Um Framework para Provisão de QoS em Ambientes Genéricos de Processamento e Comunicação[1]

ANTÔNIO TADEU A. GOMES atagomes@inf.puc-rio.br
SÉRGIO COLCHER colcher@inf.puc-rio.br
LUIZ FERNANDO G. SOARES lfgs@inf.puc-rio.br

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

Resumo

Este trabalho apresenta um framework que organiza a provisão de QoS em um domínio genérico o suficiente para incluir quaisquer ambientes de processamento e comunicação, desde subsistemas específicos até plataformas para sistemas distribuídos. Com estas características, o framework permite: (i) a construção de sistemas configuráveis no que se refere à introdução de novos serviços, e (ii) facilita a implementação dos mecanismos de compartilhamento e orquestração de recursos.

Palavras-chave: Qualidade de Serviço, Redes Multimídia, Orientação a Objetos.

Abstract

This work presents a framework that organizes QoS provision in a domain which is sufficiently generic to include any comunication and processing environment, encompassing from specific subsystems to distributed system platforms. With these characteristics, the framework allows: (i) the construction of configurable systems with regard to the introduction of new services, and (ii) eases the implementation of resource sharing and orchestrating mechanisms.

Keywords: Quality of Service, Networked Multimedia, Object Orientation.

1. Introdução

Nos últimos anos, tem-se observado uma crescente demanda por *sistemas de processamento e comunicação multimídia*, graças à utilização cada vez maior de aplicações como videoconferência, correio multimídia, etc. A diversidade de requisitos impostos pelas diferentes mídias a esses sistemas é um problema em voga na área de sistemas distribuídos[1].

É trivial para um sistema oferecer a qualidade de processamento e comunicação requerida por uma mídia específica se recursos dedicados são usados, como acontece nos sistemas telefônicos atuais. Porém, devido a restrições gerenciais e de custos, há uma necessidade cada vez maior dos sistemas terem recursos eficientemente compartilhados por diferentes mídias. Ou seja, um mesmo sistema deve ser capaz de fornecer diferentes serviços aos usuários (produtores/consumidores de dados), de acordo com as diferentes mídias de interesse [2]. É importante notar que os requisitos impostos pelas mídias a um sistema são *fim-a-fim*, ou seja, todos os subsistemas que participam do oferecimento de um serviço devem ter seus recursos orquestrados de modo que o serviço como um todo possa ser corretamente fornecido (vide Figura 1). Tanto a orquestração quanto o compartilhamento eficiente de recursos só são viáveis se os usuários puderem especificar, através de parâmetros apropriados, a *qualidade de*

1 Trabalho parcialmente financiado pela Empresa Brasileira de Telecomunicações (Embratel).

serviço (*QoS*) desejada ao sistema. Dessa forma, o sistema pode configurar de maneira adequada os mecanismos de provisão de QoS, responsáveis pela orquestração e pelo compartilhamento de recursos nos diferentes subsistemas.

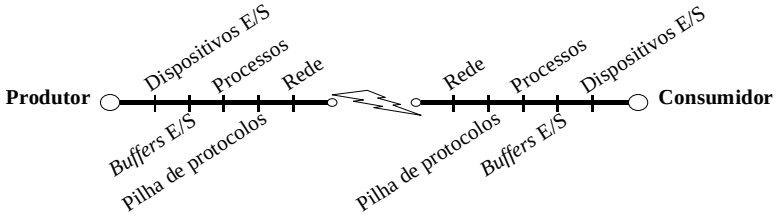

Figura 1: cenário de um sistema multimídia [3].

O cenário atual relacionado à provisão de QoS em sistemas multimídia é descrito basicamente em termos de arquiteturas para fornecimento de serviços específicos. Nessas arquiteturas, tanto os esquemas de parametrização quanto os mecanismos de provisão da QoS são predefinidos pelo projetista do sistema. Este trabalho é motivado pela crença de que tais arquiteturas não são flexíveis o bastante para dar suporte adequado a novas técnicas de tratamento de mídias que por ventura venham a ser desenvolvidas.

Este trabalho segue uma abordagem mais flexível, baseada em um framework cujo domínio de aplicação abrange não só a construção de um sistema como um todo, no que se refere à provisão de QoS, mas também a de seus subsistemas, recursivamente. A aplicação do framework viabiliza a construção de sistemas configuráveis no que se refere à parametrização e à provisão da QoS, permitindo a introdução de novos serviços, e facilita a tarefa dos projetistas no que se refere à implementação do compartilhamento e orquestração de recursos.

O restante deste artigo é organizado da seguinte forma: pesquisas relacionadas à provisão da QoS são sumariadas e avaliadas na Seção 2. O framework para provisão de QoS é detalhado na Seção 3. A Seção 4 apresenta alguns casos de uso do framework enquanto que a Seção 5 é reservada à apresentação de alguns comentários a respeito deste trabalho.

2. Trabalhos Relacionados

Nos últimos anos, boa parte do desenvolvimento relacionado ao suporte à provisão de QoS ocorreu no contexto de subsistemas isolados, abrangendo, em geral, partes específicas das arquiteturas de protocolos de comunicação e de sistemas operacionais. Mais recentemente, novas arquiteturas de sistema e modelos de computação distribuída, que abordam a questão da provisão de QoS fim-a-fim, têm sido desenvolvidos.

2.1 Sistemas Operacionais

Na área de sistemas operacionais, a maior parte dos trabalhos relacionados à provisão de QoS tem focalizado o escalonamento de processadores. Os sistemas operacionais tradicionais adotam estratégias de escalonamento fixas, normalmente baseadas em prioridades definidas pelas aplicações [15,22]. Entretanto, requisitos de processamento podem variar de aplicação para aplicação, e descrever esses requisitos somente por intermédio de prioridades inviabiliza uma provisão adequada de QoS. Por exemplo, threads periódicas requerem que o parâmetro período seja disponibilizado pelo sistema operacional. A adoção de parâmetros mais específicos permite que estratégias adequadas de escalonamento possam ser utilizadas pelo sistema operacional. Como exemplo, a estratégia *Earliest Deadline First* (*EDF*) [21] vale-se do período para escalonar threads periódicas corretamente.

A necessidade de dar suporte ao processamento integrado de diferentes mídias requer que o sistema operacional permita o uso de diferentes conjuntos de parâmetros pelas aplicações e, consequentemente, que diferentes estratégias de escalonamento possam atuar concorrentemente sobre um mesmo processador. Neste sentido, uma abordagem interessante é a do particionamento hierárquico da capacidade do processador, utilizado em trabalhos como [10] e [11]. Esta abordagem permite a um sistema operacional particionar a capacidade do processador entre diferentes categorias de processamento, através de uma determinada estratégia de escalonamento. Estas categorias, por sua vez, podem ter suas parcelas da capacidade do processador particionadas (possivelmente usando uma estratégia diferente) entre subcategorias, de modo recursivo. A abordagem de particionamento hierárquico é suficientemente flexível para permitir a introdução de novas estratégias de escalonamento. Devido a essa característica, no presente trabalho a abordagem de particionamento hierárquico é generalizada para o caso do compartilhamento de recursos quaisquer.

2.2 Protocolos de Comunicação

Na parte de protocolos de comunicação, os trabalhos relacionados à provisão de QoS têm seguido, em geral, na direção da formulação de protocolos que envolvam a negociação da QoS e a reserva de recursos. A característica desses protocolos varia de acordo com o nível de operação dos mesmos, que pode ser: (i) fim-a-fim, como os protocolos RTP [25], OSI95 [17], e as camadas de adaptação (AALs) definidas em redes com tecnologia ATM [16], ou (ii) operar em elementos intermediários, como os protocolos RSVP [26] e ST-II [27].

A definição de protocolos fim-a-fim é normalmente dependente do suporte e da flexibilidade oferecidos pela infra-estrutura de transmissão. Por exemplo, uma infra-estrutura baseada na tecnologia ATM deverá permitir uma gama maior de serviços de transmissão e um transporte mais eficiente das diferentes mídias do que outra baseada em tecnologias tradicionais de comutação de pacotes. A definição das diferentes AALs e os serviços específicos que sobre elas têm sido especificados correspondem a protocolos fim-a-fim que fazem uso da flexibilidade oferecida pela tecnologia ATM. Apesar disso, essa flexibilidade é limitada por um número finito de classes de serviço modeladas pelas AALs. O presente trabalho segue uma linha mais flexível, no sentido de que a definição do framework fornece uma capacidade de extensão que pode ser utilizada para configuração de protocolos voltados especificamente ao serviço desejado, sem a necessidade da definição de um conjunto previamente estabelecido de classes ou parâmetros que caracterizem o serviço.

Muitos trabalhos também têm investido na pesquisa de novos modelos de caracterização de tráfego (como o D-BIND [30]) e de especificação da QoS (como os definidos em [29], [18] e [23]). Em particular, a questão do mapeamento de parâmetros de QoS entre camadas, tratada em trabalhos como [29], é um problema que ainda necessita de maior aprofundamento. O mapeamento está também ligado à questão da orquestração, que é tratada de modo mais abrangente nos trabalhos ligados às arquiteturas de QoS.

2.3 Arquiteturas de QoS

Tratar a questão da QoS nos diferentes subsistemas isoladamente potencializa a geração de sistemas incompletos, pouco integrados e muito específicos [4]. Por esta razão, vários grupos de trabalho (como os de Pensylvannia [24] e Lancaster [19]) têm proposto novas abordagens de arquitetura de sistema como solução para o problema da provisão de QoS fim-a-fim, sendo coletivamente denominadas de *arquiteturas de QoS* [3].

Independente da abordagem tomada pelas diferentes arquiteturas de QoS, o que elas têm basicamente em comum é a definição de interfaces que formalizam o conceito de QoS em diferentes subsistemas. Esta formalização é semelhante a feita pelo RM-OSI [14] com relação

ao conceito de serviço. Interfaces são em geral apresentadas sob a forma de frameworks, linguagens de programação ou bibliotecas desenvolvidos sobre subsistemas com características de QoS predefinidas. Com esta abordagem, as arquiteturas de QoS permitem a um projetista integrar mecanismos de provisão de QoS presentes nos diferentes subsistemas.

2.4 Modelos de Objetos com QoS

A maior limitação da abordagem adotada pelas arquiteturas de QoS é que a provisão da QoS só é uniforme na fronteira entre os subsistemas. A definição da arquitetura interna dos subsistemas fica a cargo dos projetistas, o que é indesejável por vários motivos. Um exemplo é a ausência de um modelo para a orquestração de recursos, tanto interna a subsistemas quanto entre subsistemas. Isto dificulta a tarefa do projetista em realizar uma provisão de QoS verdadeiramente fim-a-fim.

Outro obstáculo que um projetista pode encontrar, ao se valer de uma arquitetura de QoS para a implementação de um sistema multimídia, é a diversidade de requisitos existentes. As arquiteturas de QoS sugerem, intrinsecamente, o desenvolvimento de conjuntos finitos de mecanismos para provisão de QoS, em cada subsistema, que atendam a todos os possíveis requisitos de processamento e comunicação, o que torna tais subsistemas muito complexos e pouco flexíveis. Além disso, a ausência de uma arquitetura interna impede que determinados mecanismos para provisão de QoS, potencialmente presentes em vários sistemas distintos (ou mesmo em subsistemas de um mesmo sistema), possam ser facilmente reutilizados.

Uma outra abordagem para o problema da provisão de QoS que tem sido considerada em recentes trabalhos, como o Modelo Binding [5] e o Modelo de Referência Unificado [6], é a do oferecimento de um *único* serviço configurável de acordo com a necessidade dos usuários. Tais trabalhos, em grande parte baseados no RM-ODP [12] e no CORBA [13], caracterizam-se pela definição de *ambientes genéricos de processamento e comunicação*, que permitem a implementação de sistemas a partir da conexão entre componentes OO reutilizáveis.

3. Framework Para Provisão de QoS

É com base na abordagem de serviços configuráveis em ambientes genéricos de processamento e comunicação, conforme definido em [6], que o *Framework Para Provisão de QoS* foi proposto. O framework pode ser aplicado independentemente da escala de *distribuição* dos componentes de um sistema (em um mesmo processo, em processos distintos de uma mesma máquina, em máquinas distintas de uma mesma rede ou de redes distintas interconectadas), assim como do tipo de usuário em questão (um componente, uma aplicação ou uma pessoa interagindo com o sistema). Esta característica permite ao framework alcançar uma granulosidade mais fina do que a das arquiteturas de QoS, possibilitando:

- a definição de sistemas altamente flexíveis no que concerne à provisão de QoS, através da adaptação de alguns poucos mecanismos de provisão de QoS ao fornecimento de diferentes serviços; e
- a modelagem da orquestração de recursos entre subsistemas, graças à estruturação recursiva do framework com relação às diferentes escalas de distribuição dos componentes de um sistema.

3.1 Modelo Genérico de Operação de Um Sistema Com Provisão de QoS

A orquestração de recursos em um sistema só é viável se os usuários indicarem a QoS desejada, para que o sistema possa provê-la corretamente através da configuração dos mecanismos apropriados. Um modelo genérico de operação de um sistema que aceita requisições de usuários e provê mecanismos de provisão de QoS pode ser dividido nas

seguintes fases: (i) iniciação do sistema, (ii) requisição de serviços, (iii) estabelecimento de contratos de serviço e (iv) manutenção de contratos de serviço.

3.1.1 Iniciação do Sistema

Para que um sistema possa prover QoS, é necessário primeiramente que seja definida a infra-estrutura que dará suporte aos serviços oferecidos e que determinará, juntamente com as *políticas de provisão de QoS* adotadas durante a existência do sistema, a forma de descrição do *estado interno* do mesmo. Políticas de provisão de QoS são conjuntos de estratégias que regulam a maneira como um sistema controla e gerencia a QoS por ele provida.

A definição do estado interno de um sistema inclui informações sobre recursos disponíveis no mesmo. Recursos podem ser classificados como *estaticamente* ou *dinamicamente* compartilhados. Enquanto os primeiros permanecem dedicados a um único fornecimento de serviço por vez, os últimos podem ser compartilhados por vários desses fornecimentos simultaneamente. Como exemplos de recursos dinamicamente compartilhados, podem ser citados o processador e a memória de uma máquina e os enlaces (físicos ou lógicos) de uma rede. Dispositivos de entrada e saída como câmeras, microfones, codecs, visores e alto-falantes são exemplos de recursos estaticamente compartilhados.

3.1.2 Requisição de Serviços

Após a iniciação do sistema, usuários podem requisitar-lhe serviços a qualquer momento. Uma requisição se dá através da caracterização da *carga* de processamento e comunicação a ser produzida ou consumida pelos usuários, e da especificação da QoS desejada. Carga refere-se à dinâmica dos *fluxos de dados* gerados entre os usuários e o sistema, e que estão associados ao fornecimento do serviço. São exemplos de fluxos um conjunto de instruções a serem executadas em um processador, um conjunto de pacotes a serem transmitidos através de um sistema de comunicação, ou um conjunto de quadros de vídeo a serem enviados por um sistema de distribuição de vídeo.

A interface do sistema deve permitir requisições de serviço condizentes com a *visão de QoS* de seus usuários. Isto significa que a caracterização da carga e a especificação da QoS devem ser facilmente compreendidas pelos usuários. Pessoas preocupam-se com a qualidade de percepção (e o custo associado) em um sistema de distribuição de vídeo, enquanto aplicações multimídia levam mais em consideração a qualidade de transmissão em um sistema de comunicação.

Uma vez que um sistema pode ter componentes presentes em várias escalas distintas de distribuição, o sistema pode apresentar também vários *níveis de visão de QoS*. Isto implica em traduzir ou mapear tais visões entre esses componentes. Esta função é oferecida pelos mecanismos de *mapeamento da QoS*, nos quais vários mecanismos de provisão de QoS (em especial, os que gerenciam a orquestração) se baseiam para funcionarem corretamente.

3.1.3 Estabelecimento de Contratos de Serviço

Ao receber uma nova requisição de serviço, o sistema determina se ele tem recursos suficientes para manter a QoS desejada pelo usuário, e se a admissão dos fluxos do mesmo trará algum impacto na QoS de fluxos previamente admitidos, levando em conta o estado interno atual do sistema, a caracterização da carga e a especificação da QoS. Os mecanismos de *controle de admissão de fluxos* são os responsáveis por esses testes.

Se o sistema tiver recursos suficientes, ele admitirá os fluxos do usuário e fará a *alocação dos recursos* necessários para servi-los (levando possivelmente a uma mudança de estado interno), de tal forma que a QoS especificada seja satisfeita. Senão, ele poderá ou rejeitar os fluxos do usuário ou lhe sugerir uma outra especificação de QoS factível de ser satisfeita.

Neste último caso, se o usuário aceitar a nova proposta, seus fluxos são admitidos; caso contrário, são rejeitados, podendo o usuário tentar requisitar o serviço novamente em outra oportunidade. Estes ciclos de requisição e resposta entre os usuários e o sistema são providos pelos mecanismos de *negociação da QoS*.

Durante o fornecimento do serviço, mudanças na QoS podem ser necessárias por várias razões. Em um exemplo típico, uma pessoa inicialmente requer um serviço de vídeo sob demanda de alta definição. Como ela é cobrada com base na QoS oferecida, ela pode posteriormente decidir reduzir a qualidade de definição do vídeo. Logo, mecanismos de *renegociação da QoS* também são necessários. A renegociação da QoS pode ser provocada também pelo sistema, como, por exemplo, na ocorrência de erros de processamento e comunicação que afetem a QoS.

Mecanismos de negociação e renegociação da QoS são responsáveis também pela parte da orquestração de recursos que ocorre durante a fase de estabelecimento, no caso de sistemas que gerenciam mais de um tipo de recurso em um mesmo subsistema, sendo necessária a "divisão" da responsabilidade de provisão da QoS entre esses recursos, ou que envolvem vários subsistemas, e a divisão dessa responsabilidade ocorre entre os diferentes subsistemas.

3.1.4 Manutenção de Contratos de Serviço

Após os fluxos do usuário serem admitidos pelo sistema, este último deve garantir que o usuário atenha-se à carga caracterizada, caso ele seja uma fonte, e que a QoS negociada seja mantida durante todo o tempo de uso do serviço. É essa garantia que define um *contrato de serviço*: como em todo contrato, ambas as partes interessadas devem colaborar, ou seja, o usuário deve ater-se à carga caracterizada e o sistema deve manter a QoS negociada. A quebra do contrato por uma das partes pode ocasionar desde uma mera notificação ao usuário até a interrupção do serviço para aquele usuário.

Os mecanismos de *escalonamento de recursos* permitem o compartilhamento dinâmico de recursos entre diversos fluxos de dados presentes em um sistema, de maneira tal que a QoS negociada para cada fluxo possa ser mantida. Cada fluxo "enxerga" a sua parcela de utilização de um determinado recurso (necessária para a provisão da QoS desejada) como um *recurso virtual* ao qual o fluxo tem acesso exclusivo. Os paginadores de memória e escalonadores de threads dos sistemas operacionais [15], assim como os comutadores de circuitos virtuais presentes em alguns sistemas de comunicação [16], são exemplos de mecanismos de escalonamento. Os mecanismos de alocação de recursos podem também operar com base no conceito de recurso virtual, se o recurso em questão puder ser dinamicamente compartilhado.

Os mecanismos de *monitorização de fluxos* permitem o registro da carga efetivamente gerada pelos usuários e da QoS realmente oferecida pelo sistema. Em caso de violação de um contrato de serviço, estes mecanismos podem ter seus registros repassados aos usuários de interesse sob a forma de *alertas* [17], ou então tanto os mecanismos de renegociação quanto os de *sintonização da QoS* podem ser acionados, dependendo do grau de depreciação da QoS anteriormente negociada. Mecanismos de sintonização são responsáveis pela manutenção da orquestração de recursos definida durante a negociação da QoS, sem que haja a necessidade de interrupção (isto é, renegociação) do fornecimento do serviço. Quando um sistema gerencia vários recursos, se um deles não puder suportar sua parcela de responsabilidade de provisão da QoS (graças a uma sobrecarga, por exemplo), o mecanismo de sintonização se responsabilizará por efetuar operações de ajuste nos outros recursos (por intermédio dos mecanismos de escalonamento, por exemplo), a fim de manter a QoS previamente negociada.

3.2 Definição do Framework

A idéia de se definir o framework proposto nasceu da percepção de que, em todas as fases de provisão de QoS apresentadas na Seção 3.1, há padrões de estrutura e de comportamento que independem das escalas de distribuição dos componentes de um sistema, assim como dos serviços específicos a serem fornecidos por ele. Nesse sentido, a abordagem de *patterns* [7,8] permite a descrição de frameworks de forma mais abstrata do que o código correspondente que os implementaria. Patterns permitem o delineamento dos pontos de flexibilização associados aos padrões modelados por um framework, de modo independente dos aspectos relativos à aplicação do mesmo em um projeto específico.

No presente trabalho, o uso da abordagem de patterns visa ajudar o projetista a adaptar o framework mais facilmente às suas necessidades, ocultando detalhes relativos à distribuição dos componentes de um sistema, e mostrando da maneira mais clara possível como tornar um sistema flexível com relação ao fornecimento de serviços. A abordagem de patterns permite a um projetista construir também partes isoladas de um sistema associadas à provisão de QoS, através da aplicação de um pattern específico. Essas partes poderiam se inserir em sistemas já existentes, mesmo que não modelados segundo o framework.

O framework para provisão de QoS é composto por: (i) um framework para parametrização de serviços, (ii) patterns de compartilhamento de recursos e (iii) patterns de orquestração de recursos. Os diagramas de classes que representam graficamente essas partes do framework para provisão de QoS são especificados segundo a notação UML (*Unified Modeling Language*) [20]. 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.

3.2.1 Framework de Parametrização de Serviços

A adaptação de mecanismos de provisão de QoS a serviços e escalas de distribuição específicos é essencialmente guiada segundo: (i) a forma de descrição do estado interno do sistema; (ii) a caracterização da carga dos usuários; e (iii) a especificação da QoS desejada por eles. Todas essas informações podem ser estruturadas através de *parâmetros*. Um parâmetro é uma propriedade do sistema descrita por meio de um nome e associada a um valor de um tipo qualquer (inteiro, ponto flutuante, lista de outros valores, etc.).

Parâmetros podem ser armazenados ou trocados entre os usuários e o sistema. Por exemplo, enquanto parâmetros de caracterização de carga e de especificação da QoS são, em primeira instância, originados a partir de requisitos dos usuários, parâmetros internos do sistema podem ser registros de medições relativas à real QoS sendo oferecida aos usuários, ou resultados de análises de outros parâmetros.

Nota-se que não seria trivial para um projetista de sistemas definir um conjunto finito de parâmetros concretos que representasse todos os possíveis serviços e escalas de distribuição. O propósito do framework de parametrização é definir um esquema de estruturação de parâmetros independente dos possíveis serviços a serem fornecidos por um sistema qualquer.

Dentro do esquema proposto é definido o conceito de *parâmetro abstrato*. Parâmetros abstratos podem ser usados em serviços e escalas de distribuição distintos através de especializações (criando-se *hierarquias de derivação de parâmetros*), de forma a se conseguir parâmetros concretos em uma implementação real de um sistema. Assim, parâmetros como "vazão" e "retardo" podem ser definidos abstratamente, sendo posteriormente especializados para usos específicos do framework. Estas especializações podem incluir a definição de

estatísticas (como por exemplo “variação do retardo” e “vazão média”), calculadas a partir de outros parâmetros pelos mecanismos de monitorização de fluxos.

Se, por um lado, as hierarquias de derivação de parâmetros tornam a parametrização de serviços bastante flexível, por outro lado, deixar a especialização adequada de parâmetros, em todas as escalas de distribuição presentes em um sistema, totalmente a cargo do projetista torna os frameworks de parametrização pouco práticos. Por isso, parâmetros podem ser divididos em subconjuntos (não necessariamente disjuntos) que definem *categorias de serviço* que um sistema pode oferecer. Uma categoria expressa propriedades relacionadas às mídias envolvidas no fornecimento de um determinado serviço. Como exemplo, um vídeo é caracterizado pela necessidade de reprodução de seus dados para o usuário a uma taxa constante. Logo, derivações de “retardo”, como “retardo máximo” e “variação do retardo”, devem ser usadas para descrever a categoria de serviço de vídeo.

Categorias de serviço podem ser sucessivamente refinadas, criando assim *hierarquias de categorias de serviço*, se forem consideradas, por exemplo, técnicas específicas de tratamento para uma determinada mídia. Estendendo o exemplo anterior, pode-se refinar da categoria de serviço de vídeo uma subcategoria de serviço de vídeo sem compressão e compactação e outra de serviço de vídeo com compressão e compactação. Uma importante propriedade que difere as duas relaciona-se à natureza da carga gerada por elas. Assim, derivações de “vazão” podem ser usadas para descrever essas subcategorias. Para vídeos sem compressão e compactação, “vazão máxima” é suficiente; enquanto que para vídeos com compressão e compactação, “vazão média” e “máxima variação da vazão”, entre outros, são os mais usados.

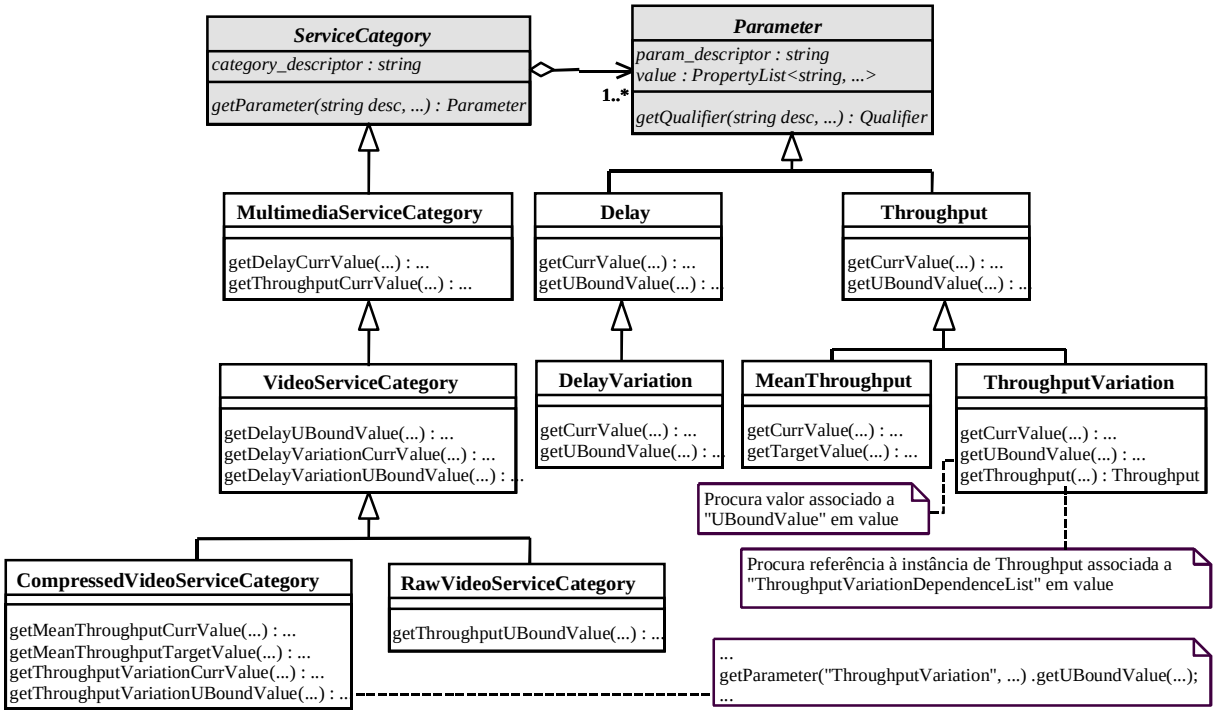

Figura 2: exemplo de instanciação do framework para parametrização de serviços.

A Figura 2 apresenta um exemplo simplificado de instanciação do framework para parametrização de serviços. As classes *ServiceCategory* e *Parameter* representam abstratamente as hierarquias de categorias de serviço e de derivação de parâmetros, respectivamente. O pattern estrutural Bridge [7] é utilizado para representar as categorias de serviço como conjuntos de parâmetros. Esta abordagem permite o desacoplamento entre ambas as hierarquias, dando

margem a diferentes formas de estruturação. No exemplo, todas as subclasses de *Parameter* representam parâmetros abstratos, e cada dupla de métodos {set*Value(), get*Value()} implementa um parâmetro concreto. Poderia também ter sido definido que somente as subclasses menos especializadas de *Parameter* simbolizariam parâmetros abstratos, enquanto os parâmetros concretos seriam representados por subclasses mais especializadas. Outra característica resultante do desacoplamento é que uma categoria de serviço pode ter um parâmetro estatístico substituído (durante o fornecimento do serviço) por outro de mesmo nome e com mesma semântica, mas cuja representação interna seja adequada a uma estratégia de monitorização que calcule o valor do parâmetro de modo mais preciso.

3.2.2 Patterns de Compartilhamento de Recursos

Uma restrição importante presente na maioria dos sistemas atuais é que eles só permitem o estabelecimento e manutenção de contratos relativos a algumas poucas categorias de serviço específicas. Objetivando possibilitar, através do uso de recursos virtuais, a definição de um conjunto amplo e flexível de serviços com características distintas de QoS em um mesmo sistema, é introduzido o conceito de *árvore de recursos virtuais*.

A cada recurso real está associada uma árvore de recursos virtuais. Como mostra a Figura 3, em uma árvore qualquer, as folhas são os recursos virtuais pelos quais os fluxos têm acesso à sua parcela de utilização do recurso real. A raiz da árvore corresponde a algum componente básico do sistema responsável por gerenciar diretamente o escalonamento do recurso real entre recursos virtuais, denominado de *escalonador de recurso real*. Os outros nós internos da árvore são recursos virtuais especializados e internos ao sistema, que se ocupam simplesmente de ceder a sua parcela de utilização do recurso real, escalonando-a entre seus recursos virtuais filhos, sendo portanto denominados de *escalonadores de recursos virtuais*. Cada escalonador mantém-se associado a uma determinada categoria de serviço e a políticas de provisão de QoS específicas. No contexto do compartilhamento e da orquestração de recursos, essas políticas definem basicamente as estratégias de admissão e escalonamento adotadas. Escalonadores de threads e comutadores de circuitos virtuais são exemplos de mecanismos de escalonamento que podem ser implementados com base em árvores de recursos virtuais.

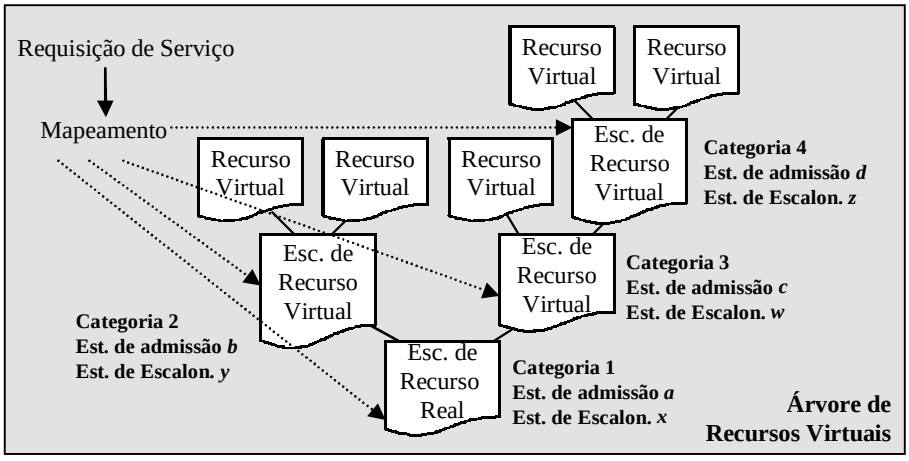

Figura 3: árvore de recursos virtuais.

Resumidamente, o processo de criação de recursos virtuais atua da seguinte forma: como parte do processo de tradução de uma requisição de serviço (efetuado pelos mecanismos de mapeamento, apresentados na subseção 3.2.3), esta requisição é associada a escalonadores presentes nas diferentes árvores que participam do oferecimento do serviço requisitado. A escolha apropriada de escalonadores é feita com base na categoria de serviço expressa pela

requisição. Os mecanismos de alocação de recursos associados a cada uma das árvores são acionados para criar os recursos virtuais, associá-los aos escalonadores escolhidos, e atualizar os parâmetros internos do sistema, que serão utilizados pela estratégia de escalonamento interna a cada um dos escalonadores.

Na Figura 4, a classe *ResourceScheduler* representa os mecanismos de escalonamento de recursos. O pattern comportamental Strategy [7] é utilizado para representar o relacionamento entre os mecanismos de escalonamento e as respectivas estratégias, representadas na figura pela classe *SchedulingStrategy*. A abordagem sugerida pelo pattern Strategy permite que políticas de provisão de QoS variem independentemente dos mecanismos que as utilizam. A definição de uma família de estratégias intercambiáveis com relação a um determinado mecanismo permite, por exemplo, que uma estratégia de escalonamento correntemente utilizada por um escalonador (associado a uma categoria de serviço) possa ser substituída, durante o fornecimento do serviço, por outra que calcule de maneira mais eficiente o próximo recurso virtual filho a ser escalonado.

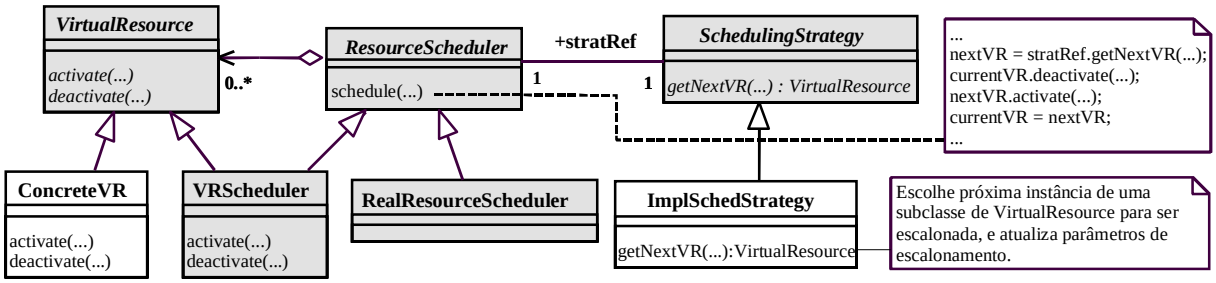

Figura 4: estrutura de escalonamento de recursos.

Os relacionamentos entre as classes VirtualResource, *ResourceScheduler*, VRScheduler e RealResourceScheduler representam a estrutura da árvore de recursos virtuais. O pattern estrutural Adapter [7] é utilizado para representar a interface genérica de escalonamento provida pela classe *ResourceScheduler*. Esta abordagem permite que os escalonadores de recurso real (representados pela classe RealResourceScheduler) e de recurso virtual (representados pela classe VRScheduler) possuam uma interface única para o resto do sistema. A compatibilização de interfaces permite, por exemplo, que: (i) recursos virtuais sejam escalonados tanto pelo escalonador de recurso real diretamente quanto por intermédio de escalonadores de recursos virtuais, indistintamente, e (ii) a infra-estrutura que dá suporte aos serviços oferecidos pelo sistema possa ser alterada, sem que isso incorra em uma alteração dos outros mecanismos de provisão de QoS.

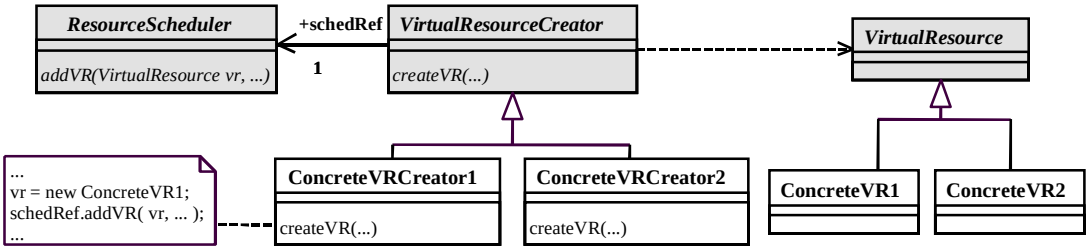

Figura 5: estrutura de criação de recursos virtuais.

Na Figura 5, é apresentada a estrutura de criação de recursos virtuais. A classe *VirtualResourceCreator* representa os componentes que, em conjunto, definem os mecanismos de alocação de recursos. O pattern de criação Factory Method [7] é utilizado para representar o

relacionamento entre cada um desses componentes e os tipos de recursos virtuais por eles criados.

A cada escalonador presente em uma árvore está ligado um componente de criação de um determinado tipo de recurso virtual. A título de exemplo, na Figura 5 pode-se considerar ConcreteVR1 e ConcreteVR2 classes que representam, respectivamente, threads periódicas e threads ligadas ao processamento de dados em lote. O pattern força que ConcreteVRCreator1 e ConcreteVRCreator2 modelem componentes de criação associados a escalonadores que usam estratégias compatíveis com ConcreteVR1 e ConcreteVR2 (como por exemplo, EDF e Round-Robin). A definição de um componente abstrato de criação de recursos virtuais permite que novas categorias de serviço possam ser introduzidas dinamicamente em um sistema, através da definição de componentes concretos de criação associados à nova categoria, em conjunto com as respectivas políticas de provisão de QoS.

3.2.3 Patterns de Orquestração de Recursos

Em um sistema distribuído, mecanismos como os de alocação e escalonamento de recursos podem estar dispersos em diversos subsistemas. Esses mecanismos devem ser gerenciados de maneira integrada, para viabilizar a orquestração de recursos. Os mecanismos de negociação e sintonização da QoS são os responsáveis por gerenciar essa integração. No presente trabalho, o conceito de fluxo é a base para a definição desses mecanismos.

Um fluxo de dados em um sistema distribuído pode ser composto de outros fluxos que atuam em escalas menores de distribuição, recursivamente. Como mostra o exemplo da Figura 6, um fluxo que represente a troca de pacotes entre duas entidades pares de um protocolo de comunicação de nível N pode ser visto como uma composição de: (i) fluxos que representem a passagem de unidades de informação entre entidades de protocolo dos níveis adjacentes N e N-1 e (ii) fluxos que representem a troca de pacotes entre entidades pares de um protocolo de nível N-1.

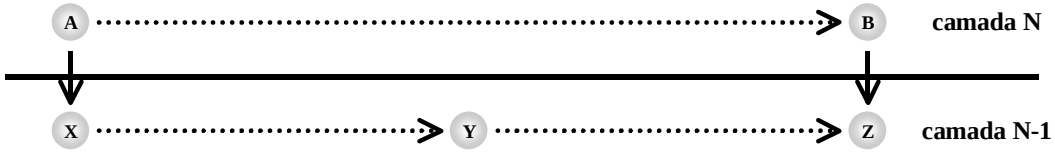

Figura 6: composição de fluxos.

Quando o mecanismo de negociação da QoS recebe uma requisição de serviço, ele aciona o mecanismo de mapeamento da QoS, que se responsabiliza pela tradução da requisição em parâmetros conhecidos internamente ao sistema. A requisição, já devidamente traduzida e associada aos escalonadores corretos, é passada ao mecanismo de controle de admissão, que valida a aceitação do novo fluxo no sistema, utilizando-se para isso das estratégias de admissão associadas aos escalonadores escolhidos nas árvores envolvidas no fornecimento do serviço. Essa validação é seguida do acionamento dos mecanismos de alocação de recursos, como foi descrito na subseção 3.2.2.

É importante perceber a característica inerentemente distribuída dos mecanismos de negociação, mapeamento e controle de admissão. Mais ainda, essa distribuição não ocorre em uma única escala específica, devido à própria definição recursiva de fluxo. Como exemplo, na Figura 6, o mecanismo de controle de admissão de um fluxo entre as entidades A e B (representado por A→B) envolve a admissão conjunta de fluxos A→X, X→Y, Y→Z e Z→B. Enquanto os fluxos X→Y e Y→Z assemelham-se a A→B quanto à escala de distribuição das entidades envolvidas, A→X e Z→B apresentam características locais, ou seja, as entidades

relacionadas são: (i) processos distintos de uma mesma máquina, ou (ii) objetos cujos métodos são executados em threads distintas de um mesmo processo, ou (iii) objetos cujos métodos são executados por uma mesma thread.

Pelo exemplo acima, nota-se a necessidade de que os três mecanismos atuem em todas as escalas de distribuição presentes em um sistema. Existem basicamente duas formas de se implementar esses três mecanismos: (i) centralizada, onde cada um desses mecanismos é implementado por um único objeto que tem acesso a todos os recursos de todos os subsistemas, e (ii) distribuída, onde cada um desses mecanismos é implementado por vários objetos presentes localmente nos vários subsistemas. Os patterns de orquestração de recursos visam definir esses mecanismos independentemente dessas duas formas de implementação.

Na Figura 7, as classes QoSNegotiator, QoSMapper e StreamAdmissionController representam os mecanismos de negociação, mapeamento e controle de admissão, respectivamente. O pattern comportamental Chain of Responsibility [7] é utilizado para representar o relacionamento recursivo definido em QoSNegotiator, que possibilita a implementação do mecanismo de negociação (e por conseqüência, os de mapeamento e controle de admissão) de forma centralizada ou distribuída. Outro pattern utilizado para representar a estrutura de estabelecimento da orquestração de recursos é o pattern estrutural Facade [7]. Este pattern possibilita a definição de uma interface única de requisição de serviços para os usuários do sistema, através da classe QoSNegotiator. O uso conjunto dos dois patterns permite que ambas as formas de implementação (centralizada ou distribuída) possam ser utilizadas de maneira transparente para os usuários.

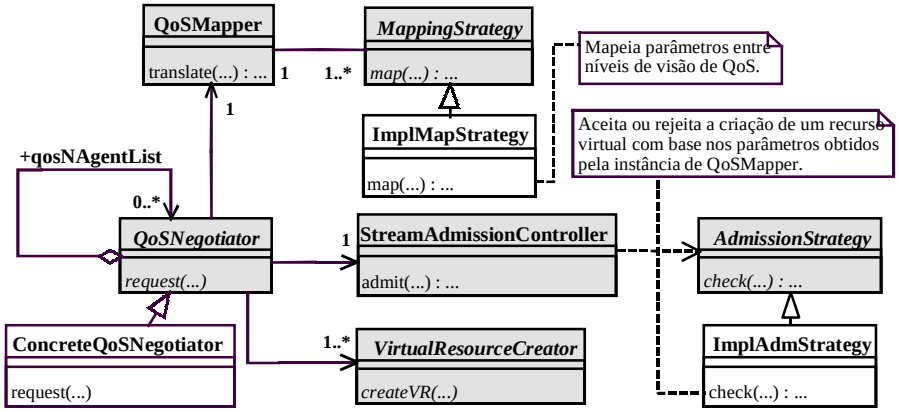

Figura 7: estrutura para o estabelecimento da orquestração de recursos.

O pattern Strategy é utilizado para representar o relacionamento entre os mecanismos de mapeamento e controle de admissão e as respectivas políticas de provisão de QoS (representadas por *MappingStrategy* e *AdmissionStrategy*). O uso desse pattern permite, por exemplo, que uma estratégia de admissão correntemente associada a um escalonador possa ser substituída por outra que gere um maior ganho estatístico de utilização do recurso, contanto que a estratégia de escalonamento associada ao escalonador seja compatível com a nova estratégia de admissão. O pattern Strategy possibilita também que políticas de provisão de QoS possam ser alteradas sem que a visão de QoS de um usuário humano com relação ao sistema precise ser também alterada, bastando para isso adotar uma nova estratégia de mapeamento condizente com as novas políticas.

Durante a manutenção de um contrato de serviço, mecanismos de monitorização associados a cada fluxo geram periodicamente registros de medições relativas à real QoS sendo oferecida aos usuários, através da atualização de parâmetros estatísticos internos ao

sistema, relacionados à categoria do serviço sendo oferecido. Posteriormente, esses mecanismos enviam alertas ao mecanismo de sintonização da QoS, que ao recebê-los, responsabiliza-se por reavaliar os parâmetros internos do sistema, podendo, em caso de detecção de uma violação de contrato: (i) repassar os alertas aos usuários, (ii) reconfigurar os mecanismos de escalonamento, ou (iii) acionar o mecanismo de negociação, para que este renegocie o contrato de serviço, dependendo do grau de depreciação da QoS anteriormente negociada.

Os mecanismos de monitorização e de sintonização também são recursivamente distribuídos. Como exemplo, na Figura 6, o mecanismo de monitorização de um fluxo A→B envolve a monitorização conjunta de A→X, X→Y, Y→Z e Z→B. Violações sucessivas de contrato no fluxo X→Y, por exemplo, podem refletir na QoS atribuída ao fluxo A→B. O mecanismo de sintonização deve responsabilizar-se então por tentar compensar essas violações, aumentando, por exemplo, a QoS associada ao fluxo Y→Z.

Na Figura 8, as classes QoSTuner e StreamMonitor representam os mecanismos de sintonização e monitorização, respectivamente. O mecanismo de sintonização se vale do pattern Chain of Responsibility para representar a sua independência (assim como a dos mecanismos de monitorização) quanto a implementações centralizadas ou distribuídas. O pattern Facade é também utilizado, através da classe QoSTuner, para definir uma interface única de sinalização de alertas aos usuários do sistema. O pattern Strategy é usado para representar o relacionamento entre os mecanismos de monitorização e as respectivas estratégias (representadas por *MonitoringStrategy*).

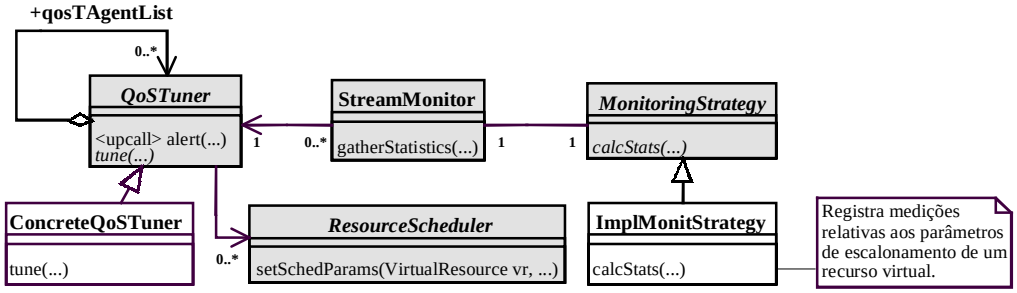

Figura 8: estrutura para a manutenção da orquestração de recursos.

4. Cenário de Utilização do Framework

Visando validar o framework em pelo menos algumas das situações possíveis de aplicação do mesmo, tem sido desenvolvido, no Laboratório TeleMídia da PUC-Rio, um protótipo de um sistema de transmissão de mídias contínuas. O ambiente de implementação utilizado consta de: (i) duas estações SPARC Ultra 1 e uma SPARC Station 5, todas com sistema operacional SOLARIS 2.5 e placas ATM Efficient 155 Mbps, e (ii) um comutador ATM IBM 8260 interligando as três estações. A Figura 9 ilustra, de modo simplificado, a arquitetura do sistema. Os seguintes elementos do sistema (hachurados na figura) correspondem a utilizações distintas do framework: (i) a biblioteca de escalonamento de threads, (ii) o controlador de processos, e (iii) o protocolo de negociação da QoS.

4.1 Biblioteca de Escalonamento de Threads

A biblioteca de escalonamento de threads permite a uma aplicação Java especificar a QoS de processamento de suas threads, e oferece mecanismos de escalonamento que permitem a provisão da QoS desejada. A biblioteca permite também que o desenvolvedor de uma

aplicação se valha de várias estratégias concorrentes de escalonamento de threads na mesma aplicação, e que defina conjuntos adequados de parâmetros que descrevam a QoS das threads da aplicação, segundo as estratégias definidas.

O framework para parametrização de serviços (Figura 2) é aplicado diretamente na estruturação de parâmetros da biblioteca. Já o mecanismo de escalonamento no qual se baseia a biblioteca corresponde à utilização da estrutura de escalonamento definida no pattern de compartilhamento de recursos (Figura 4). Este esquema de escalonamento, que estende o mecanismo de escalonamento de threads em Java proposto em [31], permite o escalonamento hierárquico de threads de maneira simples e flexível.

4.2 Controlador de Processos

Para uma aplicação que requer garantias maiores de processamento, a biblioteca de escalonamento de threads é insuficiente. Isto porque a plataforma SOLARIS vale-se de um esquema próprio de escalonamento hierárquico, onde em um primeiro nível encontram-se os *processos leves* (LWPs), associados às aplicações, e em um segundo nível encontram-se as threads, que são escalonadas pelos LWPs, que por sua vez são escalonados pelo sistema. Para que se possa garantir efetivamente o processamento de um conjunto de threads segundo os requisitos por ela impostos, é necessário, primeiramente, que as LWPs associadas a esse conjunto de threads também tenham garantias de escalonamento.

O controlador de processos foi implementado com o objetivo de permitir a gerência de escalonamento de LWPs. Esta gerência é feita de modo semelhante a da biblioteca de escalonamento de threads, correspondendo também à utilização da estrutura de escalonamento definida no pattern de compartilhamento de recursos. Este esquema de escalonamento, que estende o mecanismo de escalonamento de processos proposto em [32], permite o escalonamento hierárquico de LWPs. Em conjunto com a biblioteca de escalonamento de threads, o controlador de processos oferece às aplicações Java garantias efetivas de escalonamento de suas threads.

4.3 Protocolo de Negociação da QoS

O protocolo de negociação da QoS, em sua versão inicial, provê o mapeamento de parâmetros relativos a uma categoria genérica de serviços multimídia (como retardo, jitter e vazão) em parâmetros de contrato de tráfego utilizados nas redes ATM. O protocolo inclui também um esquema primitivo de negociação que, assim como o mapeamento, é definido com base na estrutura para estabelecimento da orquestração de recursos definida no pattern de orquestração de recursos (Figura 7).

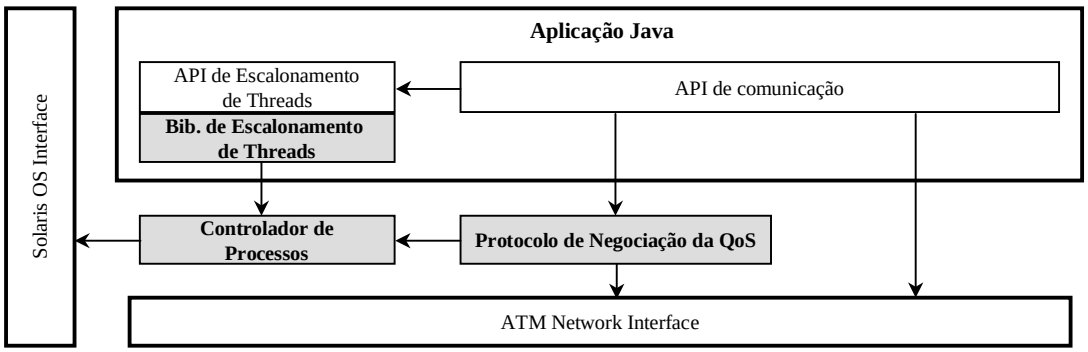

Figura 9: arquitetura do sistema de transmissão de vídeo.

5. Conclusões

Os tópicos relacionados à provisão de QoS em sistemas multimídia têm sido, em sua maioria, parcialmente tratados na literatura. A maioria das abordagens apresenta pelo menos uma das seguintes características: (i) elas são restritas a subsistemas específicos, notadamente os sistemas operacionais e de comunicação; (ii) elas tratam a questão da integração da QoS sob o ponto de vista de interfaces somente; e (iii) elas não facilitam a introdução de novos serviços. Neste artigo, foi apresentada uma abordagem que permite o tratamento da questão da QoS sob uma visão mais granular, abrangendo desde a interface com o usuário humano até a definição de mecanismos de provisão de QoS agindo sobre componentes das aplicações e dos protocolos de comunicação. A abordagem proposta consiste na definição de um framework cujo domínio de aplicação é suficientemente genérico para incluir quaisquer ambientes de processamento e comunicação, desde subsistemas específicos até plataformas para sistemas distribuídos. Com estas características, o framework possibilita: (i) a construção de sistemas configuráveis no que se refere à introdução de novos serviços, e (ii) facilita a implementação dos mecanismos de compartilhamento e orquestração de recursos.

Quanto ao estado de desenvolvimento atual do presente trabalho, ele tem sido validado basicamente através da implementação do sistema de transmissão de mídias contínuas apresentado na seção 4. Neste momento, está sendo implementada a parte referente ao protocolo de negociação da QoS. O ponto-chave desta implementação é a definição de múltiplas estratégias de mapeamento entre as requisições recebidas pelo protocolo e os parâmetros de contrato de tráfego utilizados nas redes ATM, de acordo com as diferentes categorias de serviço a serem providas pelo sistema, o que não é uma tarefa trivial.

Há três importantes áreas de trabalhos futuros necessárias ao amadurecimento deste trabalho. Primeiramente, a parte do framework relativa à sintonização da QoS e monitorização de fluxos deve ser validada através de uma implementação. Em segundo lugar, o framework deve ser reavaliado com relação à questão da comunicação multicast. O presente trabalho insere-se em um projeto que envolve também a definição de um framework para provisão de serviço de multicast [9], logo o processo de reavaliação deve consistir fundamentalmente na integração entre os dois frameworks.

Finalmente, o framework deve ser acrescido de um modelo que represente o processo de configuração do sistema. Embora os pontos de flexibilização do framework lhe concedam a característica altamente desejável de configurabilidade, tais pontos não são totalmente independentes entre si, o que dificulta a utilização do framework por um projetista de sistemas. Em um exemplo típico, quando insere-se em um sistema uma nova categoria de serviço, novas estratégias de mapeamento, admissão, escalonamento e monitorização possivelmente terão que ser também introduzidas.

6. Bibliografia

[1] SOARES, L.F.G., LEMOS, G., COLCHER, S. *Redes de Computadores: das LANs, MANs e WANs às Redes ATM*. Campus, 1995. Segunda edição.

[2] LU, G. *Communication and Computing for Distributed Multimedia Systems*. Artech House, 1996.

[3] AURRECOECHEA, C., CAMPBELL, A., HAUW, L. "A Review of QoS Architectures". *In: ACM Multimedia Systems Journal*. Novembro de 1995.

[4] HUTCHISON, D., COULSON, G., CAMPBELL, A., BLAIR, G. S. "Quality of Service Management in Distributed Systems". *Lancaster University Report Number MPG-94-02*. 1994.

[5] LAZAR, A. A., LIM, K. S., MARCONCINI, F. "Binding Model: Motivation and Description". *http://www.ctr.columbia.edu/comet/xbind/xbind/html*. Novembro de 1995.

[6] COLCHER, S., SOARES, L. F. G. "Modelo de Referência Unificado para Arquitetura de Protocolos e Programação de Aplicações Multimídia". *In: Anais/16º Simpósio Brasileiro de Redes de Computadores (SBRC'98)*. Maio de 1998.

[7] GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1995.

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

[9] RODRIGUES, M. A. A., SOARES, L. F. G. "Framework Para Provisão de Serviço de Multicast em Ambientes Genéricos de Comunicação". *In: 17º Simpósio Brasileiro de Redes de Computadores (SBRC'99)*. Maio de 1999.

[10] GOYAL, P., GUO, X., VIN, H. M. "A Hierarchical CPU Scheduler for Multimedia Operating Systems". *In: Operating Systems Design and Implementation (OSDI'96)*. Outubro de 1996.

[11] FORD, B., SAI, S. **"**CPU Inheritance Scheduling". *In: Operating Systems Design and Implementation (OSDI'96)*. Outubro de 1996.

[12] ISO/IEC. "DIS 10746-1 - Reference Model of Open Distributed Processing - Part 1: Overview". *Output from the editing meeting in Helsink (Finland)*. Maio de 1995.

[13] OMG. *The Common Object Request Broker:Architecture and Specification - Revision 2.0*. 1995.

[14] ISO/IEC. "DIS 7498-1 - Open Systems Interconnection Reference Model - Part 1: Basic Reference Model". *International Standard*, 1984.

[15] SILBERSCHATZ, A., GALVIN, P. B. *Operating System Concepts*. Addison Wesley, 1995.

[16] ATM FORUM. *ATM User-Network Interface Specification Version 3.0*. 1993.

[17] ISO/IEC. "JTC1/SC 21 N9309 - QoS: Basic Framework". *Draft Technical Report*. 1995.

[18] WADDINGTON, D. G., COULSON, G., HUTCHISON, D. "Specifying QoS for Multimedia Communications within Distributed Programming Environments". *Lancaster University Report Number MPG-96-34*.1996.

[19] CAMPBELL, A. "A Quality of Service Architecture". *Draft Technical Report*. 1996.

[20] RATIONAL SOFTWARE CORPORATION. *UML Notation Guide*. Setembro de 1997.

[21] LIU, C. L., LAYLAND, J. W. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *In: Journal of the ACM*. Janeiro de 1973.

[22] GALLMEISTER, B. *POSIX.4: Programming for the Real World*. O'Reilly & Associates, 1995.

[23] BRADEN, R., CLARK, D., SHENKER, S. "Integrated Services In The Internet Architecture: Overview". *IETF Request For Comments (RFC) 1633*. 1994.

[24] NAHRSTEDT, K. SMITH, J. M. "End-to-End QoS Guarantees: Lessons Learned from OMEGA". *Report No.UIUCDCS-R-96-1957, University of Illinois*. Maio de 1996.

[25] SCHULZRINNE, H., CASNER, S. "RTP: A Transport Protocol for Real-Time Applications". *IETF Request For Comments (RFC) 1889*. 1996.

[26] ZHANG. L., et al. "RSVP Functional Specification". *IETF Working Draft*. 1995.

[27] TOPOLCIC, C. "Experimental Internet Stream Protocol: Version 2 (ST-II)". *IETF Request For Comments (RFC) 1190*. 1990.

[28] MATHY, L., BONAVENTURE, O. "QoS Negotiation for Multicast Communications". *In: Lecture Notes in Computer Science no. 882, Int'l COST237 Workshop*. Novembro de 1994.

[29] HUARD, J. F., LAZAR, A. A. "On End-to-End QoS Mapping". *In: IFIP Fifth International Workshop on Quality of Service (IWQoS'97)*. Maio de 1997.

[30] KNIGHTLY, E., ZHANG, H. "D-BIND: An Accurate Traffic Model for Providing QoS Guarantees to VBR Traffic". *In: IEEE/ACM Transactions on Networking*. Abril de 1997.

[31] OAKS, S., WONG, H. *Java Threads*. O'Reilly & Associates, 1997.

[32] CHU, H., NAHRSTEDT, K. "A Soft Real Time Scheduling Server in UNIX Operating System". *In: European Workshop on Interactive Multimedia Systems and Telecommunication Services (IDMS'97)*. Setembro de 1997.