Uma Arquitetura Adaptável para Provisão de QoS na Internet

Oscar Thyago J. D. D. L. Mota, Antônio Tadeu A. Gomes, Sérgio Colcher, Luiz Fernando G. Soares
{thyago, atagomes, colcher, lfgs}@inf.puc-rio.br

Laboratório TeleMídia -- Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de São Vicente, 225
Rio de Janeiro - RJ
22453-900, Brasil
http://www.telemidia.puc-rio.br

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 um framework genérico, 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

1. Introdução

A grande popularidade e disseminação que a Internet vem atingindo nos últimos anos tem despertado o interesse de diversos segmentos ligados à pesquisa e desenvolvimento, especialmente no que diz respeito à adequação de um protocolo como o IP a uma série de aplicações como vídeo sob demanda, conferências multimídia, TV interativa, telemedicina, etc. Essas aplicações caracterizam-se principalmente pela utilização de tipos variados de mídia, que impõem requisitos distintos de *qualidade de serviço* (QoS) ao sistema de comunicação.

Desde sua concepção, a Internet oferecia apenas o serviço de melhor-esforço, caracterizado pela ausência de garantias de QoS às aplicações. A inadequação do serviço de melhor-esforço aos requisitos das aplicações supracitadas motivou, nos últimos anos, o desenvolvimento de várias propostas para um novo modelo de serviços que agregasse o conceito de QoS na Internet, permitindo às aplicações requisitar serviços com diferentes características de qualidade. Dentre essas propostas, destacam-se os modelos de *serviços integrados* (*intserv*) [Brad94] e *diferenciados* (*diffserv*) [Blak98].

O uso combinado dos modelos de serviços *intserv* e *diffserv* e das diversas abordagens de provisão de QoS no 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, com a identificação dos conceitos e técnicas comuns empregados, ainda não foi devidamente abordado. Diante da quantidade de opções existentes, torna-se difícil para o projetista de um provedor de serviços entender e decidir que tipo de modelo e tecnologia de sub-rede deve ser utilizado para se realizar um determinado serviço.

É importante ressaltar, porém, que o surgimento dos modelos e tecnologias citados acima não implica necessariamente que a solução proposta permaneça adequada ao suporte das novas aplicações e serviços com diferentes requisitos de QoS que provavelmente surgirão no futuro. Tanto as aplicações quanto as tecnologias de comunicação de dados encontram-se em um estágio de contínua evolução, e a tendência atual é que estrutura e funcionalidade dos subsistemas que compõem a Internet (estações, roteadores e sub-redes) venham a se tornar versáteis 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 ou ainda reinstalação completa de software.

Uma alternativa a esses procedimentos está no desenvolvimento de plataformas de comunicação que dêem suporte a algum tipo de *programabilidade* ou *adaptabilidade*, permitindo ao sistema de comunicação adequar-se, em tempo de operação, tanto às modificações nos requisitos das aplicações quanto à contínua evolução tecnológica. Vários trabalhos seguem essa abordagem [Blair98, Gome01], porém poucos tratam a questão de como regular as adaptações necessárias às funções de provisão de QoS. Objetivando preencher essa lacuna, em [Gome99] é apresentado um conjunto de frameworks que procuram generalizar a arquitetura interna dos vários subsistemas que compõem um ambiente qualquer de processamento e comunicação. Esses frameworks ilustram, especificamente, como as funções de provisão de QoS são recorrentes nesses subsistemas e como elas participam na *orquestração* dos recursos do ambiente como um todo. De forma mais específica, um *framework* [Pree95] difere de uma aplicação exatamente pelo fato de que algumas de suas partes (denominadas *pontos de flexibilização* ou *hot-spots*) são explícita e propositadamente deixadas incompletas ou marcadas como “sujeitas a variações”. Através do preenchimento dessas “lacunas” ou de particularizações nos pontos marcados, pode-se obter a adaptabilidade do serviço. A representação das regras de estruturação dessas funções sob a forma de frameworks facilita a identificação dos pontos de flexibilização onde essas funções podem ser adaptadas a situações específicas.

Para alcançar o nível de generalidade desejado, os frameworks apresentados em [Gome99] foram concebidos de forma a apenas documentar *esboços* das funções de provisão de QoS. Para poderem ser aplicados no contexto de um ambiente específico, alguns dos pontos de flexibilização devem ser implementados com informações relativas ao ambiente. O presente trabalho mostra como os pontos de flexibilização podem ser implementados no contexto dos modelos de serviços com QoS na Internet. Como resultado, 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.

É importante observar que a arquitetura geral deixa, propositalmente, vários outros pontos de flexibilização dos frameworks descritos em [Gome99] como “em aberto”. Alguns desses pontos estão relacionados à integração dos modelos de serviços com as tecnologias de sub-redes. Outros estão associados às partes das 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. Permite também, através do preenchimento de outros pontos de flexibilização, adequar a arquitetura a diferentes combinações dos modelos de serviços *intserv* e *diffserv* e das diversas abordagens de provisão de QoS no nível das sub-redes. Todos esses pontos de flexibilização são de fundamental importância para a arquitetura proposta neste trabalho, uma vez que são justamente esses pontos que permitem que a arquitetura, não importando o modelo adotado, possa ser adaptável à introdução de novos serviços.

Ainda como contribuição marginal deste trabalho, a independência da arquitetura proposta dos modelos de serviços e sub-redes existentes tem como consequência um melhor entendimento em relação ao modelo que mais se adeque a um determinado tipo de serviço que o provedor se propõe a fornecer.

O restante do artigo encontra-se estruturado da seguinte forma. A Seção 2 introduz, por meio de um modelo de serviço geral, as principais funções de provisão de QoS e a forma como os modelos *intserv* e *diffserv* particularizam essas funções a seus propósitos específicos. Essas funções de QoS, particularizadas aos modelos, se tornarão pontos de flexibilização no framework proposto. O uso combinado dos modelos de serviços *intserv* e *diffserv* e das diversas abordagens de provisão de QoS no nível das sub-redes é discutido na Seção 3, também visando a definição de pontos de flexibilização no framework proposto. A Seção 4 apresenta a arquitetura proposta, enfatizando os pontos em que ela é adaptável. Um estudo de caso é apresentado na Seção 5, através de um cenário baseado na integração de provedores *intserv* e *diffserv*. A Seção 6 é reservada às conclusões.

2. Modelos de Serviços com QoS na Internet

A provisão de QoS, de maneira geral, implica em uma série de funções que devem ser acrescentadas ao cliente e ao provedor de serviços. As funções de provisão de QoS atuam em diferentes fases do ciclo de vida de um serviço. Em [Gome01] é feito um estudo completo acerca do mapeamento das funções de provisão de QoS nessas fases. Neste trabalho, serão focadas somente três fases em particular: (i) *solicitação de serviços*, (ii) *estabelecimento de contratos de serviço* e (iii) *manutenção de contratos de serviço*. Nesta seção, as funções de QoS, presentes nessas fases, candidatas a ponto de flexibilização serão apresentadas em itálico.

No que se refere à fase de solicitação de serviços, a aplicação informa ao provedor, normalmente por intermédio de uma interface própria, a *caracterização do tráfego* correspondente aos fluxos[1] de dados a serem gerados e a *especificação da QoS* desejada. O suporte a essas especificações será provido diretamente através de especializações do framework para parametrização de serviços descrito na Seção 4, que permitirão a criação dos parâmetros específicos utilizados pelo *intserv* e pelo *diffserv* na caracterização do tráfego e na especificação da QoS.

Segue-se à fase de solicitação do serviço o estabelecimento de um contrato, ou SLA (*Service Level Agreement*), entre a aplicação e o provedor. Por esse contrato, a aplicação compromete-se a gerar fluxos de dados conforme o especificado. O provedor, por sua vez, assegura o transporte dos referidos fluxos com a QoS solicitada, desde que o tráfego introduzido esteja conforme o prometido. Durante o estabelecimento do SLA, uma série de funções (descritas nos parágrafos a seguir) deverão ser executadas, que envolverão a escolha

[1] Neste contexto, um fluxo representa um conjunto de pacotes transmitidos por uma aplicação origem em particular para uma ou mais aplicações destino.

de estratégias específicas em cima dos pontos de flexibilização definidos pelos frameworks de compartilhamento e orquestração de recursos descritos na Seção 4.

O estabelecimento de um SLA envolve, inicialmente, a determinação dos recursos necessários ao fornecimento do serviço em cada um dos subsistemas que compõem um provedor. Para isso, a função de *roteamento* identifica os subsistemas que efetivamente participarão da provisão do serviço e a função de *negociação* quantifica os recursos necessários em cada um dos subsistemas escolhidos. Tais funções dependem do *mapeamento* dos parâmetros, utilizados pela aplicação durante a solicitação do serviço, em valores que permitam quantificar os recursos que devem ser alocados nesses subsistemas. É necessário também determinar se os subsistemas possuem recursos suficientes para atender à solicitação, o que é tarefa do *controle de admissão*. Havendo recursos suficientes, os módulos de encaminhamento presentes nas estações e roteadores, compreendendo as funções de *classificação* e *escalonamento* de pacotes, devem ser apropriadamente configurados. Para que a orquestração de recursos seja efetiva, os mecanismos específicos de alocação de recursos das sub-redes envolvidas também devem ser acionados.

Caso os recursos do provedor sejam alocados por meio de uma configuração manual do administrador, diz-se que o SLA é estático. Caso esses recursos tenham sido alocados por meio de um protocolo de sinalização específico, diz-se que o SLA é dinâmico. Nesta 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*) [Jaco99], é responsável pelas decisões de alocação de recursos no provedor. O BB tem total conhecimento dos recursos disponíveis, bem como das modalidades de serviços suportadas pelo provedor. A partir destas informações, o BB pode realizar a alocação de recursos que atenda aos serviços solicitados pelas aplicações. Na *abordagem distribuída*, cada estação e roteador presente no provedor é dotado de um negociador próprio, responsável pelas decisões locais de alocação de recursos. Esses negociadores trocam informações de controle entre si, por meio de um *protocolo de sinalização*, de forma a determinar a melhor alocação de recursos possível no contexto do provedor, que seja suficiente para o atendimento do serviço solicitado.

Após o estabelecimento do SLA, o provedor de serviços deve empregar funções que mantenham a QoS requisitada pelo usuário. Uma dessas funções é a de *policiamento* dos fluxos, que determina se o tráfego entrante no provedor corresponde ao caracterizado pelos usuários na fase de solicitação de serviço. O provedor pode utilizar-se das mais variadas ações de policiamento visando penalizar aqueles fluxos considerados não-conformes, como descartar pacotes ou encerrar abruptamente o SLA, por exemplo. Em paralelo, o provedor deve se monitorar, de forma a ser capaz de responder automaticamente a degradações ocasionais na QoS provida pelos subsistemas envolvidos no fornecimento dos serviços. Funções de *ressintonização* são acionadas nos casos em que a qualidade fornecida diverge daquela contratada pelos usuários, quando então uma nova orquestração de recursos se torna necessária. O provedor pode valer-se de inúmeras ações de ressintonização, como, por exemplo, escolher subsistemas alternativos (por meio da função de roteamento) ou renegociar o SLA com a aplicação para tentar manter o SLA estabelecido.

Vários dos pontos de flexibilização citados nesta seção podem ser particularizados para os casos dos modelos *intserv* e *diffserv*. As próximas subseções discutem alguns desses pontos para os dois modelos, cujo preenchimento é mais detalhado nos exemplos da Seção 4.

2.1. Modelo de Serviços Integrados

A especificação do modelo *intserv* [Brad94] descreve as principais funções de provisão de QoS que devem estar presentes em todas as estações e roteadores, em especial o controle de admissão, a alocação de recursos e a classificação e escalonamento de pacotes. O modelo *intserv* determina que as aplicações devem solicitar serviços de comunicação com qualidade a partir de categorias padronizadas. O que vai diferenciar uma categoria de serviço de outra será o grau de comprometimento do provedor com a QoS solicitada, ou seja, o nível de garantia do serviço fornecido. Atualmente, no modelo *intserv*, encontram-se especificadas duas categorias de serviços: a *garantida* [Shen97] e a de *carga controlada* [Wroc97].

No modelo *intserv*, solicitações devem ser explicitamente efetuadas, normalmente por intermédio de uma interface de solicitação de serviços específica. No que se refere à provisão de serviços, o modelo *intserv* não faz restrição quanto à modalidade a ser implementada, porém, a abordagem dinâmica e distribuída, utilizando o protocolo RSVP [Brad97], tem sido a normalmente adotada. Caso esse protocolo seja empregado, o estabelecimento de SLAs ocorre em duas etapas: a primeira correspondente ao anúncio das características de tráfego dos fluxos gerados pelos transmissores; a segunda corresponde à efetiva solicitação de um serviço pelos receptores do tráfego anunciado, por meio da especificação do nível de garantia e da QoS desejada. Outra característica do protocolo RSVP, particularmente útil na fase de manutenção dos SLAs, é a presença de atualizações constantes das solicitações de alocação de recursos por meio da troca periódica de mensagens de sinalização. Essa característica fornece ao provedor de serviços *intserv* um mecanismo que permite a ressintonização ou renegociação automática na presença de degradações na qualidade requisitada pelas aplicações. A seleção desse conjunto de mecanismos e a sua forma de interação de acordo com a descrição acima, corresponde a especializações dos frameworks apresentados na Seção 4, em especial do framework de orquestração de recursos (descrito na Seção 4.3).

2.2. Modelo de Serviços Diferenciados

O modelo *intserv* apresenta, como uma de suas principais características, a alta granularidade na alocação de recursos, que é realizada para fluxos individuais. A provisão de serviços dinâmica, aliada a essa alocação de recursos por fluxos, contribui para uma carga de sinalização e processamento que pode ser muito alta, dificultando a aplicação do modelo *intserv* em provedores de backbone. Para contornar o problema da falta de escalabilidade do *intserv*, o modelo *diffserv* [Blak98] propõe a agregação de fluxos em classes de serviço, permitindo a redução no número de estados a ser mantido nos roteadores de um provedor.

A divisão das funções de provisão de QoS entre tipos distintos de roteadores é uma outra característica particular do modelo *diffserv*. Cabe aos roteadores de borda, ou ERs (*Edge Routers*), as tarefas mais complexas de classificação e policiamento dos fluxos entrantes no provedor de serviços *diffserv*. Nos roteadores centrais, chamados de TRs (*Transient Routers*), o processamento é simplificado devido à ausência de policiamento e à redução da complexidade da função de classificação, que passa a identificar as classes de serviço pela leitura direta do campo DS (*Differentiated Services*) do cabeçalho IP.

Comparativamente ao modelo *intserv*, os SLAs *diffserv* são mais abrangentes e, normalmente, mais estáveis, apresentando um tempo de vida maior. No modelo *diffserv* o SLA envolve não somente os aspectos técnicos da caracterização de um serviço, mas também outros aspectos, como a forma de tarifação, por exemplo. Assim como no modelo *intserv*, o estabelecimento do SLA pode ser estático ou dinâmico. No estado atual dos projetos de implementação do modelo *diffserv*, os SLAs são em geral estáticos, sendo ainda objeto de várias propostas a definição de protocolos, em sua grande maioria adotando a abordagem

centralizada (como COPS [Boyl00] e Diameter [Calh00], por exemplo), que permitam o estabelecimento de SLAs genéricos e extensíveis.

Ao contrário do intserv, 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 de serviço. A combinação de PHBs com os demais parâmetros de caracterização do SLA é determinada durante a implementação do provedor *diffserv*. Devido à ausência de padronização de serviços, o modelo *diffserv* permite que diferentes provedores concorram entre si no fornecimento de serviços que agreguem vantagens aos usuários, além de possibilitar uma maior adaptabilidade ao provedor no que se refere à introdução de novos serviços.

3. Integração entre Modelos de Serviços e Tecnologias de Sub-redes

Do ponto de vista da aplicação, os serviços solicitados são fornecidos exclusivamente pelo provedor de acesso local, ou seja, aquele que a atende diretamente. Porém, em um serviço verdadeiramente fim-a-fim, o que normalmente ocorre é a concatenação de SLAs entre os vários provedores de serviço envolvidos no transporte dos dados, entre a origem e o destino, podendo cada provedor adotar diferentes modelos de serviço. Sob essa perspectiva, uma configuração bastante discutida sugere o uso do modelo *intserv* nos provedores de borda, próximos aos usuários, e o *diffserv* nos provedores de backbone, sendo, normalmente, referenciada por *intserv* sobre *diffserv* [Bern00]. Em tais configurações, especial atenção deve ser dispensada aos pontos de interoperação de domínios diferentes, pois é essa interoperação que determinará a correta orquestração de recursos interdomínios. Nesses pontos é normalmente crítica a função de mapeamento de parâmetros correspondentes a serviços compatíveis.

Outra questão em que a orquestração de recursos, e a função de mapeamento em particular, é primordial é a da integração dos modelos de serviços com as tecnologias de sub-redes. Como exemplo, parâmetros de caracterização de um serviço garantido no modelo *intserv* devem ser, de algum modo, mapeados corretamente em parâmetros de caracterização de um circuito virtual ATM.

A variedade de combinações possíveis entre modelos de serviços e tecnologias de sub-redes sugere a definição de uma arquitetura única, capaz de englobar os conceitos e princípios de provisão de QoS já enumerados na Seção 2. Apesar dos modelos *intserv* e *diffserv* terem enfoques e objetivos próprios, pode-se afirmar que grande parte das funções de provisão de QoS presentes em ambos os modelos apresenta similaridades que justificam a definição dessa arquitetura. O presente trabalho vale-se dos frameworks apresentados em [Gome99] para definir essa arquitetura de maneira tal que as particularidades de cada modelo reduzem-se a conjuntos de algoritmos e estruturas de dados que podem ser implementados diretamente segundo as regras de estruturação definidas pelos frameworks. *Graças à recorrência dessas regras de estruturação em diferentes subsistemas, o framework permite também que a orquestração de recursos seja representada de forma recursiva, ou seja, que a combinação entre diferentes modelos de serviços e tecnologias de sub-redes seja modelada de maneira uniforme, o que facilita a adequação da arquitetura às características específicas de cada uma das combinações possíveis.* Este é um dos pontos mais importantes da arquitetura proposta.

4. Descrição da Arquitetura

A descrição da arquitetura proposta neste trabalho segue o paradigma orientado a objetos e é dividida em três partes principais, que refletem a própria divisão adotada pelos frameworks para provisão de QoS apresentados em [Gome99]: (i) parametrização de serviços, (ii) compartilhamento de recursos e (iii) orquestração de recursos.

A notação UML (*Unified Modeling Language*) [UML97] é utilizada na representação dos diagramas de classes que descrevem a arquitetura. Foi adotada uma notação diferenciada entre as classes-base (hachuradas) e as que representam possíveis pontos de flexibilização (brancas). Durante a descrição textual dessas partes, foi utilizada também uma grafia diferenciada para denominar classes abstratas (em *itálico*) e concretas.

4.1. Parametrização de Serviços

Tanto a caracterização da carga imposta pelos fluxos das aplicações, quanto a especificação da QoS desejada pelas mesmas e as informações de desempenho do provedor são descritas por meios de parâmetros, tais como a taxa média de geração de dados, o retardo máximo requerido e a banda passante disponível, respectivamente. O propósito dos frameworks para parametrização é fornecer um esquema que permita a estruturação desses parâmetros de forma independente dos serviços que possam vir a ser oferecidos por um determinado provedor. Tendo em vista este objetivo, foi desenvolvido o conceito de parâmetro abstrato, que pode ser, posteriormente, especializado em um ou mais parâmetros concretos em uma implementação real de um provedor, sendo constituída, desta forma, uma hierarquia de derivação de parâmetros.

A escolha dos parâmetros que podem ser utilizados em um determinado serviço define um ponto de flexibilização, cuja implementação é de responsabilidade do provedor. Desta forma, cabe ao provedor informar que parâmetros de QoS (retardo máximo, taxa de perdas, etc.) é capaz de suportar e como a carga deve ser caracterizada (se pela taxa de pico, se pela taxa média, etc.). Estes parâmetros, juntamente com o nível de garantia da QoS (100% garantida, carga controlada, etc.) e o estilo de compartilhamento constituem-se nas categorias de serviço disponibilizadas pelo provedor. O conceito de estilo de compartilhamento permite que um mesmo serviço seja utilizado por fluxos distintos que tenham em comum o mesmo destinatário[2].

As categorias de serviço, a exemplo dos parâmetros, também podem ser estruturadas em uma hierarquia de derivação, definindo um outro ponto de flexibilização que permite a implementação de novos serviços a partir de outros, facilitando o reuso. Cabe aos usuários informar a categoria desejada, dentre aquelas disponibilizadas pelo provedor, e os valores dos parâmetros de caracterização de tráfego e especificação da QoS correspondentes.

A Figura 1 apresenta um exemplo de aplicação dos frameworks para parametrização de serviços de um provedor qualquer na Internet. Na hierarquia de derivação de categorias de serviço, a classe *ServiceCategory* foi especializada nas classes *IntservServiceCategory* e *DiffservServiceCategory*. As categorias de serviço definidas atualmente no modelo *intserv* são representadas, no exemplo da Figura 1, através das classes GuarServiceCategory (serviço garantido) e CLServiceCategory (serviço de carga controlada). Também é ilustrado no exemplo, uma possível instanciação da categoria de serviço *diffserv* através dos serviços representados pelas classes GoldServiceCategory, SilverServiceCategory e BronzeServiceCategory. Finalmente, as estruturas *tspec* e *rspec*, utilizadas no modelo *intserv*

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

para a caracterização de tráfego e a especificação da QoS desejada, respectivamente, são representadas na Figura 1 por meio de parâmetros correspondentes. Caberá ao provedor definir os parâmetros relativos aos serviços *diffserv* oferecidos, devendo inseri-los, de forma apropriada, na árvore de derivação de parâmetros.

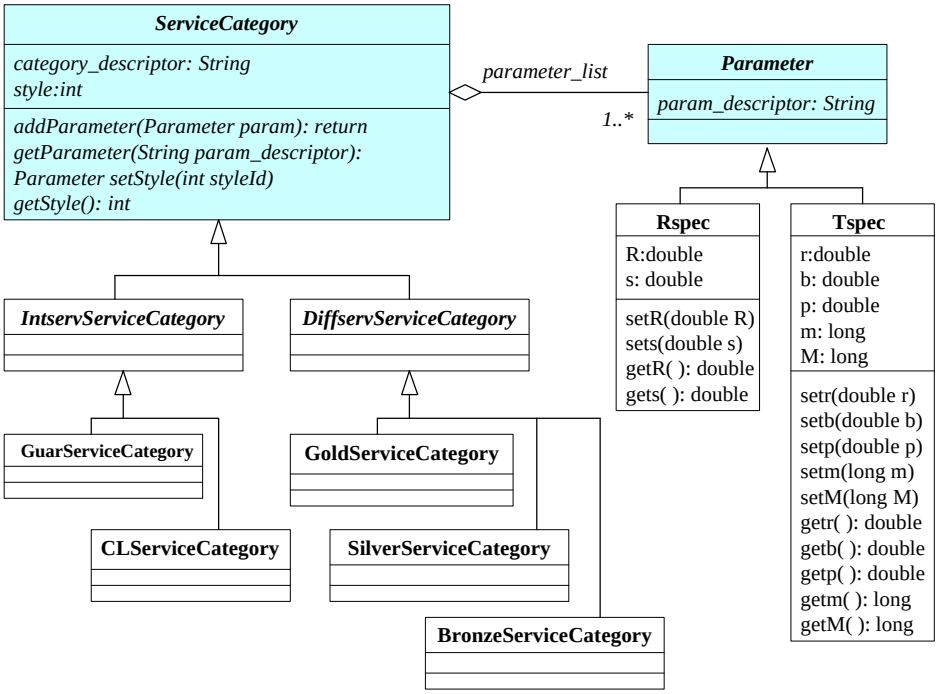

Figura 1: exemplo de instanciação do framework para parametrização de serviços.

4.2. Compartilhamento de Recursos

Várias funções de QoS, apresentadas na Seção 2, atuam sobre estruturas de dados que definem o estado interno do provedor. Essas estruturas, entre outras coisas, definem como os diversos recursos de um provedor estão sendo utilizados, isto é, como cada parcela do recurso está alocada e como é escalonada a sua utilização. Os frameworks para compartilhamento de recursos são definidos de modo a capturar essas informações. Para tanto, os frameworks usam o conceito de *recurso virtual*, que representa uma parcela de utilização de um recurso real (processador, memória, largura de banda) associada a um ou mais fluxos de dados. Esses recursos virtuais constituem-se nas folhas de uma árvore, chamada de *árvore de recursos virtuais*, sendo sua raiz representada por um escalonador de recurso real. Cabe a esse escalonador gerenciar o compartilhamento de um ou mais recursos reais entre os recursos virtuais correspondentes. Os nós internos da referida árvore também são constituídos de escalonadores, que gerenciam a sua parcela de utilização do recurso real, distribuída entre seus recursos virtuais filhos. Os frameworks para compartilhamento de recursos abrangem as fases de estabelecimento e manutenção de um SLA, sendo divididos, de acordo com a fase em que atuam, nos frameworks para alocação e escalonamento de recursos, respectivamente.

Cada escalonador presente na árvore de recursos virtuais tem associado um componente de criação de um determinado tipo de recurso virtual. Esse componente permite também a adição do recurso virtual criado à lista de recursos virtuais que o escalonador associado possui. Após essa adição, o componente de criação configura os módulos de classificação e de policiamento de fluxos de forma a ser possível a associação dos fluxos aos recursos virtuais criados e o monitoramento dos referidos fluxos, respectivamente. A Figura 2 ilustra a estruturação, em um provedor *intserv* hipotético, da árvore de recursos virtuais de um roteador. Conforme é mostrado na figura, cada escalonador de recurso virtual tem associado

uma categoria de serviço e uma parcela de utilização do recurso real que, no exemplo, corresponde à largura de banda.

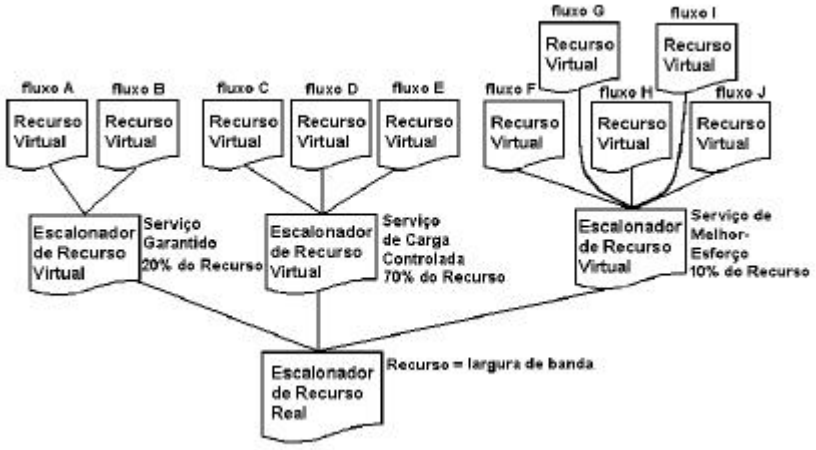

Figura 2: árvore de recursos virtuais em um provedor *intserv*.

A Figura 3 ilustra uma instanciação dos frameworks para compartilhamento de recursos capaz de modelar a árvore de recursos virtuais do provedor *intserv* exemplificado. Um criador de recurso virtual especial, representado pela classe *VirtualLinkCreator*, é ilustrado juntamente com os módulos de classificação e de policiamento, representados pelas classes *Classifier* e *PoliceAgent*, respectivamente. Os escalonadores em geral, representados pela classe *PacketScheduler*, estão associados a estratégias de escalonamento responsáveis pela divisão do percentual de uso da largura de banda disponível entre os recursos virtuais (classe *Link*). Essas estratégias são modeladas pela classe *SchedulingStrategy*, e constituem-se em pontos de flexibilização do framework de escalonamento.

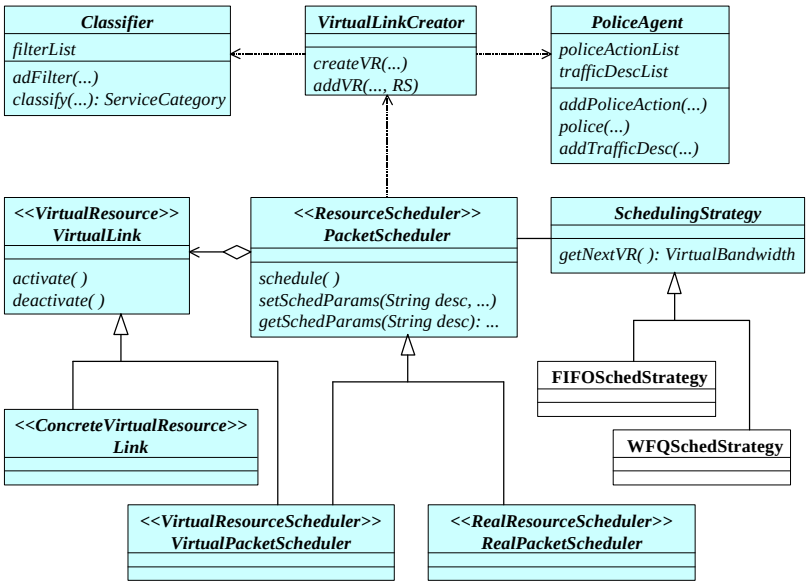

Figura 3: exemplo de instanciação dos frameworks para compartilhamento de recursos.

No exemplo do provedor *intserv*, o escalonador real de pacotes (classe *RealPacketScheduler*), está associado a três escalonadores de recurso virtual (classe *VirtualPacketScheduler*), um para cada uma das categorias de serviço padronizadas pelo modelo *intserv*. No exemplo, foi utilizada como estratégia de escalonamento o algoritmo WFQ (*Weighted Fair Queueing*) [Pare92] na divisão da banda entre as referidas categorias de serviço. Cada instância de *VirtualPacketScheduler*, por sua vez, pode adotar diferentes

estratégias de escalonamento na divisão do seu percentual de uso do recurso real. Para a categoria de serviço garantido, o algoritmo WFQ pode ser novamente utilizado na divisão do seu percentual de uso da banda entre os fluxos atendidos. Os demais escalonadores virtuais, pela própria natureza do serviço que fornecem, podem adotar estratégias mais simples, como a FIFO por exemplo.

Em geral, as especificações dos modelos de serviços com QoS na Internet não detalham as funções de escalonamento de pacotes, dando somente algumas sugestões, mas deixando a cargo do projetista a escolha da melhor forma de implementação. A divisão do uso de um recurso real entre os recursos virtuais, incluindo os percentuais e os algoritmos empregados, faz parte das políticas de provisão de QoS configuradas pelo administrador do provedor de serviços.

4.3. Orquestração de Recursos

Os frameworks para orquestração de recursos procuram modelar os mecanismos envolvidos no estabelecimento e manutenção dos SLAs como o mapeamento, o controle de admissão e a reserva de recursos. O estabelecimento de um SLA inicia-se quando um usuário[3] solicita um serviço a um provedor qualquer. O serviço requisitado é, normalmente, encaminhado a um agente de negociação de QoS presente no provedor. Conforme a Seção 2, há duas modalidades de provisão de serviços: a centralizada e a distribuída. Na provisão centralizada, o agente de negociação que recebe a solicitação de serviços simplesmente repassa o pedido a um agente central, efetivamente responsável pelas decisões relacionadas à alocação de recursos e configuração das funções de classificação e escalonamento presentes em todos os recursos do provedor de serviços. Na provisão distribuída, os agentes de negociação cooperam entre si na escolha da melhor maneira de se prover a QoS requisitada.

Nos frameworks de orquestração de recursos, os agentes de negociação são modelados pela classe abstrata *QoSNegotiator*, que disponibiliza o método *request* de forma a permitir que as características do serviço desejado possam ser informadas pela aplicação. Dependendo do modelo de serviço, da abordagem de provisão (centralizada ou distribuída) e do protocolo sinalização empregado pelo provedor, diferentes negociadores de QoS podem ser definidos por meio de especializações de *QoSNegotiator*. Essa classe abstrata constitui-se em um importante ponto de flexibilização dos frameworks de orquestração. A Figura 4 ilustra uma possível implementação desses frameworks de forma a permitir a modelagem de negociadores de QoS baseados nos modelos *intserv*, com provisão distribuída e empregando o protocolo RSVP (classe *RSVPIntservQoSNeg*), e *diffserv*, utilizando o protocolo COPS em uma negociação centralizada (classes *COPSClientDiffservQoSNeg* e *COPSServerDiffservQoSNeg*). Essa instanciação dos frameworks será empregada diretamente no cenário de uso discutido na Seção 5 deste trabalho.

Após o recebimento de uma solicitação de serviço, o agente de negociação de QoS deve, inicialmente, verificar se as políticas configuradas no provedor permitem que o referido serviço possa ser fornecido. Estas políticas podem ser as mais variadas, indo desde regras relacionadas a direitos de acesso a determinados serviços até restrições ao uso de recursos considerados especiais (um enlace intercontinental, por exemplo). O agente que mantém essas políticas em um provedor de serviços é representado pela classe *PolicyAgent*, que disponibiliza métodos para adição (*addPolicyStatement*) e verificação (*verifyPolicy*) de políticas, constituindo-se também em pontos de flexibilização dos frameworks.

[3] É importante enfatizar que, no contexto da Internet, é comum que este usuário seja representado, inclusive, por um outro provedor de serviços, o que constitui o estabelecimento de um SLA interdomínios.

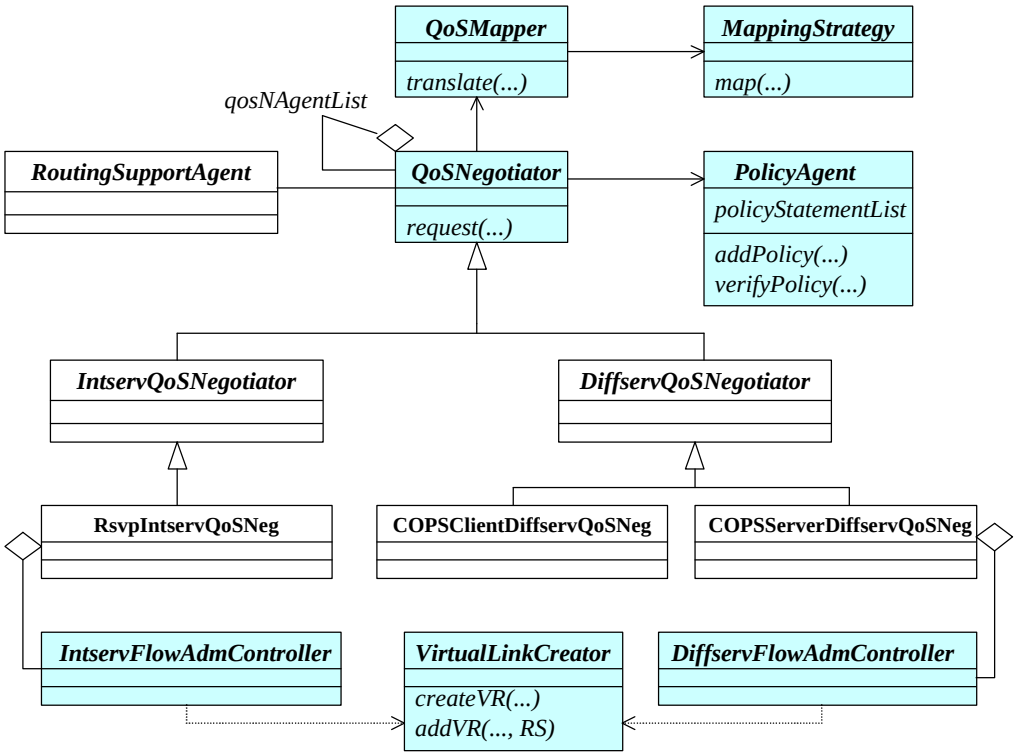

Figura 4: exemplo de instanciação dos frameworks para orquestração de recursos.

Na determinação do caminho por onde os fluxos de dados relacionados ao serviço solicitado serão encaminhados, o agente de negociação depende do suporte oferecido pelas funções de roteamento presentes no provedor, modeladas pela classe *RoutingSupportAgent*[4]. A verificação da disponibilidade de recursos no provedor é realizada pela consulta a agentes de controle de admissão. Se a provisão for centralizada, há um único agente de controle de admissão que se encontra junto ao agente de negociação central. No exemplo da Figura 4, o agente de controle de admissão para provedores *diffserv* é representado pela classe *DiffservFlowAdmController*. Nesse caso, o agente de controle de admissão é responsável por contactar os criadores de recursos virtuais (instâncias da classe *VirtualLinkCreator*, apresentada na Seção 4.2) presentes em todas estações e roteadores envolvidos com o fornecimento do serviço no provedor *diffserv*. Se a provisão for distribuída, tem-se um agente de controle de admissão local associado a cada agente de negociação presente no provedor. Na Figura 4, os agentes de controle de admissão locais para provedores *intserv* são instâncias da classe *IntservFlowAdmController*. Em cada uma das estações e roteadores presentes no provedor *intserv* encontra-se uma dessas instâncias, que se responsabilizam por contactar os criadores de recursos virtuais locais.

Independente da modalidade de provisão adotada, todo o processo de negociação e controle de admissão depende da função de mapeamento de parâmetros. Porém, no contexto dos modelos de serviços com QoS na Internet, a função de mapeamento é importante não só durante a determinação de recursos necessários. Muitas vezes a negociação interdomínios envolve também o mapeamento entre parâmetros de serviço, uma vez que os serviços oferecidos por diferentes provedores podem não ser exatamente os mesmos. É o que ocorre, por exemplo, na configuração *intserv* sobre *diffserv*, citada na Seção 3 deste trabalho. Na arquitetura proposta, o mapeamento de parâmetros é realizado pela classe *QoSMapper*. Instâncias de *QoSMapper* devem implementar traduções de parâmetros específicas para cada

[4] As funções de roteamento são tratadas com detalhes em [Rodr99].

categoria de serviço, e a classe *MappingStrategy* constitui-se no ponto de flexibilização necessário ao mapeamento.

5. Cenários de Utilização do Framework

Objetivando uma melhor compreensão de como os frameworks descritos na Seção 4, com especial destaque ao preenchimento dos pontos de flexibilização identificados, podem ser empregados na modelagem de uma real implementação de provisão de serviços com QoS na Internet, será considerado um cenário baseado na configuração *intserv* sobre *diffserv*, esquematizado na Figura 5. No cenário utilizado, a provisão de serviços no domínio *intserv* é dinâmica e distribuída, com a utilização do RSVP como protocolo de sinalização. Já o domínio *diffserv* emprega uma provisão de serviços dinâmica e centralizada, com a utilização de um BB que faz uso do protocolo COPS na troca de informações de gerenciamento de recursos com os roteadores do domínio.

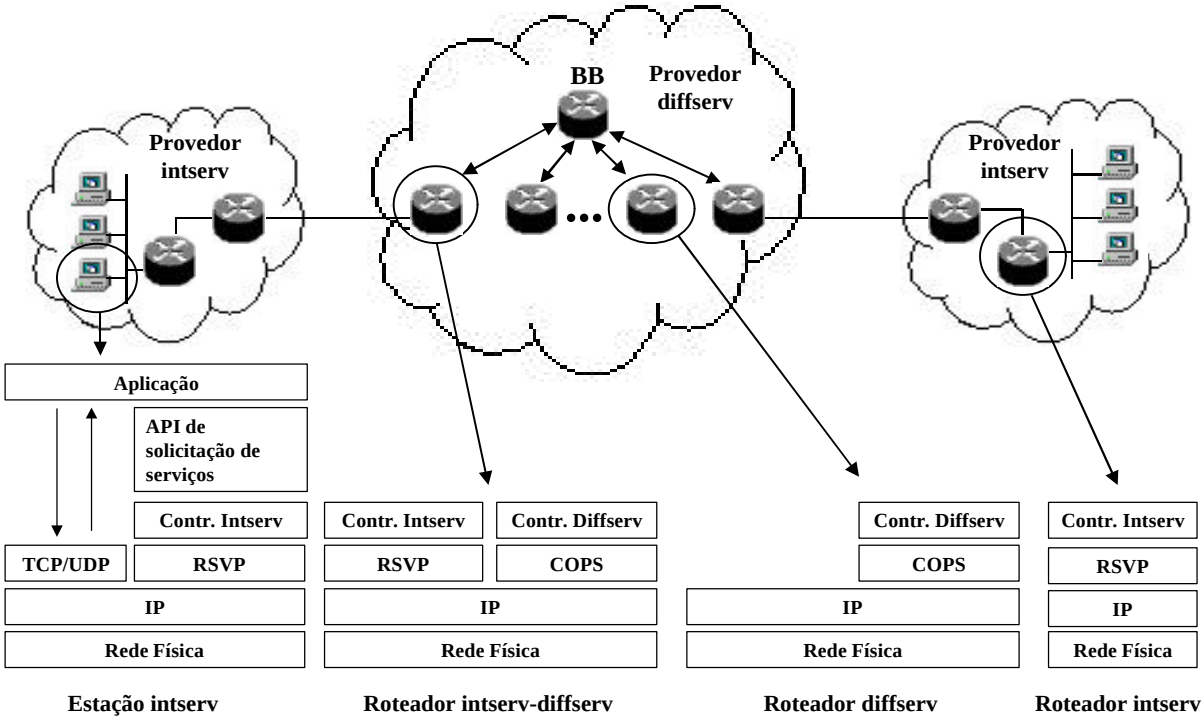

Figura 5: provisão de serviços *intserv* sobre *diffserv*.

O estabelecimento de um SLA inicia-se quando uma aplicação transmissora, em um estação *intserv* qualquer, deseja informar, a um grupo de aplicações pertencentes a uma mesma sessão de trabalho, as características dos fluxos disponíveis. Normalmente, por meio da própria interface de solicitação de serviços, a aplicação transmissora informa ao módulo de controle *intserv* local as características dos fluxos a serem anunciados na sessão. Esta caracterização é descrita por meio de uma instância da classe *TSpec*, apresentada na Seção 4.1. O módulo de controle encapsula essas informações em uma estrutura RSVP *flowspec*, que será, futuramente, encaminhada às aplicações pertencentes à sessão de trabalho por meio de uma mensagem PATH. Cabe aos roteadores *intserv* presentes no caminho entre a aplicação transmissora e as receptoras a escolha de uma rota que possua recursos suficientes para o transporte do tráfego anunciado. Adicionalmente, parâmetros de desempenho da rede são encapsulados em uma estrutura RSVP *adspec*, também encaminhada nas mensagens PATH, devendo ser atualizados por cada um desses roteadores.

No domínio de provisão de serviços *diffserv*, as mensagens PATH são encaminhadas pelos roteadores de forma transparente, não havendo qualquer interpretação ou alteração nas suas estruturas, à exceção dos roteadores de borda de entrada, que possuem, conforme ilustra a Figura 4, os módulos de controle *intserv* e *diffserv*. Cabe ao módulo de controle *intserv* presente nesses roteadores de borda a atualização da estrutura *adspec* de forma a fornecer às aplicações pertencentes à sessão de trabalho uma estimativa do desempenho relativo ao caminho a ser tomado dentro do domínio *diffserv*.

Em condições ideais, a interface de solicitação de serviços deve prover mecanismos de notificação às aplicações quando da ocorrência de eventos relacionados à negociação de QoS, como a chegada de informações relativas ao anúncio de tráfego, por exemplo. De posse dessas informações, dos requisitos próprios de QoS e do nível de garantia desejados, uma instância de uma das subclasses de *IntservServiceCategory* é criada pela aplicação receptora pertencente à sessão. O módulo de controle *intserv* no receptor encapsula essas informações em uma estrutura RSVP *flowspec*, que será encaminhada à aplicação transmissora por meio de mensagens RESV. Em cada roteador *intserv* no caminho entre a aplicação receptora e a transmissora[5], o processo de negociação de QoS é iniciado, sendo ilustrado, sob a ótica da arquitetura proposta, na Figura 6.

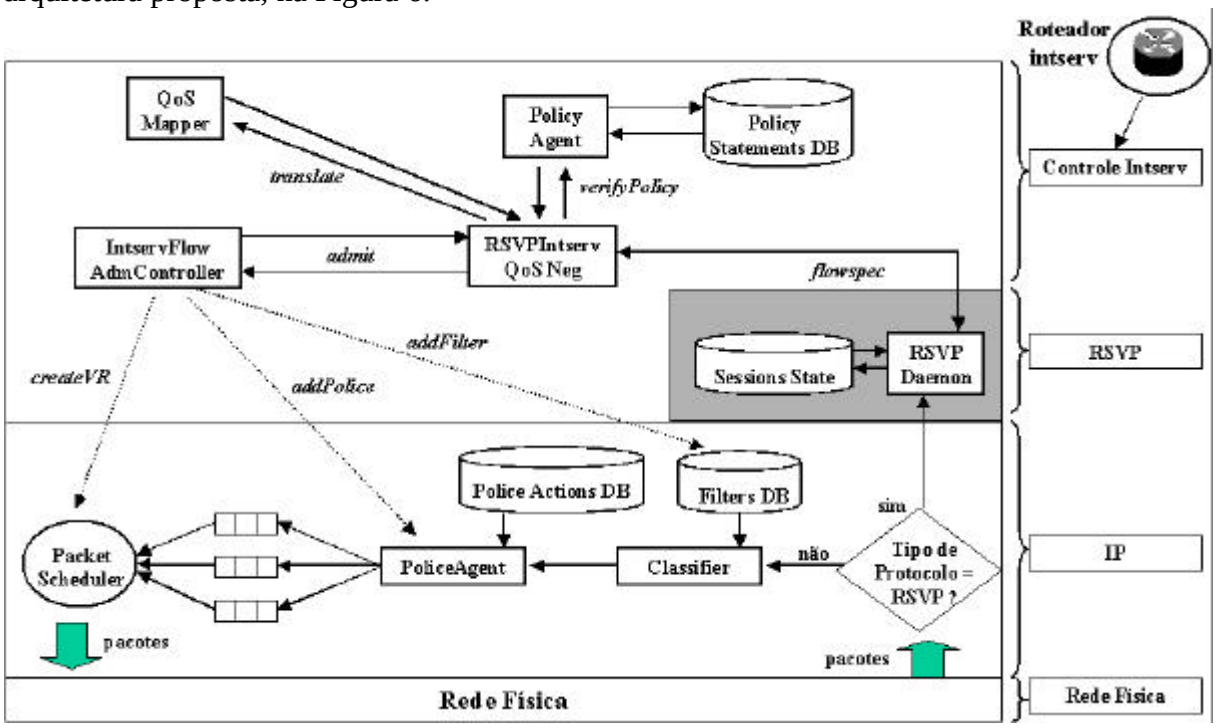

Figura 6: Provisão de QoS em um roteador *intserv*.

No roteador *intserv*, as mensagens RESV são repassadas ao módulo RSVP *daemon*, que extrai as informações relativas às estruturas encapsuladas, em especial a estrutura *flowspec*, que é informada ao módulo *RsvpIntservQoSNeg*, que representa uma instanciação da classe de mesmo nome descrita na Seção 4.3. Caso as políticas configuradas no roteador, contidas em *PolicyStatementsDB* e consultadas por intermédio do módulo *PolicyAgent*, não imponham restrições ao fornecimento do serviço solicitado, o módulo de negociação inicia o processo de mapeamento dos parâmetros de serviço em valores que possam ser interpretados pelos subsistemas envolvidos: processador, memória e rede física. Com base neste mapeamento é que será efetivado o controle de admissão no roteador. Os módulos de mapeamento

[5] Por simplificação, somente estão sendo consideradas requisições de serviço relativas a um único fluxo.

(*QoSMapper*) e de controle de admissão (*FlowAdmissionController*) ilustrados na Figura 6 correspondem às classes de mesmo nome descritas na Seção 4.3.

Caso o serviço solicitado possa ser admitido, o módulo de controle de admissão inicia a configuração dos agentes de encaminhamento dos pacotes, responsáveis pela classificação, policiamento e escalonamento, representados, no esquema da Figura 6, pelos módulos *Classifier*, *PoliceAgent* e *PacketScheduler*, respectivamente. Esses módulos também representam instanciações de classes de mesmo nome ilustradas na Seção 4.3. Ao fim do processo de configuração, o agente de negociação reformata a estrutura *flowspec* e a envia ao módulo *RSVP daemon* para posterior encaminhamento.

De forma semelhante às mensagens PATH, as solicitações de serviço *intserv* contidas nas mensagens RESV são ignoradas pelos roteadores do domíno *diffserv*, a exceção daqueles localizados na borda de entrada. O módulo *RSVPIntservQoSNeg* presente nesses roteadores deve ser capaz de traduzir, por intermédio do agente de mapeamento *QoSMapper*, os parâmetros de serviço *intserv* em parâmetros correspondentes a um serviço *diffserv* que satisfaça aos requisitos da aplicação receptora. Uma vez que esse mapeamento tenha sido realizado, o processo de negociação no domínio *diffserv* é iniciado. Conforme mencionado, o cenário discutido neste trabalho emprega, no domínio *diffserv*, a modalidade de provisão de serviços centralizada. O lado cliente da negociação encontra-se ilustrado na Figura 7.

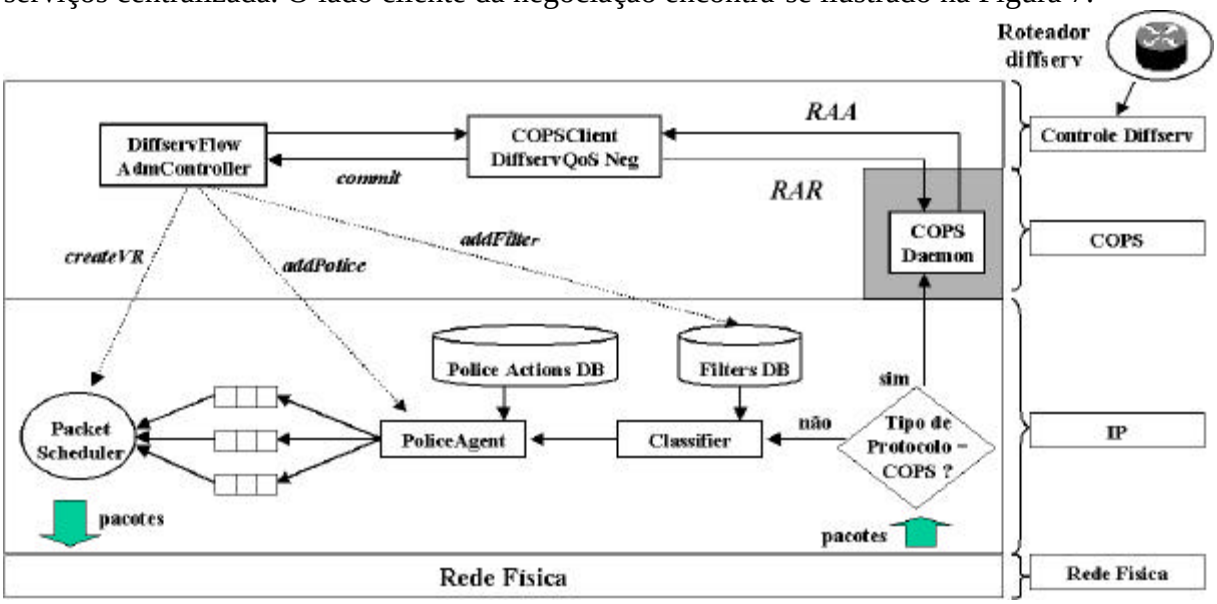

Figura 7: Provisão de QoS em um roteador *diffserv*.

Na Figura 7, uma outra derivação da classe *QoSNegotiator* foi utilizada na modelagem do cenário, sendo representada no roteador *diffserv* pelo módulo *COPSClientDiffservQoSNeg*. Esse negociador é configurado com o endereço do BB do domínio, sendo responsável por encaminhar ao mesmo a solicitação de serviço. O encaminhamento dessa solicitação é feito pela formatação de uma mensagem RAR (*Resource Allocation Request*) transmitida ao BB por intermédio do protocolo COPS. A Figura 8 ilustra os módulos presentes no BB.

O negociador de QoS no BB é representado pelo módulo *COPSServerDiffservQoSNeg*. Cabe a este módulo analisar as solicitações de serviço contidas nas mensagens RAR, verificar a disponibilidade de recursos no domínio *diffserv* e decidir pela posterior confirmação ou rejeição do fornecimento do serviço. A configuração apropriada dos roteadores *diffserv* é efetuada pelo envio de mensagens RAA (*Resource Allocation Answer*), encaminhadas pelo protocolo de sinalização COPS. Cabe aos negociadores presentes nos roteadores *diffserv* realizarem as configurações apropriadas com base nas mensagens RAA recebidas. Com uma

resposta positiva do BB, o módulo de negociação *diffserv* do roteador de borda sinaliza ao módulo de negociação *intserv* que o serviço pode ser realizado. Posteriormente, o módulo *RSVPIntservQoSNeg* presente neste roteador procederá com a formatação da estrutura *flowspec* a ser encaminhada ao transmissor através de uma mensagem RESV.

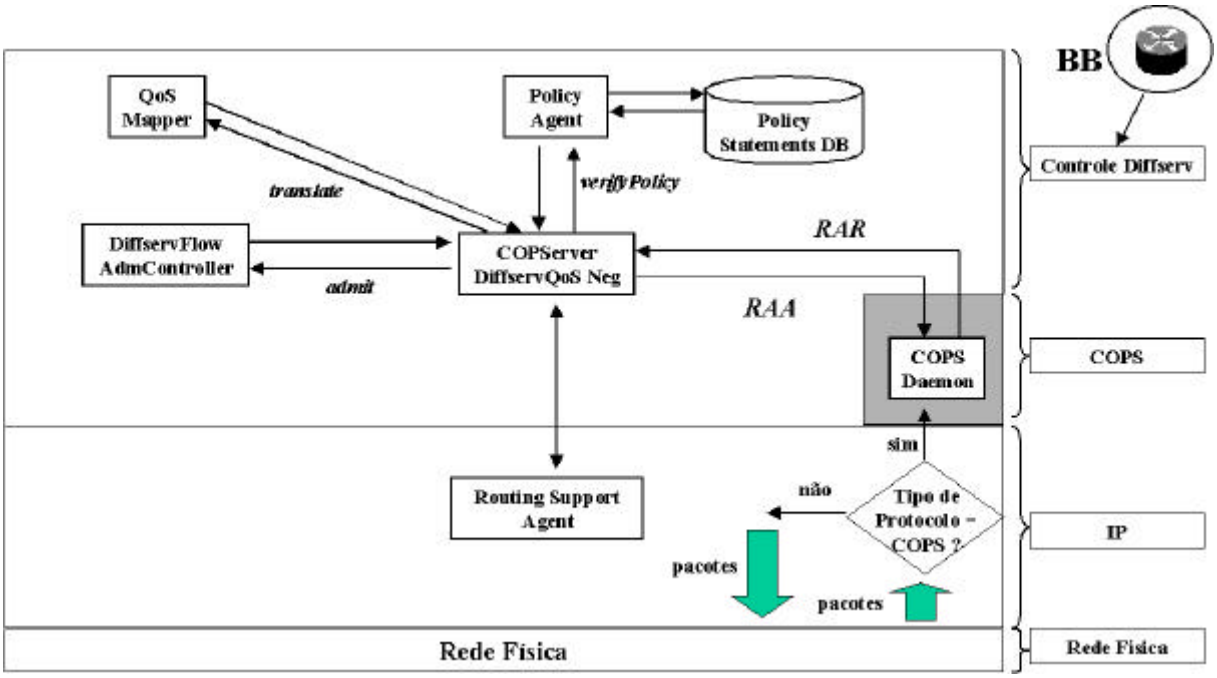

Figura 8: Provisão de QoS em um BB.

A necessidade de uma interface de solicitação de serviço independente de plataforma e de modelo de QoS motivou, durante os trabalhos de modelagem da arquitetura proposta, o desenvolvimento de uma API em Java que fornecesse suporte adequado aos projetos em curso no laboratório TeleMídia. Na sua versão inicial, a API desenvolvida utiliza negociadores *intserv* baseados no protocolo RSVP. O detalhamento completo dessa API foge ao escopo desse artigo, podendo ser obtidas maiores informações em [Mota01].

6. Conclusões

O fato de várias propostas terem sido desenvolvidas com o objetivo de se definir modelos de serviços com QoS na Internet dá margem a uma heterogeneidade de configurações que podem ser adotadas nos diversos provedores de serviço que compõem essa rede. A interoperação dessas configurações possíveis, bem como a implementação dos diferentes modelos de serviços sobre tecnologias distintas de sub-redes, sugere a definição de uma arquitetura genérica capaz de facilitar o entendimento dos projetistas de provedores de serviços acerca das opções de configuração possíveis e dos mecanismos necessários à correta interoperação dessas configurações. O objetivo deste trabalho foi propor uma arquitetura com essas características e que, principalmente, desse suporte à adaptação a novos modelos, categorias de serviços e tecnologias.

A arquitetura proposta foi parcialmente implementada no Laboratório TeleMídia da PUC-Rio. As classes referentes aos roteadores (comutadores) exigirá a utilização de equipamentos abertos e programáveis, aos quais ainda não tivemos acesso.

Bibliografia

[Bern00] Bernet, Y., et al, “A Framework for Integrated Services Operation Over Diffserv Networks”, RFC 2998, Novembro de 2000.

[Blair98] Blair, G. S., Coulson, G. “The Case of Reflective Middleware”, Technical Report MPG-98-38, Distributed Multimedia Research Group, Departament of Computing, Lancaster University, 1998.

[Blak98] Blake, S., et al, "An Architecture for Differentiated Services", RFC 2475, Dezembro de 1998.

[Boyl00] Boyle, J., et al, " The COPS (Common Open Policy Service) Protocol ", RFC 2748, Janeiro de 2000.

[Brad94] Braden, R., et al, “Integrated Services in the Internet Architecture: an Overview”, RFC 1633, Junho de 1994.

[Brad97] Braden, R., et al, "Resource ReSerVation Protocol (RSVP): Version 1 Functional Specification", RFC 2205, Setembro de 1997.

[Calh00] Calhoun, P., et al, "Diameter: Resource Management Extensions", Internet draft draft-calhoun-diameter-res-mgmt-06.txt, Setembro de 2000.

[Gome99] Gomes, A. T. A., Colcher, S., Soares, L.F.G., "Um Framework Para Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação", In: Anais do XVII Simpósio Brasileiro de Redes de Computadores (SBRC’99), Salvador-BA, Maio de 1999.

[Gome01] Gomes, A.T.A., Colcher, S., Soares, L.F.G. "Modeling QoS Provision on Adaptable Communication Environments", In: IEEE International Conference on Communications (ICC 2001), Helsinki, Finlândia, junho de 2001.

[Jaco99] Jacobson, V., et al, "A Two-bit Differentiated Services Architecture for the Internet", RFC 2638, Julho de 1999.

[Li98] Li, T., et al, “A Provider Architecture for Differentiated Services and Traffic Engineering (PASTE)”, RFC 2430, Outubro de 1998.

[Mota01] Mota, O., " IPQoS: Uma Interface em Java para Solicitação de Serviços com QoS na Internet", Projeto Final de Programação, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Janeiro de 2001.

[Pare92] 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.

[Pree95] Pree, W. “Design Patterns for Object-Oriented Software Development”, Addison Wesley, 1995.

[Rodr99] Rodrigues, M. A. A., Colcher, S., Soares, L.F.G., "Um Framework Para Provisão de Serviço de Multicast em Ambientes Genéricos de Comunicação de Dados", In: Anais do XVII Simpósio Brasileiro de Redes de Computadores (SBRC’99), Salvador-BA, Maio de 1999.

[Shen97] Shenker, S., et al, “Specification of Guaranteed Quality of Service”, RFC 2212, Setembro de 1997.

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

[Wroc97] Wroclawski, J., “Specification of the Controlled-Load Network Element Service”, RFC 2211, Setembro de 1997.