, as aplicações receptoras podem iniciar o processo de solicitação de serviços. Este processo deve levar em consideração, além dos parâmetros relacionados ao tráfego anunciado, os requisitos próprios da aplicação, como o nível de garantia desejado, parâmetros de QoS, se o serviço será compartilhado entre os fluxos anunciados, dentre outros parâmetros negociáveis do SLA a ser contratado junto com o sistema de comunicação (confiabilidade, tarifação, etc.). Todos os parâmetros de caracterização de serviços, que, em conjunto, compõem o SLA, devem ser estruturados, de acordo com a arquitetura proposta, em uma ou mais categorias de serviço. No cenário-exemplo apresentado neste capítulo, as aplicações podem solicitar serviços *intserv* estruturados de acordo com a modelagem descrita na Seção 4.3, onde foi mostrada uma aplicação do Framework para Parametrização de Serviços.

Além dos métodos sender() e dispatch(), discutidos na Seção 4.5 e utilizados nos mecanismos de anúncio de tráfego e tratamento de eventos, respectivamente, os agentes locais de negociação *intserv* / RSVP definem o método request() empregado especificamente no processo de solicitação de serviços. De fato, na implementação do cenário descrito no presente capítulo, os

métodos públicos disponibilizados pelos agentes locais de negociação definem uma *API de QoS* [MOTA01]. No desenvolvimento das primitivas dessa API, especial atenção foi dispensada à definição dos seus parâmetros, de forma a possibilitar que a referida interface fosse independente do modelo ou tecnologia de provisão de QoS empregado pelo sistema de comunicação. A Tabela 1 descreve as principais primitivas da *API de QoS* definida.

Anúncio de Tráfego: sender	
Parâmetro	**Descrição**
QoSSession	sessão de QoS
FilterSpec	identificação dos fluxos anunciados
Parameter	parâmetro de caracterização do tráfego correspondente aos fluxos anunciados
Retorno	boolean (*true* se o tráfego pôde ser anunciado; *false* caso contrário)
Solicitação de Serviço: request	
Parâmetro	**Descrição**
QoSSession	sessão de QoS
InetAddress	endereço local por onde o serviço deve ser solicitado
int	número de fluxos especificados
FilterSpec[ ]	identificação dos fluxos solicitados
int	número de categorias de serviço especificadas
ServiceCategory[ ]	categorias de serviço solicitadas
Retorno	boolean (*true* se o serviço pôde ser solicitado; *false* caso contrário)
Descontinuidade de um Serviço: requestRelease	
Parâmetro	**Descrição**
QoSSession	sessão de QoS
InetAddress	endereço local por onde o pedido de descontinuação deve ser encaminhado
int	número de fluxos especificados
FilterSpec[ ]	identificação dos fluxos sobre os quais o serviço solicitado anteriormente deve ser descontinuado
Retorno	boolean (*true* se o serviço pôde ser descontinuado; *false* caso contrário)
Verificação de Eventos: verifyEvent	
Parâmetro	**Descrição**
Retorno	boolean (*true* se ocorreram eventos de QoS; *false* caso contrário)
Tratamento de Eventos: dispatch	
Parâmetro	**Descrição**
Retorno	boolean (*true* se ocorreram eventos de QoS; *false* caso contrário) obs.: caso tenham ocorrido eventos de QoS, o método eventDispatched() é invocado no objeto tratador de eventos da sessão de QoS correspondente.
Obtenção do Código do Último Erro: getLastError	
Parâmetro	**Descrição**
Retorno	int (código do erro ocorrido na última chamada à API

Tabela 1: Principais primitivas da API de solicitação de serviços.

Como pode ser observado na Tabela 1, a forma como os parâmetros do método request() foram estruturados permite variações que flexibilizam o processo de solicitação de serviço, de acordo com o estilo de compartilhamento escolhido. Caso o serviço solicitado seja compartilhado implicitamente, por exemplo, os fluxos não precisam ser identificados (parâmetro FilterSpec[ ]), sendo assumido que o referido serviço será aplicado a todos os fluxos que foram anunciados na *sessão de QoS* escolhida. Por outro lado, na solicitação de serviços compartilhados explicitamente, como o nome já indica, os fluxos devem ser, obrigatoriamente, identificados. Finalmente, nos casos em que o serviço requisitado não é compartilhado entre os fluxos, há duas possibilidades: o mesmo serviço aplicado distintamente a cada um dos fluxos ou um serviço especificado individualmente para cada um deles. Nesse último caso, deve haver, obrigatoriamente, uma paridade entre identificadores de fluxo e categorias de serviço nos parâmetros que compõem a primitiva request().

4.7 Estabelecimento de Contratos de Serviço

Os agentes de negociação *intserv* / RSVP, localizados nas estações e nos roteadores presentes no cenário exemplificado neste capítulo, são modelados, de acordo com a arquitetura proposta, por objetos da classe *QoSNegotiator*. Mais especificamente, foram definidas especializações da referida classe na modelagem desses agentes, como pode ser visto pela hierarquia de derivação ilustrada na Figura 21, onde é mostrado como o Framework para Negociação de QoS foi aplicado na implementação do cenário discutido.

Como pode ser observado pela Figura 21, a partir das classes *QoSNAgent* e *QoSNResourceManager*, especializações de *QoSNegotiator* discutidas na Seção 3.3.1, foram derivadas, em uma seqüência hierárquica, as classes *IntservQoSNegotiator* e *RsvpIntservQoSNegotiator*. Na primeira, encontram-se definidos os mecanismos de negociação que são inerentes ao modelo *intserv*, sendo independentes do protocolo de sinalização empregado na

comunicação entre os agentes. A porção do agente de negociação que depende do protocolo RSVP foi modelada pela classe *RsvpIntservQoSNegotiator*, tendo sido, mais uma vez, especializada, de acordo com o tipo do agente ou ambiente em que foi implementado. Seguindo-se essa diretiva, foram definidos três tipos de agentes de negociação local, sendo dois deles específicos para estações e um terceiro para roteadores. Para os agentes localizados nas estações, foram implementadas duas versões: uma para o ambiente Solaris® (Sun Microsystems), denominada SolISI100QoSNegotiator, e outra para a plataforma Windows® (Microsoft Corporation), chamada de Win202QoSNegotiator. Apesar desses agentes disponibilizarem uma interface de acesso desenvolvida utilizando-se a linguagem Java, empregam, internamente, APIs RSVP nativas escritas em C++: a *Solstice Bandwidth Reservation Protocol API* [SOLSTICE01], para Solaris®, e a *Intel PC-RSVP* [PCRSVP01], para Windows®. O agente de negociação instalado nos roteadores, no entanto, é totalmente independente de plataforma, tendo sido implementado pela classe RouterRsvpIntservQoSNegotiator. Apesar de diferentes agentes terem sido definidos, todos eles respeitam a interface especificada pela *API de QoS* descrita na Seção 4.6.

Após o recebimento de uma solicitação de serviços, o agente de negociação local, presente nas estações receptoras e implementado por objetos do tipo SolISI100QoSNegotiator ou WIN202QoSNegotiator, dependendo da plataforma instalada na estação, formata uma mensagem RSVP do tipo RESV, tendo como base os parâmetros informados pelo método request(). Essa mensagem é encaminhada, em seguida, ao roteador IP mais próximo, configurado na estação no momento de instalação do agente.

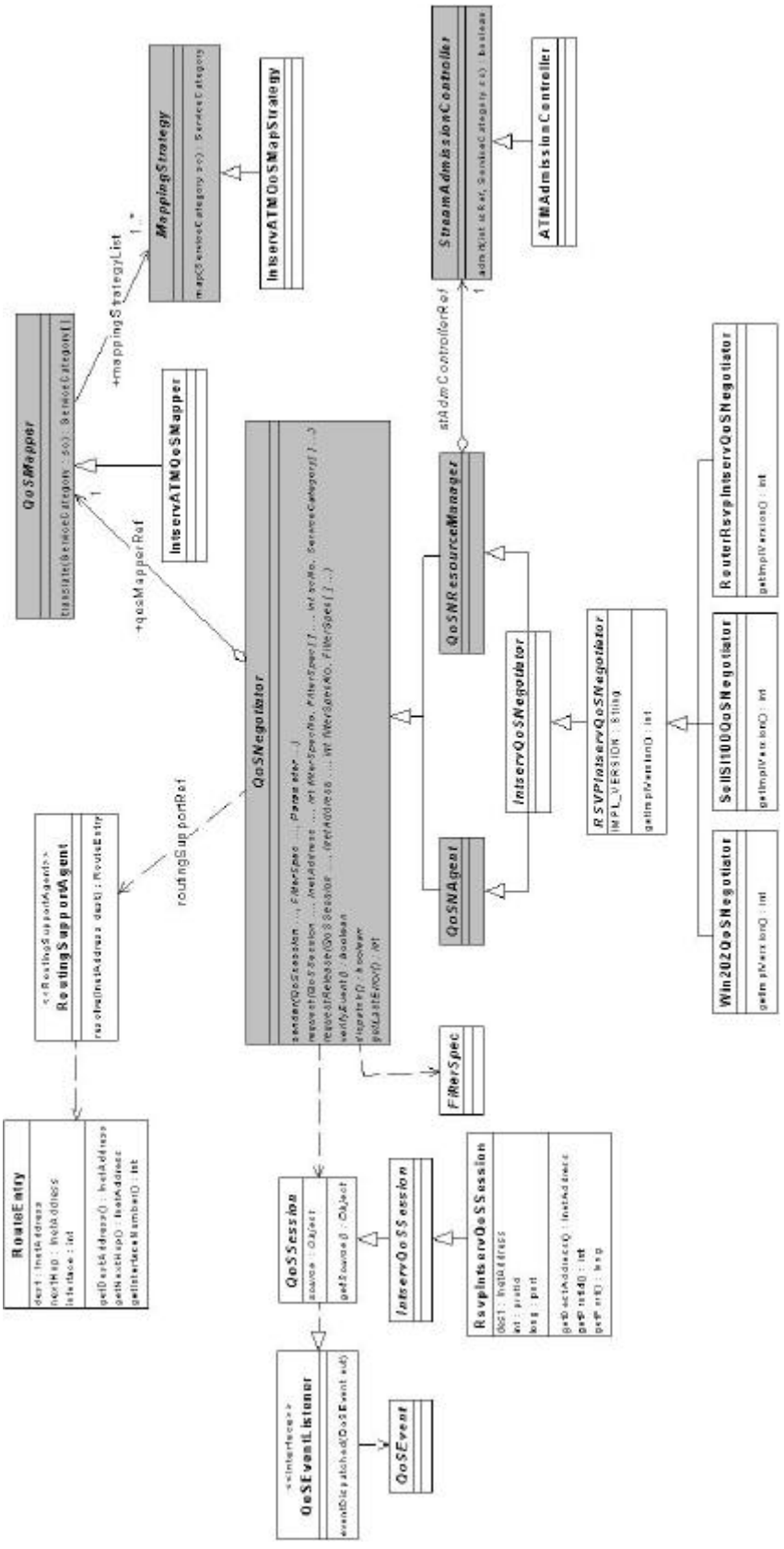

Figura 21: Negociação de QoS.

As mensagens RSVP, de acordo com as primeiras especificações do protocolo, deveriam ser transmitidas na rede encapsuladas diretamente sobre o IP, no chamado modo *raw IP*. O número 46, de acordo com a referida especificação, era reservado à identificação do tipo de protocolo RSVP. Porém, devido a maioria das estações não suportarem o modo *raw* IP, a especificação do protocolo passou a permitir que as mensagens RSVP passassem a ser transmitidas sobre o UDP, utilizando-se, para tanto, as portas reservadas 1698 ou 1699. Em particular, os agentes de negociação localizados nas estações do cenário descrito no presente capítulo empregam a porta UDP 1698 no encaminhamento das mensagens de sinalização.

No cenário implementado, todos os pacotes UDP identificados pela porta 1698 são capturados pelo agente de negociação *intserv* / RSVP localizado nos roteadores. É função desse agente analisar a PDU (*Protocol Data Unit*) UDP capturada, de forma a determinar o tipo de mensagem RSVP, para, em seguida, extrair as estruturas contidas na mesma: dados da sessão, identificação dos fluxos, categoria de serviço, parâmetros de QoS, dentre outras. Como resultado desse processo de análise, um evento de QoS apropriado é formatado.

As mensagens reconhecidas pelo agente de negociação receberão tratamento diferenciado, dependendo do seu tipo. Mensagens RSVP do tipo PATH, por exemplo, podem resultar na criação, atualização ou remoção de estados associados aos fluxos anunciados. Para a manutenção desses estados, uma classe especial, denominada FlowState, foi desenvolvida especificamente para a implementação descrita neste capítulo, de forma a serem armazenadas informações relativas à sessão de QoS ao qual cada fluxo se insere, à identificação dos fluxos, ao endereço do roteador por onde a mensagem foi recebida e à situação geral de cada fluxo (se encontra-se atendido ou não por um determinado serviço, por exemplo). A Figura 22 ilustra os atributos e métodos constituintes da classe FlowState.

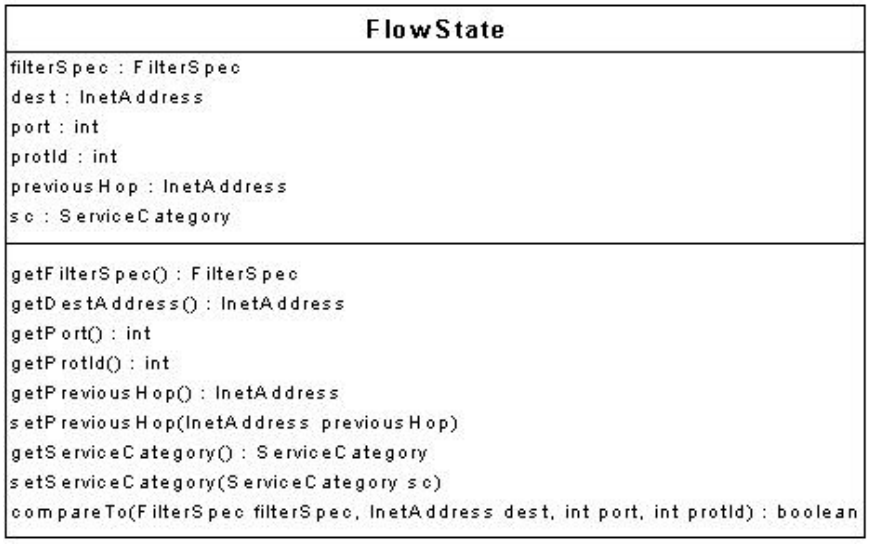

Figura 22: Classe FlowState.

O tratamento de uma mensagem RSVP do tipo RESV depende da interface de recebimento. Caso a mensagem tenha sido recebida por uma interface Ethernet, o agente de negociação simplesmente a encaminha por uma ou mais interfaces de saída, dependendo do endereço dos roteadores por onde os fluxos solicitados na mensagem de reserva foram anunciados. Esses endereços podem ser obtidos por intermédio de uma consulta à base de estados associados aos fluxos (método getPreviousHop() definido em FlowState). Conforme discutido na Seção 2.2.2, as mensagens RESV são transmitidas no caminho inverso ao das mensangens PATH correspondentes, de forma a permitir que a reserva de recursos seja feita no mesmo caminho por onde os fluxos são transmitidos.

Caso a mensagem RSVP do tipo RESV tenha sido recebida por intermédio de uma interface ATM, o agente de negociação presente nos roteadores deve iniciar o procedimento de controle de admissão do(s) serviço(s) solicitado(s). No ambiente de provisão de serviços descrito neste capítulo, o controle de admissão consiste na verificação da possibilidade de inserção de um ou mais fluxos com determinados requisitos de QoS em PVCs ATM, de acordo com a categoria de serviço solicitada. Cada PVC pode ser considerado um recurso virtual, cuja capacidade, definida em termos de vazão disponível, é determinada na fase de iniciação do provedor de serviços. O compartilhamento

desses recursos entre os fluxos admitidos é controlado por escalonadores de recursos virtuais, de acordo com a nomenclatura utilizada na Seção 3.2.

Como pode ser observado na Figura 21, o mecanismo de controle de admissão foi modelado pela classe ATMAdmissionController, uma especialização de *StreamAdmissionController*. O agente de negociação interage com esse mecanismo pela chamada ao método admit(), devendo ser informado o escalonador de recurso virtual responsável pelo atendimento ao serviço juntamente com os parâmetros de caracterização correspondentes, devidamente traduzidos em valores que possibilitem ao mecanismo de admissão deduzir a quantidade de recursos virtuais que devem ser reservados. Essa tradução, de acordo com a arquitetura proposta, é realizada pelos mecanismos de mapeamento, modelados por objetos da classe *QoSMapper*. No cenário implementado neste capítulo, uma especialização dessa classe, denominada IntservATMQoSMapper, é responsável por traduzir serviços *intserv* solicitados em um serviço ATM do tipo CBR, modelado pela classe CBRATMServiceCategory, descrita na Seção 4.3. A estratégia de mapeamento empregada, denominada IntservATMQoSMapStrategy, é bastante simples, sendo baseada unicamente na taxa de pico informada, como ilustra a Figura 23. Em relação aos parâmetros de QoS contidos na solicitação de serviços garantidos, assume-se, nesta implementação, que os valores fornecidos pelo provedor *intserv* são relativamente baixos, oferecendo, portanto, uma qualidade superior a que possa vir a ser solicitada.

Obviamente que a estratégia de mapeamento escolhida não resulta em um uso eficiente dos recursos disponíveis em cada sub-rede ATM, sendo apropriado um estudo mais aprofundado sobre as formas possíveis de mapeamento *intserv* / ATM. Porém, a referida estratégia é suficiente para o propósito de demonstração de como os mecanismos descritos pela arquitetura proposta podem ser aplicados na modelagem de um cenário de provisão de QoS na Internet.

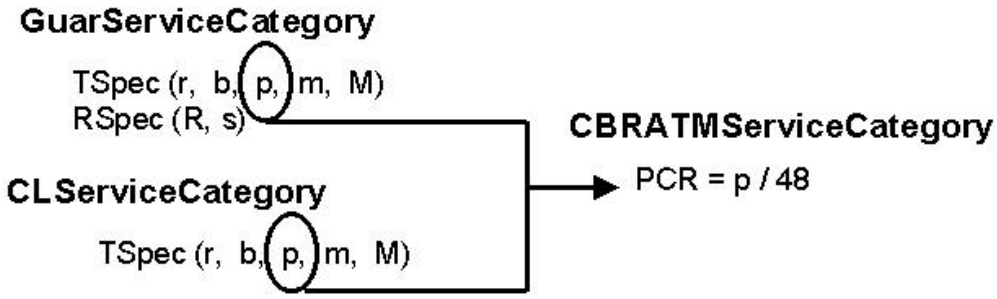

Figura 23: Estratégia de mapeamento.

Como os parâmetros que expressam taxas (**r**, **p** e **R**) nos serviços *intserv* utilizam a unidade *bytes por segundo*, a estratégia de mapeamento empregada divide o valor do parâmetro taxa de pico (**p**) por 48 (quantidade de bytes de informação transportados por cada célula ATM) de forma a se obter a mesma taxa expressa em *células por segundo*.

Dependendo da admissão ou não de um serviço solicitado, eventos do tipo ServiceRequestAdmitted ou ServiceRequestRefused, respectivamente, são notificados pelo agente de negociação. Nos casos em que o serviço solicitado possa ser admitido, o agente de negociação inicia o procedimento de configuração dos mecanismos de classificação e policiamento. De acordo com a arquitetura proposta, esses mecanismos são modelados pelos Frameworks para Compartilhamento de Recursos, discutidos na Seção 3.2. Para a configuração do módulo de classificação, os identificadores de fluxos, encapsulados nas mensagens de solicitação de serviço *intserv* na forma de estruturas do tipo FilterSpec, juntamente como as categorias de serviço associadas, são utilizados na definição de novos filtros. Em última instância, a classificação de um pacote determinará, no cenário discutido nesta seção, o circuito virtual que o transportará. Já o mecanismo de policiamento pode ser configurado pelo emprego dos descritores de tráfego (tspec) contidos na *flowspec* do serviço solicitado. Aqueles pacotes que forem identificados como fora do perfil de tráfego informado podem sofrer as mais variadas ações de policiamento, como o seu encaminhamento pelo PVC de melhor-esforço, por exemplo.

No cenário proposto, os escalonadores de recursos são configurados estaticamente, no momento de iniciação do provedor de serviços, quando são

definidos os PVCs ATM responsáveis pelo transporte dos fluxos admitidos em cada categoria *intserv*. Cabe ao agente de negociação apenas determinar, com o auxílio do mecanismo de controle de admissão, se um novo fluxo pode ser adicionado ao PVC, de acordo com a categoria de serviço escolhida.

Uma vez que o serviço solicitado tenha sido admitido e os mecanismos de controle de tráfego tenham sido configurados, o agente de negociação encaminha a mensagem RESV recebida por uma ou mais interfaces de saída, determinadas a partir do endereço dos roteadores por onde os fluxos solicitados foram anunciados. O pedido de solicitação de serviço, eventualmente, chega ao transmissores dos fluxos solicitados, sendo gerados eventos de notificação apropriados (ResvMessageArrived).

Objetivando-se verificar o funcionamento conjunto dos mecanismos envolvidos em um processo completo de solicitação e estabelecimento de contrato de serviço, foi implementada, nas estações do cenário exemplificado neste capítulo, uma aplicação de demonstração denominada IPQoSDemo. Esta aplicação possibilita a criação de sessões de QoS do tipo RSVP, anúncio de tráfego, solicitação de serviços *intserv*, bem como monitorização de eventos de QoS ocorridos na sessão criada. Em [MOTA01] pode-se obter maiores informações sobre a aplicação IPQoSDemo. Nos roteadores, por outra lado, foi implementado uma aplicação denominada IPQoSRouter e que objetiva exibir os eventos de QoS ocorridos, como a mudança do estado dos fluxos transportados, a chegada de mensagens de negociação, o encaminhamento dessas mensagens, e, finalmente, a aceitação ou rejeição de um serviço solicitado (eventos de controle de admissão).

4.8 Manutenção de Contratos de Serviço

No que se refere à manutenção do nível de QoS no provedor de serviços utilizado no exemplo deste capítulo, os mecanismos normalmente empregados, como a monitorização da QoS fornecida aos fluxos aceitos, dentre

outros, discutidos na Seção 3.3, encontram-se implementados nos subsistemas constituídos pelas sub-redes ATM. Esses mecanismos de sintonização, devido à provisão de QoS no exemplo discutido ser estática (uso de PVCs), têm a sua utilização limitada, restringindo-se à emissão de alertas ao administrador, responsável pela manutenção, incluindo o redimensionamento dos circuitos virtuais configurados nas sub-redes ATM, de acordo com a carga dos fluxos por categoria de serviço *intserv*. Neste exemplo, os mecanismos de policiamento, ao limitar a quantidade de fluxos encaminhados em cada PVC, têm uma importância fundamental à manutenção dos contratos de serviço acordados com os usuários.

4.9 Sumário

Com a implementação de um cenário de provisão de QoS baseado em uma configuração *intserv* / RSVP sobre ATM, o presente capítulo mostrou como as estruturas de modelagem da arquitetura proposta, definidas no Capítulo 3, podem ser ferramentas de grande utilidade aos projetistas de sistemas semelhantes na Internet. Também foi sugerido um esquema de notificação de eventos de QoS, implementado especificamente para o cenário descrito. Por esse esquema, as aplicações podem definir as funções que serão executadas como resposta à ocorrência de determinados eventos de QoS, como a chegada de mensagens de anúncio de tráfego, por exemplo. Uma outra contribuição significativa do presente capítulo foi a definição de uma API de solicitação de serviços independente do modelo de serviço ou tecnologia de provisão de QoS empregada internamente pelo provedor, o que é uma característica bastante desejável, uma vez que fornece um maior grau de portabilidade às aplicações que a utilizem.

Capítulo 5

Conclusões

O presente trabalhou procurou abordar as diversas propostas para provisão de QoS discutidas atualmente com uma visão generalista e sistemática. Do estudo dos aspectos comuns aos vários modelos de serviços e tecnologias de de encaminhamento ao nível das sub-redes, uma arquitetura adaptável à provisão de QoS foi definida. Valendo-se da utilização de frameworks como ferramentas de modelagem, a arquitetura definida, além de permitir a implementação de estruturas e funções de provisão de QoS, fornece mecanismos que possibilitam a a configuração de cenários mais complexos, envolvendo o uso combinado dos modelos de serviços e abordagens de provisão de QoS no nível das sub-redes. O emprego de frameworks na definição da arquitetura facilita a identificação dos pontos de flexibilização que viabilizam a sua adaptação a vários cenários possíveis de implementação. Diversos exemplos de utilização dos frameworks propostos foram descritos ao longo deste trabalho, procurando modelar as principais configurações discutidas atualmente na Internet. Ao final, um cenário mais completo, mostrando o fornecimento de serviços *intserv* com o uso combinado da tecnologia ATM na provisão de QoS, foi descrito de forma a demonstrar a aplicação da arquitetura proposta em uma configuração real.

5.1 Contribuições da Dissertação

O estudo sistemático dos diversos modelos e tecnologias de provisão de QoS, com identificação dos aspectos comuns e particulares de cada proposta, contribuiu para um estudo mais aprofundado sobre as alternativas de fornecimento de serviços com qualidade existentes atualmente. Ao serem exemplificados cenários envolvendo combinações possíveis dessas propostas, pôde-se mostrar os mecanismos principais empregados na interação entre os referidos modelos e tecnologias em um ambiente de provisão de QoS fim-a-fim.

A definição de uma arquitetura que permite a modelagem dos vários cenários de provisão de QoS discutidos atualmente na Internet é, sem dúvida, a maior contribuição do presente trabalho. Com a especificação de estruturas e mecanismos genéricos que podem ser aplicados aos vários modelos e tecnologias, ambientes com alto grau de adaptabilidade podem ser modelados. O grau de adaptabilidade desses ambientes é medido pela facilidade em se introduzir novos serviços, mecanismos de negociação, políticas de controle de acesso, estratégias de admissão, algoritmos de escalonamento, etc. Esse é um dos aspectos principais da arquitetura proposta, uma vez que as aplicações e tecnologias de comunicação de dados estão em um estágio de contínua evolução. Os subsistemas (estações, roteadores e sub-redes) que compõem a Internet devem estar preparados para acomodar, em tempo de operação, adaptações que hoje somente são possíveis por procedimentos menos dinâmicos (atualização de *hardware* ou *firmware* ou ainda a reinstalação completa de *software*).

Na implementação do cenário de provisão de QoS que demonstra a aplicabilidade da arquitetura definida, foi sugerida um API de solicitação de serviços independente do modelo ou tecnologia empregada internamente pelo provedor. Desta forma, um alto grau de portabilidade pode ser fornecido às aplicações que a utilizem. Também na implementação do referido cenário, foi definido um esquema de notificação de eventos de QoS que permite que as

aplicações possam participar, de forma mais dinâmica, no processo de negociação de serviços junto ao provedor.

5.2 Trabalhos Futuros

A implementação de cenários de provisão de QoS mais complexos com o emprego da arquitetura proposta poderia trazer contribuições e refinamentos às estruturas e mecanismos definidos. Muitas das dificuldades técnicas oriundas da implementação de cenários de maior complexidade podem ser contornadas pelo emprego de simuladores. Um estudo que se norteie pela utilização de ferramentas desse tipo na construção de cenários de provisão de QoS mais complexos, modelados pelo uso da arquitetura proposta, pode ser salientado como um trabalho futuro.

Uma questão ainda pouco explorada nos modelos de serviços propostos atualmente na Internet diz respeito aos aspectos de segurança. A habilidade em se reservar recursos no provedor de serviços implica na necessidade de autenticação, tanto do usuário que solicita a reserva, quanto dos pacotes que farão uso da mesma. Um estudo mais completo dos mecanismos relacionados à segurança em ambientes de provisão de serviços com QoS é uma área de pesquisa muito rica, e que apresenta diversos pontos que podem ser abordados futuramente.

Finalmente, a comunicação em redes móveis com QoS introduz uma série de dificuldades que devem ser consideradas pelo modelo de serviços. Renegociação de QoS, por exemplo, é um procedimento bastante comum nestas redes. Para ilustrar, consideremos o provimento de um serviço para vários usuários móveis em diferentes células ou pontos de acesso. Devido à mobilidade destes usuários, é possível que muitos deles se dirijam à mesma célula, que pode não ter recursos suficientes para atendimento a todos os usuários. Outra dificuldade consiste em se manter o serviço ininterrupto, mesmo durante os momentos críticos correspondentes às mudanças de pontos de acesso.

Finalmente, devido aos recursos de largura de banda serem mais escassos nas redes móveis, a sinalização deve ser minimizada.

Bibliografia

[ANDERSSON01] ANDERSSON, L., DOOLAN, P., FELDMAN, N., FREDETTE, A. e THOMAS, B. **LDP Specification**. IETF Request for Comments (RFC) 3036. Janeiro de 2001.

[ANDERSSON99] ANDERSSON, L., CAIN, B. e JAMOUSSI, B. **Requirement Framework for Fast Re-route with MPLS**. Trabalho em Desenvolvimento. IETF Working Draft draft-andersson-reroute-frmwrk-00.txt. Outubro de 1999.

[ATKINSON98] ATKINSON, R. e KENT, S. **IP Encapsulating Security Payload (ESP)**. IETF Request for Comments (RFC) 2406. Novembro de 1998.

[ATMFORUM96] ATM Fórum. **Private Network to Network Interface**. AF-PNNI-0055.000. Versão 1.0. Março de 1996.

[AWDUCHE01a] AWDUCHE, D., BERGER, L., GAN, D., LI, T., SRINIVASAN, V. e SWALLOW, G. **RSVP-TE: Extensions to RSVP for LSP Tunnels**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-mpls-rsvp-lsp-tunnel-08.txt. Fevereiro de 2001.

[AWDUCHE01b] AWDUCHE, D., CHIU, A., ELWALID, A., WIDJAJA, I. e XIAO, X. **A Framework for Internet Traffic Engineering**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-tewg-framework-03.txt. Março de 2001.

[BAKER00a] BAKER, F., BERNET, Y., HOFFMAN, D., SPEER, M. e YAVATKAR, R. **SBM (Subnet Bandwidth Manager): A Protocol for RSVP-based Admission Control over IEEE 802-style Networks**. IETF Request for Comments (RFC) 2814. Maio de 2000.

[BAKER00b] BAKER, F., BERNET, Y., BRADEN, R., DAVIE, B., FELSTAINE, E., FORD, P., SPEER, M., YAVATKAR, R., ZHANG, L. e WROCLAWSKI, J. **A Framework for Integrated Services Operation over Diffserv Networks**. IETF Request for Comments (RFC) 2998. Novembro de 2000.

[BAKER98a] BAKER, F., BERGER, L., BERSON, S., BORDEN, M., CRAWLEY, E. e KRAWCZYK, J. **A Framework for Integrated Services and RSVP over ATM**. IETF Request for Comments (RFC) 2382. Agosto de 1998.

[BAKER98b] BAKER, F., BLACK, D., BLAKE, S. e NICHOLS, K. **Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers**. IETF Request for Comments (RFC) 2474. Dezembro de 1998.

[BAKER99] BAKER, F., HEINANEN, J., WEISS, W. e WROCLAWSKI, J. **Assured Forwarding PHB Group**. IETF Request for Comments (RFC) 2597. Junho de 1999.

[BERGER98a] BERGER, L. **RSVP over ATM Implementation Guidelines**. IETF Request for Comments (RFC) 2379. Agosto de 1998.

[BERGER98b] BERGER, L. **RSVP over ATM Implementation Requirements**. IETF Request for Comments (RFC) 2380. Agosto de 1998.

[BERNET00] BERNET, Y. **Format of the RSVP DCLASS Object**. IETF Request for Comments (RFC) 2996. Novembro de 2000.

[BERNET99] BERNET, Y., BINDER, J., BLAKE, S., CARLSON, M., CARPENTER, B., DAVIES, E., KESHAV, S., OHLMAN, B., VERMA, D., WANG, Z. e WEISS, W. **A Framework for Differentiated Services**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-diffserv-framework-02.txt. Fevereiro de 1999.

[BLAKE97] BLAKE, S., GUERIN, R. e HERZOG, S. **Aggregating RSVP-based QoS Requests**. Trabalho em Desenvolvimento. IETF Working Draft draft-guerin-aggreg-rsvp-00.txt. Novembro de 1997.

[BLAKE98] BLAKE, S., BLACK, D., CARLSON, M., DAVIES, E., WANG, Z. e WEISS, W. **An Architecture for Differentiated Services**. IETF Request for Comments (RFC) 2475, Dezembro de 1998.

[BORDEN98] BORDEN, M. e GARRET, M. **Interoperation of Controlled-Load Service and Guaranteed Service with ATM**. IETF Request for Comments (RFC) 2381. Agosto de 1998.

[BORMANN99] BORMANN, C. **The Multi-Class Extension to Multi-Link PPP**. IETF Request for Comments (RFC) 2686. Setembro de 1999.

[BRADEN94] BRADEN, R., CLARK, D. e SHENKER, S. **Integrated Services in the Internet Architecture: an Overview**. IETF Request for Comments (RFC) 1633. Junho de 1994.

[BRADEN97] BRADEN, R., BERSON, S., HERZOG, S., JAMIN, S. e ZHANG, L. **Resource ReSerVation Protocol (RSVP): Version 1 Functional Specification**. IETF Request for Comments (RFC) 2205. Setembro de 1997.

[BRADEN98] BRADEN, R. e HOFFMAN, D. **RAPI - An RSVP Application Programming Interface - Version 5**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-rsvp-rapi-01.txt. Agosto de 1998.

[CALHOUN01] CALHOUN, P., **Diameter: Resource Management Extensions**. Trabalho em Desenvolvimento. IETF Working Draft draft-calhoun-diameter-res-mgmt-07.txt. Fevereiro de 2001.

[CALLON01] CALLON, R., ROSEN, E. e VISWANATHAN A. **Multiprotocol Label Switching Architecture**. IETF Request for Comments (RFC) 3031. Janeiro de 2001.

[CHANDRA00] CHANDRA, A., GOYAL, P., PRASHANT S. e SUNDARAM, V. **Application Performance in the QLinux Multimedia Operating System**. Em *Proceedings of the Eighth ACM Conference on Multimedia*. Los Angeles, CA. Páginas 127-136. Novembro de 2000.

[CHARNY01] CHARNY, A. e WROCLAWSKI, J. **Integrated Service Mappings for Differentiated Services Networks**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-issll-ds-map-01.txt. Fevereiro de 2001.

[CHEVAL01] CHEVAL, P., DAVARI, S., DAVIE, B., FAUCHEUR, F., HEINANEN, J., KRISHNAN, R., VAANANEN, P. e WU, L. **MPLS Support of Differentiated Services**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-mpls-diff-ext-08.txt. Fevereiro de 2001.

[CHIMENTO01] CHIMENTO, P. e TEITELBAUM, B. **QBone Bandwidth Broker Architecture**. Internet2 QoS Working Group. Trabalho em Desenvolvimento. Disponível em http://www.internet2.edu/qos/qbone/papers/sibbs/index.html. Capturado em maio de 2001.

[CLARK92] CLARK, D., SHENKER, S. e ZHANG, L. **Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism**. Em *Proceedings of SIGCOMM*. 1992.

[CLARK97] CLARK, D. e WROCLAWSKI, J. **An approach to service allocation in the Internet**. Trabalho em Desenvolvimento. IETF Working Draft draft-clark-diff-svc-alloc-00.txt. Julho de 1997.

[CRAWLEY00] CRAWLEY, E., SEAMAN, M., SMITH, A. e WROCLAWSKI, J. **Integrated Service Mappings on IEEE 802 Networks**. IETF Request for Comments (RFC) 2815. Maio de 2000.

[CRAWLEY98] CRAWLEY, E., NAIR, R., RAJAGOPALAN, H. e SANDICK, H. **A Framework for QoS-based Routing in the Internet**. IETF Request for Comments (RFC) 2386. Agosto de 1998.

[DANZIG95] DANZIG, P., JAMIN, S., SHENKER, S. e ZHANG, S. **A Measured-based Admission Control Algorithm for Integrated Services Packet Network**. Em *ACM SIGCOMM*. 1995.

[DROMS93] DROMS, R. **Dynamic Host Configuration Protocol**. IETF Request for Comments (RFC) 1541. Outubro de 1993.

[DURHAM00] DURHAM, D., BOYLE, R., COHEN, R., HERZOG, S., RAJAN, R. e SASTRY, A. **The COPS (Common Open Policy Service) Protocol**. IETF Request for Comments (RFC) 2748. Janeiro de 2000.

[GAINES98] GAINES, G. e FESTA, M. **A Survey of RSVP/QoS Implementations**. Update 2. RSVP Working Group. Julho de 1998. Disponível em http://ai.iit.nrc.ca/IETF/RSVP_survey/ietf_rsvp-qos_survey_02.txt. Capturado em maio de 2001.

[GAMA95] GAMMA E., HELM, R., JOHNSON, R. e VLISSIDES, J. **Design Patterns: Elements of Reusable Object-Oriented Software**. Addison-Wesley. ISBN 0201633612. 1a Edição. Janeiro de 1995.

[GHANWANI00] GHANWANI, A., PACE, W., SRINIVASAN, V., SMITH, A. e SEAMAN, M. **A Framework for Integrated Services Over Shared and Switched IEEE 802 LAN Technologies**. IETF Request for Comments (RFC) 2816. Maio de 2000.

[GOMES01] GOMES, A., COLCHER, S. e SOARES, L. **Modeling QoS Provision on Adaptable Communication Environments**. Em *IEEE International Conference on Communications (ICC 2001)*. Helsinki, Finlândia. Junho de 2001.

[GOMES99] GOMES, A. **Um Framework para Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação**. Dissertação de Mestrado. Departamento de Informática. PUC-Rio. Maio de 1999.

[GQoS01] Microsoft Corporation. **The Microsoft QoS Components**. White Paper. Disponível em http://www.microsoft.com/TechNet/win2000/qoscomp.asp. Capturado em maio de 2001.

[HEINANEN97] HEINANEN, J. **Use of the IPv4 TOS Octet to Support Differential Services**. Trabalho em Desenvolvimento. IETF Working Draft draft-heinanen-diff-tos-octet-00.txt. Outubro de 1997.

[IEEE98] Institute of Electrical and Eletronics Engineers / American National Standard Institute. **Information Technology - Telecommunications and Information Exchange Between Systems - Local and Metropolitan Area Networks -Common Specifications - Media Access Control (MAC) Bridges**. ANSI / IEEE Std 802.1D. 1998.

[ISSLL01] Internet Engineering Task Force. **Integrated Services over Specific Link Layers Working Group**. Disponível em http://ww.ietf.org/html.charters/issll-charter.html. Capturado em maio de 2001.

[ISO87] International Organization for Standardization / International Eletrotechnical Committee. **Information Processing Systems - Data Communications - Internal Organization of the Network Layer**. International Standard 8648. Maio de 1997.

[ITU99] International Telecommunication Union. **Recommendation I.150 (02/99) - B-ISDN Asynchronous Transfer Mode Functional Characteristics**. 1999.

[JACOBSON99a] JACOBSON, V., NICHOLS, K. e PODURI, K. **An Expedited Forwarding PHB**. IETF Request for Comments (RFC) 2598. Junho de 1999.

[JACOBSON99b] JACOBSON, V., NICHOLS, K. e ZHANG, L. **A Two-bit Differentiated Services Architecture for the Internet**. IETF Request for Comments (RFC) 2638. Julho de 1999.

[JAMOUSSI01] JAMOUSSI, B., ABOUL-MAGD, O., ANDERSSON, L., CALLON, R., DANTU, R., DOOLAN, P., FELDMAN, N., FREDETTE, A., GIRISH, M., GRAY, E., HALPERN, J., HEINANEN, J., HELLSTRAND, F., KILTY, T., MALIS, A., SUNDELL, K., VAANANEN, P., WORSTER, T. e WU, L. **Constraint-Based LSP Setup using LDP**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-mpls-cr-ldp-05.txt. Fevereiro de 2001.

[KILKKI97] KILKKI, K. **Simple Integrated Media Access (SIMA)**. Trabalho em Desenvolvimento. IETF Working Draft draft-kalevi-simple-media-access-01.txt. Junho de 1997.

[LI98] Li, T. e REKHTER, Y. **A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)**. IETF Request for Comments (RFC) 2430. Outubro de 1998.

[MALIS99] MALIS, A. e SIMPSON, W. **PPP over SONET/SDH**. IETF Request for Comments (RFC) 2615. Junho de 1999.

[MAMELI00] MAMELI, R. e SALSANO, S. **Integrated services over DiffServ network using COPS-ODRA**. Trabalho em Desenvolvimento. IETF Working Draft draft-mameli-issll-is-ds-cops-00.txt. Fevereiro de 2000.

[MOTA01] MOTA, O. **IPQoS: Uma Interface em Java para Solicitação de Serviços com QoS**. Projeto Final de Programação. Pontifícia Universidade Católica do Rio de Janeiro. Janeiro de 2001.

[MPLS01] Internet Engineering Task Force. **MultiProtocol Label Switching Working Group**. Disponível em http://www.ietf.org/html.charters/mpls-charter.html. Capturado em maio de 2001.

[PAREKH92] PAREKH, A. **A Generalized Processor Sharing Approach to Flow Control in Integrated Services Network**. Technical Report LIDS-TR-2089. Laboratory for Information and Decision Systems. MIT. 1992.

[PCRSVP01] Intel Corporation. **Intel PC-RSVP (ReSerVation Protocol)**. Versão Beta 1.0 (Build 61). Disponível em http://developer.intel.com/ial/rsvp/index.htm. Capturado em maio de 2001.

[PREE95] PREE, W. **Design Patterns for Object-Oriented Software Development**. Addison Wesley. 1995.

[QBONE01] Internet2 QoS Working Group. Disponível em http://qbone.internet2.edu. Capturado em maio de 2001.

[REKHTER01] REKHTER, Y. e ROSEN, E. **Carrying Label Information in BGP-4**. Trabalho em Desenvolvimento. IETF Working Draft draft-ietf-mpls-bgp4-mpls-05.txt. Janeiro de 2001.

[SHENKER97] SHENKER, S., GUERIN, R. e PARTRIDGE, C. **Specification of Guaranteed Quality of Service**. IETF Request for Comments (RFC) 2212. Setembro de 1997.

[SOLSTICE01] Sun Microsystems. **Solstice Bandwidth Reservation Protocol API**. Versão 1.0. Disponível em http://www.sun.com/software/bandwidth/rsvp/docs/index.htm. Capturado em maio de 2001.

[TERZIS99] TERZIS, A., OGAWA, J., ZHANG, L. e WANG, L. **A Two-tier Resource Management Model for the Internet**. Em *Global Internet*. Dezembro de 1999.

[UML97] Rational Software Corporation. **UML Notation Guide**. Setembro de 1997.

[WROCLAWSKI97] WROCLAWSKI, J. **Specification of the Controlled-Load Network Element Service**. IETF Request for Comments (RFC) 2211. Setembro de 1997.

Lista de Acrônimos

AF	Assured Forwarding
API	Application Program Interface
ATM	Assynchronous Transfer Mode
BA	Behavior Aggregate
BB	Bandwidth Broker
BGP	Border Gateway Protocol
CBR	Constant Bit Rate
CDR	Commited Data Rate
COPS	Common Open Policy Service
COS	Class Of Service
CR-LDP	Constraint-based Routed LDP
DHCP	Dynamic Host Configuration Protocol
DLCI	Data Link Connection Identifier
DS	Differentiated Services
EF	Expedited Forwarding
E-LSP	EXP inferred LSP
ER	Edge Router
FEC	Forward Equivalence Class
FIFO	First In First Out
GQoS	Generic QoS
GWSID	Global Well-known Services ID
IETF	Internet Engineering Task Force
ISO	International Organization for Standardization
ISP	Internet Service Provider

ISSLL	Integrated Services over Specific Link Layers
ITU	Internation Telecommunication Union
LAN	Local Area Network
LDP	Label Distribution Protocol
L-LSP	Label-only inferred LSP
LSP	Label Switched Path
LSR	Label Switching Router
MF	Multi-Field
MPLS	MultiProtocol Label Switching
OSI	Open Systems Interconnection
PCR	Peak Cell Rate
PDR	Peak Data Rate
PDU	Protocol Data Unit
PHB	Per Hop Behavior
PNNI	Private Network to Network Interface
PPP	Point-to-Point Protocol
QoS	Quality of Service
RAA	Resource Allocation Answer
RAR	Resource Allocation Request
RFC	Request For Comments
RSVP	Resource reSerVation Protocol
RSVP-TE	RSVP - Traffic Engineering
SBM	Subnet Bandwidth Manager
SIBBS	Simple Interdomain Bandwidth Broker Signalling
SLA	Service Level Aggrement
SLS	Service Level Specification
SONET	Synchronous Optical NETwork
SPO	Service Parametrization Objects
TC	Traffic Control
TCS	Traffic Conditioning Specification
TR	Transient Router
UML	Unified Modeling Language

VC	Virtual Circuit
VCC	Virtual Channel Connection
VCI	Virtual Circuit Identifier
VLAN	Virtual LAN
VLL	Virtual Leased Line
VPI	Virtual Path Identifier
VPN	Virtual Private Networks
WFQ	Weighted Fair Queueing