Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

Uma Arquitetura Baseada em Sistemas de Agentes para Suporte de Qualidade de Serviço em Aplicações Multimídia Distribuídas

Luiz Affonso Henderson Guedes de Oliveira

Prof. Eleri Cardozo, Ph.D.
Orientador

Campinas, SP - Brasil
Março de 1999

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

Uma Arquitetura Baseada em Sistemas de Agentes para Suporte de Qualidade de Serviço em Aplicações Multimídia Distribuídas

Luiz Affonso Henderson Guedes de Oliveira
Orientador: **Prof. Eleri Cardozo, Ph.D.**

Tese de doutorado apresentada à Faculdade de Engenharia Elétrica e de Computação como parte dos requisitos para obtenção do título de Doutor em Engenharia Elétrica.
Área de concentração: Automação Industrial

Campinas, SP - Brasil
Março de 1999

FICHA CATALOGRÁFICA ELABORADA PELA
BIBLIOTECA DA ÁREA DE ENGENHARIA - BAE - UNICAMP

OL4a

Oliveira, Luiz Affonso Henderson Guedes de
Uma arquitetura baseada em sistemas de agentes para suporte de qualidade de serviço em aplicações multimídia distribuídas. / Luiz Affonso Henderson Guedes de Oliveira.--Campinas, SP: [s.n.], 1999.

Orientador: Eleri Cardozo .
Tese (doutorado) - Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Software - Desenvolvimento. 2. Processamento eletrônico de dados – Processamento distribuido . I. Cardozo, Eleri. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Faculdade de Engenharia Elétrica e de Computação
Universidade Estadual de Campinas

Banca Examinadora

Prof. Eleri Cardozo, PhD - Orientador

Prof. Luís Carlos Trevelin, Dr - Membro Externo

Prof. Marcelo K. Zuffo, Dr - Membro Externo

Prof. Ivan L. M. Ricarte, PhD - Membro Interno

Prof. Maurício F. Magalhães, Dr - Membro Interno

Prof. Fernando Gomide, PhD - Membro Suplente

Prof. Ricardo R. Gudwin, Dr - Membro Suplente

Para minha esposa Inez Sigma e minha filha Beatriz

Agradecimentos

Ao meu orientador, Prof. Eleri Cardozo, pela forma atenciosa e competente como conduziu este trabalho.

Aos amigos Luís Fernando Faina, Paulo César Oliveira e Rodrigo Prado pelas discussões, sugestões e contribuições a este trabalho. Também agradeço a eles por me provarem que trabalhar em equipe além de mais produtivo é bem mais agradável.

Aos amigos e colegas do Laboratório de Computação e Automação (LCA), que tornaram o ambiente de trabalho bastante agradável. Dentre eles: Alexandre Pinto, André Coelho, Benito Giordani, Cláudio Rodrigues, Douglas Araújo, Eduardo Oliveira, Eurípedes dos Santos, Jussara Fardin, Luiz Gonzaga Jr. e Naur Janzantti Jr.

Ao Departamento de Engenharia de Computação e Automação Industrial (DCA) da Faculdade de Engenharia Elétrica e de Computação (FEEC) da Unicamp por ter proporcionado a infra-estrutura necessária à realização deste trabalho.

Ao Departamento de Engenharia Elétrica da Universidade Federal do Pará, por permitir meu afastamento para realização do doutorado, e à CAPES pela concessão da bolsa de estudo.

"A sabedoria de um povo não consiste no grau de conhecimento que ele detém, mas sim na forma como esses conhecimentos são utilizados."

Lista de Trabalhos Publicados

- L. A. Guedes and P.C. Oliveira and L.F. Faina and E. Cardozo, "An Agent-based Approach for Supporting Quality of Service in Distributed Multimedia Systems", Computer Communications Journal , Vol. 21 No. 14, pg. 1269-1278, September 1998.
- R. C. M. Prado and L. A. Guedes and L.F. Faina and E. Cardozo, "ODP Channel: an Open Mechanism for Supporting Media Flows in Distributed Environments", XXIV Conferencia Latinoamericana de Informatica (CLEI'98), Quito, Equador, pg. 111-122, October 21-23, 1998.
- P. Oliveira, L. A. Guedes e E. Cardozo, "Monitoramento de Qualidade de Serviço em Sistemas Multimídia Distribuídos: um Estudo de Caso para Sistemas Baseados em Agentes Móveis", Anais do XVI Simpósio Brasileiro de Redes de Computadores, SBRC 98, pages 481-500, Rio de Janeiro, Brasil, Maio 1998.
- L. A. Guedes and P.C. Oliveira and L.F. Faina and E. Cardozo "QoS Agency: An Agent-based Architecture for Supporting Quality of Service in Distributed Multimedia Systems" , IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking (PROMSMmNet'97), Santiago, Chile, pg. 204-212, November, 1997.
- R. C. M. Prado and L. A. Guedes and L.F. Faina and E. Cardozo, "Implementação de Serviços Multimídia em CORBA", XXIII Conferencia Latinoamericana de Informatica (CLEI'97), Val Paraiso, Chile, Novembro, 1997.
- L. A. Guedes e E. Cardozo, "Especificação de um Protocolo para Negociação de Qualidade de Serviço em Sistemas Multumídia", Anais do XV Simpósio Brasileiro de Redes de Computadores, SBRC 97, pages 101-117, São Carlos, Brasil, Maio 1997.
- L. A. Guedes, P. Oliveira, and E. Cardozo, "An Agent-Based Approach for Quality of Service Negotiation and Management in Distributed Multimedia Systems", Proceedings of First International Workshop in Mobile Agents, MA97, Lecture Notes in Computer Science, vol. 1219, pages 1-12, Berlin, Germany, April 1997.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Acrônimos

AAL: *ATM Adaptation Layer*

ABR: *Available Bit Rate*

ADPCM: *Adaptive Delta Pulse Code Modulation*

ATM: *Asynchronous Transfer Mode*

ATP: *Agent Transfer Protocol*

BER: *Bit Error Rate*

B-ISDN: *Broadband-Integrated Services Digital Network*

CBO: *Continuous Bit Oriented*

CBR: *Constant Bit Rate*

CCITT: *Consultative Committee for International Telegraph and Telephony*

CMTP: *Continuos Media Transport Protocol*

CORBA: *Common Object Request Broker Architecture*

CSCW: *Computer Supported Cooperative Working*

DCT: *Discrete Cosine Transform*

DMS: *Distributed Multimedia Systems*

FDDI: *Fiber Distributed Data Interface*

HDTV: *High Definition Television*

HeiTS: *Heidelberg Transport System*

IDL: *Interface Definition Language*

IP: *Internet Protocol*

IPv6: *Internet Protocol*, versão 6

ISDN: *Integrated Services Digital Network*

ISO: *International Organization for Standardization*

ITU: *International Telecommunication Union*

JPEG: *Joint Photographic Expert Group*

LAN: *Local Area Network*

LANE: *LAN Emulation*

MAC: *Medium Acess Control*

MAF: *Mobile Agents Facilities*

METS: *Multimedia Enhanced Transport System*

MPEG: *Moving Pictures Expert Group*

MSC: *Message Sequence Chart*

nrt-VBR: *non real-time Variable Bit Rate*

ODP: *Reference Model for Open Distributed Processing*

OMA: *Object Management Architecture*

OMG: *Object Manegement Group*

ORB: *Object Request Broker*

PCM: *Pulse Code Modulation*

PER: *Packet Error Rate*

QoS: *Quality of Service*

RCAP: *Real-time Channel Administration Protocol*

RMTP: *Real-time Message Transport Protocol*

RSVP: *Resource Reservation Protocol*

RTAP: *Real-Time Application Protocol*

RTCP: *Real Time Control Protocol*

RTIP: *Real Time Internet Protocol*

RTNP: *Real-Time Network Protocol*

RTP: *Real Time Protocol*

rt-VBR: *real-time Variable Bit Rate*

SBA: Sistemas Baseados em Agentes

SCMA-CD: *Carrier Sense Multiple Access - Collision Detection*

SDL: *Specification Description Language*

SMA: Sistemas Multi-Agentes

SMD: Sistemas Multimídia Distribuídos

ST-II: *Stream Protocol*, versão 2

TCP: *Transmission Control Protocol*

TINA-C: *Telecommunication Information Networking Architecture- Consortium*

TPX: *Transport Protocol with eXtension*

UBR: *Unspecified Bit Rate*

UDP: *User Data Protocol*

VBR: *Variable Bit Rate*

WAN: *Wide Area Network*

XRM: *Extended Integrated Reference Model*

XTP: *Xpress Transport Protocol*

Resumo

Oliveira, Luiz Affonso Henderson Guedes de - Uma Arquitetura Baseada em Sistemas de Agentes para Suporte de Qualidade de Serviço em Aplicações Multimídia Distribuídas.

Campinas: DCA/FEEC/UNICAMP, 1999. (Tese de Doutorado)

Sistemas multimídia têm emergido como uma grande área de pesquisa e desenvolvimento, devido à ampla possibilidade de aplicações. Em decorrência da natureza dinâmica de tais aplicações, a noção de qualidade de serviço (QoS) se tornou uma característica chave em tais sistemas. QoS intuitivamente tenta expressar quão satisfatórios são os serviços fornecidos por uma determinada aplicação. Apesar de seu caráter fortemente subjetivo, QoS pode ser expressa a partir de parâmetros bem definidos, tais como: atraso, jitter e perda de pacotes de dados.

Via de regra, QoS é estabelecida através de negociação entre usuários e provedores de serviços. O processo de suporte de QoS, que envolve negociação e gerência, é relativamente simples caso os recursos sejam gerenciados por uma entidade centralizada (sistema operacional, por exemplo) ou por um conjunto de entidades que empregam protocolos simples de negociação e gerência. Infelizmente, em sistemas multimídia distribuídos a negociação e gerência de recursos é uma atividade não trivial, uma vez que os recursos existentes são bastante diversificados, dispersos e mantidos por diferentes entidades.

Para minimizar essas dificuldades, nesta tese se objetivou basicamente a proposição de uma arquitetura para o suporte de QoS em sistemas multimídia distribuídos de modo a privilegiar na sua concepção características como encapsulação, flexibilidade e extensibilidade.

Diante da relevância desse problema, inicialmente neste trabalho procurou-se caracterizar os requisitos de QoS para os sistemas multimídia distribuídos. Foi proposto um modelo baseado em sistema de agentes fixos e móveis para o suporte de QoS e sua respectiva especificação nas linguagem formais MSC e SDL. É proposta também uma arquitetura modular para desenvolvimento de aplicações multimídia, onde o suporte de QoS é um de seus módulos. Para se avaliar a viabilidade do modelo proposto foi implementado um protótipo, o qual se utilizou em monitoramento e adaptação de QoS em aplicações do tipo teleconferência. Finalmente são tecidos comentários sobre a vialibilidade de tal proposta e sugeridos alguns desenvolvimentos futuros.

Palavras-chave: Sistemas Multimídia, Qualidade de Serviço, Sistemas Baseados em Agentes e CORBA.

Abstract

Oliveira, Luiz Affonso Henderson Guedes de - An Agent-based System Architecture for Supporting Distributed Multimedia Applications.

Campinas: DCA/FEEC/UNICAMP, 1999. (Doctorate Thesis)

Distributed multimedia systems (DMS) have emerged as an important area of research and development due to the wide range of applications that can benefit from this area. In this field, the notion of *quality of service* (QoS) is a key concept. Intuitively, QoS states how satisfactory the services offered by an application are. Although strongly subjective, QoS can be stated in terms of well defined parameters such as delay, jitter and package error rate.

Usually, QoS is negotiated between users and service providers. The negotiation process can be simple if the environment is homogeneous, or complex if the environment is heterogeneous. Since DMS runs frequently on heterogeneous environments, the negotiation and management of QoS-related parameters are non trivial. The complexity of this problem motivated us in proposing a new approach for incorporating quality of service into distributed systems.

In order to contribute to this research area, this thesis proposes an open architecture for QoS negotiation and management for DMS. This architecture is based on the Agent paradigm in order to favor encapsulation, extensibility and flexibility. In the first step this work tries to characterize the requirements necessary to incorporated QoS into distributed applications. Next, an Agent-based architecture is presented based on the requirements previously investigated. In the sequence a protocol for QoS negotiation and management among the architecture components (agents) is detailed, including its formal specification. Finally, an application in the domain of QoS monitoring and adaptation is developed following the proposed architecture.

With the results of the implementation, comments, recommendations and future work directions are also presented in the thesis report.

Keywords: Multimedia Systems, Quality of Service, Agent-based Systems and CORBA.

Capítulo 1

Introdução

Tradicionalmente, sistemas computacionais têm se dedicado exclusivamente a informações textuais e numéricas. Assim, as tecnologias de processamento de informação têm desenvolvido basicamente suporte para transmissão e manipulação para tais tipos de dados. Mais recentemente, porém, temos presenciado um enorme interesse por novos tipos de dados, tais como animação gráfica, áudio e vídeo. A estes sistemas computacionais que suportam vários tipos de dados (mídias), convencionou-se denominar sistemas multimídia.

Assim, sistemas multimídia se caracterizam pela capacidade de suportar vários tipos de mídias, desde as mais simples como textos e gráficos, até as mais ricas, como animação, áudio e vídeo. Entretanto, mídias operando isoladamente não são suficientes para caracterizar um ambiente multimídia. Para tanto, é necessário que várias formas de mídias diferentes estejam integradas em um único ambiente. Portanto, podemos definir um sistema multimídia como sendo o que permite ao usuário final criar, armazenar e comunicar uma variedade de formas de informação de uma maneira integrada.

Na essência, os problemas em sistemas multimídia estão relacionados à gerência de informação demandada pela integração de várias formas de mídias em uma infra-estrutura única de computação e comunicação. Portanto, há dois aspectos relevantes em sistemas multimídia, que são: a variedade de tipos de mídias disponíveis e a capacidade de integrá-los.

Quanto à classificação, sistemas multimídia podem ser divididos em duas classes básicas. Uma diz respeito aos sistemas que operam em uma única máquina (sistemas multimídia centralizados) e a outra concerne aos sistemas que operam em um ambiente distribuído (sistemas multimídia distribuídos) [72]. A grande maioria dos desenvolvimentos em multimídia tem sido orientada para sistemas centralizados.

O termo sistema multimídia distribuído é introduzido para descrever o caso geral onde vários ambientes multimídia centralizados estão interconectados via uma ou mais redes de comunicação. Além disso, uma aplicação multimídia distribuída é definida como sendo a que executa sobre um sistema multimídia distribuído.

A combinação de computação multimídia e sistemas distribuídos oferece grandes perspectivas de novas aplicações, dentre elas podemos citar [70]: colaboração científica, sistema de conferência e aprendizagem à distância.

As dificuldades acarretadas pela necessidade de gerência de diversas mídias em um sistema multimídia distribuído, introduziram vários problemas de pesquisa ainda não resolvidos completamente [67]. Por outro lado, as soluções relacionadas aos problemas de sistemas

multimídia centralizados são bem estabelecidas.

Nesse trabalho, estamos interessados essencialmente em enfocar os sistemas multimídia distribuídos.

1.1 Tipos de Mídias

As mídias podem ser classificadas em estáticas e contínuas. O termo "mídia estática" é utilizado para referenciar as mídias que não possuem dimensão temporal. Por outro lado, as mídias contínuas, também denominadas fluxo, possuem relações temporais, implicando que deverão ser apresentadas com taxas apropriadas em instantes de tempo bem definidos. A seguir são listados alguns componentes das duas classes de mídias.

- Mídias Estáticas:

  arquivos textuais;

  arquivos numéricos;

  gráficos;

  fotos digitalizadas.

- Mídias Contínuas:

  áudio;

  vídeo;

  animação.

A integração de mídias contínuas, devido às suas características temporais, representa o maior desafio para a computação multimídia [67].

A partir do ponto de vista de modelagem, a integração de tais tipos de mídias requer o desenvolvimento de novas formas de abstrações de comunicação e armazenamento, de modo a capturarem adequadamente o conceito de informação fluindo no tempo [72] [57]. Além disso, essas mídias também vêm requerendo o desenvolvimento de novas interfaces de apresentação ao usuário [22].

Mídias contínuas, tais como áudio e vídeo, podem ser representadas tanto na forma digital como analógica. Os primeiros sistemas multimídia manipulavam mídias contínuas de forma analógica. Nesses sistemas, áudio e vídeo eram armazenados em dispositivos analógicos os quais eram controlados por computador, sendo então transmitidos em forma analógica para apresentação. Com o avanço tecnológico, tornou-se consenso a representação digital de todos os tipos de mídias.

A partir da perspectiva computacional é mais razoável a utilização em forma digital de todas as formas de mídias, uma vez que isto permite uma integração completa dos tipos de mídias existentes, além de permitir inserção de novos tipos de mídias de forma relativamente mais fácil.

1.2 Áreas de Aplicação de Sistemas Multimídia

Há muitas áreas que podem ser enriquecidas com a introdução de sistemas multimídia, além de outras onde novas aplicações podem se tornar viáveis. A seguir são analisadas brevemente algumas dessas áreas promissoras.

1.2.1 Aplicações em Escritórios

Esta é seguramente uma das áreas que podem obter enormes benefícios com a introdução de serviços multimídia. Podemos citar alguns como serviço de correio eletrônico multimídia, de documentos multimídia e sistemas para apoio à tomada de decisões.

1.2.2 Aplicações em Educação

A utilização de recursos multimídia, tais como sons e imagens em movimento, estão provocando o desenvolvimento de novas metodologias de aprendizagem, centrada agora na forte característica de interatividade fornecida por esta nova tecnologia. Outro aspecto importante diz respeito aos sistemas de aprendizagem a distância, nos quais usuários remotos acessam bases de dados multimídia de forma interativa.

1.2.3 Aplicações em Saúde

Serviços na área de saúde consistem basicamente de vários equipamentos que produzem grande volume de informações em diversas formas, tais como gráfica e sonora. Atualmente, estas informações não estão disponíveis aos profissionais da área de saúde de forma integrada.

Sistemas multimídia distribuídos poderão num futuro próximo fornecer o suporte necessário para construção de sistemas médicos com capacidade de reter informações em várias formas, permitindo o acesso a estas de maneira rápida e fácil a partir de vários pontos.

1.2.4 Trabalho Cooperativo Suportado por Computador (CSCW)

Um dos aspectos que atrai bastante atenção em relação à área CSCW é a grande variedade de aplicações propostas, que vão desde editoração cooperativa até sistemas de teleconferências interativas.

Trabalho Cooperativo Suportado por Computador é sem dúvida uma das áreas mais propícias para incorporar técnicas multimídia. Um bom exemplo são os ambientes interativos para comunicação audiovisual entre usuários remotos, os quais causaram grandes impactos em áreas com automação de escritório, negócio, saúde e educação.

1.3 Demandas das Mídias Digitais

Com o objetivo de se avaliar as demandas de recursos de transmissão de informação requeridas pelos diversos tipos de mídias digitais, são apresentadas a seguir algumas estimativas típicas para armazenar diferentes tipos de mídias [19] [9].

- Texto: 2 KBytes por página.
- Imagens simples não comprimidas, como gráficos mapeados em bits, fotos e faxes: 64 KBytes por imagem.
- Imagem colorida ou detalhada não comprimida: 7,5 MByte por imagem.
- Uma hora de áudio mono não compactado, amostrado a 8 KHz, e codificado com 8 bits: 2,88 MBytes.
- Uma hora de música mono, amostrada a 44,1 KHz, com 16 bits de codificação: 317,52 MBytes.
- Um minuto de animação não compactada, com definição de 320 por 240 pixels, 256 cores por pixel e 16 quadros por segundo: 147,45 MBytes.
- Um minuto de vídeo digitalizado não compactado, com definição de 640 por 480 pixels, 256 cores por pixel e 30 quadros por segundo: 1,1 GBytes.

Nota-se, a partir desses valores, que à medida que se incorpora mídias mais sofisticadas, o volume de dados manipulados aumenta rapidamente, demandando sistemas com maior capacidade de armazenamento e comunicação. Para minorar estes problemas é usual se utilizar algoritmos de compressão de dados, tanto na armazenagem quanto na transmissão via rede. Dependendo do caso, estes algoritmos podem obter uma taxa de compressão de até 200 vezes [72] [57] para vídeo, viabilizando aplicações multimídia no atual estágio da indústria de informática.

O conceito de qualidade de serviço é fundamental em computação multimídia. Exemplificando, se um vídeo for exibido com um baixo número de quadros por segundo, pode-se perder a persepção de movimento contínuo, o que acarreta na perda de qualidade do sistema. No exemplo, valores típicos para exibição de vídeo estão entre 24 e 30 quadros por segundo, embora em algumas aplicações é possível se utilizar menos quadros por segundos sem deteriorar a qualidade da exibição. Já para animação, 15 a 19 quadros por segundo é satisfatório [19].

A introdução da idéia de qualidade de serviço é muito importante para se caracterizar o nível dos serviços oferecidos pelos sistemas multimídia. Entretanto, sua caracterização não é simples, uma vez que, apesar de poder ser avaliada por parâmetros quantitativos, qualidade de serviço tem uma natureza essencialmente subjetiva.

1.4 Requisitos de Sistemas Multimídia

Uma característica fundamental de sistemas multimídia é que eles incorporam mídias contínuas, tais como: áudio, vídeo e animação gráfica. O uso de tais mídias em sistemas computacionais distribuídos implica em capturar, processar, transmitir e exibir dados continuamente por longos períodos de tempo. Além disso, devido às suas características, requer-se a garantia de que os dados possam ser entregues e exibidos de forma cadenciada no tempo. Para suportar esta característica isócrona das mídias contínuas, sistemas multimídia requerem desenvolvimento de novos modelos de sistemas de processamento e de comunicação.

A partir de uma perspectiva temporal, aplicações multimídia podem ser consideradas como uma subclasse de uma grande classe de aplicações de tempo real, a qual inclui também computação distribuída de tempo real, sistemas de controle remoto e aplicações interativas.

Porém, apesar de possuírem características temporais, sistemas multimídia não podem ser caracterizados propriamente como sistemas de tempo real crítico (*hard real time*), uma vez que toleram ocasionais atrasos provocados por latências de armazenamento, comunicação e transmissão, sem que ocorra resultados catastróficos ao sistema.

No outro extremo, em determinadas situações, sistemas multimídia são descritos apropriadamente pela semântica *melhor-nunca-do-que-tarde* (um exemplo seria o caso em que um quadro de vídeo se encontra tão atrasado que é melhor não exibi-lo). Assim, sistemas multimídia não podem ser caracterizados completamente como sistemas de tempo real flexível (*soft real time*).

Em vista disso, assumimos aqui que sistemas multimídia sejam caracterizados mais adequadamente como sistemas de tempo real orientados à qualidade de serviço, sendo os seus requisitos temporais de desempenho descritos a partir de índices de qualidade de serviço, os quais o sistema procura atender. Desse modo, a noção de qualidade de serviço (QoS) se torna fundamental, o que torna indispensável caracterizá-la da forma mais adequada possível.

QoS não é uma concepção nova, uma vez que vem sendo empregada no domínio de redes de computadores para especificar um conjunto de parâmetros tipicamente relacionados às conexões de transporte. Via de regra, QoS é estabelecida através de negociação entre usuários e provedores de serviços.

O processo de negociação e gerência de QoS é relativamente simples se os recursos são gerenciados por uma entidade centralizada (sistema operacional, por exemplo). Porém, em sistemas multimídia distribuídos a negociação e gerência de QoS é uma tarefa não trivial, uma vez que os recursos existentes são bastante diversificados, dispersos e mantidos por diferentes entidades. Tudo isto leva à necessidade de implementação de políticas de gerência de recursos orientadas a reserva, configuração de serviços, monitoramento da qualidade de serviço, otimização de recursos e padronização, sendo flexibilidade e capacidade de adaptação características desejáveis.

1.5 Objetivos e Contribuições do Trabalho

O principal objetivo desta tese é estudar os sistemas multimídia distribuídos quanto ao aspecto de qualidade de serviço e propor e validar um modelo para negociação e gerência de qualidade de serviço nestes sistemas.

A nossa proposta é um modelo para negociação e gerência de QoS baseado em sistemas de agentes, que incorpora tanto agentes fixos quanto agentes móveis [24]. Esse modelo visa privilegiar características como autonomia e flexibilidade no processo de negociação e gerência da qualidade de serviço em aplicações multimídia distribuídas, além de princípios de modelagem como: encapsulação, distribuição e simplicidade.

Como principais contribições deste trabalho podemos citar os seguintes pontos:

- Caracterização dos sistemas multimídia quanto aos seus requisitos de qualidade de serviço.

- Proposição de um modelo baseado em sistemas de agentes para o suporte à qualidade de serviço em aplicações multimídia.
- A especificação via as linguagens formais MSC e SDL do protocolo de negociação e gerência de QoS de uma sessão multimídia.
- A proposicição de uma arquitetura modular para implementação de aplicações multimídia distribuídas, onde o serviço de suporte à qualidadede de serviço baseados em sistemas de agentes é um de seus módulos.
- O desenvolvimento de um protótipo do modelo proposto com o objetivo de avaliar sua adequação em cenários de monitoramento e adaptação de QoS de aplicações de teleconferência.

1.6 Divisão do Trabalho

O trabalho está dividido da seguinte maneira:

- No Capítulo 2 é apresentada uma discussão sobre qualidade de serviço em sistemas multimídia distribuídos.
- No Capítulo 3 são caracterizados os principais requisitos para suporte de qualidade de serviço. Também é realizada uma análise sobre os principais trabalhos desenvolvidos na área.
- No Capítulo 4 é apresentada a nossa proposta de modelo, baseado em sistemas de agentes, para a realização da negociação e gerência da qualidade de serviço em sistemas multimídia distribuídos.
- No Capítulo 5 é especificado um protocolo de negociação e gerência de qualidade de serviço para o nosso modelo, através das linguagens formais SDL e MSC.
- No Capítulo 6 é apresentada a arquitetura de implementação do nosso modelo para negociação e gerência de qualidade de serviço, contextualizando-a dentro de uma plataforma de suporte de aplicações multimídia distribuídas.
- No Capítulo 7 é apresentada a implementação de um protótipo da arquitetura proposta, com o propósito de validar alguns dos conceitos expostos nos capítulos 4, 5 e 6.
- No Capítulo 8 são apresentadas as principais conclusões e contribuições do trabalho, além da indicação de possíveis futuros trabalhos.

Capítulo 2

Qualidade de Serviço em Sistemas Multimídia

Este capítulo é destinado à caracterização dos principais requisitos das aplicações multimídia. Neste sentido, introduz-se o termo qualidade de serviço (QoS) para sistemas em geral e sistemas multimídia distribuídos (SMD) em particular. A qualidade de serviço da aplicação está associada a diversos parâmetros que denotam o desempenho do sistema, denominados de parâmetros de qualidade de serviço. Também são apresentados os principais tipos de serviços suportados por sistemas multimídia distribuídos, como serviços de dispositivo, sincronismo e comunicação, por exemplo. Por fim, qualidade de serviço é associada ao problema de configuração de serviço, com respectiva reserva de recursos.

2.1 Introdução

Uma característica fundamental de sistemas multimídia é incorporar mídias contínuas tais como: áudio, vídeo e animação gráfica. O uso de tais mídias em sistemas computacionais distribuídos implica em capturar, processar, transmitir e exibir dados continuamente e por longos períodos de tempo. Além disso, requer-se, devido às suas características temporais, a garantia de que os dados possam ser entregues e exibidos de forma cadenciada no tempo. Para suportar completamente esta característica isócrona das mídias contínuas, sistemas multimídia distribuídos requerem a utilização de sistemas operacionais de tempo real, mecanismos de transporte com garantia de qualidade de serviço fim-a-fim e políticas de reserva e compartilhamento de recursos eficientes. Porém, estes requisitos usualmente são satisfeitos apenas parcialmente, com impacto direto na qualidade do serviço oferecido pelo sistema.

Sob esta ótica, podemos vislumbrar três classes de serviços multimídia: serviço garantido (ou determinístico), serviço com garantia estatística e serviço por melhor-esforço [22]. O primeiro tipo de serviço corresponde aos sistemas que satisfazem todos os requisitos de *hardware*, *software* e sistema de comunicação, de tal modo que as características de seus serviços podem ser previstas deterministicamente. Um exemplo seria um sistema que garante que nenhum quadro de um determinado fluxo de vídeo irá sofrer atraso de transmissão maior que 200ms [19]. O segundo tipo de serviço corresponde aos sistemas que satisfaçam parcialmente àqueles requisitos, uma vez que o serviço é garantido em termos de um valor

estatístico, como por exemplo: o sistema garante que pelo menos 60% dos quadros de um determinado fluxo de vídeo não irão sofrer atrasos de transmissão maiores que 200ms. Já o terceiro tipo de serviço está associado aos sistemas que não garantem previamente os níveis de qualidade dos seus serviços, e por isso atuam de forma a prover de melhor maneira possível às demandas de qualidade requeridas pela aplicação.

Analisando-se esses tipos de serviços, observa-se que aplicações multimídia apresentam fortes características de tempo real. Por isso, assumimos no capítulo anterior que sistemas multimídia sejam caracterizados mais adequadamente como sistemas de tempo real orientado a qualidade de serviço, onde os seus requisitos temporais de desempenho são descritos a partir de índices de qualidade de serviço, os quais o sistema irá procurar atender. Deste modo, a noção de qualidade de serviço torna-se fundamental, sendo indispensável caracterizá-la de forma mais adequada possível.

O termo QoS é bastante intuitivo para o usuário de aplicações multimídia, expressando basicamente uma noção subjetiva de quão bons estão os serviços fornecidos pelo sistema. O então CCITT, e atual ITU, definiu QoS de uma forma geral como sendo "o efeito coletivo do desempenho do sistema o qual determina o grau de satisfação do usuário do sistema" [32]. Já o modelo de referência ODP (*Open Distributed Processing*) da ISO [71] define qualidade de serviço como sendo "um conjunto de qualidades requerido para o comportamento de um ou mais objetos". Uma definição de QoS mais específica para aplicações multimídia foi fornecida por Vogel e outros em [67] como: "Qualidade de serviço representa o conjunto das características qualitativas e quantitativas de um sistema multimídia necessário para obter as funcionalidades requeridas pela aplicação".

Porém, para avaliarmos objetivamente o desempenho do sistema é necessário que se disponha de padrões mensuráveis. Visando isto, descreveremos qualidade de serviço em sistemas multimídia a partir de um conjunto de parâmetros bem definidos, os quais buscam caracterizar satisfatoriamente a noção subjetiva de qualidade de serviço fornecida por um dado sistema multimídia. Estes parâmetros são conhecidos na literatura como parâmetros de qualidade de serviço.

2.2 Parâmetros de Qualidade de Serviço

Há vários parâmetros que podem ser utilizados na caracterização de qualidade de serviço das diversas mídias suportadas por um ambiente de computação multimídia. Estes parâmetros podem ser agrupados em diversos níveis, desde os mais baixos, como os relacionados aos sistemas operacionais, até os mais altos, como os destinados aos usuários da aplicação. Optamos aqui por agrupá-los em dois níveis, a saber: nível de sistema e nível de usuário.

Os parâmetros do nível de sistema estão mais relacionados com as características requeridas pelo sistema de informação, caracterizando-se assim, como um nível mais baixo de abstração. Já os parâmetros do nível de usuário dizem respeito aos requisitos relacionados com a parte externa do sistema, expressando assim as características mais perceptíveis do sistema. Porém, é importante ressaltar que os parâmetros do nível de sistema estão relacionados aos do nível de usuário, e vice- versa, como é ilustrado na Fig. 2.1.

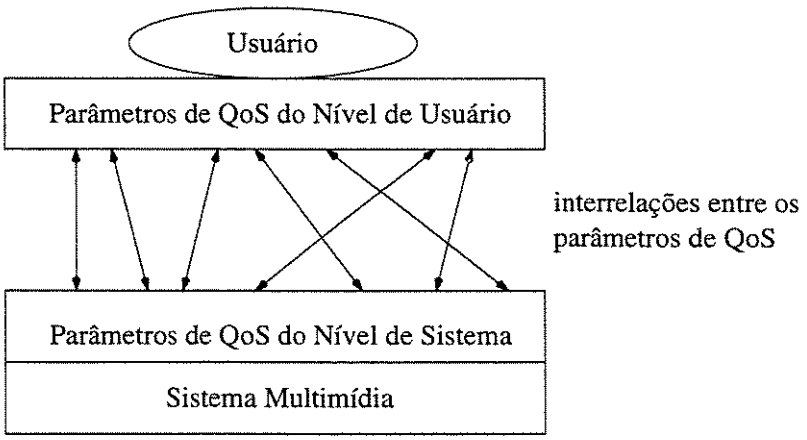

Fig. 2.1: Interrelações entre os níveis de abstrações dos parâmetros de QoS.

2.2.1 Parâmetros de QoS no Nível de Sistema

No nível de sistema, os parâmetros de QoS são bem estabelecidos na literatura [57] [19]. Eles são listados a seguir:

- atraso fim-a-fim (*end-to-end delay*);
- jitter fim-a-fim (*end-to-end* jitter);
- razão de pacotes perdidos (PER, *Packet Error Rate*);
- razão de bits errados (BER, *Bit Error Rate*).

O atraso fim-a-fim é o tempo gasto desde a geração de um dado até sua exibição. Por geração entendemos a captura do dado por um periférico (câmera e microfone, por exemplo) ou sua ativação numa base de dados. Este atraso é devido às latências na captura, ativação, processamento e transmissão dos dados. Em aplicações multimídia, os requisitos de atrasos fim-a-fim máximos, via de regra, estão entre 100 e 250ms [19].

O jitter fim-a-fim é a variação média do atraso fim-a-fim entre duas exibições sucessivas de uma dada mídia. Para uma boa qualidade na apresentação, mídias contínuas requerem baixos níveis de jitter fim-a-fim. Idealmente, o jitter fim-a-fim deveria ser nulo, ou seja, a razão de apresentação dos dados das mídias contínuas deveria ser constante. Porém, devido a compartilhamento de recursos de processamento e comunicação, temos que o atraso fim-a-fim não é constante. Em aplicações multimídia típicas, o jitter fim-a-fim máximo tolerado se encontra na faixa entre 5 a 10ms [19].

A razão de pacotes perdidos e a razão de bits errados são parâmetros de QoS que visam caracterizar o grau de confiabilidade do sistema de comunicação. Estes parâmetros são bastantes importantes, uma vez que para a maioria das aplicações multimídia distribuídas mecanismos de retransmissão de dados não são factíveis. Estes parâmetros são obtidos de forma estatística, o que confere ao conjunto de parâmetros de QoS uma característica estocástica. Valores típicos de PER máximos requeridos por aplicações multimídia estão entre 0,001% e 0,01%. Já valores máximos de BER máximo estão entre 0,01 e 0,1% [19].

Devido à alta confiabilidade das redes de computadores na atualidade o parâmetro BER não é muito utilizado na caracterização de qualidade de serviço. Por isto, utilizaremos apenas o parâmetro PER para caracterizar o confiabilidade do sistema de comunicação.

2.2.2 Parâmetros de QoS no Nível de Usuário

Nesta seção são apresentados os parâmetros de qualidade de serviço no nível de usuário. Estes parâmetros destinam-se a caracterizar a qualidade dos serviços fornecidos por um sistema multimídia de forma mais abstrata que os enfocados na seção anterior, ou seja, numa ótica mais próxima do usuário final. Deste modo, para expressar QoS no nível de usuário, propomos os seguintes parâmetros:

- qualidade de mídia de vídeo;
- qualidade de mídia de áudio;
- qualidade de mídia estática;
- qualidade de sincronismo.

Na realidade, estes parâmetros são estruturas compostas de outros parâmetros. No caso, mídias de vídeo e de áudio possuem as mesmas estruturas, porém seus parâmetros têm natureza distintas. A mesma estrutura deve-se ao fato que ambos pertencem à classe das mídias contínuas. Já as mídias estáticas possuem uma estrutura de parâmetros mais simples, decorrência de sua maior simplicidade em relação às mídias contínuas. A seguir estas estruturas serão descritas em detalhe.

Qualidade de Mídia de Vídeo

A qualidade de serviço das mídias de vídeo é caracterizada pela seguinte estrutura:

- definição (pixel/quadro, cores/pixel);
- fidelidade (distorção, confiabilidade);
- dinâmica (quadros/segundo, variação de quadros/segundo);
- interatividade.

O item definição diz respeito ao grau de detalhe da imagem exibida, para tanto é expresso por dois parâmetros: número de pixels por quadros e número de cores por pixels. Esse parâmetro tem impacto direto na largura de banda requerida para transmissão de tal fluxo.

O item fidelidade visa capturar o nível de fidelidade dos dados de vídeo, sendo descrita por dois parâmetros: distorção e confiabilidade. O primeiro diz respeito ao nível de distorção ocorrido na imagem, desde a sua geração até a sua exibição. A distorção pode ter sido provocada pelo algoritmo de compactação de dados, por exemplo. O segundo parâmetro denota quão confiável é o sistema de comunicação, estando portanto relacionado aos parâmetros PER e BER do nível de sistema.

O item dinâmica se destina a caracterizar o grau de precisão com que o movimento dos segmentos de vídeo são exibidos. Seus parâmetros são números de quadros por segundo e variação de quadros por segundo. Em termos estatísticos, eles podem ser descritos respectivamente como média e variância de uma distribuição. A variação da freqüência de exibição tem relação direta com o jitter fim-a-fim.

O item interatividade é uma medida da utilidade do fluxo em aplicações interativas, como por exemplo teleconferência e trabalho cooperativo por computador. Este parâmetro tem relação direta com o atraso fim-a-fim.

Qualidade de Mídia de Áudio

A qualidade de áudio possui a mesma estrutura da qualidade de vídeo, porém, seus parâmetros não têm as mesmas dimensões, como pode ser visto a seguir:

- definição (bits/amostra, padrão de áudio);
- fidelidade (distorção, confiabilidade);
- dinâmica (amostra/segundo, variação de amostra/segundo);
- interatividade.

O parâmetro padrão de áudio, em definição, denota se o áudio é mono ou estéreo. Já os outros parâmetros seguem, em linhas gerais, as mesmas interpretações da qualidade de vídeo.

Qualidade de Mídia Estática

A qualidade das mídias estáticas possui uma estrutura mais simples do que as das mídias contínuas. Seus itens são:

- tipo (tipo de mídia estática);
- fidelidade (distorção, confiabilidade).

O item tipo tem um único parâmetro que caracteriza o tipo de mídia, que pode ser texto, gráfico ou imagem. O item fidelidade é similar aos definidos anteriormente, porém, o parâmetro distorção não possui significado quando o tipo de mídia estática for texto.

Qualidade de Sincronismo

Esta estrutura visa capturar o grau de sincronismo entre os diversos fluxos de dados das diversas mídias. Aqui, só há sentido em expressar sincronismo entre fluxos que realmente têm alguma relação temporal entre si, as quais podem ser de precedência, conseqüência ou simultaneidade. Assim, qualidade de sincronismo é composta de parâmetros que caracterizam o grau de sincronismo pertinente aos diversos fluxos em uma aplicação. É importante ressaltar que os fluxos que possuem relações de sincronismo devem ser especificados no ato da configuração da aplicação. Outro aspecto importante a ressaltar é que o item qualidade de sincronismo tem por finalidade apenas expressar o nível de sincronismo entre os diversos fluxos, e não propor algoritmos de sincronização intra e inter-fluxos.

2.2.3 Relação Entre Parâmetros de Qualidade de Serviço nos Níveis de Usuário e de Sistema

A Fig. 2.2 ilustra em termos de diagrama de blocos as relações entre os parâmetros de QoS nos níveis de sistema e usuário para um fluxo de vídeo. Alguns parâmetros de QoS no nível de usuário não têm relação direta com os do nível de sistema, pois, com será visto mais a frente, estes parâmetros estão associados diretamente aos recursos do sistema.

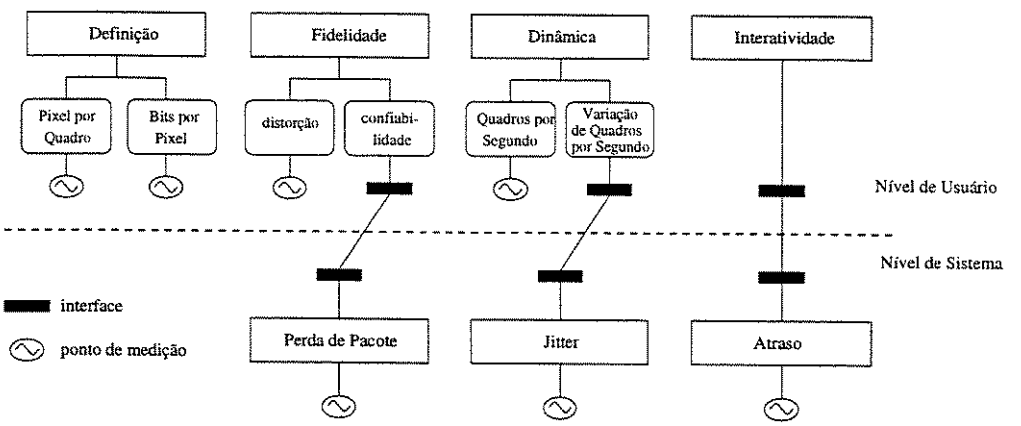

Fig. 2.2: Relações entre os parâmetros de QoS do nível de sistema e do nível de usuário para um fluxo de vídeo.

Um aspecto importante a ser observado é que essas estruturas de qualidade de serviço aqui definidas podem ser bastante úteis, tanto para especificação quanto para verificação da qualidade de serviço em sistemas multimídia distribuídos. Deste modo, na próxima seção abordaremos os tipos de serviços multimídia, com seus respectivos requisitos de qualidade de serviço e recursos de *software* e *hardware*.

2.3 Qualidade em Serviços Multimídia

Para que se entenda os tipos de serviços e recursos requeridos por aplicações multimídia distribuídas é necessário primeiro analisar as diversas etapas por que passa um fluxo, desde sua captura até a consequente exibição. Desse modo, a Fig. 2.3 ilustra de forma simplificada o caminho percorrido por um fluxo em uma sessão multimídia distribuída

A partir da Fig. 2.3 podemos notar que ao longo das várias etapas das quais percorre, o fluxo de informação vai incrementando seu atraso e jitter fim-a-fim. O atraso fim-a-fim se deve à necessidade de um tempo finito para que as etapas constituintes de uma sessão multimídia se completem, que no caso são tempo de processamento, tempo de espera em *buffers* e tempo de transmissão através da rede. Tempo de CPU, banda de transmissão e *buffers* são recursos disponíveis no sistema. Assim, para diminuir o atraso fim-a-fim deve-se aumentar o tempo de CPU e a largura de banda de transmissão do fluxo em questão, além de

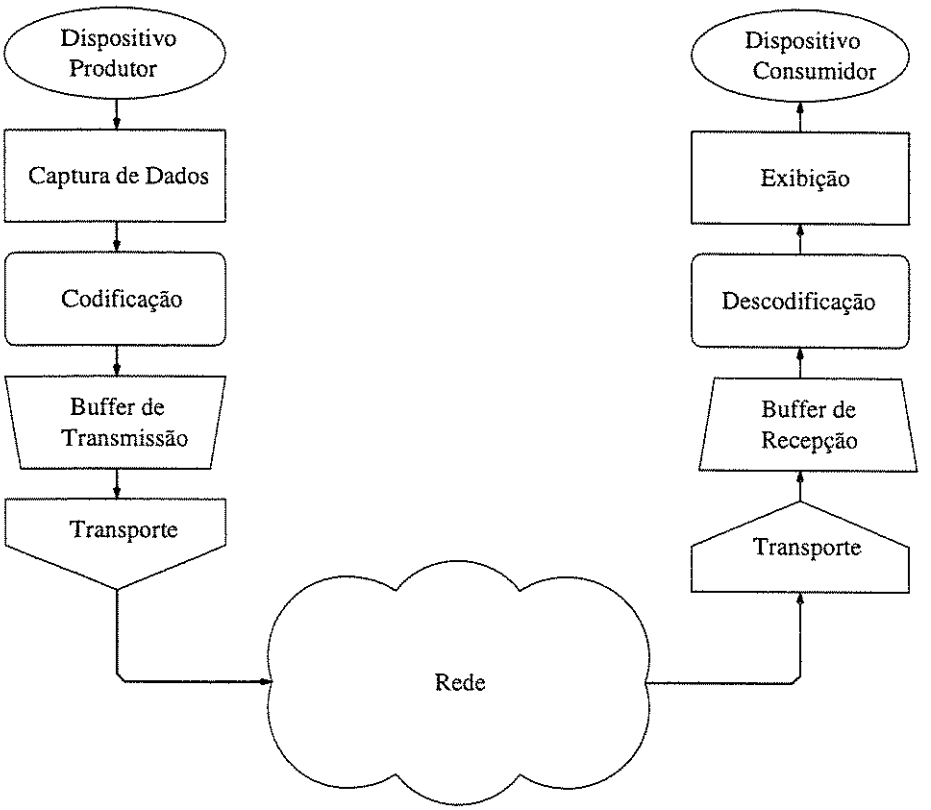

Fig. 2.3: Diagrama simplificado de um fluxo de informação distribuída.

diminuir os *off-set*[1] dos *buffers* de transmissão e recepção (estes parâmetros são responsáveis por atrasos de espera nos *buffers*, uma vez que somente são transmitidos ou recebidos dados quando se atinge nos *buffers* os valores estabelecidos pelos *off-sets*).

Como visto no capítulo anterior, o suporte de dados de vídeo e áudio requer uma grande largura de banda de transmissão, de modo a garantir vazão para o alto volume de dados produzidos. Para diminuir a excessiva largura de banda de transmissão requerida por tais mídias, tornou-se usual a utilização de técnicas de compressão de dados. Em geral, quanto maior a taxa de compressão obtida por um algoritmo, maior o grau de supressão de informação (este é o conflito chave nos algoritmos de compressão de dados). Os algoritmos da classe MPEG [20] são padrões de compressão de dados bem estabelecidos em multimídia. Para vídeo com movimento intenso, é usual obter-se taxas de compressão de 50 a 200 vezes, enquanto para áudio, as taxas de compressão típicas alcançadas estão entre 5 e 10 vezes [19].

Como várias aplicações de sistemas multimídia distribuídos são de natureza ponto-multiponto, como é o caso de teleconferência, mecanismos de transmissão em *multicast* se apresentam como serviços bastante relevantes para otimizar recursos de rede.

Porém, a dificuldade no suporte de mídias contínuas não se limita somente às grandes demandas de larguras de banda de transmissão, mas também pela necessidade de se ter uma razão de apresentação constante, ou seja, jitter fim-a-fim nulo. O jitter fim-a-fim surge devido ao compartilhamento de recursos do sistema, como tempo de CPU e banda de transmissão,

[1]Valor necessário que se atinja no *buffer* para que se comece a transmissão e ou recepção de dados.

o que leva a uma variação no atraso fim-a-fim (jitter fim-a-fim) do fluxo. Para se obter exibição isócrona é necessário que se tenha: processamento e transmissão isócronos. O primeiro requer capacidade de processamento adequada aliada a um sistema operacional de tempo real, por exemplo, enquanto o segundo requer, no atual estágio da tecnologia de redes de computadores, a utilização de protocolo de reserva de largura de banda associado com *buffers* de transmissão e recepção dimensionados adequadamente. Deste modo, os fatores que dificultam a obtenção do requisito de "isocronismo" são: i) as vazões das mídias contínuas não são constantes ao longo de seu ciclo de vida e ii) os protocolos de transporte convencionais não suportam adequadamente transmissão contínua por longos períodos de tempo. O fato das mídias contínuas não apresentarem vazões constantes é devido ao uso de técnicas de otimização de recursos, como por exemplo as técnicas de compressão de dados. Já o segundo fator é devido ao fato dos protocolos de transporte e serviços de rede convencionais terem sido projetados visando transmissão de mídias estáticas, como texto, onde o mais importante é a ausência de erros.

Uma maneira de garantir que sempre será possível atender à demanda de vazão de transmissão de um dado fluxo de mídia contínua é via reserva da largura de banda de transmissão máxima requerida pelo fluxo. Aqui a opção é pelo superdimensionamento de reserva de recursos. Porém, isto não é suficiente para obter uma razão de apresentação constante dos dados (jitter fim-a-fim nulo). Para tal, também faz-se necessário a utilização de *buffers* de dimensões adequadas na transmissão e recepção, além de mecanismos de sincronização de relógios.

Entretanto, muitas vezes, por limitação de recursos, não é possível reservar previamente a largura de banda de transmissão máxima requerida por uma dada aplicação. Então, torna-se necessário optar por uma solução de compromisso, qual seja: reservar um tamanho de banda de transmissão intermediário com o risco de perda de qualidade de serviço. Aqui há o conflito entre a utilização de *buffers* de tamanhos elevados (ou até mesmo de tamanhos variáveis), de modo a diminuir o jitter devido a etapa de transmissão em rede, mas aumentando o atraso fim-a-fim; ou a utilização de *buffers* menores, com menores atrasos fim-a-fim e maiores jitter e PER, devido respectivamente a prováveis situações de esvaziamento e transbordamento de dados nos *buffers* (sinalização de buffer-cheio e buffer-vazio).

Há duas possibilidades de descarte de informação devido à falta de espaço de armazenamento nos *buffers* de transmissão e recepção (buffer-cheio). No primeiro caso o descarte é decorrente do desbalanceamento entre as velocidades de geração e transmissão dos dados (possivelmente devido a sobrecarga no sistema de comunicação), enquanto no segundo o descarte é devido ao desbalanceamento entre as velocidades de transmissão e exibição dos dados (possivelmente devido a sobrecarga no sistema de processamento do nó consumidor de informação). A perda total é a composição das perdas ocorridas ao longo do processo de geração e consumo de informação. Uma solução imediata para a diminuição de perda de informação por buffer-cheio seria o aumento dos tamanhos dos *buffers*, acarretando um maior consumo de recursos do sistema.

Em suma, a utilização de *buffers* melhora a qualidade do serviço em relação aos parâmetros jitter fim-a-fim e PER, mas piora em relação ao atraso fim-a-fim. A forma de se administrar esses conflitos vai depender de cada caso, por exemplo, a tolerância a erros de bits e perdas de pacotes em vídeo depende da técnica de compressão de dados adotada. Além disto, cada tipo de mídia possui características próprias. Por exemplo, vídeo requer maior vazão que

áudio, porém o áudio é mais sensível a jitter, erros de bits e perdas de pacotes que vídeo [64].

Deste modo, podemos caracterizar o caminho percorrido por um fluxo de informação em uma sessão multimídia ponto-ponto como dois problemas de produtor-consumidor em série, onde o produtor primário seria o capturador de dados (ou o ativador na base de dados), a transmissão em rede funcionaria como consumidor para a captura e produtor para a exibição, sendo esta o consumidor final. Caracteriza-se, assim, definitivamente a necessidade de cadência entre todas as etapas pelas quais um determinado fluxo de informação atravessa.

A partir dessas considerações, podemos visualizar dois níveis de serviços multimídia. Um relacionado ao processamento e armazenamento de informações local a cada máquina e outro relacionado à comunicação destas informações entre os diversos nós envolvidos na aplicação multimídia distribuída, denominados aqui respectivamente de serviços da aplicação e serviços de comunicação. A Fig. 2.4 refaz a ilustração da Fig. 2.3 a partir dessa ótica.

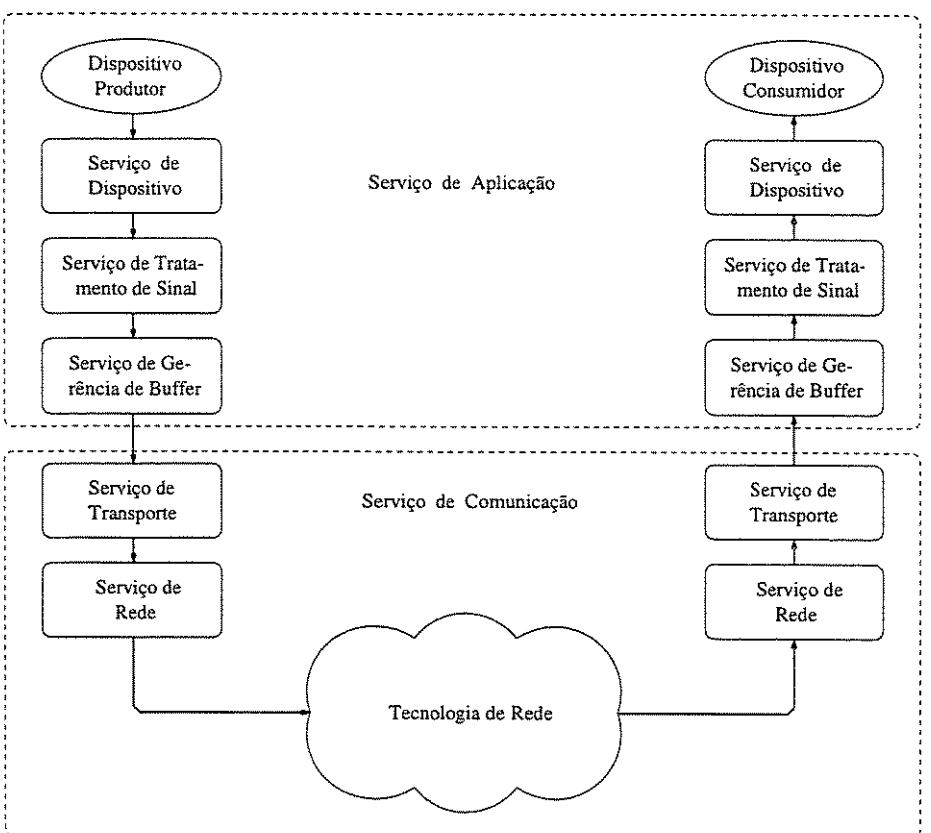

Fig. 2.4: Exemplo dos níveis de serviços utilizados por um fluxo de informação.

2.4 Serviços da Aplicação

Os serviços considerados como da aplicação são basicamente serviços de dispositivos (captura e exibição), de base de dados (armazenamento e ativação), de tratamento de si-

Freqüência de Amost. (KHz)	Precisão por Amostra (Bits)	Tipo de Codificação	Tipo de Configuração	Tipo de Qualidade
44,1	16	PCM	CD	Muito Boa
32	16	PCM	FM	Boa
8	8	PCM	Telefone	Média
4	4	PCM	Metálica	Baixa

Tab. 2.1: Alguns formatos típicos para áudio

Formato	Quadro/segundo	Pixel/quadro	Bits/Pixel
NTSC	30	391.680	8
PAL	25	576.384	7
HDTV	60	1.440.000	24

Tab. 2.2: Alguns formatos usuais para vídeo.

nal (compactação, descompactação, codificação e filtragem), de sincronismo, de gerência de *buffer* e de escalonamento de tarefas. Estes serviços são detalhados a seguir.

2.4.1 Serviços de Dispositivos

Consideramos serviços de dispositivos como aqueles relacionados com a captura e exibição de áudio e vídeo. Para áudio, os serviços de dispositivos estão relacionados respectivamente aos dispositivos físicos microfone e alto falante, enquanto que para vídeo estão relacionados à câmera e ao monitor de vídeo.

Estes serviços caracterizam os formatos de captura e conseqüente exibição dessas mídias. Os formatos caracterizam aspectos temporais (freqüência de amostragem), precisão (bits por amostra) e tipo de codificação.

Para áudio, a faixa da freqüência de amostragem é de 4kHz a 44,1 Khz, a precisão de 4 a 16 bits por amostra, e o método de codificação mais usual é o PCM (*Pulse Code Modulation*). A Tab. 2.1 mostra algumas configurações típicas. Por tal tabela, temos que áudio com qualidade de CD gera uma quantidade de dados (sem compactação) de 705,6 Kbps (Kilobits por segundo), enquanto que com qualidade de telefone cai para 64 Kbps.

Para vídeo, a precisão é caracterizada pela quantidade de pixeis por quadro (número de pixeis por linha multiplicado pelo número de pixeis por coluna) e número de bits por pixel. A Tab. 2.2 mostra alguns formatos padrões para vídeo com qualidade de TV [22]. A partir da Tab. 2.2, verifica-se que o volume de dados gerados pelos dispositivos de vídeo são extremamente elevados.

2.4.2 Serviços de Base de Dados

Serviços de base de dados são um componente importante em aplicações multimídia. Estes serviços fornecem armazenamento persistente e coerente aos dados multimídia. O problema fundamental no desenvolvimento destes sistemas de armazenamento, conhecidos como

servidores multimídia, é que as mídias contínuas têm características bastante diferentes dos convencionais, como texto, requerendo técnicas especializadas para a organização e gerência da informação [21]. Além dos grandes volumes de memória demandados para o armazenamento das mídias contínuas, tem-se, devido às suas características temporais, a necessidade de recuperá-las de forma cadenciada no tempo de modo a manter a integridade da informação [58]. Deste modo, um servidor multimídia deve assegurar que a escrita e a leitura de fluxos contínuos sejam realizadas em conformidade não somente com suas características espaciais, mas também com suas características temporais.

2.4.3 Serviços de Tratamento de Sinal

Serviços de tratamento de sinal são basicamente atividades de filtragem, criptografia, compactação e descompactação de dados. A filtragem se destina a retirar freqüências espúrias do sinal de fluxo, melhorando sua apresentação, enquanto a criptografia visa a segurança da transmissão do sinal.

Já as atividades de compactação e descompactação se justificam principalmente pelos altos volumes de dados gerados pelas mídias contínuas, que demandam altas larguras de banda de transmissão de rede e enormes espaços de memória para o seu armazenamento. Isto é especialmente válido para fluxos de vídeo.

Algoritmos de compactação de dados digitais necessitam ser eficientes não somente no nível de compactação, mas também no seu desempenho computacional, pois as aplicações multimídia possuem características de tempo real. E justamente para atender a esses requisitos temporais é que em geral o par compactação/descompactação possui característica assimétrica, sendo mais custoso computacionalmente o processo de compactação (geralmente realizado por um *hardware* especial) que o processo de descompactação (geralmente efetuado por *software*).

As técnicas de compactação/descompactação podem ser classificadas como sem perdas e com perdas [19]. As técnicas sem perdas recuperam completamente a informação original. Já as técnicas com perdas recuperam a informação original com algum nível de distorção, porém, apresentam altas taxas de compressão, sendo de uso mais freqüente em compactação de vídeo que as técnicas sem perdas. De forma geral, quanto maior o fator de compressão de uma técnica, maior é o seu nível de perda.

Para fluxos de áudio é usual se utilizar o ADPCM (*Adaptive Delta Pulse Code Modulation*), ou variações suas, como método de compactação de dado. Nele se transmite apenas a variação do sinal codificado em PCM. Além disto, consegue-se bom nível de economia de recursos de transmissão quando se utiliza técnicas de detecção de silêncio. Essas técnicas consistem basicamente em monitorar o nível de intensidade do sinal de áudio de modo a caracterizar períodos de silêncio na transmissão de áudio. Estes períodos de silêncio não são transmitidos, economizando-se, assim, recursos de transmissão. A dificuldade aqui é justamente detectar de forma eficiente esses períodos de silêncio, pois um procedimento ineficiente de detecção leva a uma degradação na qualidade do áudio.

Para vídeo, os métodos de compactação e descompactação mais usuais são: JPEG, MPEG, H.261 e o CellB.

JPEG

O padrão JPEG (*Joint Photographic Expert Group*) foi desenvolvido pela ISO originalmente para imagem estática, obtendo nestes casos taxas de compactação usualmente na faixa de 15 vezes [68]. É uma técnica com perdas que realiza somente compressão espacial da imagem (intra-quadro), sendo também utilizado para compactação de fluxo de vídeo (neste caso é conhecido como MJPEG). Um fluxo de vídeo MJPEG é uma seqüência de quadros JPEG.

Para seqüências de vídeo, cada quadro é dividido em blocos de 8x8 ou 16x16 pixeis, e então é aplicada a cada bloco uma transformação DCT (*Discrete Cosine Transform*) para o domínio da freqüência, de modo a produzir coeficientes que retém a informação espacial da imagem. Muitos destes coeficientes não necessitam ser codificados, pois possuem valores nulos ou muito pequenos. Este procedimento introduz perda de informação, pois apenas os aspectos mais relevantes da imagem são preservados. A codificação de Huffman [68] também é usada para aumentar o grau de compactação do método. Para descompactar a imagem é realizado o procedimento inverso.

JPEG apresenta um característica quase simétrica, uma vez que o esforço computacional de compactação e descompactação são equivalentes, sendo a compactação um pouco mais custosa computacionalmente que a descompactação.

MPEG

MPEG (*Moving Pictures Expert Group*) é um padrão da ISO para compactação e descompactação de fluxo de vídeo [20] [40], tanto compactação espacial (intra-quadro) quanto temporal (inter-quadro). Na realidade há um série de padrões MPEG (MPEG-1, MPEG-2, MPEG-3, MPEG-4 e MPEG-7).

MPEG-1, que é essencialmente um exibidor de vídeo digital, trabalha com quadros de vídeo de até 320 x 240 pixeis, se destina a aplicações como multimídia interativa e seu requisito mínimo de geração de dados é de 1,5 Mbps. MPEG-2 prevê quadros de até 720 x 480 pixeis, destina-se a compactação e transmissão de vídeo digital com qualidade de TV e produz dados na faixa de 4 a 10Mbps. MPEG-3 foi concebido para suporte a fluxo de vídeo com qualidade de HDTV (*High Definition Television*) e gerar dados na faixa de 5 a 20 Mbps, porém por sua siminaridade com o MPEG-2, o padrão MPEG-3 foi abandonado e o suporte para HDTV foi incorporado no MPEG-2. MPEG-4 se destina a aplicações como videofone e aplicações interativas, que utilizam quadros de tamanhos pequenos a baixa taxa de amostragem com produção de dados na faixa de 9 a 40 Kbps. MPEG-7 se destinará à padronização de descrição de material multimídia, como imagem em movimento. MPEG também possui uma especificação de compactação de áudio, que não se tornou muito popular.

No MPEG-1, a compactação espacial é realizada de forma similar ao JPEG, enquanto a compactação temporal é advinda da utilização de quadros intermediários gerados por intepolações. No caso, o método gera seqüências compostas por três tipos de quadros: I, P e B.

Os quadros do tipo I são similares aos do JPEG, e exploram a compactação espacial da imagem. Os dos tipos P e B são gerados por interpolações preditivas e bidirecionais, respectivamente, gerando um alto grau de compactação temporal do fluxo. Quanto mais quadros B e P forem inseridos entre dois quadros I sucessivos, maior será a compactação temporal do fluxo de vídeo, mas por outro lado, maior será a distorção durante o processo

de descompactação do fluxo. O nível de compactação no MPEG-1 depende da intensidade de movimento da imagem, mas valores na faixa de 200 vezes são factíveis [20].

No MPEG-1, o processo de interpolação é computacionalmente bastante custoso, já o seu processo de descompactação é mais simples. Caracterizando-o como um método assimétrico. Porém, ainda assim, o processo de descompactação MPEG-1 é mais custoso que o do JPEG. Usualmente, a compactação MPEG é efetuada por um *hardware* especial, enquanto a descompactação é realizada tanto por *software* quanto por *hardware*.

H.261

H.261 é um padrão de compactação de vídeo projetado para ser utilizado em linha de comunicação com capacidade múltipla de 64 Kbps [43], sendo por esta característica também conhecido com o px64 (sendo p = 1,2,...,30). Deste modo, este padrão pode gerar até 1,92 Mbps de dados. Ele se destina a aplicações de videofone e videoconferência. Valores de 1 e 2 para p correspondem a aplicações de videofone, enquanto aplicações de videoconferência necessitam de valores de p acima de 6.

H.261 utiliza compactação espacial e temporal. Similarmente ao MPEG e ao JPEG, H.261 utiliza a transformação em freqüência DCT para compactação espacial da imagem. A compactação espacial também é realizada por interpolação. Porém, seu algoritmo de interpolação é mais simples que o do MPEG, o que reduz a carga computacional de compactação e descompactação, mas, por outro lado, reduz a qualidade do vídeo após a sua descompactação. Por isto, H.261 é mais indicado para vídeo de baixa intensidade de movimento. Nestes casos, níveis de compactação na faixa de 100 a 200 vezes são factíveis.

CellB

CellB é um padrão de compactação e descompactação de vídeo proprietário da SUN [62], que foi projetado para ser utilizado em aplicações de videoconferência. CellB efetua compactação espacial e temporal. A compactação espacial é conseguida representando cada bloco de 4x4 pontos por uma máscara de bit de 2 bytes e dois vetores de 1 byte cada. Como um pixel é representado por um byte, a taxa de compactação intra-quadro é de 4 vezes. A compactação temporal é realizada por um procedimento de comparação dos blocos entre quadros sucessivos. Quando a informação num bloco sofre variação menor que um certo valor de *off-set* entre quadros adjacentes, não há necessidade de transmiti-lo. Este *off-set* é o parâmetro de ajuste do algoritmo, sendo que quando maior for o *off-set* maior será a taxa de compactação temporal, mas pior será a qualidade do fluxo de vídeo após a sua descompactação.

CellB apresenta taxas de compactação (espacial + temporal) tipicamente na faixa de 10 a 20 vezes. Como a sua compactação espacial não utiliza DCT (computacionalmente custosa), sua carga computacional é bem menor que a do MPEG.

2.4.4 Serviços de Sincronismo

Sistemas multimídia são caracterizados por gerar, armazenar, comunicar, manipular e apresentar diversos tipos de mídias de forma digitalizada e integrada. Neste aspecto, a manutenção das características temporais dos fluxos de informação é fundamental. Serviços

de sincronização são responsáveis por esta integridade, estando assim relacionados aos requisitos temporais inerentes em tais sistemas. Estes requisitos podem ser divididos em três categorias: sincronização de fluxo, sincronização de eventos e sincronização de grupo. Por sua vez, sincronização de fluxo pode ser dividida em sincronização intra-fluxo e sincronização inter-fluxos [64] [6].

A sincronização intra-fluxo, ou sincronização serial, se refere à manutenção da relação temporal entre as várias unidades de um fluxo contínuo, ou seja, manter sua característica isócrona. Está portanto relacionada ao parâmetro jitter de QoS. De modo a manter a cadência na apresentação dos fluxos contínuos, o serviço de sincronização pode descartar unidades de informação que apresentem atrasos excessivos, acarretando o aumento de perda de informação. Nesta perspectiva, o parâmetro de QoS atraso também desempenha papel fundamental.

A sincronização inter-fluxos, ou sincronização paralela, se refere à manutenção da relação temporal que pode existir entre unidades de informação de diferentes fluxos contínuos. Sincronização entre áudio e vídeo é o exemplo mais comum de sincronismo entre fluxos. Como os fluxos atravessam vários componentes desde a captura até a apresentação, fluxos relacionados temporalmente podem experimentar atrasos distintos, causando perda de sincronismo. No caso, o serviço de sincronização atua de modo a minimizar este fenômeno. Este serviço pode se utilizar basicamente de duas estratégias: sincronização na origem e sincronização no destino.

Na sincronização na origem, o procedimento, em geral, é multiplexar os fluxos relacionados temporalmente, para depois demultiplexá-los no destino. Porém, esta abordagem apresenta algumas deficiências, como [5]:

- a sobrecarga e a complexidade do procedimento de multiplexação e demultiplexação, principalmente quando diferentes esquemas de codificação e compactação são utilizados pelas mídias envolvidas na sincronização;
- a qualidade dos diversos fluxos é tratada de forma homogênea, de modo que as mídias mais sensíveis são penalizados em seus parâmetros de QoS (atraso, jitter e perda);
- para mídias que se originam de fontes localizadas em diferentes pontos na rede, a multiplexação na origem não é factível.

Na sincronização no destino, mantém-se integridade dos fluxos pela utilização de canais individuais para cada fluxo participante do processo de sincronização. No caso, em geral usa-se *buffers* na recepção de cada fluxo, sendo o sincronismo obtido através de operações de suspensão e avanço na leitura destes *buffers*. Uma estratégia de mestre-escravos é uma abordagem usual. Nela, escolhe-se um fluxo para ditar a cadência de apresentação (fluxo mestre), enquanto os outros fluxos (fluxos escravos) devem seguir essa cadência, devendo realizar, caso necessário, operações de *pause* e *skew* em seus *buffers*. Deste modo, a qualidade do fluxo mestre é privilegiada em detrimento da qualidade dos fluxos escravos.

A sincronização de eventos está relacionada à programação temporal de ações em resposta à ocorrência de eventos no sistema.

A sincronização de grupo diz respeito à manutenção da relação temporal da apresentação de um fluxo em diversos pontos. Ela segue o princípio de que um fluxo deve ser apresentado

idealmente ao mesmo tempo para todos os participantes de um sessão multimídia. Um número importante de aplicações, como teleconferência e trabalho cooperativo, necessita de sincronização de grupo. No caso, o serviço de sincronização deve providenciar que os atrasos sofridos pela informação, ao percorrer os diversos caminhos, sejam similares.

2.4.5 Serviços de Gerência de *Buffers*

Como visto, um fluxo contínuo atravessa várias etapas desde a sua geração até o seu consumo. A interligação dessas etapas são, quando locais, realizadas por *buffers*. Porém, devido aos elevados volumes de dados gerados pelas fontes de mídias contínuas, procedimentos usuais de cópia de dados entre *buffers* não são eficientes em aplicações multimídia. Assim, técnicas de gerência que manipulam endereços de *buffer* são mais apropriadas.

Os serviços de gerência de *buffer* têm tipicamente a responsabilidade de controlar o acesso de leitura (consumo) e escrita (geração) nos *buffers*, além de seu dimensionamento e gerência de região de controle [46].

Como a velocidade de consumo de dados não necessariamente é a mesma da velocidade de geração, é usual a utilização de valores iniciais de *off-set* nos *buffers* de modo a atenuar o jitter devido a buffer-vazio. Porém, seu efeito colateral negativo é o incremento do atraso fim-a-fim do fluxo, devido ao aumento médio de espera no *buffer*.

O dimensionamento dos *buffers* é fundamental, e suas relações de compromisso são as seguintes:

- a utilização de *buffers* de dimensões reduzidas economiza recursos de memória do sistema, porém aumenta a possibilidade de ocorrência de *buffer-cheio* (situação na qual a velocidade de geração de dados é maior que a de consumo). Em situação de *buffer-cheio* há duas possibilidades: o descarte a informação mais recente ou o descarte da informação mais antiga. Em aplicações multimídia, a segunda opção é a mais adequada. Em última análise, a ocorrência de *buffer-cheio* implica em descarte de informação, deteriorando a qualidade do serviço.

- a utilização de *buffers* de tamanhos elevados diminui a possibilidade de ocorrência de *buffer-cheio*, porém há a necessidade de maior quantidade de recursos de memória, o que nem sempre é possível, além da possibilidade de se utilizar recursos com informações que poderão não ser utilizadas devido aos requisitos temporais da aplicação, ou seja, informações que possivelmente serão descartadas na apresentação devido ao seu elevado atraso.

A partir das análises anteriores, verifica-se a importância de se manter a cadência entre as velocidades de geração e consumo de informação.

2.4.6 Serviços de Escalonamento de Tarefas

Baseado nas análises realizadas nas seções anteriores, verifica-se que é de fundamental importância o sincronismo e o controle de velocidade dos serviços que atuam sobre os fluxos contínuos. Esses serviços são incorporados em tarefas no sistema operacional e para se garantir a qualidade desses serviços há a necessidade de utilizar políticas de escalonamento de

tarefas típicas de sistemas operacionais de tempo real [59]. Essas políticas de escalonamento têm o tempo de utilização da CPU como recurso gerenciado.

Devido às suas características temporais peculiares, aplicações multimídia não demandam toda a rigidez necessária em aplicações de tempo real convencionais, como é o caso de controle de processos em tempo real, por exemplo. Assim, aplicações multimídia apresentam restrições temporais maleáveis, podendo admitir certo nível de violação de seus requisitos o que acarreta na deterioração da qualidade de seu serviço.

Processamento de fluxo contínuo necessita ser efetuado periodicamente em intervalo previamente definidos (múltiplo do período de amostragem do sinal contínuo). As operações sobre estes dados devem ser, assim, concluídas nestes intervalos de tempo e o escalonador de tarefas deve prover uma programação de execução de tarefas que permita que todas as atividades sobre um segmento de fluxo contínuo sejam cumpridas dentro do seu intervalo de amostragem. A dificuldade aqui é justamente encontrar a seqüência de execução das tarefas que permita que todas possam cumprir suas restrições temporais. Os algoritmos de escalonamento de tarefas mais usados em aplicações multimídia ainda são variações dos algoritmos taxa-monotônica e *earliest-deadline* [59].

Outro aspecto importante é que devido à necessidade de intensa mudança de contexto, em aplicações multimídia é preferível a utilização de *threads* em lugar de processos. Isto é devido ao fato de *threads* apresentarem menor *overhead* na mudança de contexto que processos.

2.5 Serviços de Comunicação

Quando um fluxo de informação atravessa um sistema de comunicação, ele se utiliza basicamente os serviços de transporte e de rede, considerados aqui como serviço de comunicação. A atribuição desses serviços de comunicação refere-se basicamente a transportar as informações dos fluxos contínuos desde a origem até o ponto de consumo (*unicast*), ou pontos de consumo (*multicast*), mantendo a integridade da informação. Para fluxos contínuos, não só a integridade e ordenação dos dados são importantes, mas também os aspectos temporais desse serviço. Outro fator que dificulta o serviço de comunicação é o fato das mídias contínuas requererem altas taxas de transmissão com baixos níveis de atraso e jitter. Isto indica a necessidade de suporte à qualidade de serviço também para os serviços de comunicação.

O atraso introduzido pelo sistema de comunicação para processar uma unidade de dado depende de vários fatores, tais como o tamanho da unidade de dado, atrasos de propagação, estratégias de retransmissão e espera em *buffers*. Os dados podem ser corrompidos ou perdidos durante o processo de transmissão, devido a ruídos no meio físico ou congestionamento no sistema de comunicação, acarretando uma degradação do serviço. Porém, aplicações multimídia em geral podem tolerar determinados níveis de perda de informação sem necessitar de mecanismos de retransmissão e correção. Em alguns casos, para satisfazer requisitos temporais (atraso limitado) ou obter sincronização (jitter baixo) alguns pacotes de dados podem vir a ser descartados. Deste modo, sistemas de comunicação multimídia não fazem uso de estratégias de retransmissão, uma vez que introduzem atrasos inaceitáveis. Além disto, devidos às características de algumas aplicações, como teleconferência, serviços de *multicast* tornam-se fundamentais para a otimização de banda de transmissão.

Em suma, os serviços de comunicação podem ser caracterizados pelos seguintes parâmetros de QoS:

- atraso de transmissão: tempo transcorrido desde a entrada do dado no sistema de comunicação até a sua retirada;
- jitter de transmissão: variação do atraso de transmissão;
- razão de perda: razão do número de bits perdidos pelo número total de bits enviados.

Geralmente, utiliza-se a razão de pacotes perdidos para caracterizar a informação de perda no serviço de comunicação.

Os recursos utilizados pelo serviço de comunicação são basicamente banda de transmissão e *buffers* de transmissão e recepção. Os níveis destes recursos têm impacto direto nos parâmetros de QoS da comunicação.

2.5.1 Serviço de Transporte

A responsabilidade deste serviço é efetuar o transporte dos dados da origem até o seu destino. Protocolos orientados a conexão com garantia de entrega de dados não são convenientes para transportar fluxos contínuos, pois esses protocolos utilizam mecanismos de retransmissão, de modo a terem entrega garantida. E, como já mencionado, devido às suas caraterísticas (altos volumes de dados e bom nível de redundância de informação), retransmissão não é factível em fluxos contínuos. Assim, protocolos do tipo TCP (*Transmission Control Protocol*) não são adequados para o transporte de fluxos contínuos.

No outro extremo temos os protocolos não orientados a conexão mais apropriados para efetuarem o transporte de fluxos contínuos que os orientados a conexão, pois não utilizam mecanismos de retransmissão de dados. Porém, não garantem a entrega dos dados, o que compromete a qualidade de serviço da aplicação como um todo. O protocolo UDP (*User Data Protocol*) é um representante típico desta classe de protocolos.

Os requisitos das aplicações multimídia demandam protocolos de transporte leves e com algum nível de suporte a qualidade de serviço. Entende-se por protocolos leves aqueles que apresentam baixo custo de transmissão, em geral advindo das seguintes características:

- utilização de baixa carga de informação de controle sobre os dados que transportam;
- não utilização de mecanismo de retransmissão;
- minimização de cópia de dados.

O suporte à qualidade de serviço pode ser fornecido através dos seguintes mecanismos:

- controle de *buffers* de recepção e transmissão;
- priorização de tráfego;
- monitoramemto de qualidade de serviço (atraso, jitter e perda de pacotes).

Visando atender aos requisitos dos fluxos contínuos, recentemente tem surgido várias propostas de protocolos de transporte projetados especialmente para esta finalidade. Dentre elas podemos destacar as seguintes: TPX, CMTP, HeiTS e RTP.

TPX (*Transport Protocol with eXtension*) [10] desenvolvido no projeto Esprit OSI 95, provê suporte a serviços orientados a conexão com entrega ordenada, QoS configurável e renegociável, além de notificação de erro.

CMTP (*Continuos Media Transport Protocol*) [15], protocolo de transporte da arquitetura Tenet da Universidade de Berkeley, Califórnia, fornece entrega ordenada e periódica, além de notificação de todos os dados não entregues ou entregues com erro.

HeiTS (*Heidelberg Transport System*) [30], da IBM de Heidelberg, foi projetado de modo a privilegiar a integração da qualidade de serviço do sistema de transporte com a gerência de recursos locais (principalmente o escalonamento de CPU). HeiTS emprega ênfase especial sobre um conjunto de *buffers* que minimiza cópias de dados e também permitem uma transmissão de dados eficiente entre dispositivos locais.

RTP (*Real Time Protocol*) [56], o mais popular deles, na realidade não é um protocolo de transporte propriamente dito mas sim um cabeçalho sobre um protocolo de transporte, TCP ou UDP, por exemplo. A sua configuração mais usual em aplicação multimídia é sobre o protocolo UDP. Seu nome se deve ao fato de seu cabeçalho só carregar informações de controle essenciais, sendo estas informações relacionadas a marca de tempo de transmissão, tipo de dado transmitido e número do pacote.

Como RTP não garante a entrega e nem a seqüência dos dados, assumindo que a qualidade do serviço será provida pelas camadas de comunicação mais baixas, torna-se necessário o monitoramento da qualidade de seu serviço. Para isto, há nos destinos dos fluxos RTP um protocolo de monitoramento de qualidade de serviço: o RTCP (*Real Time Control Protocol*). Este protocolo calcula estatísticas sobre a qualidade de serviço do transporte (jitter e perda de pacotes, por exemplo) a partir do cabeçalho do RTP e as transmite periodicamente num relatório à fonte de fluxo. RTCP também tem outras atribuições, como por exemplo desconectar receptores associados às fontes.

Outro protocolo utilizado para o transporte de dados multimídia é o XTP (*Xpress Transport Protocol*), desenvolvido pelo XTP Forum [17]. Apesar de não ter sido projetado especialmente para o transporte de dados multimídia, apresenta algumas características interessantes, como:

- independência entre paradigma e política de transmissão;
- separação entre controle de fluxo e controle de taxa de transmissão;
- suporte para comunicação *multicast*;
- escalonamento e priorização de mensagens;
- independência do serviço de entrega de dados.

2.5.2 Serviço de Rede

Serviços de rede basicamente incorporam atividades de endereçamento e roteamento, além de algum mecanismo de controle de erro. Para aplicações multimídia, particularmente, suporte a comunicação *multicast* é muito importante.

Além disto, para fornecer garantia de qualidade de serviço no nível de rede faz-se necessário a utilização de mecanismos de gerência de recursos, sendo que estes mecanismos

realizam especificação, controle de admissão, reserva de recursos (largura de banda e *buffers* de transmissão) e gerência de fila.

O protocolo IP (*Internet Protocol*), que é um padrão de fato, possui suporte a endereçamento *multicast*, porém não garante a qualidade de seu serviço. E visando justamente o suporte de qualidade de serviço, surgiram na literatura várias propostas de novos protocolos de rede. A maior parte deles foi desenvolvida em universidades e centros de pesquisas sem pretensões de utilização em larga escala.

No âmbito da Internet, houve propostas para suporte à qualidade de serviço. Dentre elas se destacam os protocolos ST-II e RSVP.

ST-II (*Stream Protocol*, versão 2) [65] é um protocolo orientado a conexão destinado a serviços *multicast* com facilidade de negociação e reserva de recursos. ST-II modela a reserva de recursos como um fluxo de dado relacionado à origem e estendido a todos os destinos via uma árvore de distribuição *multicast*, ou seja, o nível de reserva de recursos é definido pela fonte. Seu desenvolvimento foi descontinuado devido a nova versão do protocolo IP (IPv6).

RSVP (*Resource Reservation Protocol*) [73] é o mais promissor dos protocolos de reserva de recursos. Diferentemente do ST-II, RSVP é um protocolo não orientado a conexão, mantendo-se assim compatível com a filosofia do IP.

A principal inovação do RSVP é que a qualidade de serviço não é especificada pela fonte, mas sim pelo destino. O seu mecanismo de reserva de recursos opera da seguinte maneira: as fontes regularmente emitem uma mensagem especial denominada de *path*, que são transportadas como pacotes *multicast*; os destinos respondem com uma mensagem *reserve*, que especifica a qualidade de serviço requerida. Os roteadores pertencentes ao *path*, ao receberem a mensagem *reserve*, reservam os recursos requeridos pela mensagem.

2.5.3 Tecnologias de Rede

Aqui serão abordadas as principais tecnologias de redes, caracterizando a viabilidade de sua utilização em aplicações multimídia distribuídas. Para avaliar o quão apropriadas são estas diversas tecnologias de rede para atender aos requisitos das mídias contínuas, serão consideradas as seguintes características: banda de transmissão, estratégia de acesso ao meio, atraso de transmissão e comunicação *multicasting*. A confiabilidade é uma característica importante em aplicação multimídia, pois o processo de retransmissão é custoso, porém não será abordada aqui por considerarmos que a atual tecnologia de redes provê produtos confiáveis.

Para efeito de análise, as tecnologia de rede serão agrupadas em redes locais (LANs) e redes de longa distância (WANs). O termo rede para redes locais típicas refere-se claramente às camadas física e de acesso ao meio, sendo este termo menos claro no contexto de redes de longa distância.

Redes Locais (LANs)

O termo LAN designa redes que interligam computadores dispersos até algumas centenas de metros. Por sua limitação geográfica, tais redes são mais baratas e mais rápidas que as de longa distância. Em termos de funcionalidades, LANs incluem basicamente serviços da camada física e de acesso ao meio.

Quanto à estratégia de acesso ao meio, LANs atuais podem ser classificadas basicamente em: acesso múltiplo com detecção de colisão (CSMA-CD) e acesso por passagem de *token*. Redes locais que utilizam a estratégia CSMA-CD têm tipicamente estrutura física em barramento, enquanto as que se utilizam da estratégia de passagem de *token* possuem estrutura em anel. Ethernet e Fast Ethernet, com banda de transmissão respectivamente de 10 e 100Mbps (megabits por segundo) são padrões que utilizam a estratégia CSMA-CD; enquanto Token Ring e FDDI (*Fiber Distributed Data Interface*), com banda de transmissão respectivamente de 16 e 100Mbps, são exemplos de padrões que utilizam a estratégia por passagem de *token*.

CSMA-CD [60] é um método não determinístico e em situações de sobrecarga o número de colisões aumenta enormemente, acarretanto severa perda de desempenho ao sistema. Além disto, os padrões Ethernet e Fast Ethernat não provêem nenhum mecanismo de prioridade de acesso a meio, inviabilizando qualquer tratamento preferencial de tráfego de tempo real sobre o tráfego convencional.

Teoricamente, Ethernet pode distinguir até 246 diferente endereços de *multicast*, mas na prática um adaptador Ethernet gerencia um número limitado de endereços de grupo (*multicast*). Uma vez que a tecnologia Ethernet não exerce controle sobre o acesso ao meio e nem garante banda de transmissão, não é uma tecnologia de rede adequada para transmissão de fluxos contínuos. Porém, possui banda de transmissão suficiente para um número pequeno de fluxos, além de possuir função de *multicast*, o que a torna uma tecnologia aceitável apenas em aplicações modestas.

Fast Ethernet possui as mesmas limitações com respeito às características de acesso ao meio apresentada pela rede Ethernet e banda efetiva de transmissão raramente excede a 50 Mbps devido a presença de colisões. Devido a sua largura de banda máxima de 100 Mbps, Fast Ethernet pode suportar um maior número de fluxos contínuos que Ethernet. Porém, a rede não garante atraso máximo de acesso ao meio nem reserva de banda de transmissão, de modo que fluxos contínuos podem deteriorar em situações de sobrecarga. Assim, Fast Ethernet é uma escolha possível para configurações de pequeno a médio porte, não sendo uma boa alternativa para aplicações multimídia em geral, por não prever garantias qualidade de serviço.

Estratégia de acesso ao meio via passagem de *token* é mais aceitável que CSMA-CD para transmissão de dados multimídia, pois garante um tempo máximo de acesso ao meio para o pior caso e apresenta uma degradação de desempenho menos aguda em caso de sobrecarga que a estratégia CSMA-CD. Além disto, possui um mecanismo de prioridade na sub-camada de acesso ao meio (MAC), permitindo separar tráfego de alta prioridade dos de baixa. Porém, protocolos de rede como IP não utilizam este esquema de prioridade.

Token Ring suporta um esquema de endereçamento similar ao da Ethernet, ou seja, cada nó pode gerenciar um pequeno número de endereços *multicast*. A rede FDDI pode ser considerada conceitualmente como uma "Fast Token Ring" [54]. Além de tráfego prioritário, FDDI também suporta uma classe de tráfego síncrono, com conseqüente limitação de atraso de transmissão máximo. Porém, apesar de haver uma especificação do protocolo de tráfego síncrono este modo de operação não é disponibilizado nas redes FDDI comerciais.

Devido a essas características, redes Token Ring são viáveis para aplicações multimídia de pequeno porte. Já a rede FDDI, por possuir uma grande banda de transmissão, apresenta-se como uma boa alternativa para aplicações multimídia, mesmo sem disponibilidade da classe de tráfego síncrono.

Redes de Longa Distância (WANs)

Serviços de redes de longa distância incluem uma mistura de camadas e funções; ATM provê serviço de camada 2 ou 3, dependendo de sua utilização; Frame Relay uma mistura de serviços das camadas de enlace e de rede e a X.25 os serviços das camada física, enlace e rede.

As redes X.25 não provêem tráfego *multicasting* e se destinam a entrega confiável de pacotes X.25 sobre *links* de velocidades relativamente baixas. Seus requisitos de processamento e mecanismos de janelamento tornam o X.25 somente conveniente até velocidade de transmissão de 2 Mbits/s. Além disto, os mecanismos de recuperação de erro e controle de fluxo baseado em janela deslizante tornam os atraso de transmissão não controláveis, gerando alto *overhead* como baixa taxa de acesso. Assim sendo, X.25 não é uma rede aceitável para aplicações multimídia.

Frame Relay é uma tecnologia de rede por chaveamento de pacotes desenvolvida para transpor as limitações da rede X.25 em *links* de alta velocidade. Esta tecnologia de rede se constitui de um protocolo da camada de enlace, com provisão de serviços sobre diferentes redes físicas. A eficiência da rede Frame Relay é propiciada pela comutação rápida de pacotes, ou seja, pacotes são inspecionados somente no destino. Pacotes com erro são descartados (a rede não utiliza retransmissão com a rede X.25). Frame Relay tira proveito das baixas taxas de erro da transmissão óptica. Velocidades de acesso situam-se tipicamente entre 1,5 e 45 Mbits/s.

Frame Relay implementa políticas de policiamento de tráfego, mas, no que tange à qualidade de serviço, a rede procura honrar de forma estatística a banda de transmissão associada à conexão. Assim, mesmo tendo banda de transmissão satisfatória, Frame Relay não se apresenta como uma boa infra-estrutura de comunicação para fluxos multimídia.

As redes digitais de serviços integrados (ISDN) são WANs concebidas para suportar uma grande variedades de serviços, que vão desde dados de mídia estática até áudio e vídeo. ISDN é constituída de canais síncronos de 64 Kbps, que podem ser utilizados para tráfegos de razão de bits constante (CBR) e orientado a fluxo contínuo (CBO), ou ainda para comunicação por pacotes.

Quando se utiliza ISDN para tráfego CBO fim-a-fim, o atraso é constante com valor muito pequeno. Entretanto, num contexto de interconexão de LAN com WAN tem-se de considerar o dado multimídia empacotado, uma vez que LANs atuais não suportam tráfego CBO. Neste caso, quando maior o tamanho do pacote e menor a banda de transmissão maior será atraso de empacotamento dos dados.

ISDN suporta banda de transmissão de até 30 canais de 64 Kbps (1.920 Kbps) com características isócronas, o que é plenamente satisfatório para uma ampla gama de aplicações multimídia. Uma vez que nem X.25 e nem Frame Relay satisfazem este requisito, ISDN é atualmente uma opção disponível para comunicação multimídia interativa em WAN. Porém, devido a carência de serviços *multicast*, seu uso é mais voltado a aplicações ponto-ponto que aplicações de teleconferência com vários participantes.

Asynchronous Transfer Mode (ATM) é uma técnica chaveamento baseada em célula, que está sendo a base para a construção dos novos serviços de rede. A ITU declarou ATM como a tecnologia base para as futuras redes digitais de serviços integrados de faixa larga (B-ISDN) [11] [1]. Ao mesmo tempo, o ATM *Forum*, com centenas de membros, dentre eles os principais fornecedores de equipamentos de telecomunicações, se concentrou sobre a

definição de redes ATM privadas tanto para aspecto de LAN quanto de WAN. A arquitetura ATM constitui-se em um modelo de três camadas: camada física, camada ATM e camada de adaptação ATM. Este modelo não corresponde às três primeiras camadas do tradicional modelo de referência da ISO para redes de computadores.

A camada física ATM, a de mais baixo nível, define a transmissão física. A ITU definiu inicialmente um transmissão ótica de 155 Mbps. Por outro lado, o ATM *Forum* definiu uma interface de 100 Mbps para rede local. Interfaces de 622 Mbps e 2,4 Gbps de taxa de transmissão já são disponíveis.

A camada ATM é uma camada de chaveamento e multiplexacão independente da camada física. Ela define a estrutura da célula ATM. Por sua vez, células são cápsulas de informação de 53 bytes, sendo 48 bytes de dados e 5 bytes de cabeçalho. ATM é fundamentalmente uma rede orientada a conexão. A conseqüência disto é que antes de qualquer transferência de dado é necessário estabelecer um circuito virtual. Devido ao pequeno tamanho da célula, alta velocidade de transmissão dos *links* e chaveamentos velozes nos nós, redes ATM fornecem níveis de atraso de transmissão muito baixos. As pequenas células ATM podem ser entrelaçadas com um grau de granularidade bastante fino, o que reduz possíveis atrasos por espera em *buffers*. O ATM *Forum* definiu cinco categorias de conexões para esta camada [1]:

- Taxa de bit constante (CBR): Destina-se a aplicações que requeiram uma banda de transmissão constante e baixo atraso na comunicação, como transferência em tempo real de áudio e vídeo não compactado.

- Taxa de bit variável para tempo real (rt-VBR): Destina-se a aplicações com restrições de tempo real que requeiram banda de transmissão variável garantida e baixo atraso na comunicação, como é o caso de transmissão em tempo real de áudio e vídeo compactado.

- Taxa de bit variável - não tempo real (nrt-VBR): Destina-se a aplicações sem restrições de tempo real mas que requeiram banda de transmissão média garantida e comunicação confiável (baixa perda de células).

- Taxa de bit não especificada (UBR): Destina-se a serviço tipo melhor-esforço e sem garantia de qualidade de serviço.

- Taxa de bit disponível (ABR): Este serviço oferece às aplicações uma banda de transmissão que varia em função da carga da rede, sendo valores mínimo (eventualmente zero) e máximos estipulados pela aplicação e negociados com a rede. Há um mecanismo de realimentação para informar à aplicação a banda de transmissão disponível no momento.

A camada de adaptação ATM (AAL) foi concebida para tornar compatíveis a camada ATM com os requisitos da aplicação, de modo a prover as redes ATM com característica de multi-serviço.

Há várias propostas para camada de adaptação ATM, que em linhas gerais detinam-se a oferecer suporte aos serviços de conexão da camada ATM. A ITU propôs as seguintes camadas de adaptação: AAL 1, para serviços tipo CBR; AAL 2 (especificação ainda não concluída), para serviço tipo rt-VBR, e AAL 3/4 para serviços tipo nrt-VRB, UBR e ABR. O ATM *Forum* propôs a AAL 5 (equivalente à AAL3/4) e está estudando a especificação da

Tecnologia de Rede	Banda de Transmissão	Tipo de Uso do Meio	Atraso de Transmissão	Disponibilidade de *Multicast*
Ethernet	10 Mbps	compartilhado	limitado	sim
Fast Ethernet	100 Mbps	compartilhado	não limitado	sim
Token Ring	16 Mbps	compartilhado	limitado	sim
FDDI	100 Mbps	compartilhado	limitado	sim
X.25	menor que 2 Mbps	dedicado	não limitado	não
Frame Relay	menor que 50 Mbps	dedicado	não limitado	não
ISDN	múltiplo de 64 Kbps	dedicado	fixo	não
ATM	25-2400 Mbps	dedicado	limitado	sim

Tab. 2.3: Resumo das características de diversas tecnologias de rede.

AAL 6, destinada ao suporte de fluxo contínuo em pacotes, particularmente vídeo MPEG 1 e MPEG 2. Os atuais produtos comerciais provêem suporte para a AAL5, que de modo geral é uma camada mais leve que as demais. Outro aspecto importante é que ATM provê comunicação *multicast*. De forma geral, devido às altas taxas de transmissão, baixos atrasos e jitter e diversidade de tipos de tráfegos, ATM se apresenta com a tecnologia de rede que melhor atende aos requisitos dos sistemas multimídia.

Resumo das Características das Tecnologias de Rede

A Tab. 2.3 [60] resume as considerações sobre tecnologia de rede realizadas nas seções anteriores. A partir da Tab. 2.3, podemos observar que embora várias tecnologias de rede prometem suporte para tráfego multimídia, somente poucas dessas estão disponíveis atualmente. E dentre essas, ATM se apresenta com a tecnologia mais atraente para o suporte de aplicações multimídia.

2.6 Comentários Finais

Como analisado neste capítulo, aplicações multimídia distribuídas podem requerer e apresentar vários tipos de níveis de qualidade de serviço, que vão desde os serviços garantidos até os de melhor-esforço, passando pelos de garantia estatística. A qualidade de serviço está diretamente associada à quantidade de recursos destinada à aplicação e aos tipos de serviços disponíveis nos sistemas de computação e comunicação.

Além disso, como os fluxos de uma aplicação multimídia atravessam várias etapas durante o seu trajeto, que vai da captura à apresentação, a qualidade de serviço fim-a-fim é função da qualidade de serviço de cada etapa. Deste modo, a qualidade de serviço final depende não somente de reserva de recursos, mas também de uma política de seleção de serviços.

Uma vez que aplicações multimídia distribuídas executam sobre vários domínios e há uma grande variedades de padrões de serviços (compactadores, transporte e tecnologia de rede, por exemplo), a seleção de configuração e conseqüente reserva de recursos se tornam bastante complexas. Isto caracteriza a necessidade de procedimentos eficientes de especificação da qualidade de serviço em tais tipos de aplicações. E como habitualmente os serviços não são

deterministicamente garantidos, pois seria necessário reservar altos volumes de recursos com baixa utilização, há a necessidade de se gerenciar os recursos disponíveis da melhor forma possível.

Assim, o suporte à qualidade de serviço pode ser visto como a composição das seguintes atividades: especificação de qualidade de serviço, especificação de serviços, reserva de recursos e gerência de recursos.

No próximo capítulo serão abordados os requisitos necessários para o suporte adequado de qualidade de serviço das diversas atividades que compõem as aplicações multimídia distribuídas.

Capítulo 3

Suporte à Qualidade de Serviço em Aplicações Multimídia Distribuídas

Como analisado no capítulo anterior, os fluxos de informação em aplicações multimídia distribuídas atravessam várias etapas, desde a sua produção (captura) até seu consumo (apresentação), utilizando-se de vários serviços (compactação, transporte e descompactação, por exemplo) com vários níveis de qualidade de serviço. Assim, a qualidade de serviço fim-a-fim de uma aplicação multimídia é a composição da qualidade de serviço de cada componente da aplicação. Deste modo, a qualidade de serviço de uma aplicação multimídia distribuída está relacionada não somente à quantidade de recursos a ela destinados, mas também a uma política eficiente de configuração de serviços, pois a qualidade de serviço de uma aplicação multimídia é avaliada fim-a-fim. Isto caracteriza a necessidade de suporte à qualidade de serviço fim-a-fim em aplicações multimídia distribuídas.

Devido à sua relevância, analisaremos neste capítulo os requisitos para o suporte de qualidade de serviço em aplicações multimídia. Para isto, primeiramente será apresentado um ciclo de vida para aplicações multimídia, descrevendo-se as diversas atividades desempenhadas em cada etapa. Depois, serão analisados os conflitos inerentes a este processo. E, finalmente, serão apresentadas algumas das mais relevantes propostas de arquitetura para suporte à qualidade de serviço de aplicações multimídia disponíveis na literatura, caracterizando-se os seus princípios conceituais.

3.1 Ciclo de Vida de Aplicações Multimídia

Utilizaremos aqui um modelo de ciclo de vida de aplicações multimídia proposto por Hafid e von Bochmman [27]. Este modelo divide uma sessão multimídia em três fases sucessivas: estabelecimento, atividade e encerramento. Cada fase possui várias funções de gerência de QoS.

3.1.1 Fase de Estabelecimento de Sessão

Nesta fase são realizadas várias atividades com o intuito de preparar a sistema para a execução da aplicação. As atividades desenvolvidas nesta fase são: especificação de QoS, mapeamento de QoS, negociação de QoS e reserva de recursos.

Especificação de Qualidade de Serviço

A especificação de QoS está relacionada com a captura dos requisitos e políticas de gerência de qualidade de serviço no nível de aplicação. Assim, o usuário da aplicação deve estar habilitado a especificar os requisitos desejados em termos abstratos, o que confere à especificação de QoS uma natureza fortemente declarativa. A especificação de QoS poderá englobar os seguintes aspectos [2]:

- Especificação de sincronismo entre fluxos: especifica o grau de sincronismo entre vários fluxos relacionados temporalmente.
- Especificação de desempenho de fluxos: caracteriza os requisitos do usuário para o desempenho dos fluxos, descritos por parâmetros de QoS. Estes parâmetros podem ser expressos em várias formas, como em faixas de valores (valores mínimos e máximos).
- Nível do serviço: especifica o grau de garantia da qualidade de serviço e pode ser caracterizado por determinístico, estatístico ou melhor esforço.
- Política de gerência de qualidade de serviço: captura o grau de adaptação, discreto ou contínuo, que um fluxo pode experimentar, além da forma como será realizado o monitoramento de QoS (monitoramento síncrono ou assíncrono).
- Custo do serviço: especifica o preço que o usuário está disposto a pagar pelo serviço, sendo este um fator limitante da qualidade de serviço. Caso não haja custo de serviço envolvido na especificação, o sistema tentará prover o máximo da qualidade especificada.

Mapeamento de Qualidade de Serviço

Esta atividade engloba funcionalidades de translação automática de representações de QoS nos diversos níveis da aplicação, de modo a liberar o usuário de encargo de especificar a qualidade de serviço em termos de característica de baixo nível do sistema. Assim, torna-se necessário que o sistema possa manipular e gerenciar várias formas de representação de QoS.

De forma geral, há basicamente três tipos de mapeamento de representação de QoS:

- Mapeamento entre parâmetros de QoS: mapeia parâmetros de QoS descritos no nível do usário em parâmetros de QoS descritos no nível do sistema.
- Mapeamento entre parâmetros de QoS e recursos do sistema: mapeia parâmetros de QoS descritos no nível do sistema em quantidade de recursos necessária para atender tais requisitos de qualidade.
- Mapeamento entre serviços e componentes do sistema: determina em quais componentes do sistema os diversos serviços serão realizados, realizando também a configuração desses serviços de modo a atender aos requisitos de QoS.

Negociação de Qualidade de Serviço

O objetivo da negociação de QoS é encontrar uma configuração de serviços de modo que atenda aos requisitos de qualidade de serviço da aplicação. O processo de negociação de QoS é realizado com todos os componentes e usuários de uma aplicação multimídia. Os aspectos envolvidos na negociação são basicamente os parâmetros de QoS, limites de custos e tipos dos serviços. Esta atividade de negociação é desempenhada após a especificação de QoS e antes da reserva de recursos, caso a negociação seja bem sucedida.

Há várias possibilidades de abordagem para o processo de negociação de QoS [10], porém o princípio do procedimento de negociação é basicamente o mesmo: atender de melhor forma possível aos requisitos da especificação de QoS da aplicação em face das limitações de recursos e ou custos.

Por sua relevância no suporte à qualidade de serviço, a atividade de negociação de QoS será abordada em detalhes mais a frente.

Reserva de Recursos

Após um bem sucedido procedimento de negociação de recursos, há a necessidade de reservar recursos em quantidade suficiente para atender aos níveis de QoS e tipos de serviços negociados. Os recursos reservados são basicamente tempo de CPU, dimensões de *buffers*, banda de transmissão e periféricos.

Há duas abordagens possíveis para reserva de recursos [46]: pessimista e otimista. Na abordagem pessimista a reserva de recursos é feita com base no pior caso de utilização de recursos. Já na abordagem otimista, reserva-se recursos tomando como base valores médios de utilização de recursos. A reserva pessimista é típica de serviços com QoS garantidos deterministicamente e a otimista é mais própria dos serviços com garantia de QoS estatística, sendo que a primeira abordagem leva a uma sub-utilização dos recursos, enquanto que a segunda utiliza os recursos de forma mais racional.

3.1.2 Fase de Atividade de Sessão

É nesta fase que que as aplicações multimídia são executadas. As atividades realizadas nesta fase são: monitoramento de QoS, adaptação de QoS, mapeamento de QoS, renegociação de QoS, policiamento de fonte, controle de admissão e tarifação de QoS.

Monitoramento de Qualidade de Serviço

Esta atividade se destina a mensurar os vários níveis de QoS da aplicação. Estas medidas podem servir como realimentação para as atividades de tarifação, adaptação e renegociação de QoS, a medida que sejam detectadas violações de qualidade de serviço. Os valores medidos são basicamente os parâmetros de QoS.

Os procedimentos de monitoramento de QoS são de natureza síncrona, porém suas notificações podem ser síncronas, assíncronas ou uma composição de ambas.

Adaptação de Qualidade de Serviço

A atribuição da adaptação de QoS é manter da melhor forma possível os níveis de qualidade de serviço acordados na fase de negociação. Neste sentido, uma atividade de adaptação de QoS deve ser hábil para atuar no sistema sempre que violações de QoS sejam detectadas.

Os procedimentos de adaptação de QoS atuam basicamente nas reservas de recursos e necessitam de mecanismos e políticas eficientes de redistribuição de recursos. Porém, caso a adaptação de QoS não consiga manter os níveis de QoS acordados, deve ser gerada uma notificação de violação de QoS com propósito de renegociação de QoS.

Mapeamento de Qualidade de Serviço

Nesta fase, as atividades de mapeamento de QoS estão basicamente relacionadas ao mapeamento dos parâmetros de QoS mediados na atividade de monitoramento de QoS para os parâmetros de QoS no nível do usuário. Esse tipo de mapeamento de QoS se destina a manter informado o usuário sobre o nível de qualidade corrente da aplicação.

Em situações de renegociação de QoS, há atividades de mapeamento de QoS similares às desempenhadas na fase de estabelecimento de sessão.

Renegociação de Qualidade de Serviço

Atividades de renegociação de QoS podem ser iniciadas pelos usuários da aplicação ou pelo próprio sistema. Quando iniciada por um usuário, a renegociação geralmente tem caráter de melhoria dos requisitos de qualidade de serviço, enquanto que quando iniciada pelo sistema provavelmente é decorrente da detecção de violação do nível de QoS acordado durante a atividade de negociação de QoS. Tipicamente, este segundo caso de renegociação de QoS deverá ser iniciado somente quando a adaptação automática não é factível.

Em certa medida, os procedimentos de renegociação de QoS são similares aos de negociação de QoS, porém na renegociação parte-se de um nível de recursos que já foram reservados na fase de negociação.

Policiamento de Fonte

Esta atividade concerne a um conjunto de ações de monitoramento e controle, realizadas pelo sistema de modo a proteger o usuário da informação de fontes não confiáveis. Um cenário típico seria o caso de uma fonte estar transmitindo informação com qualidade bem acima dos níveis acordados, o que poderia causar sobrecarga no consumidor, com conseqüente comprometimento dos níveis de qualidade de serviço dos outros eventuais fluxos da aplicação.

Controle de Admissão

A responsabilidade do controle de admissão durante a fase de atividade da sessão é comparar os recursos requeridos pela aplicação com os disponíveis pelo sistema. Sendo que a decisão de aceitar ou não um novo serviço irá depender do nível de recursos disponíveis na ocasião.

A atividade de controle de admissão é fortemente relacionada às atividades de renegociação de QoS e reserva de recursos pois, antes de se efetivar a reserva de recursos, a

renegociação de QoS se utiliza de mecanismos de controle de admissão para verificar se os requisitos de qualidade de serviço requeridos pelo novo serviço podem ser atendidos.

Tarifação de Qualidade de Serviço

A atividade de tarifação concerne à forma como calcular os custos dos serviços utilizados. Este cálculo é função de vários parâmetros, como por exemplo: quantidade de recursos reservados, tempo de reserva dos recursos, tipo de garantia de QoS e horário de utilização do serviço.

3.1.3 Fase de Encerramento de Sessão

Nesta fase são realizadas atividades que visam manter a integridade do sistema após o término da sessão, pois quando um serviço é terminado, todos os recursos reservados devem ser liberados. A atividade de término deve enviar uma notificação para os componentes do sistema envolvidos na provisão do serviço para que liberarem os recursos e atualizarem a quantidade de recursos disponíveis.

3.1.4 Relações Entre as Atividades do Ciclo de Vida

As diversas atividades componentes de um ciclo de vida de uma sessão multimídia estão inter-relacionas entre si. E são justamente as interações entre essas atividades que caracterizam a dinâmica de uma sessão multimídia.

Em relação à fase de estabelecimento, primeiramente temos a especificação de QoS dos fluxos de dados da aplicação multimídia, que é descrita através de parâmetros de QoS (possivelmente em forma declarativa), limite de custos e tipo de garantia de qualidade de serviço. Depois, é realizado o mapeamento entre parâmetros de QoS e entre parâmetros de QoS e recursos do sistema. Em seguida é realizada a negociação de QoS com todos os componentes e usuários envolvidos na aplicação, com conseqüente estimativas de recursos a serem utilizados, reserva de recursos e seleção de configuração. Estas diversas atividades que compõem a fase de estabelecimento de uma sessão multimídia podem ser visualizadas de forma mais nítida através de um diagrama de fluxo, como mostrado na Fig. 3.1. No caso, devido à sua complexidade, a atividade de negociação é dividida em várias sub-atividades, que envolvem negociação ao nível de sistema, mapeamento de parâmetros de QoS, seleção de configuração e previsão de desempenho da aplicação.

Já a fase de atividade de uma sessão multimídia, devido às suas características dinâmicas, é melhor modelada via máquina de estados, como é mostrado na Fig. 3.2.

A fase de encerramento de uma sessão multimídia é composta basicamente pela liberação de recursos e sinalização de fim de sessão, caracterizados por término de sessão.

Então, a partir das análises realizadas sobre as fases de uma sessão multimídia podemos visualizar três aspectos fundamentais relacionados ao suporte de qualidade de serviço de aplicações multimídia. Estes aspectos são: i) caracterização de qualidade de serviço; ii) negociação de qualidade de serviço entre os componentes da aplicação e iii) gerência da qualidade de serviço. A primeira questão está relacionada a como realizar adequadamente o mapeamento entre parâmetros de qualidade de serviço de diferentes níveis, além de relacionar a qualidade de serviço aos recursos disponíveis. O segundo problema está relacionado a como

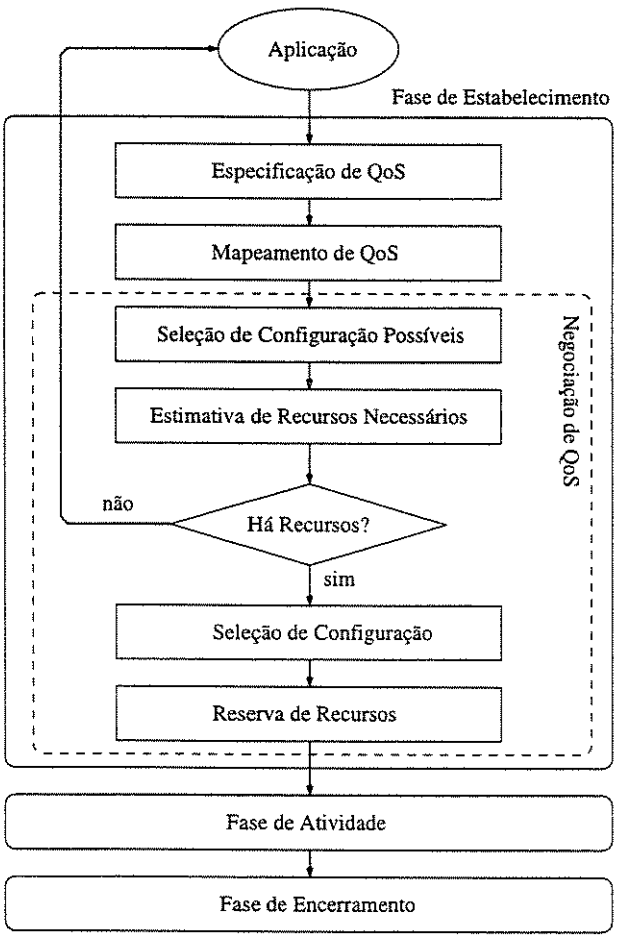

Fig. 3.1: Diagrama de fluxo das atividades de especificação de sessão multimídia.

efetuar a distribuição dos recursos entre os diversos fluxos de informação, com posterior seleção de uma configuração adequada (problema da negociação de qualidade de serviço). Finalmente, a terceira questão concerne à sustentação dos níveis de QoS acordados na etapa de negociação.

3.2 Conflitos no Processo de Negociação de Qualidade de Serviço

Como já abordado, a negociação de QoS se apresenta como uma questão central tanto para o estabelecimento quanto para a manutenção de uma sessão multimídia. Por negociação de QoS entende-se estabelecer uma política de reserva de recursos e seleção de configuração da aplicação. Por isto, torna-se míster caracterizar de forma adequada as premissas que constituem a atividade de negociação de QoS em uma sessão multimídia. Para tanto, neste trabalho definimos a entidade "negociador de QoS".

O negociador de QoS é uma entidade que visa atuar no sistema de modo a melhor atender às especificações de QoS da aplicação, ou seja, negocia reserva de recursos a partir de uma

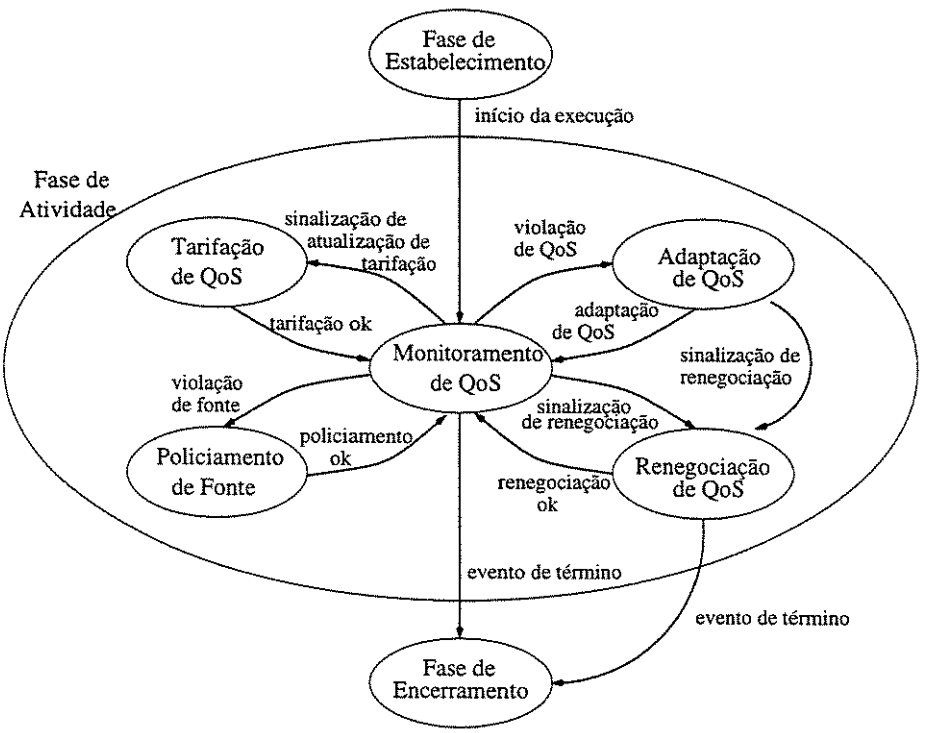

Fig. 3.2: Máquina de estados da Fase de Atividade de uma sessão multimídia.

política de otimização da qualidade de serviço provida pelo sistema. Para tanto, o negociador de QoS se vale de um conjunto de parâmetros que caracteriza a qualidade do serviço, como os parâmetros de QoS do nível de sistema (atraso fim-a-fim, jitter fim-a-fim e perda de pacotes) e do nível de usuário (definição, fidelidade, dinâmica e interatividade), ambos descritos no Capítulo 2.

Melhorar a qualidade de serviço da aplicação consiste em melhorar tais parâmetros. Porém, um impasse decorre do fato de que as melhorias individuais desses parâmetros são conflitantes, ou seja, a melhoria de um parâmetro de QoS pode vir a prejudicar o desempenho de outros. Esta característica nos leva a uma solução de compromissos.

O aumento de recursos (banda de transmissão e tempo de CPU) para um dado fluxo de mídia contínua implica na melhoria dos parâmetros de QoS (atraso fim-a-fim, jitter fim-a-fim e perda de pacotes). Neste caso não há relações de conflitos, mas apenas de benefícios. Porém, infelizmente, em sistemas reais não é possível se aumentar indiscriminadamente a quantidade de recursos destinada a um dado fluxo, pois há outros fluxos de dados competindo pelos recursos limitados.

Caso se mantenha a mesma quantidade de recursos de processamento e de comunicação, ainda podemos afetar os parâmetros de QoS via a utilização de *buffers* tanto na recepção quanto na transmissão. No caso, as relações de compromissos apresentam as seguintes características:

- Aumentando-se a capacidade dos *buffers* teremos um atraso fim-a-fim maior, porém com menor jitter fim-a-fim.

- Diminuindo-se a capacidade dos *buffers* teremos um atraso fim-a-fim menor, mas com o compromisso de termos maiores taxas de jitter fim-a-fim.

Ciente da dimensão dos recursos disponíveis para cada fluxo de dados ainda é importante que o negociador de QoS selecione uma configuração adequada, caso contrário pode haver uma deterioração nos parâmetros de QoS no nível de sistema. Um bom exemplo deste fato seria a seleção de uma taxa de exibição de vídeo que produza vazão muito superior à largura de banda de transmissão disponível a este fluxo, levando invariavelmente a uma elevação na taxa de descarte de quadros devido a *buffer-cheio*.

Assim, a eficácia do negociador de QoS está condicionada à sua habilidade em conciliar a política de reserva de recursos como os critérios de seleção de configuração. Observa-se que os recursos do sistema (largura de banda de transmissão, tamanho de *buffer* e *slots* de CPU) estão mais associados aos parâmetros de QoS do nível de sistema, enquanto a seleção de configuração (algoritmo de compactação de dados, qualidade de exibição de áudio e vídeo, etc.) está mais vinculada aos parâmetros de QoS do nível de usuário.

O nível de sincronismo entre fluxos de áudio e vídeo tem grande impacto na caracterização da qualidade de serviço de uma dada aplicação, o que o credencia como um parâmetro de QoS. De uma forma geral, o procedimento de sincronismo entre fluxos de informação é realizado a partir da determinação de um fluxo mestre e de fluxos escravos. No caso de sincronismo entre áudio e vídeo o fluxo mestre é, via de regra, o fluxo de áudio, enquanto o de vídeo é designado como escravo. Isto decorre do fato de que o fluxo de áudio é mais sensível a jitter e a perda de informação do que o fluxo de vídeo [64]. Então, a partir desse procedimento, o algoritmo de sincronização atua de forma a cadenciar o fluxo escravo em função do fluxo mestre, o que pode vir a provocar acréscimo de atraso fim-a-fim, jitter fim-a-fim e descarte de quadros de informação no fluxo escravo, deteriorando sua qualidade de exibição.

Assim, pode-se notar que esse procedimento de sincronismo atua no sistema de modo a cadenciar o fluxo mestre, o que implica na melhoria da qualidade de sua apresentação, porém isto é conseguido, via de regra, às custas da deterioração da qualidade de exibição do fluxo escravo. Logo, este cenário caracteriza uma relação de conflito entre as melhorias no sincronismo e na qualidade do fluxo escravo.

Outro aspecto de grande importância é a existência de dois estágios de negociação de QoS em aplicações multimídia distribuídas, que são: negociação local e negociação remota. Primeiramente o negociador de QoS realiza uma negociação local, de modo a verificar se há recursos e infra-estrutura locais que atendam aos requisitos da especificação de QoS da aplicação. Caso haja infra-estrutura local, o negociador verifica se há infra-estrutura remota, de modo que os requisitos de qualidade de serviço fim-a-fim sejam atendidos.

3.3 Arquiteturas de Suporte de Qualidade de Serviço

Tem surgido recentemente várias propostas de arquitetura para suporte de qualidade de serviço de aplicações multimídia distribuídas. Propostas iniciais se destinavam ao suporte de qualidade de serviço apenas de alguns componentes do sistema, como rede ou sistema operacional. Já as mais recentes se preocupam com suporte de QoS fim-a-fim.

Atualmente não há consenso sobre qual abordagem é mais adequada ao suporte de QoS para sistemas multimídia distribuídos. Porém, Aurrecoechea e outros em [2] propõem cin-

co princípios funcionais que devem nortear a construção de uma arquitetura para suporte de QoS em sistemas multimídia distribuídos. Estes princípios são: integração, separação, transparência, gerência assíncrona de recursos e desempenho.

O princípio da integração estabelece que a qualidade de serviço deve ser passível de configuração, predição e gerência em todas as camadas da arquitetura de modo a suportar QoS fim-a-fim.

O princípio da separação estabelece que as funcionalidades de gerência, controle e processamento/transmissão do fluxo devem ser atividades distintas na arquitetura. Isto decorre do fato dessas atividades terem natureza distintas. Por exemplo, transferência de fluxo geralmente requer altas bandas de transmissão, baixos atrasos e serviço de transporte não garantido, além de algum mecanismo de controle de jitter; enquanto atividades de controle requerem baixa banda de transmissão, serviço de transporte garantido e nenhum mecanismo de controle de jitter.

O princípio da transparência estabelece que a aplicação não deve se envolver com detalhes das atividade de estabelecimento e gerência de QoS, tais como mapeamento, negociação e monitoramento de QoS. Os benefícios da transparência são basicamente três: redução da necessidade de se embutir funcionalidades de QoS na aplicação; esconder da aplicação detalhes de serviços de baixo nível e delegar para as camadas mais baixas a complexidade das atividades de gerência de QoS.

O princípio de gerência assíncrona de recursos objetiva a divisão das atividades em módulos para fins de controle e gerência de QoS. Este princípio se baseia no fato de que um sistema multimídia distribuído é constituído de vários componentes que possuem restrições de tempo particulares, além de executarem concorrentemente.

E por fim, o princípio de desempenho prescreve que o desenvolvimento de uma arquitetura de suporte de QoS deve incorporar técnicas que privilegiem o desempenho das suas aplicações.

A seguir apresentaremos resumidamente algumas das propostas mais representativas de arquitetura para suporte de QoS existentes na literatura.

3.3.1 A Arquitetura Tenet

A arquitetura Tenet [14], desenvolvida na Universidade da Califórnia-Berkeley, EUA, tem por objetivo prover garantia de QoS até o nível de transporte. Mais precisamente, esta arquitetura é uma composição de vários protocolos ao nível de rede e de transporte. Os protocolos ao nível de transporte são: RMTP (*Real-time Message Transport Protocol*) e CMTP (*Continuos Media Transport Protocol*). No nível de rede é utilizado o protocolo de rede RTIP (*Real Time Internet Protocol*).

RMTP é um protocolo leve orientado a conexão e com garantia de desempenho, mas entrega de dados não confiável. Suas principais funcionalidades são o controle de fluxo, via controle de vazão, e a fragmentação e remontagem de mensagens.

CMTP foi projetado para o suporte de mídias contínuas. É um protocolo leve que fornece entrega periódica e ordenada de amostras de dados de mídias contínuas com controle de QoS sobre atraso, banda de transmissão utilizada e percentual de perda.

RTIP é um protocolo orientado a conexão e com garantia de desempenho, mas sem entrega confiável de pacotes.

Todos os protocolos de transmissão obedecem a administração de recursos feita pelo

protocolo RCAP (*Real-time Channel Administration Protocol*), que tem funcionalidades de reserva de recursos, controle de admissão e gerência de QoS de rede. Este protocolo também fornece garantia de serviços com restrições determinísticas e estatísticas. Durante a transmissão o usuário pode requerer informações ao RCAP sobre o estado dos recursos. Entretanto, os protocolos de transporte e de rede não interagem com o RCAP.

Um grande problema com esta abordagem é a utilização protocolos que não têm grande difusão, o que dificulta sua utilização com as aplicações já existentes.

3.3.2 A Arquitetura QoS-A

Esta arquitetura de suporte de QoS foi proposta pelo grupo de sistemas distribuídos da Universidade de Lancaster [7], Inglaterra, com cooperação da Netcomm Ltd., fabricante de chaves ATM. QoS-A propõe uma visão integrada de suporte de QoS, que inclui tanto rede quanto processamento. Mais especificamente, ela é composta de quatro camadas: plataforma de sistema distribuído; orquestração; transporte e rede. Este modelo é ilustrado na Fig. 3.3.

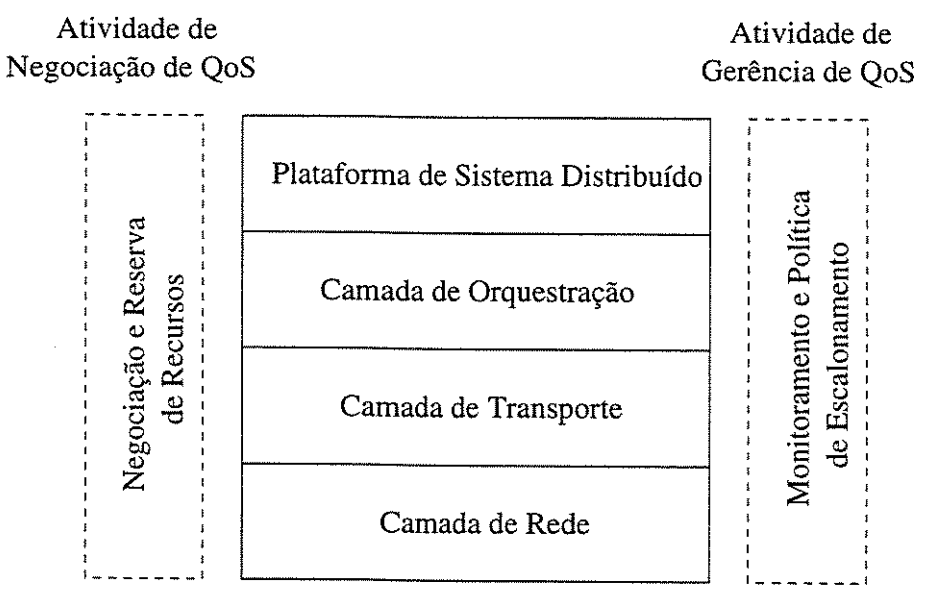

Fig. 3.3: Modelo da arquitetura QoS-A.

A camada de mais alto nível consiste de uma plataforma orientada a objeto para suporte a aplicações distribuídas, acrescida de serviços de comunicação de fluxos contínuos e especificação de QoS.

A camada de orquestração fornece serviços de sincronização entre fluxos, controle de jitter (sincronização intra-fluxo) e regulação da periodicidade dos fluxos contínuos. De forma geral, esta camada oferece um conjunto de primitivas de suporte à sincronização de fluxos.

A camada de transporte, que é a principal contribuição desse modelo, é constituída do protocolo de transporte METS (*Multimedia Enhanced Transport System*) projetado especificamente para o transporte de mídias contínuas. METS incorpora compartilhamento de *buffer*, regulação de tráfego, escalonamento e monitoramento de fluxo. Para reservar e adaptar o protocolo de recursos, uma gerência de recursos é utilizada. Os parâmetros de QoS

relevantes são: vazão de transmissão, atraso fim-a- fim, jitter, prioridade, PER e política de gerência de erro. Três tipos de serviços são fornecidas: determinístico, estatístico e melhor esforço.

A camada de rede é uma infra-estrutura de comunicação adaptada para suportar os requisitos dos fluxos contínuos. A principal funcionalidade desta camada é fornecer gerência de recursos para garantia de QoS segundo três classes: QoS determinístico, QoS estatístico e melhor-esforço.

No caso, admite-se que a comunicação de mídias contínuas é feita por uma rede ATM, que impõe uma limitação à arquitetura.

A noção de fluxo é o conceito arquitetônico mais relevante nessa abordagem. O conceito de fluxo considera a manipulação de uma mídia contínua como uma atividade integrada. Os requisitos de QoS do usuário e o grau de compromisso de serviço do provedor são formalizados num contrato de serviço acordado por ambas as partes. QoS-A suporta especificação de QoS, mapeamento de QoS, renegociação de QoS e adaptação de QoS.

3.3.3 A Arquitetura OMEGA

A arquitetura OMEGA [45], desenvolvida na Universidade da Pennsylvania, EUA, assume a existência de um subsistema de rede que fornece atrasos e erros limitados de modo a atender às demandas de banda de transmissão e um sistema operacional capaz de fornecer garantias de QoS em tempo de execução. Assim, a essência da arquitetura OMEGA são a reserva de recursos e gerência de QoS fim-a-fim. A arquitetura se constitui da integração de dois modelos: modelo de comunicação e modelo de recursos, que são respectivamente os seus componentes distribuído e local.

O modelo de comunicação, Fig. 3.4, por sua vez, possui dois subsistemas: subsistema de aplicação e subsistema de transporte. O subsistema de transporte incorpora funcionalidades das camadas de rede e de transporte através do protocolo RTNP (*Real-Time Network Protocol*), que tem atribuições de gerência de conexões, correção de erro, deteção de falhas e transporte de dados.

O subsistema de aplicação possui o protocolo RTAP (*Real-Time Application Protocol*), que contém as funcionalidades de aplicação e sessão, de modo a suportar os requisitos da aplicação. Exemplos desses serviços são: gerência de chamada e controle de vazão de dados nos dispositivos. Os serviços oferecidos pelo subsistema de aplicação fazem uso dos serviços oferecidos pelo subsistema de transporte.

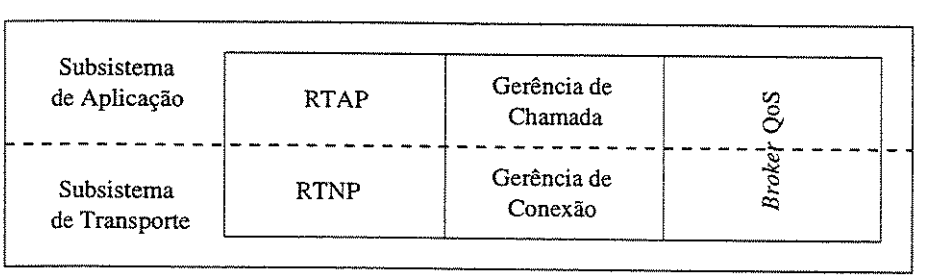

Fig. 3.4: Modelo de comunicação da arquitetura OMEGA.

Neste sentido, ambos os subsistemas devem oferecer serviços com QoS garantidos para chamadas e conexões especificadas pela aplicação. Mais precisamente, a garantia de QoS é negociada por uma entidade denominada de Broker QoS durante a fase de estabelecimento de sessão. O Broker QoS é responsável pela gerência dos recursos locais e globais.

A garantia de QoS é advinda de mecanismos de reserva de recursos. Esta atividade é suportada pelo modelo de recursos, Fig. 3.5, que possui três componentes: aplicação, sistema operacional e sistema de comunicação. A aplicação especifica seus requisitos de recursos usando parâmetros de QoS de alto nível, como freqüência de amostragem e resolução da amostra. O sistema operacional fornece recursos de processamento e comunicação local, como tempo de CPU e *buffers*. O sistema de comunicação fornece recursos de rede, como banda de transmissão. Com relação ao modelo de comunicação, a aplicação está localizada na camada do subsistema de aplicação e gerencia dispositivos multimídia; o sistema de comunicação está localizado na camada do subsistema de transporte, enquanto o sistema operacional possui funcionalidades nas duas camadas do modelo de comunicação.

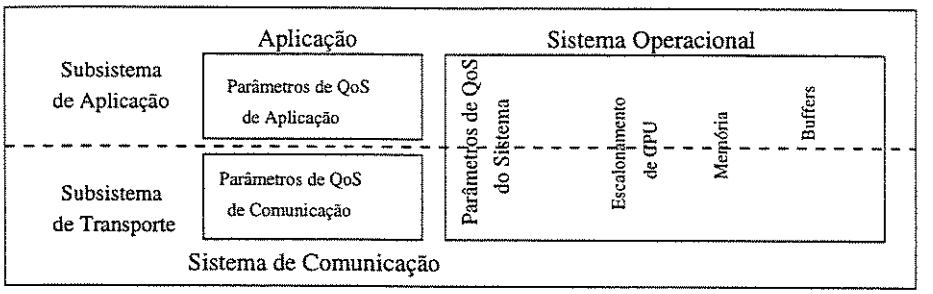

Fig. 3.5: Modelo de recursos da arquitetura OMEGA.

Este modelo utiliza protocolo de reserva de recursos e de comunicação próprios, que assumem a existência de sistema operacional e sistema de comunicação com QoS garantido, via reserva de recursos. Tais características limitam o escopo de aplicações dessa arquitetura.

3.3.4 Outras Propostas

As arquiteturas apresentadas nas seções anteriores têm em comum o fato de serem oriundas de projetos que não tinham a preocupação de seguir padrões. Como isto, essas abordagem desenvolveram modelos e protocolos de comunicação específicos, que dificultam sua utilização em larga escala.

Por outro, há na literatura várias outras propostas de modelo para o suporte de QoS em sistemas multimídia distribuídos que tem forte preocupação com aspectos de padronização dos serviços envolvidos nas aplicações multimídia. Estas propostas surgiram de diversos segmentos, como comunidade de telecomunicações, comunidade de computação, entidades internacionais de padronização e universidades e centros de pesquisas.

Da comunidade de telecomunicações a proposta mais relevante é a do TINA-C (*Telecommunication Information Networking Architecture - Consortium*) [47] [12], que é um consórcio de empresas da telecomunicações e de computação. Este consórcio propôs uma arquitetura para suporte de serviços distribuídos abertos para telecomunicações, considerando multimídia

distribuída como uma aplicação de grande relevância. Essa arquitetura, que é fortemente baseada no modelo de referência ODP (*Open Distributed Processing*) da ISO [71], provê uma especificação de serviços de suporte da qualidade de serviço num ambiente distribuído. Porém, o seu modelo de serviços não endereça aspectos quantitativos do suporte de QoS, caracterizando-se, assim, mais propriamente como um modelo de referência.

Uma das principais propostas oriundas das entidades de padronização para o suporte da QoS foi o QoS *Framework* da ISO[39]. Esse modelo define terminologia e conceitos para qualidade de serviço e fornece um modelo que identifica objetos de interesse para o suporte da QoS em sistemas abertos. A qualidade de serviço associada com objetos e suas interações é descrita através da definição de um conjunto de características que incorporam o conceito de especificação, monitoramento, controle e manutenção de QoS fim-a-fim.

Há ainda várias propostas provenientes de universidade e centro de pesquisa, como a arquitetura XRM (*Extended Integrated Reference Model*) [42], desenvolvida na Universidade de Columbia, EUA, que guarda muita equivalência com a arquitetura QoS-A.

Diante das várias propostas de modelos e arquiteturas de suporte de QoS para sistemas multimídia distribuídos, não seria apropriado buscar identificar a melhor abordagem, pois elas possuem propósitos e enfoque diversos entre si. Assim, elas devem ser analisadas mais propriamente a partir de seus requisitos de concepção, de modo a verificar se eles foram alcançados. Porém, é necessário que se possa determinar alguns critérios que indiquem o quão convenientes, amplas e eficientes são essas diversas abordagens.

Deste modo, os princípios de integração, separação, transparência, gerência assíncrona de recursos e desempenho, proposto em [2], se mostram valiosos indicadores funcionais no projeto de arquiteturas de suporte de QoS de sistemas multimídia distribuídos. Porém, como tais sistemas necessitam fornecer uma vasta diversidade de serviços num ambiente distribuído e heterogêneo, as arquiteturas de QoS necessitam incorporar nas suas concepções não somente conceitos funcionais, mas também conceitos de modelagem que propiciem flexibilidade, modularidade, encapsulação e simplicidade. Assim, a partir dessa perspectiva, acreditamos que as abordagens existentes, mesmo que possam atender aos requisitos funcionais do suporte de QoS, não atendem satisfatoriamente àqueles requisitos de modelagem, o que pode dificultar severamente as atividades de ampliação e manutenção de suas funcionalidades.

Apresentaremos no próximo capítulo uma arquitetura baseada em sistemas de agentes para provisão de QoS em sistemas multimídia distribuídos, que foi concebida justamente para privilegiar a flexibilidade, modularidade, encapsulação e simplicidade do modelo. Além disto, esta arquitetura não é vinculada a nenhuma tecnologia em especial, podendo assim operar sobre sistemas orientados a melhor-esforço, como redes TCP/IP e sistemas operacionais convencionais (*Unix-like*).

Capítulo 4

Um Modelo Baseado em Agentes Para Suporte à Qualidade de Serviço

Neste capítulo será apresentado um modelo para negociação e gerência de qualidade de serviço de sistemas multimídia. Visando privilegiar a autonomia e flexibilidade dos processos de negociação e gerência de qualidade de serviço nas aplicações multimídia distribuídas, este modelo utiliza o paradigma de sistemas baseados em agentes. Alguns agentes são móveis, ou seja, eles podem suspender suas execuções numa máquina e reiniciá-las em outra máquina; enquanto outros são fixos, isto é, executam integralmente em uma só máquina.

O capítulo está dividido da seguinte forma: primeiramente é realizada uma breve abordagem sobre sistemas baseados em agentes (SBA). A seguir é apresentado o modelo baseado em agentes móveis e fixos para suporte à qualidade de serviço em sistemas multimídia, sendo descritas as suas funcionalidades ao longo de todo ciclo de vida da aplicação multimídia.

4.1 Sistemas Baseados em Agentes (SBA)

Sistemas baseados em agentes (SBA) são sistemas onde as entidades básicas de computação são agentes, os quais trabalham cooperativamente na solução de problemas. O ambiente onde agentes habitam e trocam informações é denominado agência. O conceito de agente, por sua vez, foi primeiramente empregado na área de inteligência artificial, para a solução de problemas distribuídos [31]. Esta área de aplicação ficou conhecida como sistemas multi-agentes (SMA). Neste contexto, agentes são vistos como entidades autônomas dotadas de certo grau de inteligência, que trabalham de forma coordenada e cooperativa na busca de soluções para problemas comuns.

Atualmente, o paradigma de sistemas baseados em agentes é empregado na solução de problemas em diversas áreas, como robótica, otimização, interface homem-máquina, simulação de comportamento social e comércio eletrônico. Nwana em [48] apresenta uma vasta taxonomia de agentes. Porém, apesar de toda atividade nesta área, ainda não há consenso na literatura sobre uma definição para agentes, podendo-se, entretanto, detectar um conjunto de características bem aceito para caracterizá-los.

Para Hewitt [31] agente é uma entidade de *software* autocontida que possui estado interno, identidade e comportamento próprio, e que se comunica com outros agentes ou com seu ambiente através de mensagens. Uma definição mais atual, fornecida por Virdhagriswaran

[66], considera o termo agente como designador de dois conceitos básicos: i) a habilidade do agente para a execução autônoma e ii) a habilidade do agente em reagir em função do ambiente. Nesta definição, autonomia é um aspecto chave. Já para Russel e Norvig [55], um agente é algo que pode perceber seu ambiente, por meio de seus sensores, e atua sobre ele através de seus atuadores. Esta definição equivale ao segundo conceito da definição anterior e de certo modo é uma conseqüência da característica de autonomia dos agentes, pois sem autonomia o agente não poderia reagir ao seu ambiente.

Oliveira e Cardozo [50] e Franklin e Graesser [18] apresentam uma série de características bem aceitas para agentes, tais como:

- Autonomia: um agente pode tomar suas próprias decisões, baseado sobre seus objetivos, preferências, limitações e políticas. Assim, ele exerce controle sobre suas ações.
- Delegação: usuários e outros programas podem delegar tarefas para um agente, atribuindo a ele autoridade para atuar sobre o domínio destas tarefas.
- Comunicação: um agente pode se comunicar com outros elementos em seu ambiente.
- Cooperação: agentes podem atuar em cooperação com outros agentes de modo a alcançarem objetivos não conflitantes.
- Sensibilidade ao ambiente: um agente é sensível às mudanças de seu ambiente.
- Orientação a objetivo: um agente não atua somente em reação ao ambiente, mas também em função de seus objetivos.
- Adaptabilidade: um agente pode mudar seu comportamento baseado em experiências passadas.
- Flexibilidade: agentes não assumem papeis fixos, podendo atuar como clientes, servidores, observadores, etc., dependendo dos seus objetivos no momento.

Neste trabalho consideraremos agente como sendo uma entidade de *software* autônoma, que possui estado, habilidades e objetivos próprios e que se comunica com seu ambiente e outros agentes através de sensores e atuadores (mensagens recebidas e enviadas, respectivamente). O estado corresponde aos dados persistentes associados ao agente. As habilidades caracterizam o quê um agente tem capacidade de realizar. E por fim, o objetivo diz respeito às atividades que o agente pode ou deve desempenhar para resolver uma determinada tarefa. Este modelo de agente é ilustrado em Fig. 4.1.

Oliveira e Cardozo [50] citam duas habilidades fortemente relacionadas a agentes, mas não consideradas como fundamentais, que são:

- Inteligência: a habilidade de um agente de reagir e aprender a partir de interações com outros agente e com seu ambiente.
- Mobilidade: habilidade de um agente de suspender sua execução em uma máquina e se transportar para uma outra, mantendo sua integridade.

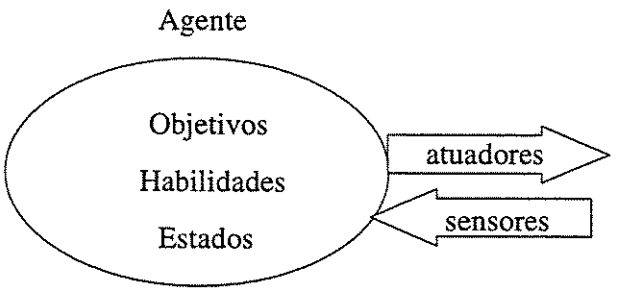

Fig. 4.1: Modelo de agente.

A habilidade de inteligência foi atribuída largamente a agentes para solução de problemas na área de inteligência artificial distribuída; neste caso agentes são comumente denominados de agentes inteligentes. Para Hayes-Roth [29] agentes inteligentes desempenham continuamente três funções: percepção das condições dinâmicas do ambiente; ações para afetar condições no ambiente e reações às ações do ambiente, resolvendo problemas, realizando inferências e determinando ações. Os principais problemas relacionados a agentes inteligentes são basicamente os seguintes:

- Aprendizagem: como capacitar um agente a aprender com suas experiências anteriores.
- Adaptação: como tornar um agente sensível ao contexto de seu ambiente.
- Coordenação: como gerar estratégias de coordenação para os agentes, de modo que possam cooperar efetivamente na solução de problemas comuns.

Agentes com habilidade de mobilidade são denominados de agentes móveis [69]. Agentes móveis, diferentemente de agentes fixos (aqueles que não possuem capacidade de migração), quando necessitam se comunicar com outro agente ou serviço localizado em um outro ambiente (possivelmente em uma outra máquina) podem se mover (migrar) para aquele ambiente de modo a realizar uma interação local. As Figs. 4.2 e 4.3 ilustram o processo de comunicação entre agentes remotos nos paradigmas de agentes fixos e agentes móveis, respectivamente.

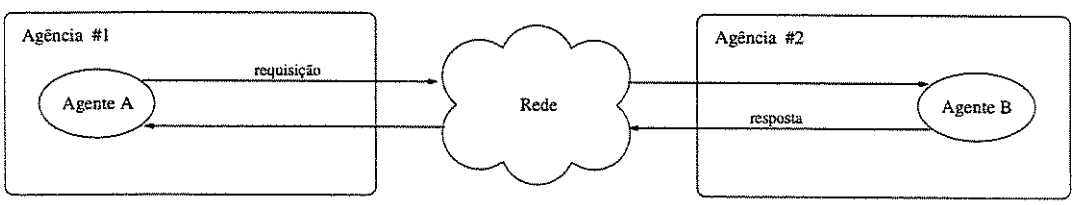

Fig. 4.2: Comunicação entre agentes remotos no paradigma de agentes fixos.

O uso de agentes móveis pode propiciar algumas vantagens, como [28] [8] [41]:

- Redução do tráfego de comunicação, uma vez que as interações entres agentes e agentes e ambiente são sempre locais.

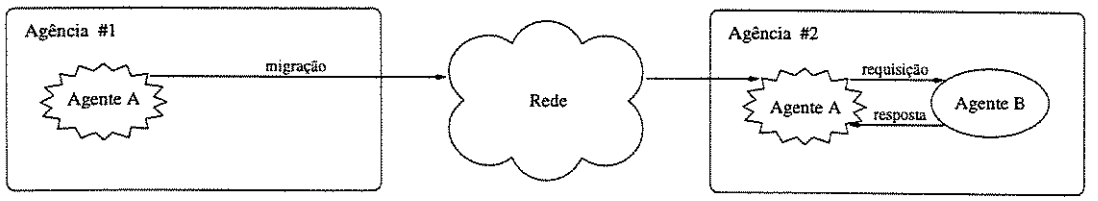

Fig. 4.3: Comunicação entre agentes remotos no paradigma de agentes móveis.

- Suporte à operação desconectada, uma vez que após a migração do agente não há a necessidade de se manter a conexão entre as máquinas remotas.
- Facilidade de distribuição de serviços, uma vez que a mobilidade do agente por natureza incorpora a integração de serviços distribuídos.
- Balanceamento de carga, pois um agente móvel executa seu código em várias máquinas.

Por outro lado, a habilidade de mobilidade requer que agentes executem em ambientes heterogêneos e seguros [28], uma vez que estes agentes devem executar seus códigos em várias máquinas. Além disto, o processo de migração deve ser eficiente e seguro.

Como agentes são entidades de *software* autônomas e autocontidas, eles possuem por essência as características de encapsulação, modularidade e distribuição. Essas características são fundamentais para o desenvolvimento eficiente de sistemas, pois privilegiam a flexibilidade, além do aumento da facilidade de integração e expansão dos componentes. Deste modo, o paradigma de sistemas baseados em agentes propicia uma abordagem única para o desenvolvimento de sistemas distribuídos.

Por essas características sistemas baseados em agentes se apresentam como uma abordagem capaz de modelar eficientemente as atividades complexas de negociação e gerência de qualidade de serviço em sistemas multimídia distribuídos descritas nos capítulos anteriores. Com este intuito, na próxima seção será apresentado um modelo baseado em agentes para negociação e gerência de qualidade de serviço em aplicações multimídia distribuídas combinando agentes móveis e fixos.

4.2 Arquitetura Baseada em Agentes Para Suporte à Qualidade de Serviço

A arquitetura de negociação e gerência de qualidade de serviço baseada em agentes é composta de uma agência de QoS para cada nó da aplicação, a qual suporta dez tipos de agentes, sendo três móveis e sete fixos. Além dos agentes, a agência de QoS possui um servidor de contratos e uma fábrica de agentes [24] [26]. Uma ilustração de uma agência de negociação e gerência de QoS, com seus componentes e suas interligações, é mostrada na Fig. 4.4.

O servidor de contratos tem por finalidade a gerência dos contratos em curso na sua agência. Um contrato é composto de estrutura e parâmetros, onde a estrutura descreve quais fluxos compõem a aplicação com suas respectivas configurações, enquanto os parâmetros

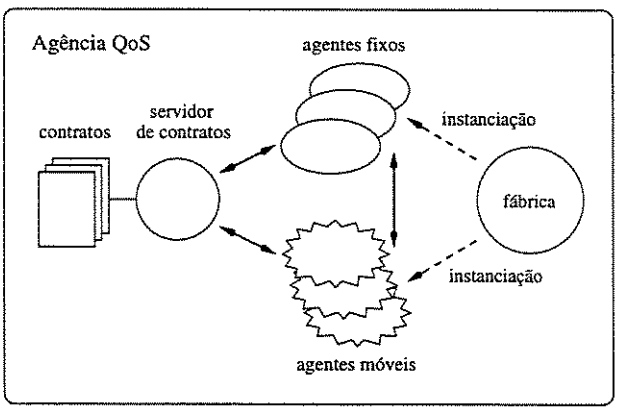

Fig. 4.4: Arquitetura baseada em agentes para suporte de QoS.

se referem às características dos fluxos, como níveis aceitáveis de qualidade de áudio, por exemplo. Assim, quem propõe um contrato deve especificar o que deseja receber e o que pode fornecer, ou seja, quais recursos de hardware e *software* possui e deseja. Para tal, o protocolo de negociação de contrato deve constar das seguintes etapas: proposta, contraproposta e aceitação ou não de contrato. O contrato ainda possui um identificador de sua versão, para propósito de consistência das suas cópias.

A fábrica de agentes é o componente da agência responsável pela instanciação (criação) e desativação de agentes.

A agência suporta os seguintes tipos de agentes: agente `interface`, agente `mapeador_de_qos`, agente `negociador_local`, agente `negociador_de_contrato`, agente `estimador_de_recursos`, agente `gerente_de_recursos`, agente `renegociador_de_contrato`, agente `monitor_de_contrato`, agente `monitor_de_qos` e agente `adaptador_de_qos`.

O agente `interface` é o representante do usuário na agência e interage com o usuário da aplicação para fins de especificação de contrato, renegociação de contrato e informar os níveis da qualidade do serviço durante a execução da aplicação. Este agente não tem capacidade de mobilidade.

O agente `negociador_de_contrato` é um agente com capacidade de migração. Este agente é criado na agência onde a aplicação é iniciada e persiste durante todo o ciclo de estabelecimento de sessão. Antes de ser extinto, o agente gera uma cópia de contrato para cada participante da sessão. Sua função é negociar o contrato com todos os participantes da aplicação, valendo-se de um protocolo de negociação bem definido. Este agente interage com os agentes `negociador_local` de cada agência envolvida na aplicação.

O agente `negociador_local` não possui capacidade de migração, sendo responsável pela negociação e renegociação local do contrato em cada agência envolvida na aplicação.

O agente `mapeador_de_qos` não possui capacidade de migração e tem as seguintes atribuições: i) na fase de estabelecimento de sessão, ele transforma especificações do usuário em especificações ao nível de sistema e ii) na fase de execução ele interage com o agente `monitor_de_qos` realizando o mapeamento inverso ao realizado na fase anterior, ou seja, mapeia parâmetros do nível de sistema em parâmetros subjetivos do nível do usuário, de modo a informar ao agente `interface` do desempenho da aplicação durante a fase de execução.

O agente `estimador_de_recursos` tem por finalidade estimar *a priori* os recursos que

serão necessários para cumprir os requisitos de QoS estabelecidos no contrato.

O agente `monitor_de_qos` não tem capacidade de migração e a sua função é mensurar o desempenho local do sistema a partir de medidas dos parâmetros QoS ao nível de sistema, que já foram apresentados. Este agente mede o desempenho do sistema a partir de medidas dos parâmetros QoS provenientes de sensores locais, mas externos à agência QoS. O `monitor_de_qos` interage com os agentes `mapeador_de_qos` e monitores_de_contratos para atualização de informações locais.

O agente `monitor_de_contrato` é um agente com capacidade de migração que permanece ativo durante toda a fase de execução da aplicação. Este agente tem por função monitorar se os fluxos nos destinos estão cumprindo os termos do contrato. Há um agente monitor_de_ contrato para cada agência produtora de fluxo. Sua atribuição é visitar ciclicamente todas as agências que recebem fluxos da agência a ele associada. Para tal, o agente possui um itinerário com os endereços de todas as agências que deve visitar. Ao chegar em uma agência, o agente `monitor_de_contrato` interage com o agente `monitor_de_qos` local de modo a obter informações sobre os fluxos sob sua responsabilidade. Se for detectada alguma violação nos limites do contrato, o `monitor_de_contrato` informa a violação ao agente `adaptador_de_qos` da agência produtora do fluxo. Após cumprir sua atribuições numa agência, o `monitor_de_contrato` parte para próxima agência de seu itinerário circular. Quando há uma renegociação do contrato, com conseqüente atualização de versão, estes agentes (`monitor_de_contrato`) são avisados por suas respectivas agências de que eles devem atualizar seus objetivos. Então, ao receber tal notificação o agente `monitor_de_contrato` interage com o servidor de contrato para obter a nova versão do contrato. Se a mudança do contrato for apenas paramétrica, então há mudança apenas nos níveis de aceitáveis de qualidade de serviço. Porém, se a mudança for estrutural com inclusão ou exclusão de algum nó recebedor de fluxo na aplicação, há a necessidade da atualização do itinerário do agente `monitor_de_contrato`. Caso haja a inclusão de um nó produtor de fluxo a agência produtora instancia um novo agente `monitor_de_contrato` para monitorar este fluxo. Caso não seja detectada nenhuma violação de contrato durante um certo intervalo de tempo, o agente `monitor_de_contrato` deve reportar o fato com respectivo relatório à sua agência de origem (isto fornece uma garantia de que este agente continua em seu funcionamento normal). Ao final da fase de execução de uma sessão todos os agentes `monitor_de_contrato` são sinalizados por suas respectivas agências para serem extintos. O procedimento é similar quando um determinado nó produtor de fluxo se retira de uma sessão.

Cada estação envolvida na aplicação possui um agente `gerente_de_recursos` e cada estação produtora de fluxo possui um agente `adaptador_de_qos`, ambos sem capacidade de migração. O `gerente_de_recursos` tem por função apenas reservar e liberar recursos do sistema aos fluxos, a partir de solicitações dos agentes `negociador_local` e `adaptador_de_qos`. Já o agente `adaptador_de_qos` tem atribuições de manter os índices de qualidades dentro dos limites estabelecidos no contrato. Para tal, atua junto ao agente `gerente_de_recursos` local e ao seu agente `monitor_de_contrato` remoto da seguinte forma: i) é informado pelo seu `monitor_de_contrato` caso haja alguma violação de contrato no destino de algum de seus fluxos, ii) interage com o `gerente_de_recursos` para tentar honrar o contrato localmente, iii) dependendo da informação que obtém do `monitor_de_contrato` remoto, o agente pode efetuar uma ação de adaptação de QoS nos fluxos destinos desta agência, via gerente de recursos local e iv) quando as ações anteriores não surtirem os efeitos desejados o

agente `adaptador_de_qos` pode solicitar ao agente `interface`, via agente `mapeador_de_qos`, que dispare um agente móvel `renegociador_de_contrato`, caracterizando-se uma violação severa dos termos do contrato.

O agente `renegociador_de_contrato` possui capacidade de migração. Este agente atua na fase de execução da sessão, sendo instanciado pela fábrica a pedido do agente `negociador_local` com o propósito de renegociar os termos do contrato (estrutura ou parâmetros). Uma renegociação de contrato para mudança de parâmetros pode solicitar aumento de QoS, possivelmente sugerida pelo usuário, ou um diminuição de QoS, usualmente promovida pelo sistema em decorrência de sobrecarga. Já uma renegociação de contrato para mudança de estrutura se destina a inclusão ou exclusão de fluxos na ou da aplicação, o que pode causar uma renegociação de parâmetros do contrato. Este agente deve interagir com todos os negociadores_locais envolvidos na aplicação, sendo extinto no final do processo de renegociação de contrato após atualizar todas as cópias do contrato renegociado.

Para que não haja conflito na negociação e renegociação, todos os agentes envolvidos nestas atividades viajam utilizando sempre um mesmo sentido de rota. Para efetuar esta ordenação na rota de viagem pode-se utilizar os números IP das máquinas envolvidas na aplicação, por exemplo.

Além de uma agência de QoS em cada nó da aplicação, o modelo ainda prevê um *trader*, no sentido do modelo ODP (*Open Distributed Processing*) [71], com o propósito de armazenar informações sobre os recursos e serviços de todas as agências que podem participar de aplicações conjuntas. Estas informações seriam de caráter estruturais, como tipo de dispositivos de entrada e saída, tipos de serviço de transporte, infra-estrutura de rede e serviços de compressão de áudio e vídeo. Este componente do modelo, o *trader*, é especialmente útil em aplicações onde a localização dos serviços não é explicitada no ato da proposição do contrato, como é típico em aplicações de vídeo sob-demanda [53]. Nessas aplicações geralmente não se impõe a princípio a localização dos recursos, mas sim o tipo de serviço com respectiva qualidade pretendida. No caso, o processo de negociação se caracteriza basicamente em determinar a melhor oferta do serviço levando em conta a qualidade e o custo do mesmo. Por outro lado, aplicações tipo teleconferência, onde o localização dos serviços (participantes da aplicação) são bem determinados no ato da proposição do contrato, a utilidade do *trader* torna-se menor, ou até desnecessária, uma vez que não compromete o processo de negociação de contrato.

A seguir apresentaremos o modelo baseado em agentes para negociação e gerência de QoS, explicitando-se as funcionalidades e interações dos agentes em cada etapa do ciclo de vida de uma sessão multimídia, ou seja, as etapas de estabelecimento, execução e encerramento de sessão.

4.3 Estabelecimento de Sessão

A fase de estabelecimento de sessão é composta de duas atividades: especificação de contrato e protocolo de negociação de contrato. Nesta fase os agentes ativos nas agências de QoS são os seguintes: `interface`, `mapeador_de_qos`, `negociador_local`, `negociador_de_contrato`, `gerente_de_recursos` e `estimador_de_recursos`.

4.3.1 Especificação de Contrato

A especificação de contrato corresponde à apresentação de uma proposta de contrato por parte do usuário da aplicação. A proposta é constituída de estrutura e parâmetros. A estrutura especifica os fluxos envolvidos na aplicação, caracterizando-os quanto ao tipo, origem e destino(s). Os tipos são vídeo, áudio, animação e dados (mídias estáticas como texto, gráficos e imagens digitalizadas). Já os parâmetros no contrato especificam as características dos fluxos envolvidos na proposta de contrato e podem ser do tipo *hard* ou *soft*.

Os parâmetros *hard* estão relacionados com as infra-estruturas de processamento e comunicação do sistema. No caso, teríamos para cada fluxo os seguintes parâmetros:

- tipo de codificador e decodificador;
- tipo de mecanismo de transporte e infra-estrutura de rede;
- tipo de dispositivos de entrada;
- tipo de dispositivos de saída;
- característica de processamento (sistema operacional, por exemplo) e
- serviços oferecidos.

Cada estação possui um registro associado ao agente `gerente_de_recursos`, que contém todas as opções disponíveis para os tipos de parâmetros *hard* numa dada estação. Este registro guarda as configurações possíveis para o suporte de áudio e vídeo de uma dada estação. Registro de possíveis recursos de *hardware* e *software* para um fluxo de Vídeo é caracterizado na Tab. 4.1, enquanto a Tab. 4.2 caracteriza um registro de possíveis recursos de *hardware* e *software* para um fluxo de Áudio. É usual que uma estação possua apenas uma opção de configuração para os itens Disp. de Entrada, Disp. de Saída e Suporte de Processamento, descritos nas Tabs. 4.1 e 4.2. Porém, freqüentemente há mais de uma opção de configuração para os itens Tipo de Codificação e Sistema de Comunicação, sendo que este último caracteriza aspectos de infra-estrutura de rede e mecanismos de transporte.

É importante notar que as características *hard* são justamente aquelas que não provêem transparência de portabilidade, por exemplo, um fluxo codificado em MPEG não pode ser descodificado como se fosse CellB. É justamente por este motivo que estas características necessitam ser explicitadas.

Os parâmetros *soft* são justamente os parâmetros de QoS do nível de usuário, descritos no Capítulo 2: fidelidade, dinâmica e interatividade.

O protocolo de especificação de contrato é composto das seguintes etapas:

- Especificação da estrutura da aplicação: especificar todos os fluxos que constituem a aplicação, explicitando o tipo, a origem e o(s) destino(s).
- Especificação dos parâmetros de cada fluxo.

  Parâmetros *soft*: valores desejáveis e faixas aceitáveis.

Disp. de Entrada	Disp. de Saída	Tipo de Codificação	Sistema de Comunicação		Suporte de Processamento
Câmera 1 Câmera 2 . . . . Câmera n	*Display* 1 *Display* 2 . . . . *Display* k	Sem codificação	ATM	ATM puro IP/ATM XTP/ATM	Sist. Oper. Não Tempo Real
		MPEG	FDDI	TCP/IP RTP/UDP XTP/IP	Sist. Oper. Não Tempo Real Multithreaded
		MJPEG	Ethernet	RTP/UDP RTP/UDP XTP/IP	
		CellB	Token Ring	RTP/UDP RTP/UDP XTP/IP	Sist. Oper. Tempo Real

Tab. 4.1: Alguns formatos típicos para Vídeo.

- Transladar os parâmetros de QoS do nível de usuário para os parâmetros de QoS do nível de sistema, via o agente `mapeador_de_qos`. Os parâmetros de QoS ao nível de sistemas são: atraso fim-a-fim, jitter fim-a-fim e taxa de descarte de pacote.

- Obter um subconjunto viável no registro de parâmetros *hard* local, via agente `gerente_de_recursos`, que atenda aos requisitos de QoS, ou seja, cada campo no registro de parâmetros *hard* deve possuir ao menos uma opção de configuração. Caso não seja possível, notificar ao usuário e retornar um indicativo de erro.

- Caso positivo, retornar ao usuário as opções de configuração, com as respectivas estimativas de utilização de recursos, para serem homologadas.

- Consultar o *trader* para se certificar se há infra-estrutura global para a aplicação, caso positivo, criar um agente `negociador_de_contrato` e começar a etapa de negociação de contrato com as outras agências participantes da sessão multimídia.

A configuração da agência QoS durante a etapa de estabelecimento de contrato é apresentada na Fig. 4.5, via diagrama de interações. As interações da agência de QoS na etapa de estabelecimento de contrato são descritas na Fig. 4.5 por setas. As transições, correspondendo às numerações nas setas da Fig. 4.5, descrevem a dinâmica do processo de estabelecimento de contrato:

1. Proposta de contrato do agente `interface`, contendo estrutura e parâmetros, descrita por valores subjetivos.

2. Proposta de contrato descrita em faixas de valores dos parâmetros de qualidade de serviço do nível de sistema. Estas faixas caracterizam os valores desejados e mínimos aceitos pela proposta de contrato.

<table>
<tr><th>Disp. de Entrada</th><th>Disp. de Saída</th><th>Tipo de Codificação</th><th colspan="2">Sistema de Comunicação</th><th>Suporte de Processamento</th></tr>
<tr><td rowspan="10">Microfone 1
Microfone 2
.
.
.
.
.
.
.
Microfone n</td><td rowspan="10">Speaker 1
Speaker 2
.
.
.
.
.
.
.
Speaker k</td><td rowspan="10">PCM

A-Law

μ-Law</td><td rowspan="3">ATM</td><td>ATM puro</td><td rowspan="10">Sist. Oper.
Não Tempo Real

Sist. Oper.
Não Tempo Real
Multithreaded

Sist. Oper.
Tempo Real</td></tr>
<tr><td>IP/ATM</td></tr>
<tr><td>XTP/ATM</td></tr>
<tr><td rowspan="3">FDDI</td><td>TCP/IP</td></tr>
<tr><td>RTP/UDP</td></tr>
<tr><td>XTP/IP</td></tr>
<tr><td rowspan="3">Ethernet</td><td>RTP/UDP</td></tr>
<tr><td>RTP/UDP</td></tr>
<tr><td>XTP/IP</td></tr>
<tr><td rowspan="3">Token Ring</td><td>RTP/UDP</td></tr>
<tr><td>RTP/UDP</td></tr>
<tr><td>XTP/IP</td></tr>
</table>

Tab. 4.2: Alguns formatos típicos para Áudio.

3. Solicitação de estimativa de recursos locais com respectivas configurações, que atendam aos requisitos da proposta de contrato. A estimativa diz respeito apenas aos fluxos associados com este nó, ou seja, fluxos que tenham origem ou destino nesta estação.

4. Solicitação de reserva de recursos ao `gerente_de_recursos`. A solicitação é feita em relação aos valores máximos.

5. Reserva de recursos bem sucedida.

6. Confirmação de reserva de recursos local.

7. Sinalização de viabilidade local da proposta de contrato e possíveis configurações necessárias para atender aos requisitos da aplicação. Para cada possível configuração, são associadas faixas de valores de recursos, que correspondem respectivamente às estimativas para atender aos requisitos desejados e aos mínimos aceitos pela proposta de contrato.

8. Consulta ao *trader* sobre a viabilidade, em termos de configuração da aplicação. É passado ao *trader* uma lista de possíveis configurações para a aplicação, a partir das restrições locais.

9. Resposta afirmativa do *trader* contendo a lista de configurações viáveis.

10. Solicitação à fábrica de agentes que crie um agente `negociador_de_contrato`.

11. Instanciação de um agente `negociador_de_contrato`.

12. Confirmação da criação de um agente `negociador_de_contrato`.

13. Sinalização de que a negociação global da proposta de contrato está em curso.

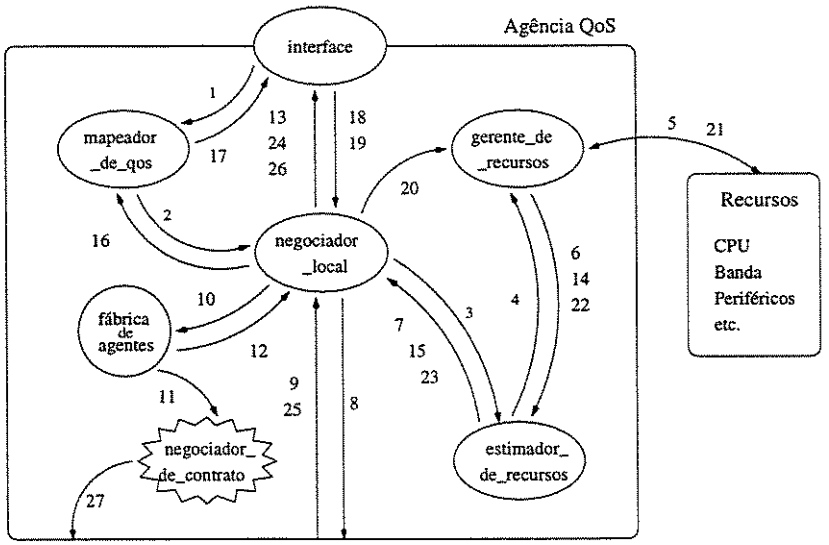

Fig. 4.5: Atividades da agência de QoS na etapa de estabelecimento de contrato.

14. Sinalização de que apenas parte dos recursos são disponíveis.

15. Contraproposta de contrato sugerida pelo agente `estimador_de_recursos`, após a estimativa de que não haverá recursos locais suficiente para atender à demanda de QoS requerida pela proposta de contrato original. A contraproposta inclui as possíveis configurações com as respectivas estimativas de recursos descritas em faixas de valores, além dos níveis de qualidade de serviço associados, que por sua vez são descritos em faixas de valores de parâmetros mensuráveis de qualidade de serviço.

16. Contraproposta de contrato descrita em faixas de valores dos parâmetros de QoS do nível de sistema.

17. Contraproposta de contrato descrita em valores dos parâmetros de QoS do nível de usuário.

18. Aceite da contraproposta local pelo agente `interface`.

19. Desistência do agente `interface` em continuar a negociação de contrato.

20. Sinalização para liberação de recursos reservados.

21. Liberação de recursos reservados.

22. Sinalização de que não há recursos disponíveis.

23. Aviso por parte do agente `estimador_de_recursos` da impossibilidade de atender aos requisitos da proposta de contrato, pois não há recursos no registro *hard* local, como por exemplo: microfone, câmera e mecanismos de transporte. Este caso é definido como limitação estrutural.

24. Sinalização ao agente `interface` de que não há infra-estrutura local para suportar a proposta de contrato.

25. Sinalização ao agente `negociador_local` de que não há configuração possível para viabilizar a aplicação globalmente.

26. Sinalização ao agente `interface` de que não há infra-estrutura global para suportar a proposta de contrato.

27. Partida do agente `negociador_de_contrato`, dando início a negociação global (processo de migração).

Assim, as transições 1 a 4 correspondem à apresentação de proposta de contrato, as transições 5 a 13 correspondem ao cenário de existência plena de recursos, as transições 14 a 21 correspondem ao cenário de existência de recursos parciais e as transições 22 a 26 correspondem ao cenário de inexistência de recursos.

4.3.2 Negociação de Contrato

No final da fase anterior, a agência proponente do contrato gera um agente `negociador_de_contrato`, que tem por objetivo negociar um contrato viável com todos as outras agências de QoS participantes da sessão. Esse agente possui uma proposta de contrato, que se constitui de estrutura e parâmetros, além de uma lista das possíveis configurações. A estrutura caracteriza o tipo, a origem e o(s) destino(s) de cada fluxo constituinte da aplicação, enquanto os parâmetros compreendem os valores de QoS de cada fluxo, com suas respectivas margem de negociação e a configuração caracteriza a infra-estrutura (como sistema operacional, tipo de codificadores e protocolo de transporte, por exemplo). É importante notar que a política de negociação é de responsabilidade do agente `negociador_de_contrato`, pois este tem consigo além do contrato, os limites máximos e mínimos de negociação de QoS.

Ao chegar a uma agência de QoS, o agente `negociador_de_contrato` apresenta uma proposta de contrato ao representante do usuário dessa agência. Esta proposta é apresentada da seguinte forma:

- O agente `negociador_de_contrato` apresenta sua proposta ao agente `negociador_local`, especificando os requisitos de QoS em termos de parâmetros do nível do sistema e infra-estrutura de comunicação necessária.

- Caso não haja em curso nenhuma negociação que envolva esta agência, o agente `negociador_local` apresenta, via agente `mapeador_de_qos`, a proposta remota de contrato ao agente `interface`.

- O usuário, ao obter a proposta de contrato via agente `interface`, primeiramente verifica se está de acordo com os índices de qualidade propostos para os parâmetros dos fluxos que deverá receber. Caso esteja, libera o início do processo de negociação e reserva de recursos locais. Por outro lado, caso não esteja de acordo com o proposta de contrato inicial, especifica os valores desejados e as margens de negociação numa contraproposta e apresenta-a ao agente `negociador_local`, via o agente `mapeador_de_qos`, e este por sua vez repassa-a ao agente `negociador_de_contrato`.

- O agente `negociador_de_contrato`, se houver contraproposta à sua proposta inicial, pode incorporá-la se achar pertinente.
- O agente `negociador_de_contrato` começa a negociar com o `negociador_local` a viabilidade da proposta de contrato, de modo similar ao realizado na etapa de especificação de contrato.
- Dependendo do desfecho da negociação, o agente `negociador_de_contrato` pode tomar várias atitudes, como por exemplo:

  - se os índices de qualidade não podem ser garantidos por esta agência, o agente obtém os valores possíveis de garantias, reservando recursos para futura decisão e assinala erro na negociação nesta agência.

  - se os índices de qualidade garantidos estiverem nos limites aceitáveis, então o agente reserva os recursos pertinentes e assinala sucesso na negociação.
- O agente `negociador_de_contrato` parte para outra agência, de modo a continuar o processo de negociação.
- Após visitar todas as agências que participarão da sessão, o agente `negociador_de_contrato` volta para a agência que propôs o contrato inicial. Então, comunica ao usuário dessa agência, via o `mapeador_de_qos` local, sobre o contrato conseguido. De posse desta informação, o usuário atua, via agente `interface`, da seguinte forma:

  - caso concorde com o contrato conseguido, sinaliza para o agente `negociador_local` para que este providencie a confirmações de reservas de recursos negociados e dê início à fase de atividade de sessão.

  - caso não concorde com esta proposta de contrato, sinaliza para o agente `negociador_local` para que este providencie a liberação dos recursos locais e remotos já negociados, indicando desistência da sessão.

A Fig. 4.6 apresenta diagrama de interações as atividades desempenhadas em cada agência QoS durante o protocolo de negociação de contrato. As interações da Fig. 4.6 têm os seguintes significados:

1. Chegada do agente `negociador_de_contrato` em uma agência QoS.
2. Apresentação de proposta de contrato, descrita em termos de faixas de parâmetros mensuráveis e possíveis configurações para a aplicação.
3. Repasse da proposta remota de contrato para o agente `mapeador_de_qos`.
4. Apresentação da proposta remota de contrato em termos subjetivos ao agente `interface`.
5. Aceite de negociação por parte do agente `interface`.
6. Solicitação de estimação de recursos com restrições de configurações.
7. Solicitação de reserva de recursos ao agente `gerente_de_recursos`.

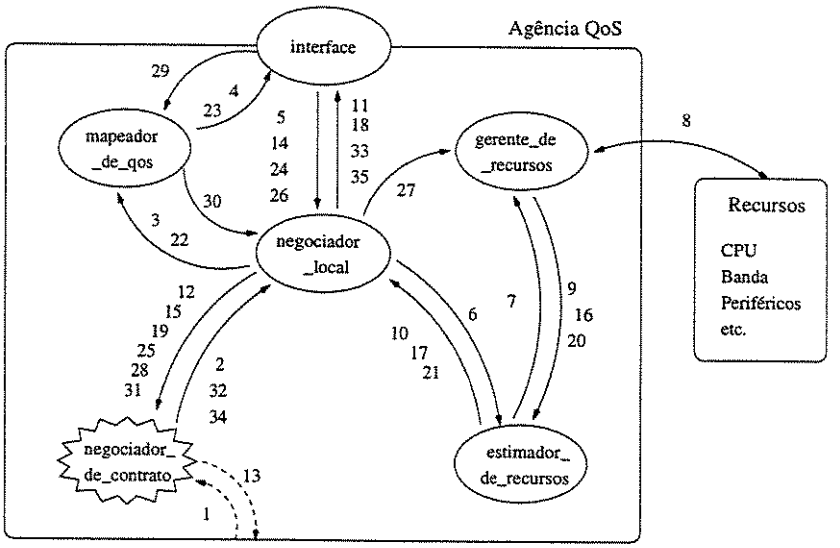

Fig. 4.6: Atividades de uma agência QoS durante o processo de negociação de contrato.

8. Reserva de recursos.

9. Reserva de recursos bem sucedida.

10. Sinalização de que há uma estimativa favorável de se atender localmente aos requisitos da proposta remota de contrato. Informa ainda as estimativas dos níveis de QoS e a lista de configurações viáveis, que é um subconjunto da lista apresentada pelo agente `negociador_local`.

11. Aviso ao agente `interface` de que a parte local da negociação de contrato foi cumprida e que o `negociador_local` aguarda o desfecho da negociação global.

12. Aviso ao agente `negociador_de_contrato` de que o processo de negociação local foi bem sucedido; informando ainda os níveis de recursos reservados e as configurações viáveis.

13. Migração do agente `negociador_de_contrato` para outra agência.

14. Aviso de recusa em participar da sessão.

15. Aviso ao agente `negociador_de_contrato` de que o usuário se recusa a participar desta negociação de contrato.

16. Aviso ao agente `estimador_de_recursos` de que não há configuração possível para esta aplicação.

17. Aviso do agente `negociador_local` de que não há configuração possível para esta aplicação.

18. Aviso ao agente `interface` de que não há configuração possível para atender a aplicação.

19. Aviso ao agente `negociador_de_contrato` de que não há configuração possível para atender a aplicação.

20. Sinalização de que os recursos solicitados não foram possíveis ser reservados, mas indica a quantidade de recursos possíveis de serem alocados.

21. Envio de contraproposta de contrato por parte do agente `estimador_de_recursos`, especificando os níveis de QoS possíveis com suas respectivas configurações.

22. Repasse ao `mapeador_de_qos` da contraproposta de contrato feita pelo agente `estimador_de_recursos`.

23. Repasse ao agente `interface` da contraproposta de contrato feita pelo agente `estimador_de_recursos`, descrita em termos subjetivos.

24. Sinalização de aceite em participar da sessão nas condições da contraproposta de contrato acima.

25. Sinalização de que a proposta não foi possível ser atendida, mas indica a contraproposta possível.

26. Sinalização de desistência do processo de negociação de contrato por parte do usuário.

27. Liberação de recursos reservados a esta negociação.

28. Sinalização de desistência do processo de negociação de contrato por parte do usuário.

29. Apresentação de contraproposta de contrato, descrita em termos subjetivos.

30. Apresentação de contraproposta de contrato, descrita em termos de faixas de parâmetros mensuráveis.

31. Apresentação de contraproposta de contrato do usuário local para a agente `negociador_de_contrato`.

32. Sinalização de aceite pelo agente `negociador_de_contrato` da contraproposta feita pelo usuário local.

33. Sinalização indicando que o agente `negociador_de_contrato` incorporou a contraproposta feita pelo usuário local.

34. Sinalização de desistência de negociação do agente `negociador_de_contrato`, por não aceitar a contraproposta feita pelo usuário local.

35. Sinalização indicando que o agente `negociador_de_contrato` desiste da negociação por não aceitar a contraproposta feita pelo usuário local.

As transições 1 a 4 correspondem à etapa de apresentação de proposta remota. As transições 5 a 12 correspondem ao cenário de negociação remota bem sucedida. As transições 14 e 15 correspondem à recusa de participação por parte do usuário local. As transições 16 a 19 corespondem ao cenário de inexistência de recursos locais para satisfazer ao requisitos

do contrato. As transições 20 a 23 correspondem ao cenário de existência parcial de recursos locais, sendo que as transições 24 e 25 correspondem ao aceite do usuário local das condições existentes para o contrato, enquanto as transições 26, 27 e 28 correspondem à desistência do usuário local diante das condições existentes para o contrato. Finalmente, as transições 29 a 35 dizem respeito ao cenário de contraproposta de contrato feita pelo usuário local.

Ao término de uma negociação bem sucedida, o agente `negociador_local` fica à espera do agente `negociador_de_contrato` para a confirmação ou não do contrato. Caso este não confirme a reserva dentro de um determinado intervalo de tempo, o `negociador_local` assume que a negociação global não se completou. Neste caso, o agente libera os recursos reservados para aquele contrato e avisa o agente `interface` que o processo de negociação de contrato não foi bem sucedido. Por outro lado, se o `negociador_de_contrato` for bem sucedido em sua missão, o `negociador_local` confirma a sua reserva de recursos e avisa o agente `interface` que a negociação global de contrato foi bem sucedida estando a sessão apta a entrar em sua fase de execução.

4.4 Execução de Sessão

Ao final da fase de estabelecimento e antes da fase de atividade de sessão, cada agência participante da sessão cria os agentes fixos `adaptador_de_qos` e `monitor_de_qos`. Os outros dois agentes fixos, `gerente_de_recursos` e `mapeador_de_qos`, já foram criados na fase de estabelecimento da sessão.

Para começar a fase de atividade é necessário um protocolo simples de sinalização, que é gerado pela agência que propôs a aplicação. O protocolo se constitui das seguintes etapas:

- A agência proponente difunde para todas as outras agências participantes da sessão uma sinalização de começo de execução de sessão e espera as respostas destas por um prazo determinado.
- Caso a agência proponente receba todas as confirmações em tempo hábil, é difundida uma sinalização de início de fase de execução de sessão.
- Caso falhe o processo de confirmação, a fase de execução não pode ser iniciada e é gerada uma mensagem de erro para o usuário da agência proponente.
- Após transcorrer um certo prazo do início da fase de execução, cada agência cujo nó é fonte de fluxo(s) cria um agente `monitor_de_contrato` (móvel) a ela associado, com a finalidade de monitorar os fluxos nos seus diversos destinos.

Nas Figs. 4.7 e 4.8 são mostradas, via diagrama de interação, as atividades desempenhadas por uma agência de QoS durante o processo de monitoramento na fase de atividade de uma sessão multimídia. No caso, a Fig. 4.7 representa uma situação de adaptação de QoS, e as interações têm os seguintes significados:

1. Medição dos parâmetros de QoS dos fluxos que têm destino neste nó.
2. Atualização das medidas de QoS locais, em termos de parâmetros de QoS no nível de sistema.

3. Atualização das medidas de QoS locais, em termos de parâmetros de QoS no nível de usuário.
4. Chegada do agente `monitor_de_contrato` em uma Agência de QoS.
5. Atualização das medidas de QoS locais, em termos de parâmetros de QoS no nível de sistema.
6. Sinalização para adaptação de contrato, por violação moderada da qualidade de serviço.
7. Migração do agente `monitor_de_contrato` para outra agência.
8. Sinalização para adaptação de recursos.
9. Adaptação de recursos.
10. Aviso de adaptação da qualidade de serviço, em termos de parâmetros mensuráveis.
11. Aviso de adaptação da qualidade de serviço, em termos de parâmetros subjetivos.

Já a Fig. 4.8 representa uma situação de violação severa de contrato, com conseqüente renegociação de contrato, e as interações têm os seguintes significados:

1. Medição dos parâmetros de QoS dos fluxos que têm destino neste nó.
2. Atualização das medidas de QoS locais, em termos de parâmetros de QoS no nível de sistema.
3. Atualização das medidas de QoS locais, em termos de parâmetros de QoS no nível de usuário.
4. Chegada do agente `monitor_de_contrato` em uma Agência de QoS.
5. Atualização das medidas de QoS locais, em termos de parâmetros de QoS no nível de sistema.
6. Sinalização de ocorrência de violação severa da qualidade de serviço.
7. Migração do agente `monitor_de_contrato` para outra agência.
8. Aviso da necessidade de renegociação de contrato, descrita em termos de parâmetros mensuráveis.
9. Aviso da necessidade de renegociação de contrato, descrita em termos de parâmetros subjetivos.
10. Sinalização de liberação para renegociação.
11. Proposta de renegociação de contrato.
12. Solicitação de criação de um agente `renegociador_de_contrato`.
13. Criação do agente `renegociador_de_contrato`.
14. Migração do agente `renegociador_de_contrato`.

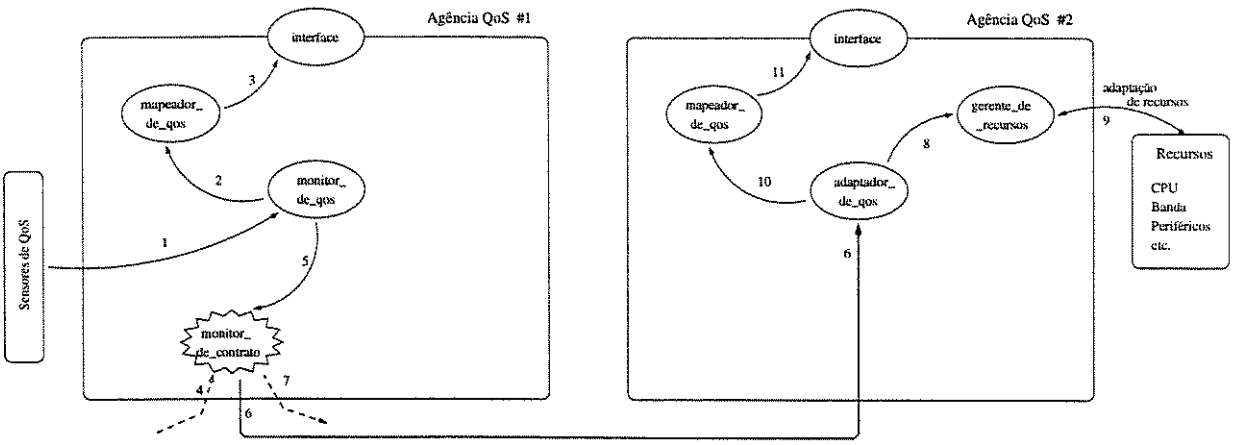

Fig. 4.7: Representação de uma situação de adaptação de QoS durante a fase de execução de uma sessão multimídia.

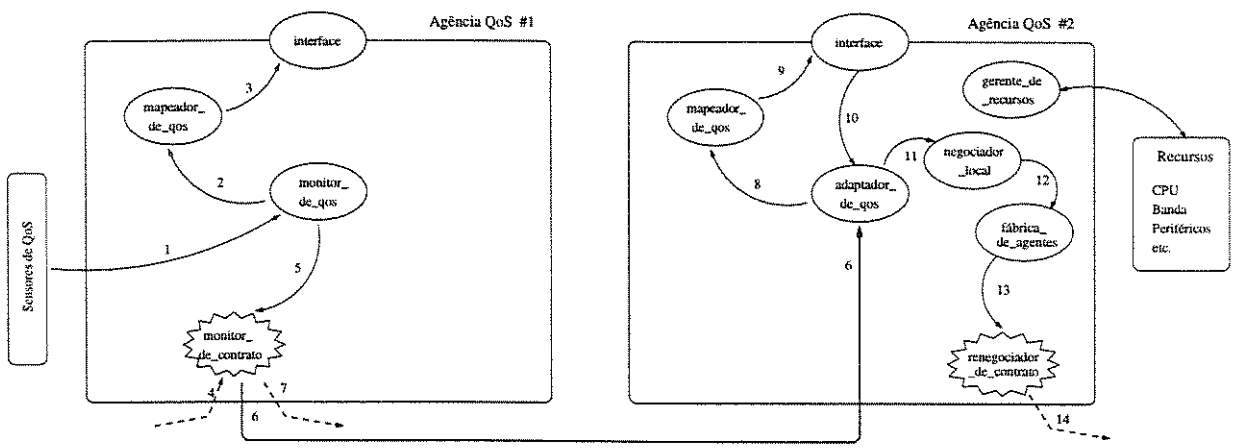

Fig. 4.8: Representação de uma situação de renegociação de contrato durante a fase de execução de uma sessão multimídia.

4.5 Encerramento de Sessão

Quando algum usuário deseja se retirar de uma sessão, ele sinaliza essa sua decisão ao seu agente `interface`, que por seu turno cria um agente `renegociador_de_contrato` para este propósito. Este agente deve visitar todas as outras agências de QoS que fazem parte desta sessão. As agências de QoS, ao receberem a proposta de retirada, devem atender este pedido. Após interagir com todas as agências envolvidas no processo de retirada, o agente `renegociador_de_contrato` volta à agência de origem para ser extinto. Então, é gerada uma nova versão do contrato pela agência proponente de contrato para todas as outras agência que participam da aplicação. Assim, o processo de saída de um participante é caracterizado como uma atividade de renegociação de contrato. Por questão de consistência, quando um `monitor_de_contrato`, que tem em seu itinerário a agência que se retirou da sessão, chega a uma agência remota, ele é avisado pelo agente `monitor_de_qos` local da necessidade de

atualização da versão do seu contrato. Além disto, antes de se retirar de uma sessão, a agência elimina seu agente `monitor_de_contrato` móvel.

Ao término da aplicação, é gerada uma sinalização a partir da agência proponente do contrato para todas as outras agência participantes da aplicação. Como conseqüência, todos os recursos alocados à aplicação são liberados em cada agência.

Capítulo 5

Especificação Formal do Protocolo de Negociação e Gerência de Qualidade de Serviço

Neste capítulo será apresentado o protocolo de negociação e gerência de qualidade de serviço para o modelo baseado em agentes descrito no capítulo anterior. Devido à riqueza de seus procedimentos de troca de mensagens, sua especificação será efetuada via as linguagens formais SDL (*Specification Description Language*) [3] e MSC (*Message Sequence Chart*) [44], ambas padronizadas pela ITU (*International Telecommunication Union*), [34], sob as descrições Z100 e Z120, respectivamente.

O capítulo está dividido em três seções, sendo uma para cada fase do modelo de ciclo de vida de uma sessão multimídia, ou seja: fase de estabelecimento, fase de atividade e fase de encerramento.

5.1 Especificação do Protocolo para Estabelecimento de Sessão

Nesta seção será apresentada a especificação em SDL e MSC do protocolo de estabelecimento de sessão, que corresponde basicamente ao procedimento de negociação de contrato [23]. Na especificação assumimos que cada nó da aplicação possui uma agência de QoS, que corresponde por sua vez a um bloco em SDL e cada agente, por seu turno, será implementado como um processo SDL. A fábrica de agentes também é implementada como um processo SDL.

As visões hierárquica e de interconexão, em SDL, da agência de QoS são mostrada nas Figs. 5.1 e 5.2, respectivamente. Da visão hieráquica (Fig. 5.1) podemos verificar que o sistema (`Agencia_QoS`) é composto de uma área de declaração de variáveis globais (`Pr declaration`) e de um único bloco (`Agencia_de_QoS`), este por sua vez contém os processos SDL. Os processos em SDL são máquinas de estados que trocam mensagens entre si através de rotas de sinais. A troca de mensagem com o ambiente externo ao bloco é realizada através de canais que são conectados às correspondentes rotas. A visualização de com quem e quais sinais cada processo pode trocar é fornecida pela visão de interconexão (Fig. 5.2). Porém, esta

representação não fornece as possíveis ordens temporais das trocas de mensagens. Os pares ordenados associados aos processos na Fig. 5.1 correspondem respectivamente aos números inicial e máximo de instâncias do processo no modelo.

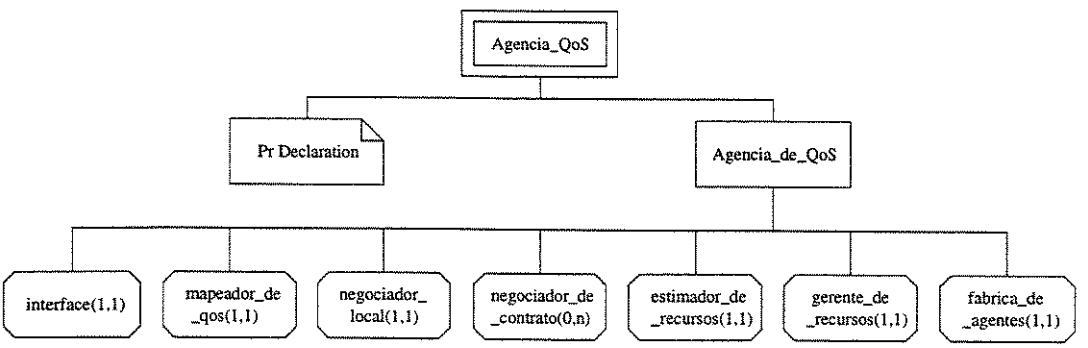

Fig. 5.1: Visão hierárquica em SDL do modelo da agência de QoS.

Como a visão de interconexão em SDL não é clara sobre as possíveis seqüências temporais de troca de mensagens entre os processos SDL, tornou-se habitual a utilização da linguagem formal MSC juntamente com SDL. MSC é fundamentalmente uma linguagem para descrição de troca de mensagens entre entidades (no caso processos em SDL) e seu ambiente. A principal vantagem da MSC se deve a uma representação gráfica simples que fornece a compreensão intuitiva imediata do comportamento temporal do sistema. MSC é utilizada mais especificamente com os propósitos de geração de possíveis cenários de seqüências temporais das trocas de mensagens entre processos e ambiente, e para simulação e verificação de consistência da especificação em SDL. O primeiro é referente à especificação de requisitos do processo de comunicação do modelo, enquanto o segundo diz respeito à etapa de validação e teste do sistema.

Por questão de simplificação iremos apresentar aqui somente os principais cenários de troca de mensagens especificados para o processo de negociação de QoS, que são divididos em dois grandes grupos: situação de sucesso na negociação e situação de falha na negociação.

5.1.1 Cenários de Sucesso na Negociação

No nosso modelo de agentes, um processo de negociação de QoS em uma aplicação multimídia distribuída é composto de três etapas: i) negociação da proposta localmente na agência proponente do contrato de QoS, ii) verificação de disponibilidade estrutural global da proposição, via consulta ao *trader* e iii) negociação da proposta com cada agência de QoS envolvida na aplicação. Para uma negociação obter sucesso é necessário que todas as três etapas do processo sejam bem sucedidas. Na primeira etapa há ainda a possibilidade de contraproposta do sistema à proposta inicial, sendo que isto ocorre nos casos em que o sistema detecta a impossibilidade de atender localmente aos requisitos da proposta de contrato.

Assim, a etapa de negociação de QoS na própria agência proponente do contrato pode ser bem sucedida com ou sem contraproposta do sistema. O primeiro caso é mostrado na Fig. 5.3 através de diagrama MSC, onde as barras verticais mais espessas correspondem ao ambiente. A Fig. 5.3 descreve a situação em que o agente `interface` recebe do

usuário da aplicação, via ambiente, uma proposta de contrato (proposta_local_s), descrita em termos de estrutura, fluxos constituintes da aplicação, e parâmetros de QoS do nível do usuário, que são descritos em termos de valores subjetivos (contrato_s). O agente interface repassa esta proposta ao agente mapeador_de_qos, que por sua vez a retransmite ao agente negociador_local, porém em termos de parâmetros de QoS do nível de sistema, que são descritos por faixas de valores aceitáveis (contrato_m). A seguir, o negociador_local solicita ao estimador_de_recursos uma estimativa da quantidade de recursos que será necessária para atender aos requisitos de qualidade requeridos pelos fluxos associados com esta agência de QoS (lista_de_fluxos). O agente estimador_de_recursos após efetuar a estimativa solicitada verifica junto ao gerente_de_recursos se há recursos disponíveis. Como a solicitação é viável, o agente gerente_de_recursos faz a reserva requerida, retornando ao estimador_de_recursos uma lista de possíveis configurações para a aplicação e um inteiro que corresponde ao número da reserva.

Caso seja detectada a inviabilidade da proposta, o agente estimador_de_recursos faz uma contraproposta, que pode ser aceita ou não pelo proponente do contrato. A Fig. 5.4 mostra o cenário em que a contraproposta é aceita. No caso, ao ser detectado a inviabilidade da proposta o gerente_de_recursos fornece o máximo de recursos disponíveis ao agente estimador_de_recursos, para que este produza uma contraproposta de contrato estendido (contrato_e) ao agente negociador_local. Esta contraproposta de contrato estendido contém estrutura e parâmetros de QoS do nível de sistema descritos em faixas de valores, além das possíveis configurações para a aplicação.

A etapa de verificação de viabilidade de infra-estrutura global do contrato, via consulta ao *trader*, para de sucesso é mostrada na Fig. 5.5. Esta etapa é importante porque garante que somente será iniciado um processo de negociação remota caso haja disponibilidade de infra-estrutura para suportar as demandas da aplicação.

Após esta etapa, a agência proponente do contrato cria um agente negociador_de_contrato móvel, que irá negociar a proposta de contrato em todas as outras agências participantes da aplicação. A Fig. 5.6 mostra o diagrama MSC do processo de instanciação do agente negociador_de_contrato. No caso, para efeito de especificação estamos utilizando um mecanismo de replicação que simula o processo de migração do agente negociador_de_contrato, causando o mesmo comportamento lógico no modelo. Como processos em SDL somente podem instanciar outros que estejam no mesmo bloco, a fábrica de agentes (fabrica_de_agentes) envia um sinal para uma outra fábrica remota (migracao_de_negociador) solicitando a instanciação de um processo negociador_de_contrato, de tal modo a emular o procedimento de migração desse agente.

A etapa de negociação remota de contrato terá êxito somente se o agente negociador_de_contrato for bem sucedido em todas as negociações que efetuar durante a sua passagem pelas agências de QoS envolvidas na aplicação. A Fig. 5.7 mostra o diagrama MSC para o caso de sucesso de negociação de contrato em uma agência remota, ou seja, uma agência não proponente de contrato.

Na Fig. 5.7, a seta pontilhada e o "x" na extremidade inferior da barra temporal do processo negociador_de_contrato significam respectivamente instanciação e morte do processo. Já o segmento pontilhado na barra temporal do negociador_local corresponde a uma co-região, ou seja, região onde a ordem temporal dos sinais é irrelevante.

Observando-se ainda a Fig. 5.7, podemos verificar o mecanismo de emulação do processo

de migração do agente `negociador_de_contrato`, no caso via um procedimento de replicação com passagem de informação através da variável `contrato_e` (contrato estendido). Essa variável contém a estrutura, os parâmetros de QoS do nível de sistema descritos em faixas de valores aceitáveis e as possíveis configurações para a aplicação. É importante ressaltar que a rota de viagem do agente é extraída a partir da estrutura do contrato.

Após o agente `negociador_de_contrato` ter obtido sucesso em todas as suas negociações, o mesmo volta a visitar todas as agências de QoS remotas para confirmar a reserva de recursos e habilitá-las para o início da fase de atividade de sessão. Este procedimento é mostrado na Fig. 5.8. Após este procedimento, o agente `negociador_de_contrato` volta a agência proponente do contrato para efetuar procedimento similar de confirmação de reserva Ao final do procedimento de reserva o agente é extinto. Caracteriza-se assim o final da fase de negociação de contrato.

Desse modo, todas as agências que irão participar da aplicação se encontram prontas para o início da fase de atividade, que será iniciada por uma sinalização da agência proponente do contrato.

5.1.2 Cenários de Falha na Negociação

Ao contrário de uma negociação de contrato bem sucedida, para se caracterizar uma negociação fracassada basta que haja falha em pelo menos uma das etapas de negociação descritas na seção anterior. No caso, a ocorrência de falha aborta todo o processo de negociação. Assim, pode-se ter falha na negociação devido a falta de recursos locais e/ou remotos à agência proponente da aplicação.

A falha na etapa de negociação local à agência proponente do contrato pode ser devido à falta de infra-estrutura local para atender aos requisitos da proposta ou à recusa pelo usuário de uma eventual contraproposta do sistema. O diagrama MSC para o primeiro caso é exibido na Fig. 5.9.

Caso se obtenha sucesso localmente (Figs. 5.3 e 5.4), a próxima etapa de negociação corresponde à consulta ao *trader*, para propósito de verificação de viabilidade da proposta em termos de infra-estrutura global. Caso o *trader* constate que não há opções que atendam aos requisitos da proposta, a negociação falha. O diagrama MSC deste caso é mostrado na Fig. 5.10. Nessa circunstância, a agência proponente do contrato deve liberar os recursos previamente reservados.

Na última etapa do processo de negociação, ou seja, negociação com todas as agências não proponentes do contrato, há a possibilidade de falha por recusa de alguma agência em participar da sessão ou ainda por falta de recursos para atender aos requisitos da proposta de contrato. O segundo caso é retratado na Fig. 5.11, que é o cenário de falha dual ao de sucesso descrito na Fig. 5.7. Porém, neste caso (Fig. 5.11), quando o `gerente_de_recursos` detecta que não há disponibilidade de recursos, os sinais gerados provocam a migração do `negociador_de_contrato` com o objetivo de cancelar eventuais reservas efetuadas em outras agências.

5.2 Especificação do Protocolo para Execução de Sessão

A visão hieráquica em SDL da agência de QoS para a fase de atividade é apresentada na Fig. 5.12.

Após obter sucesso na fase de estabelecimento de sessão, o sistema encontra-se apto para passar à fase de atividade de sessão. Porém, antes da iniciação da fase de atividade propriamente dita há a necessidade de uma etapa de preâmbulo, onde são instanciados os agentes necessários ao suporte das atividades a serem desempenhadas nesta fase da sessão.

A etapa de preâmbulo se constitui basicamente de sinalizações para instanciação de agentes e ativação dos recursos reservados na fase anterior, como servidores de fluxos de áudio e de vídeo, para todas as agências de QoS que fazem parte do contrato. Este procedimento, modelado via MSC, é apresentado nas Figs. 5.13 e 5.14 para uma agência proponente do contrato e uma agência remota, respectivamente. Nestas figuras são apresentados os casos gerais onde as agências são tanto produtoras quanto receptoras de fluxo. Quando uma agência não for receptora de fluxo o agente `monitor_de_qos` é dispensável. Por outro lado, quando uma agência não for fonte de nenhum fluxo os agentes `adaptador_de_qos` e `monitor_de_contrato` são dispensáveis.

Como apresentado no capítulo anterior, durante a fase de atividade, uma agência de QoS cujo nó possui recepção de fluxo desempenha basicamente as atividades de monitoramento de QoS e monitoramento do contrato negociado. Dependendo do resultado dessas atividades as agências de QoS cujos nós são fonte de fluxo podem ter atividades de adaptação de QoS e até renegociação de contrato.

A Fig. 5.15 apresenta o diagrama MSC de um cenário contendo atividades de monitoramento de QoS e de monitoramento de contrato para uma agência de QoS cujo nó é receptor de fluxo. O procedimento de migração do agente `monitor_de_contrato` é emulado através de replicação, similarmente ao realizado com o agente `negociador_de_contrato` na fase de estabelecimento de sessão. O agente `monitor_de_contrato`, ao chegar em uma agência receptora de fluxo, interage com o agente `monitor_de_qos` local de modo a verificar se não houve violação dos níveis de QoS estabelecidos no contrato para todos os fluxos que tem este nó como destino. Caso não se verifique nenhuma violação nesses níveis de QoS, o agente `monitor_de_contrato` parte para a próxima agência de seu itinerário (como descrito no capítulo anterior, por questão de confiabilidade é necessário que periodicamente o agente `monitor_de_contrato` envie um relatório à sua agência de origem).

Caso o `monitor_de_contrato` verifique que há uma violação moderada nos níveis de QoS de algum conjunto de fluxos, esse agente informa ao agente `adaptador_de_qos` da necessidade de se efetuar algum procedimento de adaptação na(s) fonte(s) de tal conjunto. As Figs. 5.16 e 5.17 exibem este cenário em MSC respectivamente para uma agência receptora e uma agência produtora de fluxo. É importante resaltar que os algoritmos de deteção de violação de contrato e adaptação de QoS são próprios dos agentes de monitoramento de contrato e adaptação de QoS, não sendo caracterizada assim pelo protocolo de troca de mensagens.

Caso o agente `adaptador_de_qos` receba de seu agente `monitor_de_contrato` remoto uma mensagem sinalizando uma violação severa de contrato, o primeiro pode sinalizar esta ocorrência ao agente `interface` local, via agente `mapeador_de_qos`, que por sua vez pode dar início a um processo de renegociação de contrato, através da criação de um agente `renegociador_de_contrato`. O diagrama MSC deste cenário é apresentado na Fig. 5.18. O

processo de renegociação de contrato é similar ao de negociação, cujo protocolo de mensagens foi apresentado na seção anterior.

5.3 Especificação do Protocolo para Encerramento de Sessão

Ao final de uma sessão há a necessidade de liberação dos recursos alocados aos fluxos constituintes da aplicação. Neste caso, cada agência deve liberar os seus recursos alocados, eliminar, caso tenha, seu agente `monitor_de_contrato` móvel e sinalizar o encerramento à agência proponente do contrato.

Quando um participante deseja se retirar isoladamente de uma sessão, sua agência deve gerar um agente `renegociador_de_contrato` que visitará todas as outras agências que tenham interação com esta para fins de liberação de recursos. Então, esse agente retorna a sua agência de origem, sendo que antes de se retirar da sessão a agência ainda deve eliminar seu agente `monitor_de_contrato móvel`.

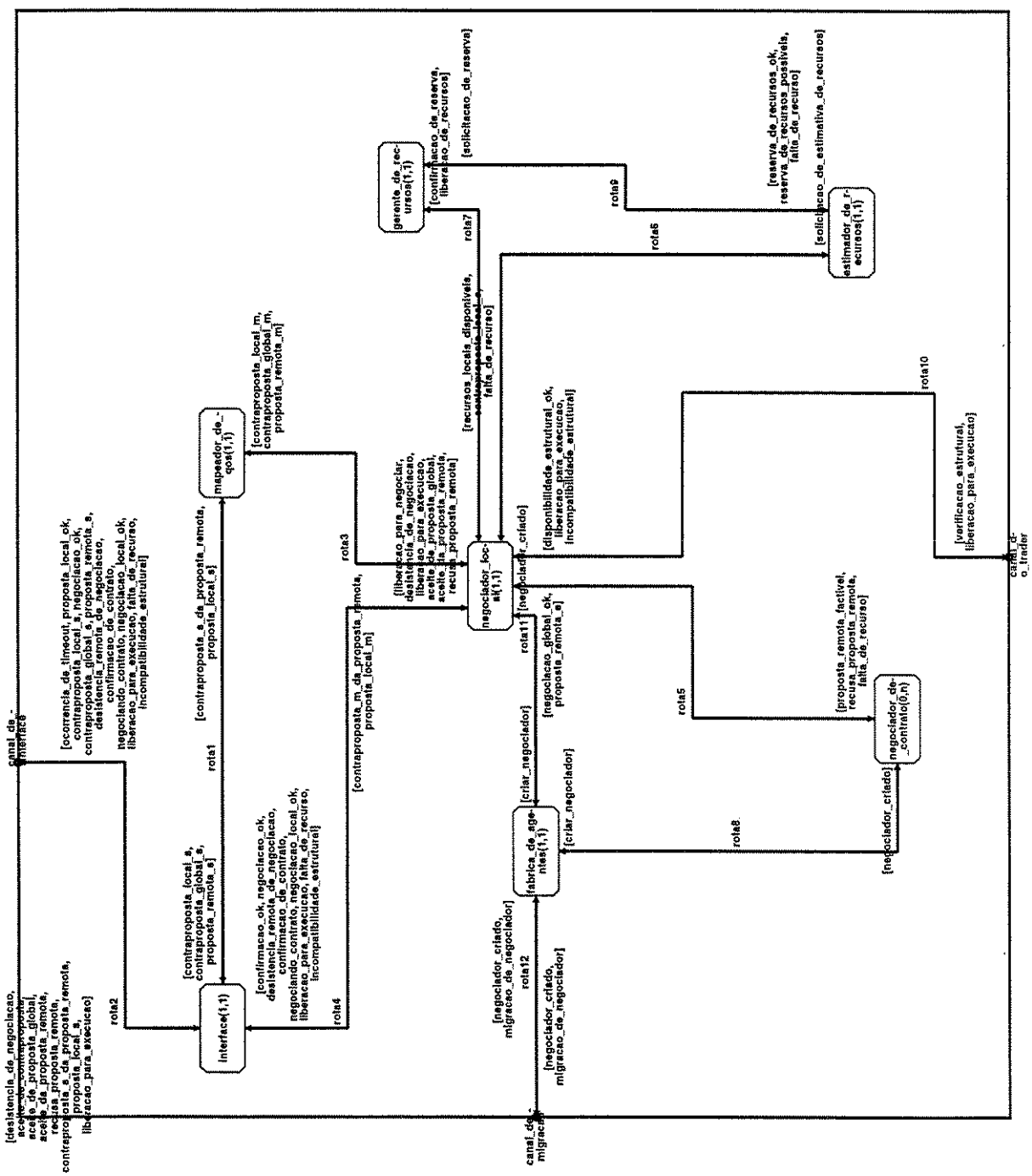

Fig. 5.2: Visão de interconexão em SDL do bloco Agencia_de_QoS, para a fase de estabelecimento de sessão.

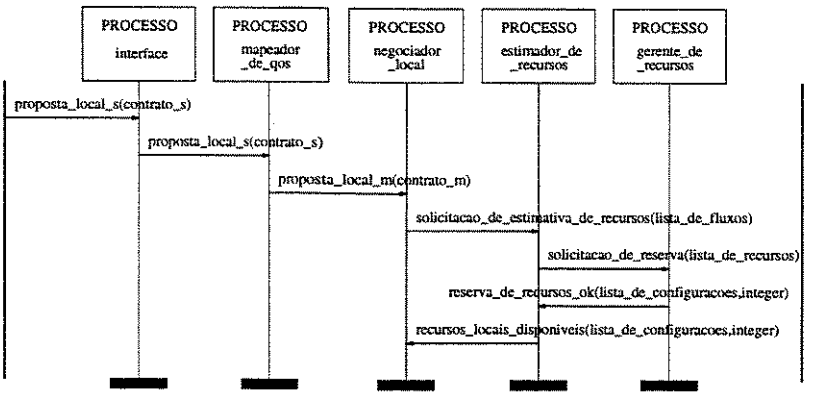

Fig. 5.3: Diagrama MSC da etapa de negociação local bem sucedida.

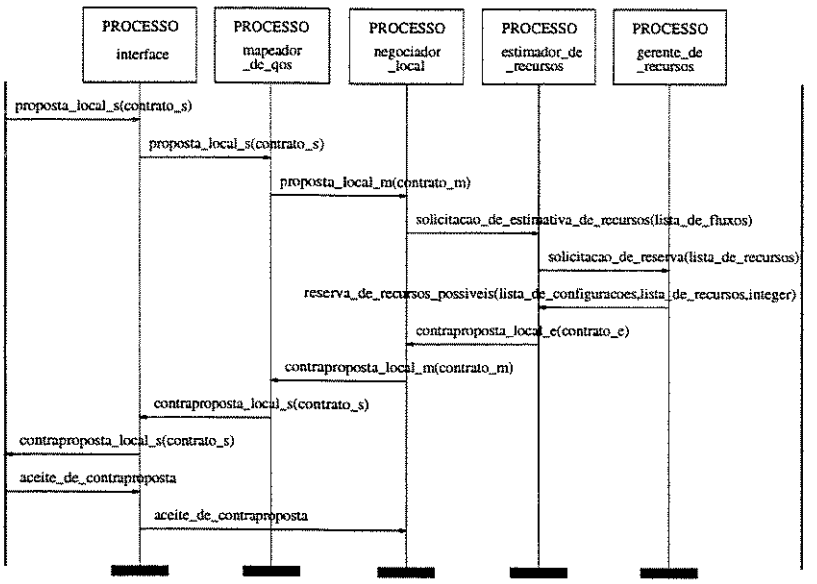

Fig. 5.4: Diagrama MSC do aceite de contraproposta de contrato.

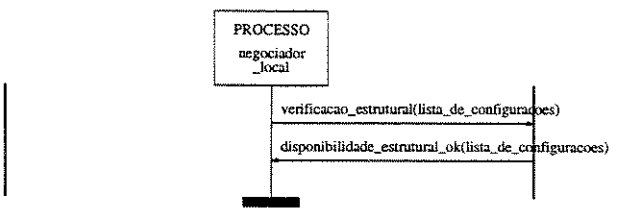

Fig. 5.5: Diagrama MSC da verificação de viabilidade de infra-estrutura global da proposta de contrato.

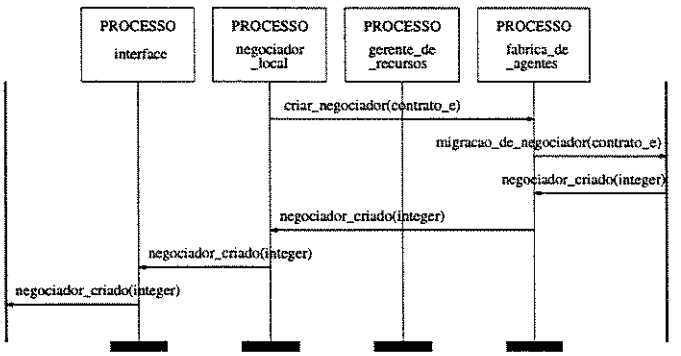

Fig. 5.6: Emulação do procedimento de migração do agente `negociador_de_contrato`.

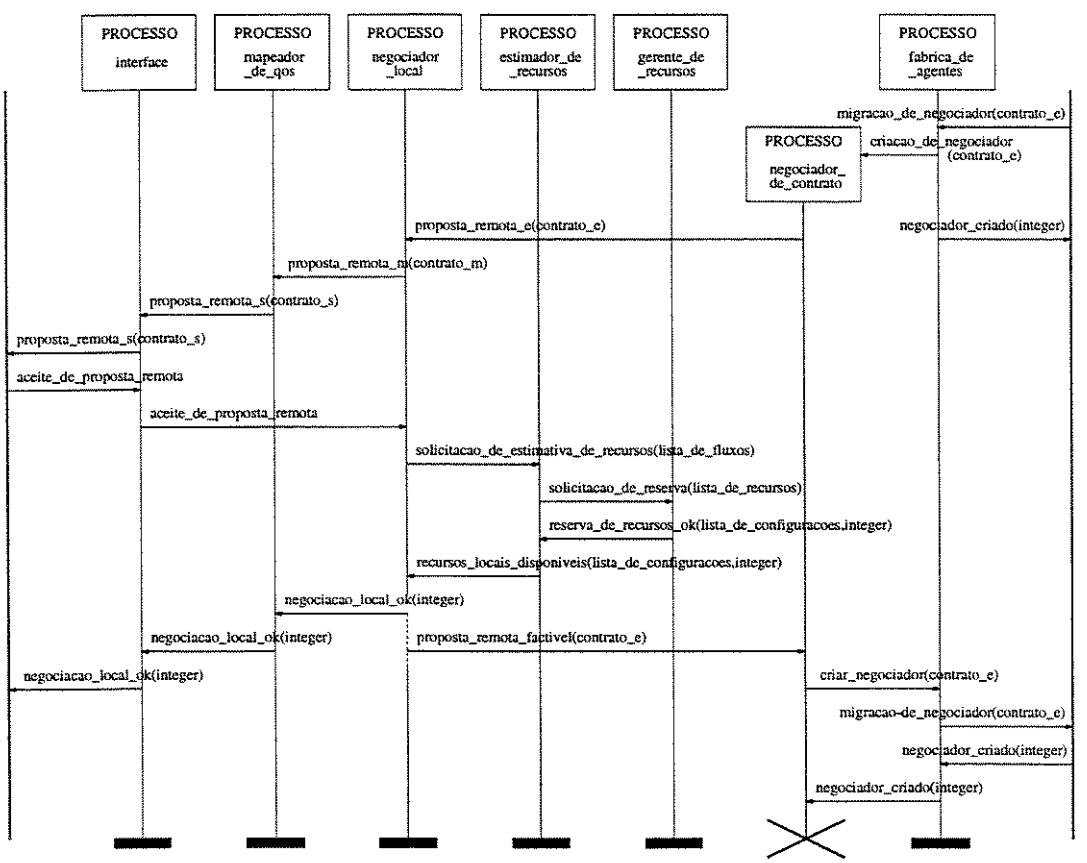

Fig. 5.7: Diagrama MSC de uma etapa de negociação de contrato bem sucedida em uma agência remota.

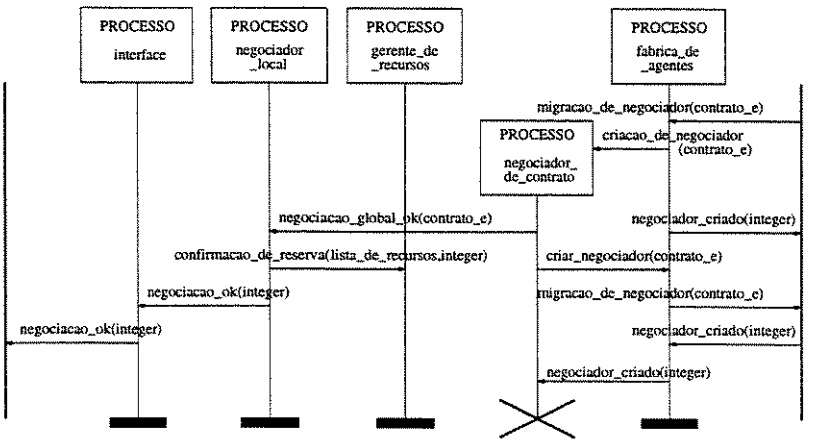

Fig. 5.8: Diagrama MSC do processo de confirmação de reserva de recursos da fase de negociação de contrato.

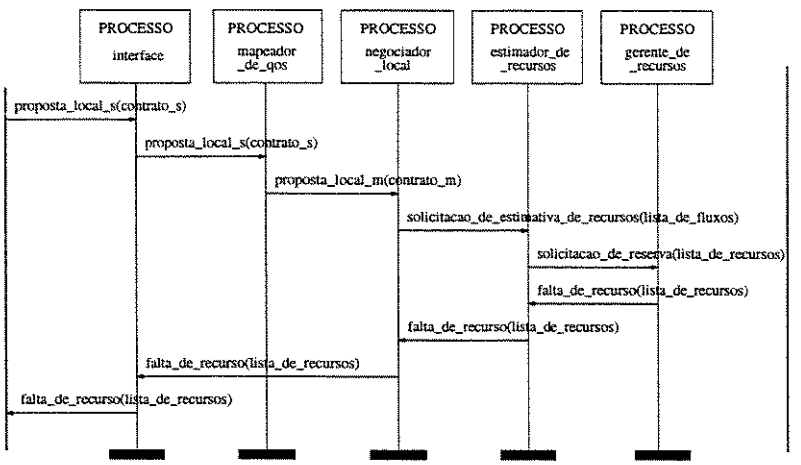

Fig. 5.9: Diagrama MSC de um cenário de falha local de negociação devido à falta de infra-estrutura.

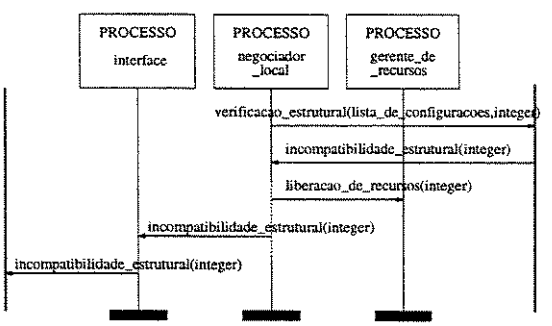

Fig. 5.10: Diagrama MSC do cenário de falha de negociação na etapa de consulta ao *trader*.

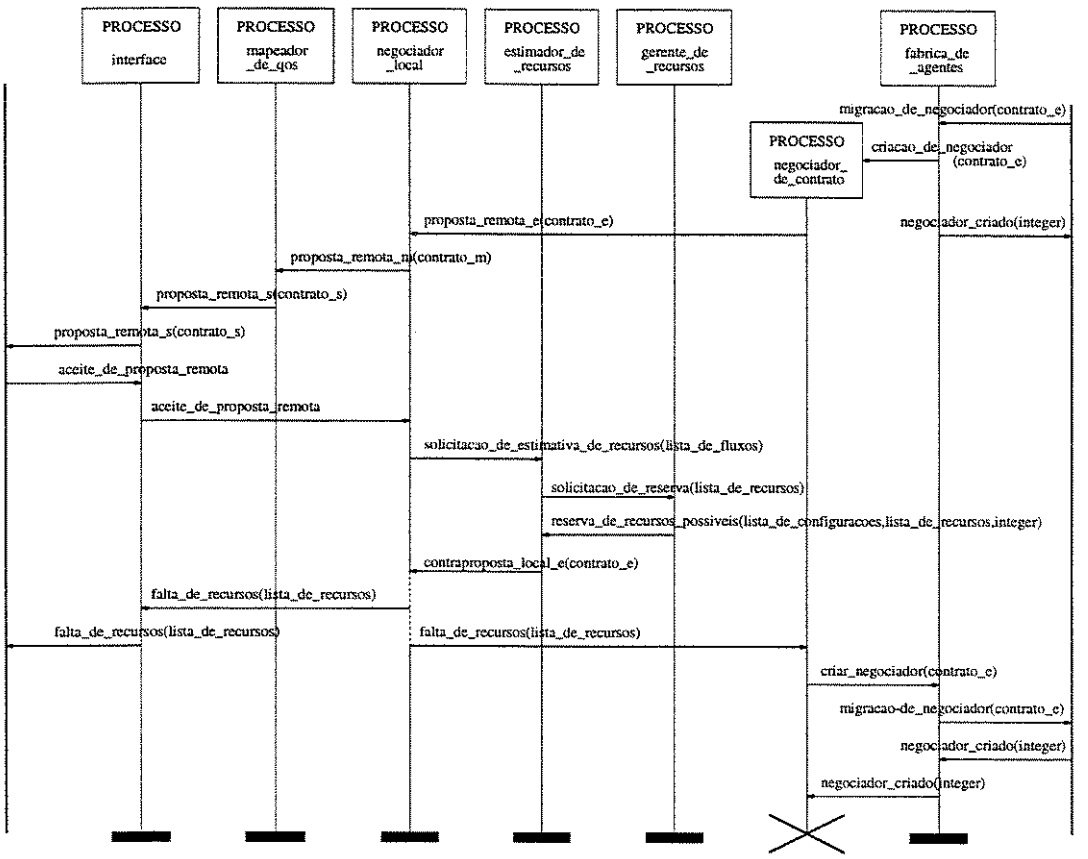

Fig. 5.11: Cenário de falha de negociação de contrato em uma agência remota.

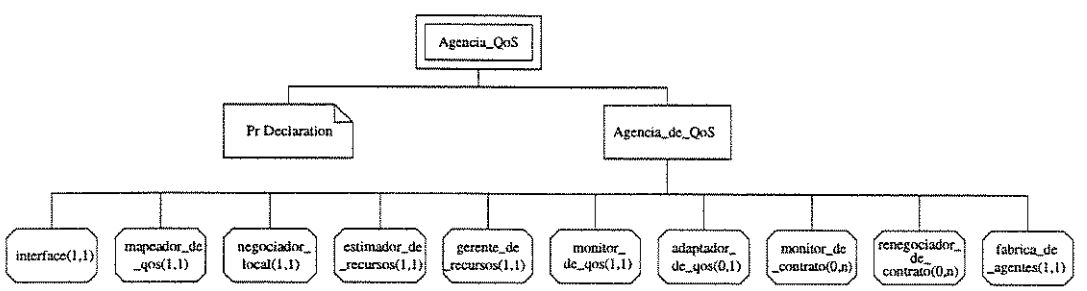

Fig. 5.12: Visão hieráquica em SDL da agência de QoS para a fase de atividade.

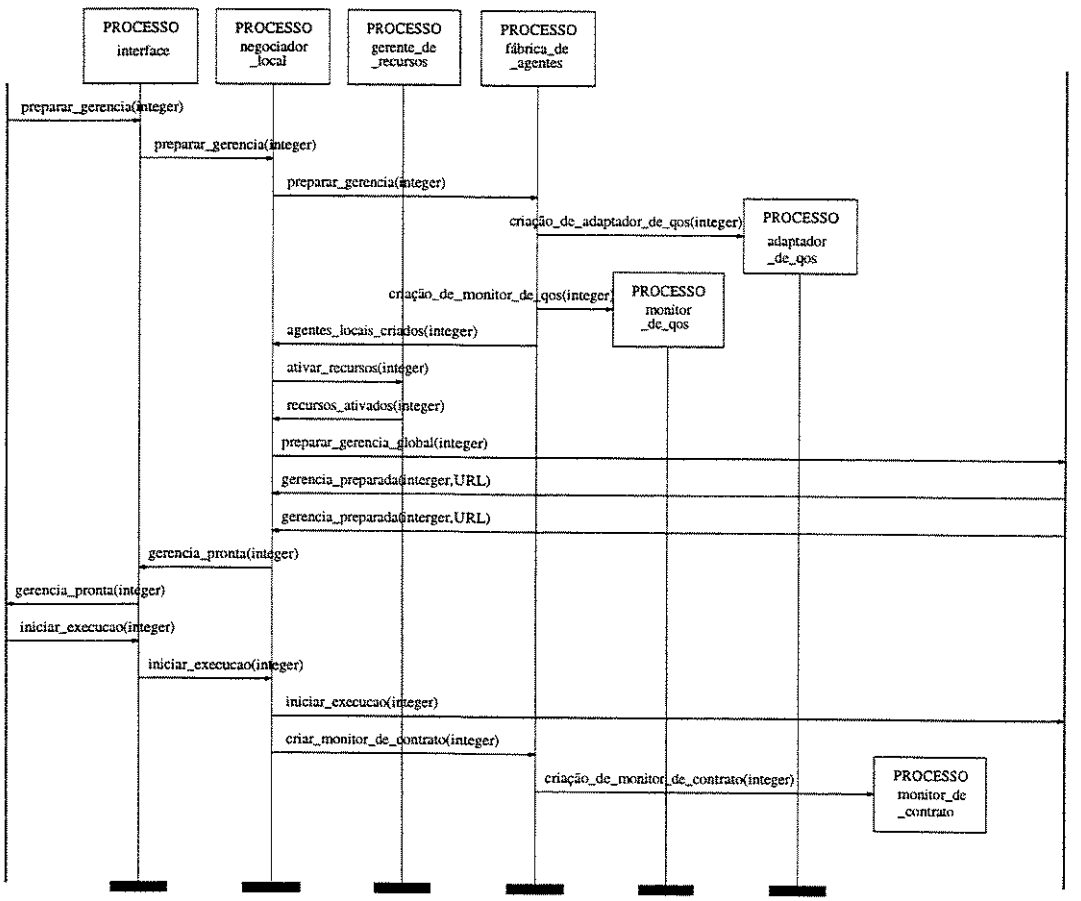

Fig. 5.13: Preâmbulo na agência proponente do contrato.

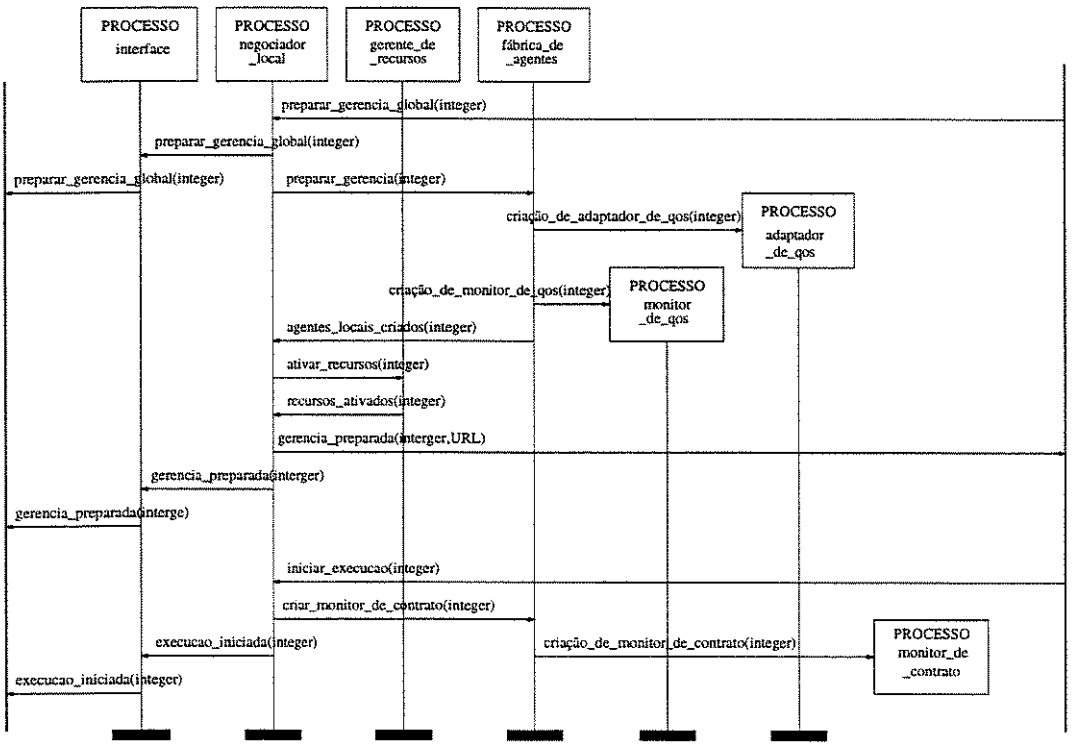

Fig. 5.14: Preâmbulo numa agência remota.

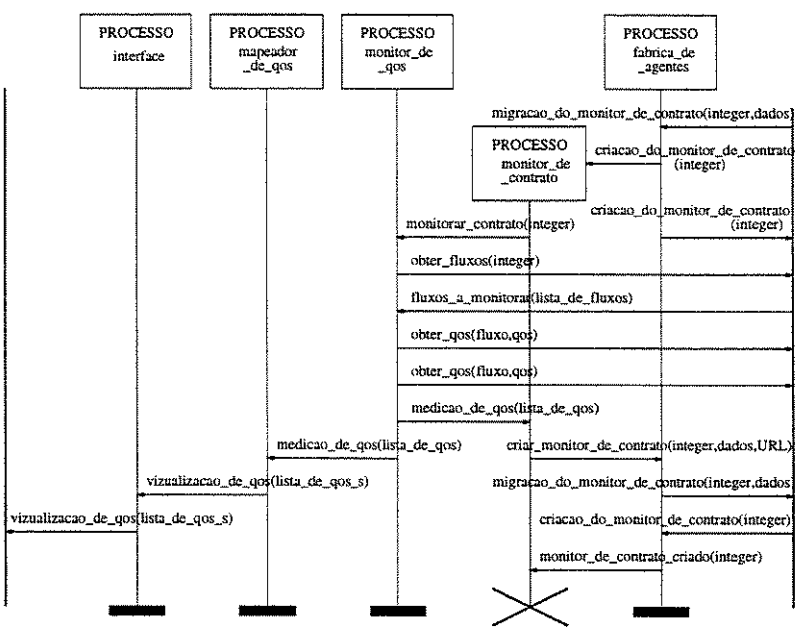

Fig. 5.15: Cenário de monitoramento de contrato numa agência receptora de fluxo.

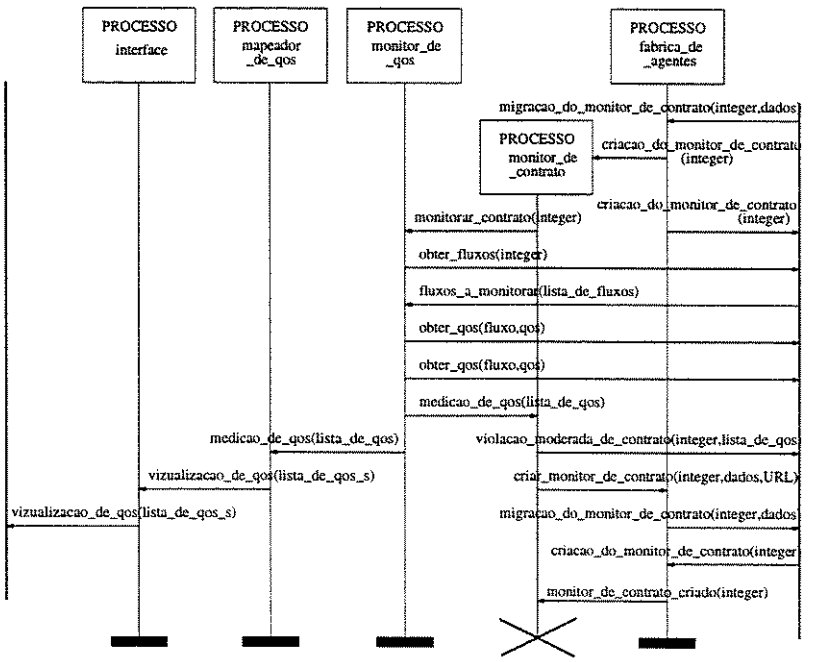

Fig. 5.16: Cenário de violação moderada de contrato numa agência receptora de fluxo.

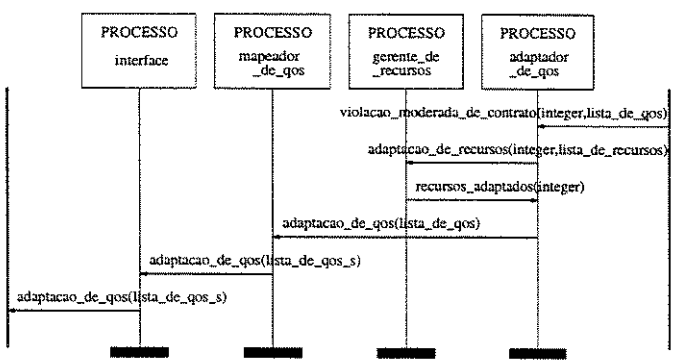

Fig. 5.17: Cenário de violação moderada de contrato numa agência produtora de fluxo.

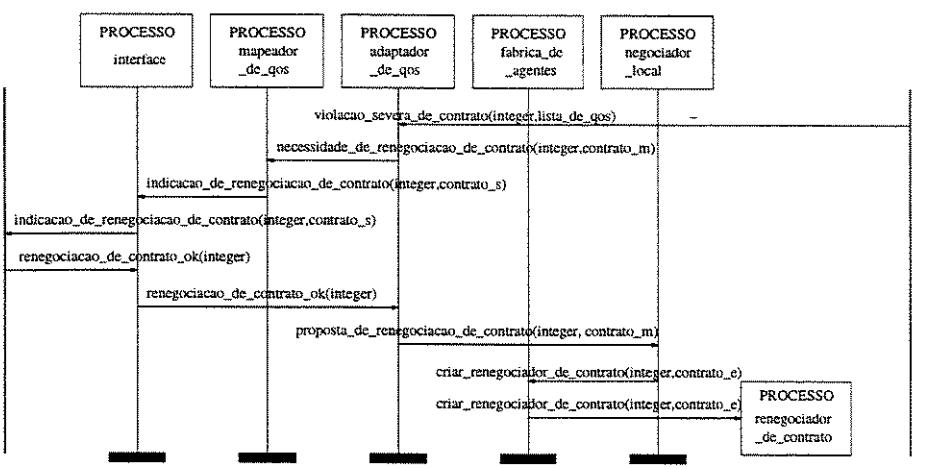

Fig. 5.18: Cenário de renegociação de contrato, numa agência produtora de fluxo, devido a violação severa de contrato.

Capítulo 6

Arquitetura de Implementação

Neste capítulo será descrito um modelo de arquitetura de implementação para aplicações multimídia distribuídas, com suporte a qualidade de serviço [25]. A Fig. 6.1 apresenta os componentes dessa arquitetura. O módulo de sistemas distribuídos provê funcionalidade de comunicação entre os diversos módulos. O restante do capítulo está dividido em seções que analisam mais detalhadamente cada módulo dessa arquitetura de implementação.

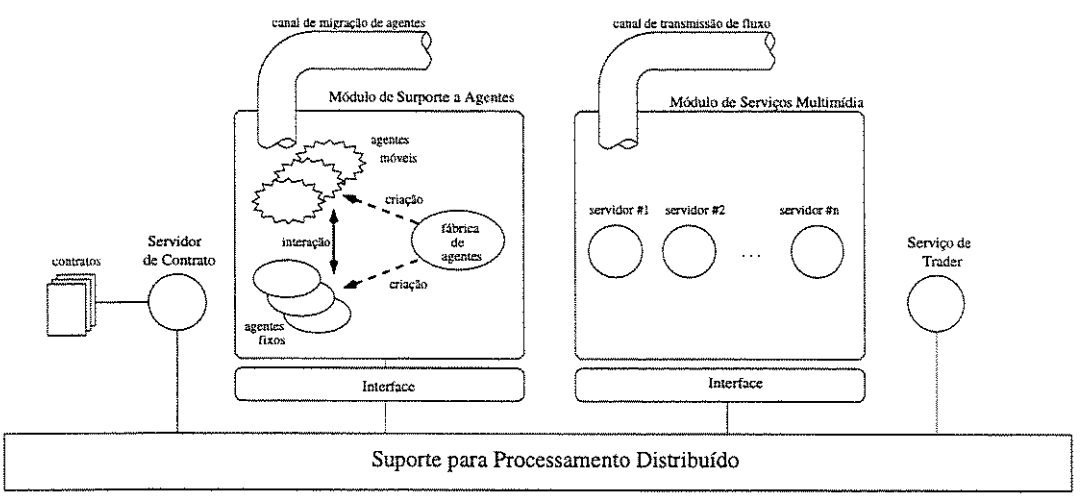

Fig. 6.1: Arquitetura de suporte à aplicação multimídia distribuída.

6.1 Suporte para Processamento Distribuído

Este módulo tem por função servir de meio de integração dos outros componentes da arquitetura de implementação, fornecendo suporte a serviços de distribuição, uma vez que boa parte das aplicações multimídia são de natureza distribuída, como teleconferência e vídeo sobdemanda. Além disto, este componente deve possuir característica de interação padronizada, de modo a permitir a interoperabilidade entre os outros componentes do sistema e tornar a aplicação multimídia mais independente possível da infra-estrutura de implementação. Por isto, optamos aqui por utilizar o padrão CORBA [4] do consórcio OMG (*Object*

Management Group) como modelo para suporte a processamento distribuído, pois acreditamos que tal modelo de objetos distribuídos se apresenta bastante apropriado para suportar os diversos tipos de serviços demandados pelas aplicações multimídia distribuídas.

A especificação CORBA (*Common Object Request Broker Architecture*) tem sua origem na necessidade de uma padronização de sistemas distribuídos orientados a objeto. A arquitetura distribuída orientada a objeto surgiu como uma solução para os problemas relacionados à complexidade e diversidade do *software*. Este paradigma possibilita a diversas aplicações, escritas em diferentes linguagens, executar sobre múltiplos sistemas operacionais e tecnologias de rede. Sistemas distribuídos oferecem a infra-estrutura básica que permite abstrair as camadas de comunicação enquanto a orientação a objeto fornece um ambiente para encapsulamento e reuso necessários à integração de aplicações. A especificação *Object Management Architecture* (OMA) é a visão completa do OMG de um ambiente distribuído. Esta visão é composta de cinco componentes principais, conforme mostra a Fig. 6.2.

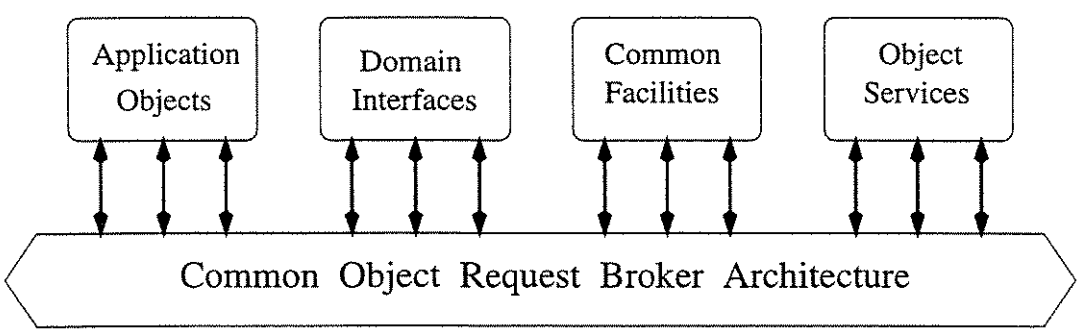

Fig. 6.2: Especificação da *Object Management Architecture*.

Object Request Broker (ORB) é o mecanismo de comunicação da especificação. ORB fornece uma infra-estrutura que permite a interação entre objetos independente da plataforma e técnicas utilizadas na implementação destes objetos. A conformidade com a especificação CORBA garante portabilidade e interoperabilidade de objetos sobre uma rede heterogênea.

Object Services definem a gerência do ciclo de vida de objetos. Interfaces são fornecidas para criar objetos, controlar o acesso a objetos, manter o mapeamento de objetos relocados e controlar a relação entre tipos de objetos (gerência de classe). Também são providos ambientes genéricos nos quais os objetos podem executar suas tarefas. Exemplos de *Object Services* são: *Object Naming Service*, *Event Notification Service*, *Persistent Object Service* e *Transaction Service*. Os *Object Services* fornecem consistência às aplicações e auxiliam o aumento da produtividade do programador.

Common Facilities correspondem a um conjunto de aplicações genéricas que podem ser configuradas de acordo com os requisitos específicos de uma determinada aplicação. Estas facilidades se situam no nível de usuário, como por exemplo, facilidades de impressão, correio eletrônico e gerência de documentos e banco de dados.

Domain Interfaces representam áreas verticais que fornecem funcionalidades de interesse do usuário final em domínios de aplicação particulares. As interfaces podem combinar algumas *Common Facilities* e *Object Services*, mas são projetadas para realizar tarefas particulares dentro de um determinado mercado ou indústria específicos.

Application Objects representam componentes (objetos) que realizam tarefas particulares ao usuário final. Uma aplicação é tipicamente construída a partir de um grande número de objetos básicos - alguns específicos à aplicação, outros ao domínio, alguns construídos a partir de *Object Services* e outros de um conjunto de *Common Facilities*.

A especificação CORBA utiliza para construção e integração de aplicações distribuídas os paradigmas de orientação a objetos e cliente-servidor. Neste modelo, os objetos são os componentes de distribuição e podem desempenhar o papel de cliente, quando requisitam uma operação, ou de servidor, quando recebem, executam e respondem à requisições (Fig 6.3).

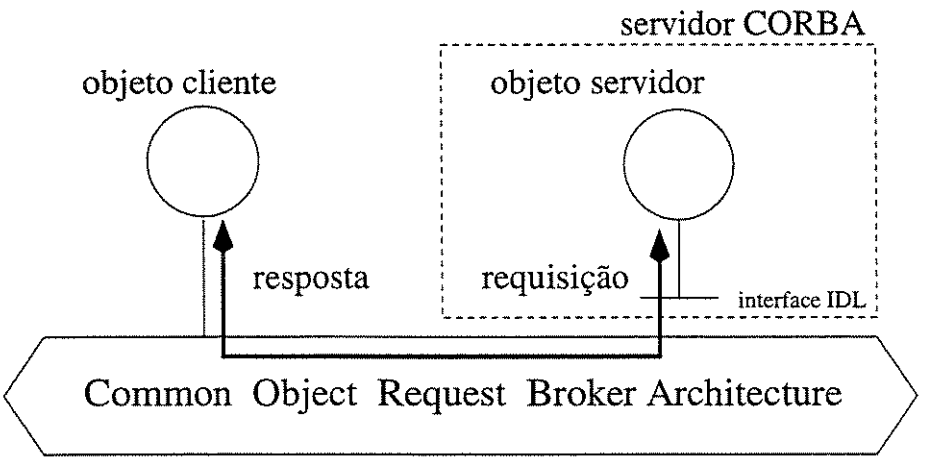

Fig. 6.3: Interação de objetos Cliente/Servidor via CORBA.

Para permitir interações entre objetos, cada objeto deve notificar aos seus potenciais clientes quais operações são possíveis e como elas devem ser invocadas. Em outras palavras, devem definir suas interfaces. A linguagem *Interface Definition Language* (IDL), também especificada pelo OMG, define o tipo do objeto, seus atributos, métodos e os parâmetros dos métodos.

Cada objeto está associado a um servidor, podendo o servidor gerenciar um conjunto de objetos com diferentes ou idênticas interfaces. Os servidores provêem objetos a serem manipulados por um ou mais clientes. Se um servidor não estiver ativo o mesmo pode ser ativado quando um de seus objetos recebe uma requisição. Para que isto seja possível todos os servidores devem ser registrados num repositório de implementações.

Atualmente há várias implementações CORBA, tanto comerciais quando acadêmicas, sendo que a maioria delas suporta implementações nas linguagens C++ e Java, isto é, possuem compiladores de IDL para C++ e Java.

6.2 Módulo Servidor de Contrato

Este componente da arquitetura tem por funcionalidade oferecer serviços de armazenamento persistente, atualização e consulta de contratos de QoS para as aplicações multimídia distribuídas. Este componente é fundamental para a realização das atividades de negociação e de gerência da qualidade de serviço de uma aplicação multimídia.

Aqui o servidor de contrato é modelado como um servidor CORBA, cuja interface de serviço, descrita em IDL, é:

```
interface ServidorContratoQoS {

//   definicao dos tipos de dados manipulados pela interface
// sequencia de strings
typedef sequence <string>seqStrings;

//   parametros de QoS que serao monitorados
struct fluxoParamsStruct {

   unsigned long packetRate;
   unsigned long minPacketRate;
   unsigned long maxPacketRate;

   unsigned long delay;
   unsigned long minDelay;
   unsigned long maxDelay;

   double jitter;
   double minJitter;
   double maxJitter;

   unsigned long packetErrorRate;
   unsigned long minPacketErrorRate;
   unsigned long maxPacketErrorRate;
};

// definicao de tipo fluxoParamsQoS
typedef fluxoParamsStruct fluxoParamsQoS;

//   formato do fluxo
struct fluxoFormatoStruct {
string tipoCompactador;
string padraoCompactador;
float amostrasPorSegundo;
char padraoDaAmostra;
}

// definicao de tipo  fluxoFormato
typedef fluxoFormatoStruct fluxoFormato;


// estrutura que define fluxo multimidia para gerencia de QoS
struct fluxoStruct {
```

```
unsigned short tipo;
string origem;
seqStrings destinos;
fluxoParamsQoS params;
fluxoFormato formato;
string tipoTransporte;
};

// definicao de tipo fluxoQoS
typedef fluxoStruct fluxoQoS;

// definicao de tipo sequencia de fluxos
typedef sequence <fluxoQoS>              seqFluxoQoS;

// estrutura que define contrato para gerencia de QoS
struct contratoStruct {
string nome;
string versao;
seqFluxoQoS fluxosQoS;
};

// definicao do tipo contratoQoS
typedef contratoStruct   contratoQoS;

// definicao das operacoes da interface
contratoQoS getContratoQoS(in string nomeContrato, in string versao);
boolean     lerContratos(in string arquivo);
boolean     alterarContrato(in string nomeContrato, in string versao,
    in contratoQoS novocontrato);
string     getUltimaVersao(in string nomeContrato);
};
```

Pode-se verificar a partir da descrição de sua interface que um contrato de QoS (`contratoQoS`) é composto de nome, versão e uma seqüência de fluxos (`seqFluxosQoS`). Por sua vez, um fluxo (`fluxoQoS`) se constitui de tipo da mídia (que pode ser vídeo ou áudio por exemplo), tipo de protocolo de transporte (`tipoTransporte`) e endereços das máquinas onde estão localizados a fonte e os destinos do fluxo (`origem` e `destinos`, respectivamente), além de parâmetros de formato (`fluxoFormato`) e de qualidade de sua transmissão (`fluxoParamsQoS`). O `fluxoFormato` está mais associado às características com que a mídia foi produzida, enquanto o `fluxoParamsQoS` está mais associado às características de qualidade do meio de transmissão.

6.3 Módulo de *Trading*

Este serviço corresponde ao *trading object service* especificado pelo OMG [63], que basicamente corresponde a facilidades de se ofertar e descobrir serviços de tipos pré-estabelecidos. Deste modo, o *trader* se caracteriza como um objeto CORBA através do qual outros objetos podem tanto cadastrar como consultar referências de serviços, denominados respectivamente de exportador e importador de serviço.

A Fig 6.4 ilustra as interações que ocorrem entre um *trader* e seus clientes. Para exportar (interação 1), um objeto fornece ao *trader* uma descrição do serviço junto com a localização da interface na qual o serviço é disponível. Para importar (interação 2), um objeto pergunta ao *trader* pelo serviço que tem determinadas características. Então, o *trader* verifica se há algum serviço que atenda tal descrição e responde ao importador com a localização da interface do serviço selecionado. A partir daí o importador está habilitado a interagir com o serviço (interação 3).

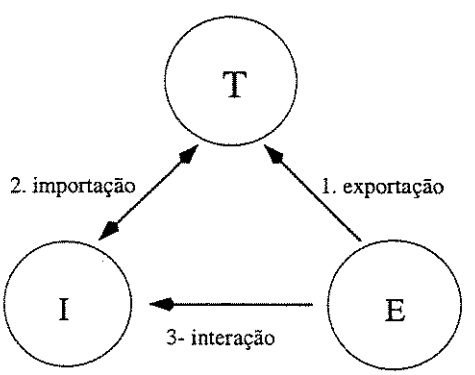

Fig. 6.4: Interações entre um *trader* e seus clientes.

Este tipo de serviço se destina principalmente a aplicações como vídeo sob demanda, onde pode-se ter a princípio mais de um servidor de vídeo exportando serviços similares. No caso, o serviço é escolhido dentre um grupo de possíveis serviços, sendo que esta atividade é realizada pelo agente negociador_local (vide capítulo 4). Porém, para aplicações do tipo video-conferência, onde as localizações dos serviços são pré-estabelecidas, o serviço de *trader* não se apresenta como fundamental.

6.4 Módulo de Serviço Multimídia

Como abordado no Capítulo 2, há vários tipos de serviços multimídia. Aqui, por questões de simplicidade, classificamo-os como: serviços de dispositivos; serviços de transporte e serviços de controle e configuração. Estes serviços são implementados como servidores CORBA.

Os serviços de dispositivos modelam componentes de *hardware* e *software* do sistema que são fontes ou destinos de fluxos, como microfone, alto falante e base de dados multimídia. As funcionalidades desses serviços são basicamente captura, exibição e ativação de base de dados, além de tratamento de sinal, como compactação e descompactação de dados.

Os serviços de transporte têm como responsabilidade transportar os dados de mídias contínuas desde sua fonte até seus receptores. Os serviços de transporte podem ser orientados a conexão (com garantia de entrega) ou não orientados a conexão. Como já mencionado no Capítulo 2, particularmente para os fluxos de mídias contínuas, o processo de retransmissão, típicos dos serviços orientados a conexão, pode acarretar efeitos indesejáveis na qualidade do serviço de transporte, devido ao aumento do jitter e atraso. Além disto, pode adicionar uma carga desnecessária ao transporte de dados de tempo crítico se os dados estão sendo transmitidos por um sistema de transporte de alta confialibilidade.

A atual versão da especificação CORBA não provê suporte a interfaces do tipo fluxo contínuo. Assim, não é possível a utilização de IDL para definir uma interface de modo a se utilizar o ORB como meio de transmissão de fluxos contínuos. Outros aspecto é que a comunicação entre os objetos CORBA é tipicamente orientada a conexão e para o caso de fluxo contínuo, como já ressaltado, o uso deste tipo de serviço não é o mais indicado. Por esta razão optou-se por modelar o transporte dos dados multimídia "por fora" do ORB, utilizando-se um protocolo de transporte multimídia com característica de protocolo leve, isto é, não orientado a conexão e com baixa sobrecarga adicional [52].

Os serviços de configuração e controle se caracterizam por tratarem dos aspectos de gerência dos outros dois tipos de serviços previamente abordados. Por configuração, entende-se a seleção de padrões compatíveis para todas as etapas por que percorrem os fluxos contínuos. Por exemplo, um vídeo compactado num padrão MPEG não pode ser descompactado via algoritmo CellB. Outro exemplo, um fluxo de áudio capturado a uma taxa de 8 KHz e com 16 bits de precisão não pode ser exibido a 16 KHz, mesmo mantendo-se o nível de precisão, pois suas características sonoras seriam totalmente deturpadas. No caso, para se manter a integridade da informação há a necessidade, antes de sua exibiçao, de se converter o sinal amostrado em 8 KHz para 16 KHz, via um processo de interpolação, por exemplo.

Os serviços de controle, por sua vez, correspondem à atividade de monitoramento dos níveis de qualidade de serviço dos fluxos, medidos a partir dos parâmetros de QoS, além das atividades que atuam propriamente sobre as características dos fluxos contínuos. Esses serviços são vistos pela agência de QoS como sensores e atuadores de QoS, respectivamente.

A Fig 6.5 ilustra interações dos serviços multimídia entre módulos de serviços multimídia.

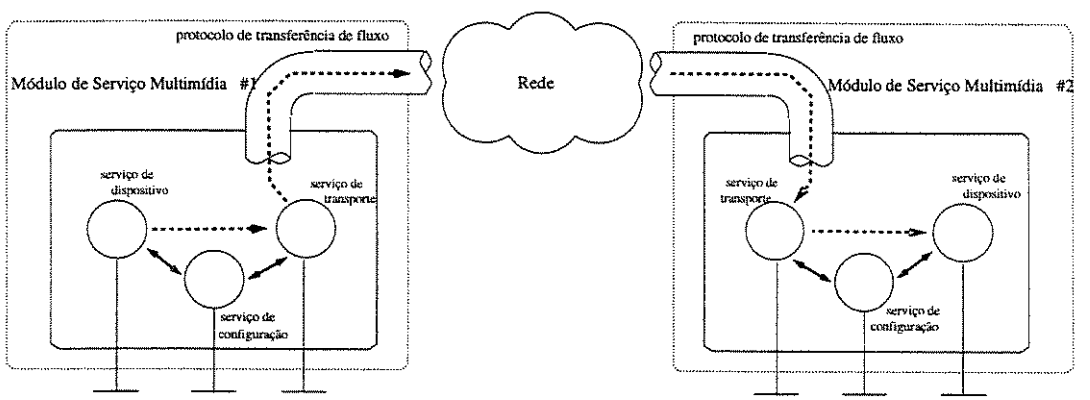

Fig. 6.5: Interação dos componentes do módulo de serviço multimídia.

Para que se possa utilizar esses serviços multimídia eficientemente, é necessário que eles

apresentem interfaces bem especificadas e padronizadas, além de facilidade de distribuição, de tal modo que aplicações multimídia tenham seus custos de desenvolvimento reduzidos consideravelmente.

Nessa perspectiva, o OMG especificou seu serviço CORBA para gerência e controle de fluxos de áudio e vídeo denominado *Control and Management of Audio/Video Streams* (AVStreams) [35]. Essa especificação contém as interfaces IDL com sua estrutura de herança, as quais implementam um ambiente para desenvolvimento de aplicações distribuídas que contém fluxos de áudio e ou vídeo. Em linhas gerais, esses serviços podem ser agrupados de maneira similar à feita nesta seção, ou seja: serviços de dispositivos, serviços de configuração e controle e serviços de transporte. O transporte dos fluxos de áudio e de vídeo são realizados por fora do ORB, utilizando-se tanto protocolos leves (UDP, por exemplo) quanto protocolos confiáveis (TCP, por exemplo).

6.5 Módulo de Serviço de Agentes

Este módulo tem a responsabilidade de prover a infra-estrutura necessária ao suporte de execução dos agentes que foram especificados no capítulo anterior. Tal infra-estrutura deve permitir criação, execução, migração e término de agentes, bem como a comunicação entre agentes, agentes e serviços localizados em outros módulos do sistema e agentes e seres humanos. A comunicação entre agentes e serviços localizados em outros módulo deve se dar via ORB, o que torna necessário que este módulo seja interoperável com CORBA.

Apesar do OMG definir uma "facilidade para agentes móveis" (MAF) [16], não existe até o momento qualquer implementação desta e as infra-estruturas disponíveis, tanto comerciais quanto não-comerciais, não utilizam CORBA como meio de migração de agentes e sim protocolos próprios. em vista disso, no nosso modelo de implementação optamos por realizar a migração dos agentes por um canal específico, `canal de migração de agentes` (Fig. 6.1), e não via ORB. Tal escolha de projeto se deve ao fato que a utilização do CORBA como meio de transmissão dos agentes implicaria em construir todo o protocolo para migração de agente, não aproveitando os protocolo já existentes.

Devido ao grande interesse por aplicações baseadas em agentes, houve um grande esforço, tanto na academia quanto na indústria, por desenvolvimento de infra-estruturas para suportar tais tipos de aplicações. A Tab. 6.1 apresenta um conjunto de implementações existentes de infra-estruturas para o suporte a sistemas baseados em agentes móveis, indicando sua origem e linguagens suportadas [51].

Como pode-se observar a partir da Tab. 6.1, a linguagem Java é a preferida para o desenvolvimento de infra-estrutura de suporte a agentes móveis. Isto se deve às características dessa linguagem, como: portabilidade, suporte a distribuição, suporte à *multithreading*, orientação a objeto, segurança e suporte a serialização de objetos [61]. Além disso, seu ambiente de execução é baseado numa arquitetura de máquina virtual, característica fundamental para a execução de agentes móveis pois implica em código independente de plataforma. Outro aspecto importante é que a partir da versão 1.2, Java já incorpora um ORB CORBA, tornando a interoperabilidade com CORBA embutida na própria linguagem.

Em nossa implementação, escolhemos a infra-estrutura Aglets [33] para o suporte a serviço de agentes, a qual foi desenvolvida no laboratório de Pesquisa da IBM em Tóquio. Tal escolha deveu-se a vários fatores, como: modelo abrangente e bem projetado, facilidade de

Infra-estrutura	Origem	Linguagem
Tabriz	Indústria	Telescript
Odyssey	Indústria	Java
Agent Tcl	Academia	Tcl/Tk
Aglets	Indústria	Java
Voyager	Indústria	Java
TACOMA	Academia	C e Tcl/Tk
Mole	Academia	Java
Ara	Academia	Tcl/Tk e C/C++

Tab. 6.1: Implementações disponíveis de infra-estruturas para suporte a agentes móveis.

operação, boa documentação, utilização da tecnologia Java, e por se encontrar num estágio de desenvolvimento relativamente estável.

A infra-estrutura Aglets é implementada completamente em Java e composta basicamente de um servidor; de um ambiente de gerência de agentes (que em sua terminologia é denominado de contexto) e de um protocolo de transferência de agentes denominado *Agent Transfer Protocol* (ATP). Um agente, denominado de *aglet*, é implementado como uma classe em Java que herda as característica de uma classe básica Aglets, sendo que essa classe básica implementa funcionalidades de suporte a serialização de código, que é a base do mecanismo de migração deste modelo. Além disto, a infra-estrutura Aglets suporta um mecanismo bastante versátil de troca de mensagem entre agentes.

O seu ambiente de gerência de agentes suporta criação, migração, duplicação e remoção de agentes, além de mecanismo de troca de mensagens síncronas e assíncronas entre agentes locais e remotos. A Fig 6.6 ilustra a arquitetura da infra-estrutura Aglets.

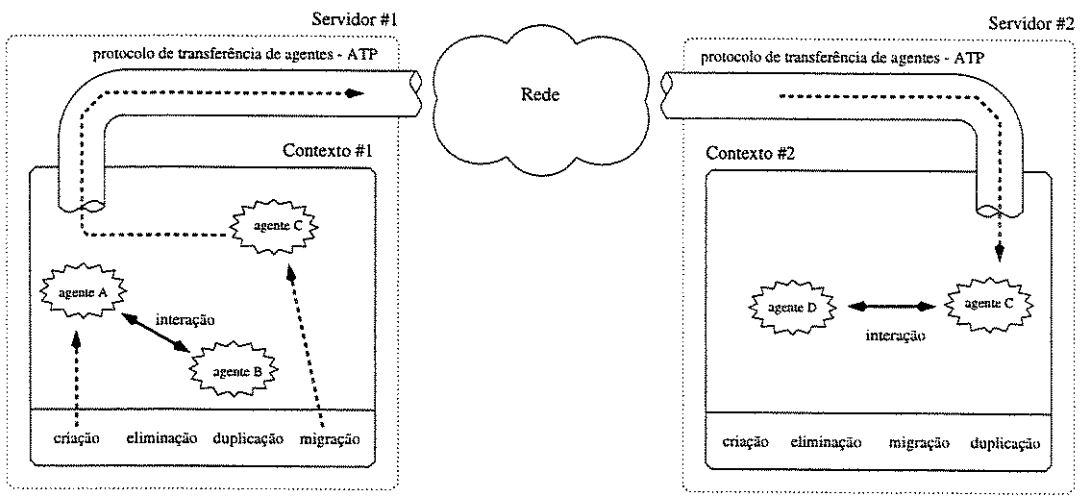

Fig. 6.6: Arquitetura Aglets.

6.6 Infra-estrutura da Arquitetura de Implementação

A infra-estrutura de implementação da qual obtivemos os resultados expostos no próximo capítulo é ilustrada em Fig 6.7.

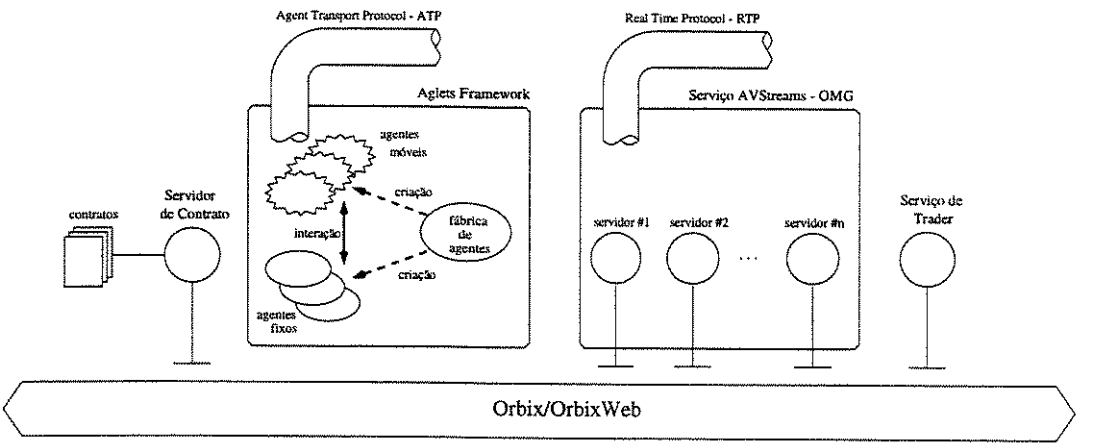

Fig. 6.7: Infra-estrutura de implementação.

Utilizamos os produtos da família Orbix da Iona Technologies, Inc [36] [38] como infra-estrutura CORBA. Esses produtos estão em conformidade com a especificação CORBA 2.0 e provêem suporte para implementações de servidores e clientes nas linguagens C++ (Orbix) e Java (OrbixWeb).

O servidor de contrato foi implementado em Java como um servidor OrbixWeb. O agentes de QoS e a fábrica de agentes são objetos implementados em Java com extensão Aglets, sendo que eles são somente clientes dos outros serviços CORBA.

Como serviço multimídia foi utilizada uma implementação do serviço AVStreams do OMG [13], onde os objetos servidores foram desenvolvidos na linguagem C++, sendo que os sensores e os atuadores de QoS são métodos implementados pelas interfaces desse serviço.

Utilizamos uma implementação do RTP (*Real Time Protocol*) [56] como protocolo de transporte dos fluxos multimídia. Como usual, o RTP foi utilizado sobre o protocolo UDP. A escolha do RTP deveu-se às suas características de protocolo leve, baixo *overhead*, além de possuir campos em seu cabeçalho destinados a aspectos temporais e de ordenação, tais como tempo de criação do pacote RTP e número do pacote. São justamente desses campos que são extraídas informações através das quais obtém-se os parâmetros de QoS da rede (atraso, jitter e PER).

O sistema foi desenvolvido e testado para plataforma Solaris/Unix em *workstations* SPARC e ULTRA, interligadas por uma rede Ethernet.

Para o serviço de captura e exibição de vídeo foi utilizada a biblioteca XIL da SUN [62], codificada em C++, que utiliza CellB como algoritmo de compactação e descompactação de vídeo. O serviço de captura e exibição de audio foi desenvolvido a partir do disposivo nativo /dev/audio do sistema operacional Solaris.

Em nosso protótipo não utilizamos os serviços do *trader*, pois nossas aplicações foram voltados para teleconferência. Porém, já há algumas implementações do *trader* OMG dis-

poníveis, tanto comerciais quanto acadêmicas. A própria Iona Technologies, Inc já comercializa um serviço de *trader* OMG [37].

No próximo capítulo serão apresentados e analisados alguns cenários de utilização desta infra-estrutura em teleconferência.

Capítulo 7

Aspectos de Implementação e Resultados Obtidos

Neste capítulo serão apresentados e analisados alguns resultados obtidos com um protótipo da arquitetura de implementação descrita no capítulo anterior. Os cenários de teste foram realizados sobre aplicações de teleconferência.

O objetivo principal para o desenvolvimento do protótipo foi avaliar a adequação do modelo baseado em agente para a gerência de qualidade de serviço em aplicações multimídia distribuídas. Por adequação entende-se características como: simplicidade, abrangência, modularidade, flexibilidade e desempenho.

No protótipo foi desenvolvido o servidor de contrato e parte da agência de QoS [51] [49]. Como módulo de serviço multimídia foi utilizada uma implementação da especificação AVStreams do OMG [13]. Com relação à agência de QoS, os agentes implementados foram aqueles relacionados à fase de execução da sessão, a saber: agente `interface`, agente `monitor_de_qos`, agente `monitor_de_contrato` e agente `adaptador_de_qos`, além da fábrica de agentes, que também foi implementada como um agente. Como o agente `mapeador_de_qos` não foi implementado, o agente `interface` se comunica diretamente com os agentes `adaptador_de_qos`, `monitor_de_contrato` e `monitor_de_qos`.

A implementação desse conjunto mínimo de agentes possibilitou a realização de testes enfocando aspectos de monitoramento e adaptação de qualidade de serviço em aplicações de teleconferência. A escolha por tal conjunto deveu-se ao fato de não dispormos de infra-estrutura que garantisse níveis de qualidade de serviço previamentes estabelecidos, como sistemas operacionais de tempo real e infra-estrutura de rede com garantia de banda de transmissão. Deste modo, aplicações relacionadas com reserva de recursos, típicas da fase de negociação de contrato, ficaram inviáveis.

O servidor de contratos foi implementado na linguagem Java, a partir da interface IDL descrita no capítulo anterior. Os contratos de QoS são cadastrados via este servidor e armazenados em uma base de contratos. Esta base de contratos é um arquivo seqüencial de objetos do tipo contrato. A Fig 7.1 ilustra a estrutura de um contrato de QoS.

Procedimentos de inserção, modificação e exclusão de contratos na base são de responsabilidade dos agentes de negociação e renegociação de contrato. Entretanto, como esses agentes não foram implementados no protótipo, optou-se por desenvolver uma aplicação de cadastro de contrato [49], também escrita em Java, funcionando como cliente do servidor

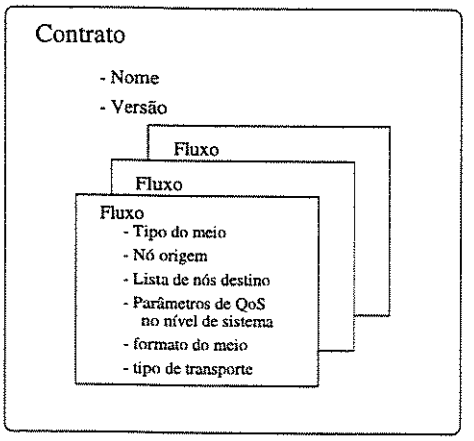

Fig. 7.1: Estrutura de um contrato de QoS.

de contrato. A Fig 7.2 ilustra este mecanismo de cliente-servidor via CORBA, enquanto as Figs 7.3 e 7.4 mostram algumas interfaces gráficas que compõem a aplicação de cadastro de contratos.

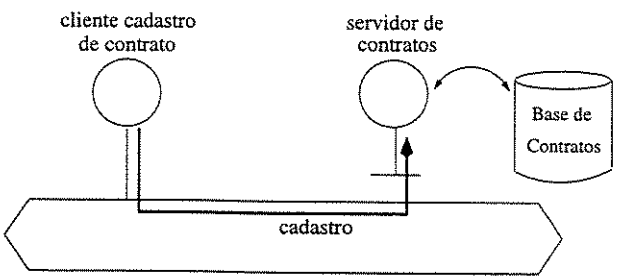

Fig. 7.2: Modelo cliente-servidor da aplicação de cadastro de contrato.

A interface gráfica principal da aplicação de cadastro de contrato, Fig. 7.3, permite a manipulação de dados de identificação de contratos, como nome e versão. A interface de cadastro de contrato, Fig. 7.4, permite a manipulação de dados de identificação de fluxos, como tipo da mídia e nó de origem. As demais informações pertinentes a um fluxo podem ser acessadas por outras interfaces gráficas, que são geradas a partir da interface de cadastro de contrato.

Para começar uma aplicação multimídia faz-se necessário ter uma instância de agência de QoS em cada nó participante da aplicação. No caso, essa instância se constitui do ambiente Aglets contendo uma instância de um agente fábrica de agentes.

Cada fábrica de agentes recebe uma sinalização para configurar a aplicação contendo o nome e a versão do contrato. Como conseqüência, cada fábrica de agentes consulta seu servidor de contratos, obtendo o contrato correspondente, para em seguida instanciar seus agentes necessários à aplicação: agente `interface`; agente `monitor_de_contrato` e agente `adaptador_de_qos` para as agências que têm fontes de fluxos, e agente `monitor_de_qos` para as que têm consumidores de fluxos. Também são instanciados os servidores multimídia necessários à aplicação, com configuração de formato apropriada. A Fig. 7.5 ilustra esse

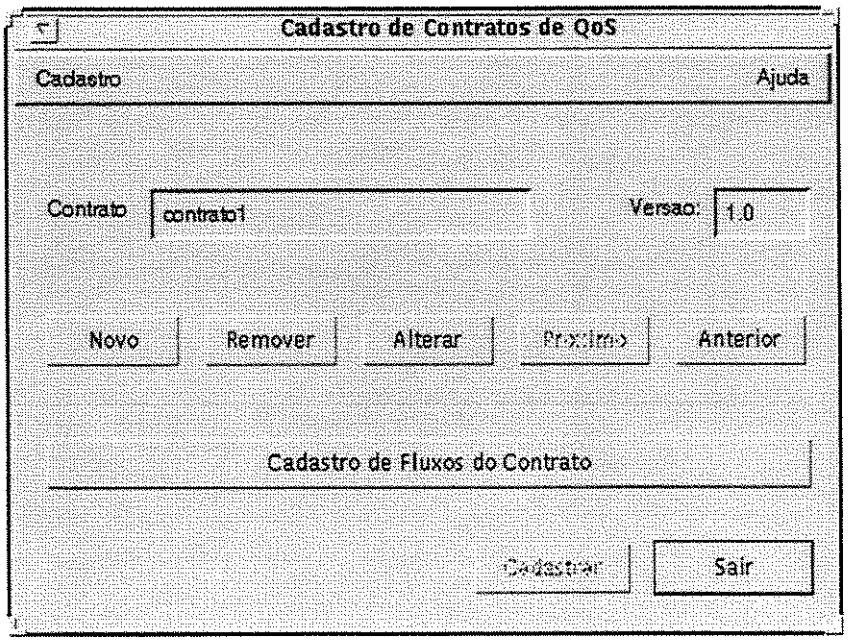

Fig. 7.3: Interface principal da aplicação de cadastro de contratos.

procedimento de configuração da agência de QoS. No caso, a sinalização de começo da aplicação é proveniente de um agente sistema localizado em uma outra agência genérica, que se comunica com as fábricas de agentes através do mecanismo de troca de mensagem nativo do aglets. Após este procedimento, as agências de QoS estão prontas para começarem a sessão, que se dá pelo recebimento de uma sinalização de início de execução de sessão vinda do agente sistema.

No restante deste capítulo serão apresentados e analizados alguns resultados de monitoramento e adaptação de QoS obtidos com esse protótipo da arquitetura de implementação. Os casos de teste foram realizados sobre aplicações de teleconferência. As estações de trabalho que foram utilizadas nos experimentos estão interligadas por redes locais ATM (LAN Emulation -LANE) e FDDI (Fig. 7.6). Suas características são apresentadas na Tab. 7.1. No caso, o nome do domínio de todas as máquinas é: dca.fee.unicamp.br

Nome da Máquina	**Tipo de *Hardware***	**Sistema Operacional**	**Placa de Rede**
botafogo	SUN - Ultra1	Solaris 2.6	ATM-LANE
enseada	SUN - Ultra1	Solaris 2.6	ATM-LANE
aracati	SUN - SPARC 5	Solaris 2.5.1	ATM-LANE
tambau	SUN - SPARC 5	Solaris 2.5.1	ATM-LANE
juquei	SUN - SPARC 5	Solaris 2.5.1	FDDI
itapua	SUN - SPARC 20	Solaris 2.5.1	FDDI

Tab. 7.1: Características das estações de trabalho utilizadas nos experimentos.

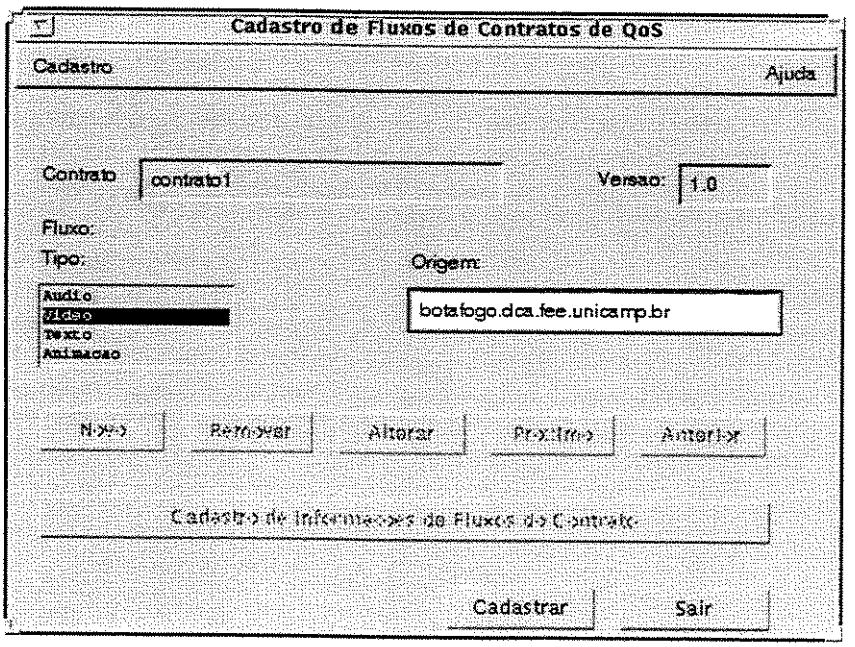

Fig. 7.4: Interface de manipulação de fluxos presentes na aplicação de cadastro de contratos.

7.1 Monitoramento de QoS

Nesta seção serão apresentados e analisados alguns resultados obtidos com monitoramento de qualidade de serviço em cenários de teleconferência. O mecanismo de adaptação de qualidade de serviço foi desabilitado, pois os objetivos nestes casos de estudo foram analisar o comportamento do mecanismo de monitoramento de QoS por agentes e avaliar seus impactos.

Nestes casos de estudo foram utilizados três cenários: dois utilizando transmissão de áudio e um utilizando transmissão de vídeo. Em todos os casos se utilizou um mecanismo de endereçamento *multicast*.

No primeiro cenário de transmissão de áudio foi enviado um fluxo de áudio de uma estação para outras duas e no outro se transmitiu um fluxo de áudio de uma para três estações. Em ambos os casos, os fluxos de áudio tinham as mesmas características (vide a Tab. 7.2). No

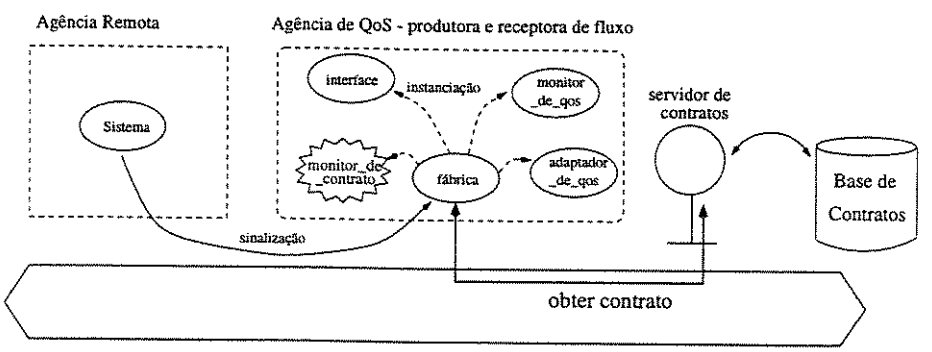

Fig. 7.5: Esquema de configuração de uma agência de QoS.

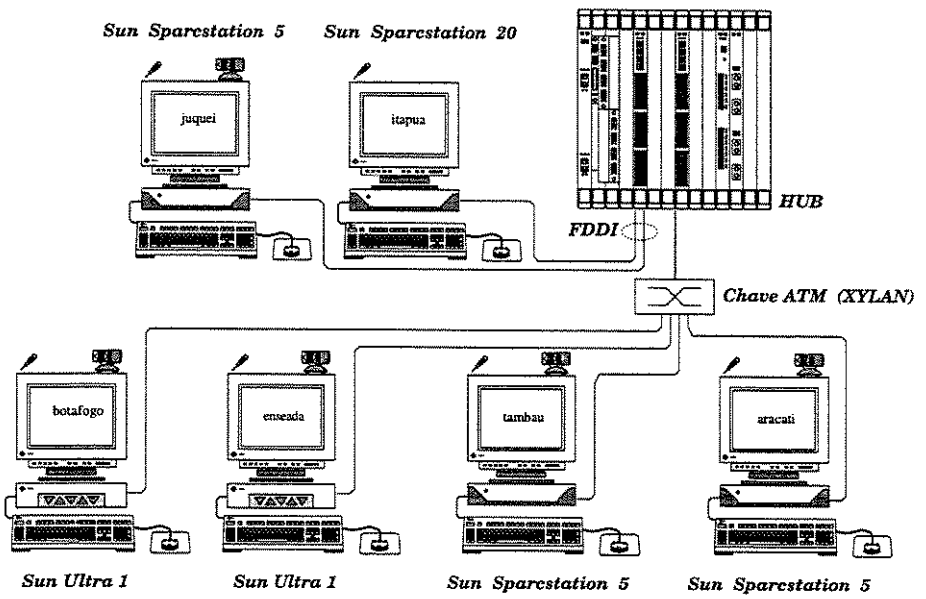

Fig. 7.6: Esquema de configuração da rede local.

cenário de transmissão de vídeo se transmitiu vídeo de uma estação para outras duas. As características do fluxo de vídeo são descritas na Tab. 7.3.

Bits por Amostra	Amostras por Segundo	Tipo de Codificação	Tipo de Compactação
8	8000	PCM	Nenhuma

Tab. 7.2: Característica do fluxo de áudio.

No primeiro cenário de transmissão de áudio, a produtora de fluxo foi a estação `juquei`, e as receptoras foram as estações `botafogo` e `aracati`. Os resultados de monitoramento obtidos neste cenário para as estações `aracati` e `botafogo` são mostrados nas Figs. 7.7 e 7.8, respectivamente. O segundo cenário de transmissão de áudio é similar ao primeiro acrescido de mais uma estação receptora de áudio, no caso foi a estação `itapua`. Os seu resultado estão exibidos nas Figs. 7.9, 7.10 e 7.11. Para o cenário de transmissão de vídeo se utilizou a estação `botafogo` como produtora de fluxo e as estações `enseada` e `aracati` como receptoras de fluxos, e os resultados obtidos são apresentados nas Figs. 7.12 e 7.13.

Nessas figuras, o tempo de migração corresponde ao intervalo de tempo em segundos transcorrido no transporte de um agente `monitor_de_contrato` entre dois nós consecutivos do seu itinerário, enquanto o tempo de resposta corresponde ao intervalo de tempo, dado em segundos, transcorrido entre dois recebimentos de valores de parâmetros de QoS com respeito a um destino específico de um fluxo.

As Tabs. 7.4 e 7.5 apresentam, respectivamente, estatísticas dos tempos de migração e de resposta para os dois cenários de monitoramento de áudio, enquanto a Tab. 7.6 se refere

Pixeis por Quadro	Bits por Pixel	Quadros por Segundo	Tipo de Codificação	Tipo de Compactação
320 X 240	8	30	NTSC	CellB

Tab. 7.3: Característica do fluxo de vídeo.

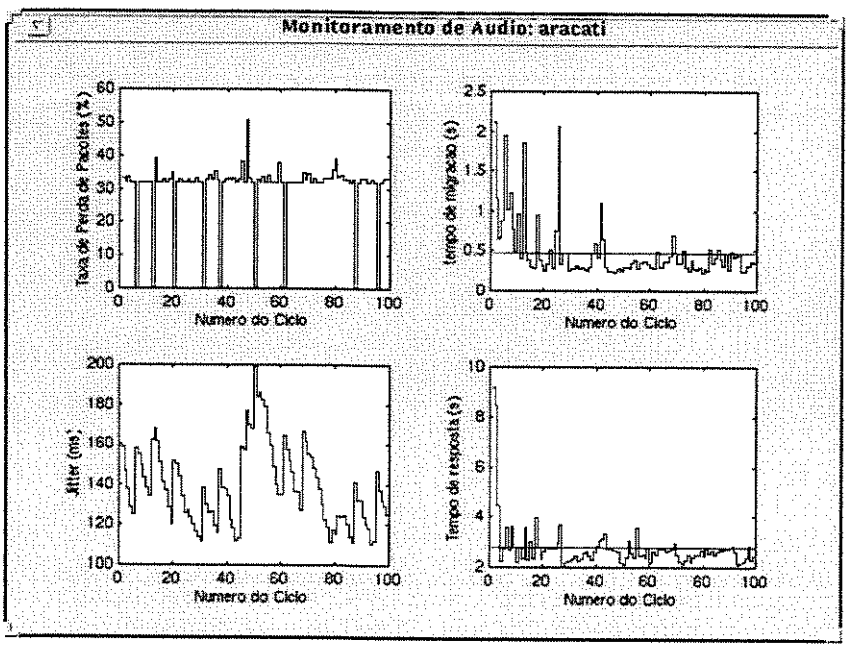

Fig. 7.7: Resultados de monitoramento da estação `aracati` no primeiro cenário de transmissão de áudio.

às estatísticas do cenário de monitoramento de vídeo. O tempo de migração está associado à estação de onde o agente `monitor_de_contrato` parte, por exemplo: o tempo de migração da estação `botafogo` na Tab. 7.4 corresponde ao tempo transcorrido na migração do agente `monitor_de_contrato` entre a estação `botafogo` e `aracati`.

Para os cenários de áudio, o tempo de migração médio ficou em torno de 0,5 segundos enquanto para o cenário de vídeo este tempo médio foi em torno de 0,9 segundos. Isto se deve ao fato de que no cenário de vídeo há bem mais tráfego de dados na rede e mais atividade de processamento que nos cenários de monitoramento de áudio. As variações nos tempos de migração dentro de um mesmo cenário possivelmente se devem mais ao fato das diferenças de poder de processamento entre as estações de trabalho utilizadas nos testes do que nas diferenças de rede. Este fato pode ser verificado pela equivalência dos tempos de migração da estação `botafogo` no primeiro cenário de monitoramento de áudio, Tab. 7.4, e da estação `itapua`, Tab. 7.5, no segundo cenário de monitoramento de áudio. As perdas de pacotes mais acentuadas nas estações com menos poder computacional contribuem com esta hipótese.

Comparando-se os resultados obtidos no primeiro cenário de monitoramento de áudio com

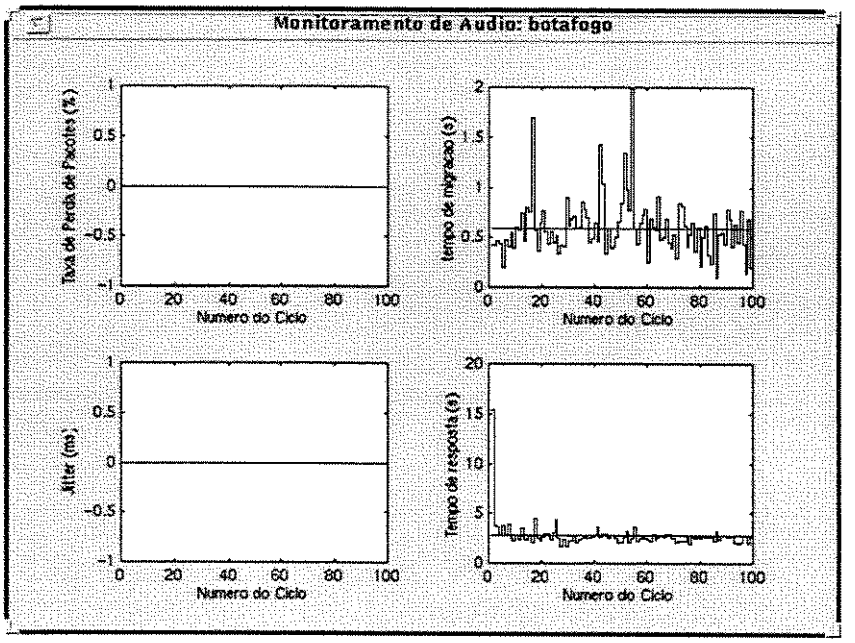

Fig. 7.8: Resultados de monitoramento da estação `botafogo` no primeiro cenário de transmissão de áudio.

relação aos obtidos no cenário de vídeo se observa que houve um aumento maior no tempo médio de resposta que o aumento ocorrido no tempo médio de migração, indicando que o tempo médio de permanência do agente `monitor_de_contrato` em cada agência para o caso de monitoramente de vídeo foi maior que para o caso de áudio. Isto é decorrência do fato de que recepção e exibição de vídeo requerem mais esforço computacional que os equivalentes de áudio, o que leva o sistema operacional a destinar menos tempo de processamento ao processo que contém a agência.

7.2 Adaptação de QoS

Nesta seção serão apresentados dois cenários de adaptação de QoS em transmissão de vídeo. Em ambos os cenários se transmitiu vídeo em modo *multicast* de uma estação para outras duas, com as mesmas características utilizados para o cenário de monitoramento de vídeo, Tab. 7.3, apresentado na seção anterior. No caso, a estação produtora de fluxo foi a `tambau` e as receptoras foram as estações `botafogo` e `enseada`.

Nesses dois cenários, o agente `monitor_de_contrato` ao detectar uma violação de contrato informa a ocorrência ao agente `adaptador_de_qos` da agência fonte de fluxo. Por sua vez, esse agente adapta as características do fluxo de vídeo, que no caso corresponde a diminuir ou aumentar a freqüência de transmissão do fluxo de vídeo. Por limitações do sistema de vídeo, a sua freqüência de captura de vídeo é de 30 quadros por segundo, de modo que as outras freqüência são obtidas a partir de um mecanismo de descarte de quadros baseado naquela freqüência. Assim, só se pode obter freqüências que sejam resultado da divisão de 30 por um número inteiro, como por exemplo: 30; 15; 10; 7,5; 6; 5; 3; 2; 1 e 0,5 quadros por

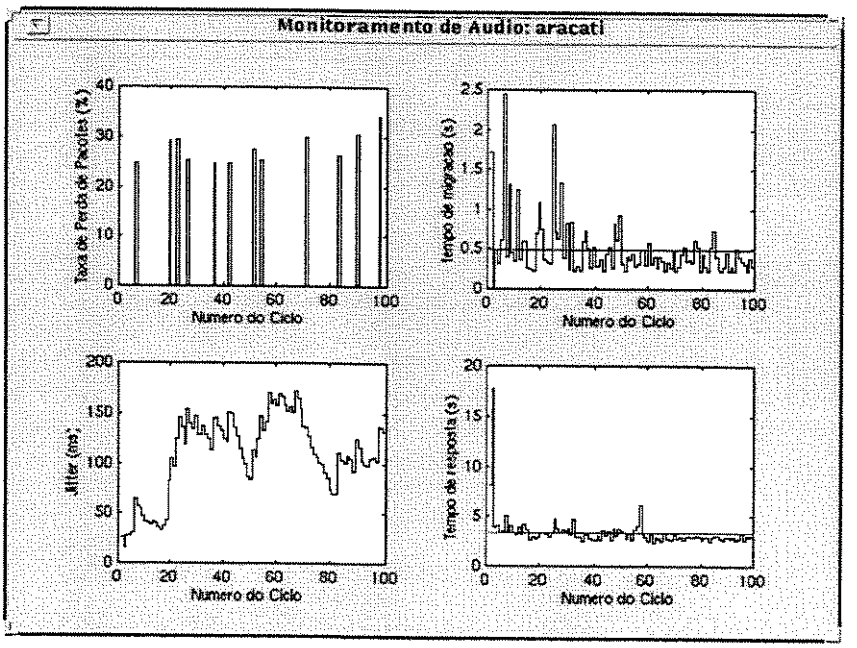

Fig. 7.9: Resultados de monitoramento da estação `aracati` no segundo cenário de transmissão de áudio.

segundo. O valor da freqüência de amostragem inicial foi de 15 quadros por segundo para os dois cenários.

Em ambos os cenários de adaptação de QoS, o agente `monitor_de_contrato` sinaliza ao seu correspondente `adaptador_de_qos` sob duas circunstâncias: violação inferior de contrato e violação superior de contrato. O primeiro caso corresponde à situação em que os níveis de qualidade monitorados estão abaixo dos mínimos especificados no contrato, enquanto o segundo caso corresponde à situação em que os níveis monitorados de QoS estão acima dos níveis máximos especificados. A primeira situação levará o agente `adaptador_de_qos` a aumentar, se for possível, a freqüência de amostragem dos quadros, enquanto o segundo caso levará esse agente a tomar uma atitude oposta àquela, ou seja, diminuir a freqüência de amostragem. Nos dois cenários, o agente `adaptador_de_qos` utiliza a mesma estratégia de adaptação: conforme o tipo de sinalização, aumenta ou diminui a freqüência de amostragem do vídeo para o valor factível mais próximo.

No primeiro cenário de adaptação, Figs. 7.14 e 7.15, o agente `monitor_de_contrato` foi programado para detectar violação inferior de contrato quando ocorrer em alguma das agências perdas de pacotes, em três amostragens sucessivas de monitoramento, menor que o nível mínimo de perda de pacotes estabelecido no contrato. Já a violação superior de contrato ocorre quando há a detecção por três vezes sucessivas de perda de pacotes acima do nível máximo estabelecido em alguma das agências monitoradas. Duas sinalizações sucessivas de violação de contrato só podem ocorrer num intervalo mínimo de três ciclo de monitoramento. O nível mínimo de perda de pacotes estabelecido para este contrato é de 5% e o nível máximo é de 15%. Neste caso, pode-se observar a partir da Fig. 7.15 que basta haver violação de contrato em uma agência para que seja acionado o mecanismos de adaptação de QoS. Para os gráficos de violação de contrato na Fig. 7.15, o valor 1 corresponde a uma violação do limite

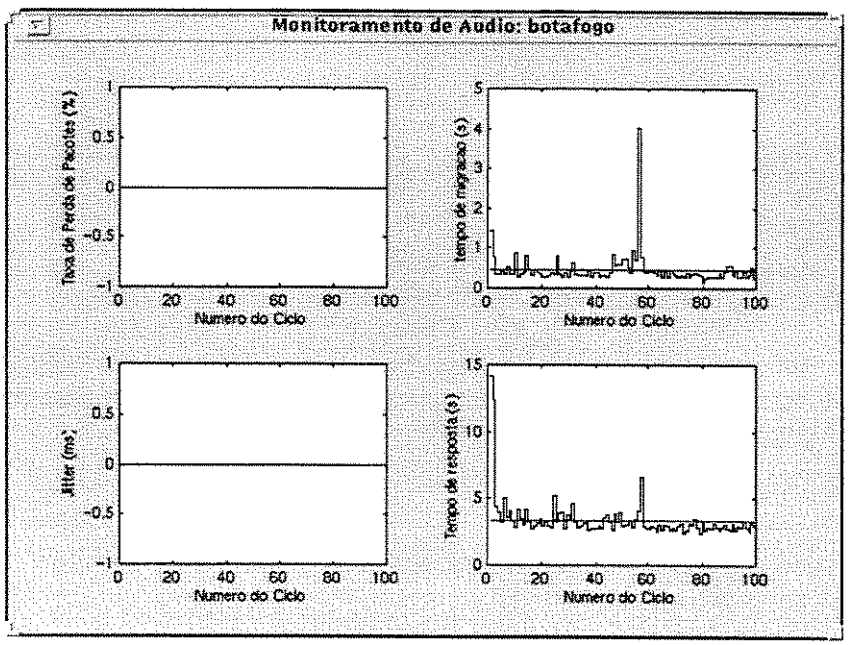

Fig. 7.10: Resultados de monitoramento da estação `botafogo` no segundo cenário de transmissão de áudio.

superior do contrato, enquanto o valor -1 corresponde a uma violação do limite inferior.

No segundo cenário de adaptação, Figs. 7.16 e 7.17, o procedimento de detecção de violação de contrato é similar ao caso anterior, porém aqui, diferentemente do primeiro caso, só há a sinalização da violação se houver simultaneamente o mesmo tipo de violação em todas as agências monitoradas. Isto pode ser observado na Fig. 7.17, onde há a detecção de violação de contrato por algumas vezes nas agências monitoradas, mas não foi acionado o mecanismo de adaptação de QoS porque as violações de contrato não ocorreram simultaneamente.

Assim, no primeiro cenário a estratégia de adaptação utilizada previlegia a estação com menos disponibilidade computacional, enquanto no segundo cenário a estratégia de adaptação visa não penalizar a estação com mais disponibilidade computacional.

7.3 Análise dos Resultados

O protótipo implementado apresentou comportamento robusto e realizou as tarefas de monitoramento e adaptação de QoS conforme programado. A integração entre as diversas infra-estruturas utilizadas se mostrou consistente e com alto nível de transparência.

Agentes são entidades autocontidas, com interfaces bem definidas e com a comunicação se dando através de troca de mensagens. Estas características inerentes ao paradigma de sistemas baseados em agentes levam a propriedades desejáveis na construção de qualquer sistema de *software*, tais como: divisão em módulos, flexibilidade e encapsulação. Esta afirmação pôde ser comprovada durante o processo de implementação dos protótipos dos agentes, que se deu de forma bastante intuitiva e natural. Isto possibilita que evoluções no protótipo possam ser feitas com impactos reduzidos no sistemas como um todo.

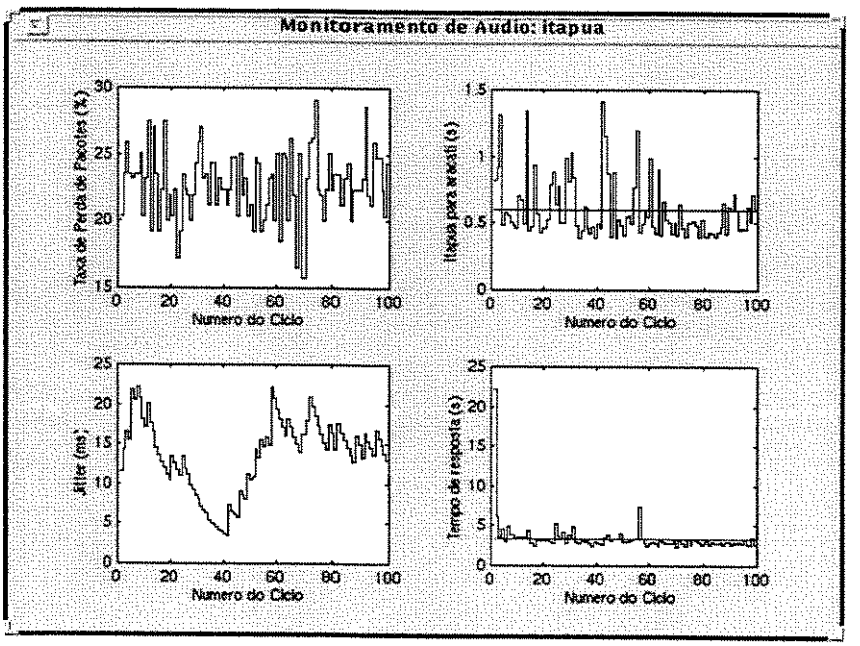

Fig. 7.11: Resultados de monitoramento da estação `itapua` no segundo cenário de transmissão de áudio.

Para efeito de comparação, o tempo médio de acesso a um servidor CORBA via OrbixWeb, em circunstâncias equivalentes ao do primeiro cenário de transmissão de áudio, ficou em torno de 0,02s [51], o que corresponde a 25 vezes menos que o tempo de migração médio do agente `monitor_de_contrato`. Embora os tempo de migração e de resposta para os casos de teste sejam bem elevados, quando comparados com o tempo de resposta de uma requisição via CORBA, a utilização da abordagem baseada em agentes não se torna inviável, uma vez que os tempo de monitoramento obtidos com a estratégia de migração são compatíveis com o dinâmica das aplicações cooperativas.

Outro fator é que a infra-estrutura Aglets, utilizada aqui como suporte a agente, está ainda em fase de amadurecimento e encontra-se ainda em versão *alfa* teste. A própria IBM reconheceu que o seu protocolo de migração e comunicação de agentes, o ATP, ainda é bastante ineficiente. Além disto, na construção do protótipo da Agência de QoS não se teve a preocupação de otimização de código e sim a implementação das funcionalidades propostas no modelo, de modo a analisar sua viabilidade.

Em termos de modelagem, a estratégia de se utilizar um agente móvel para realizar todo o monitoramento de um fluxo do contrato, no caso o agente `monitor_de_contrato`, se mostrou com algumas deficiências. Uma delas é que o tempo de resposta entre dois monitoramentos sucessivos em uma determinada agência é dependente da carga do sistema (estações e rede) e do número de agências em seu itinerário. No caso, o tempo de resposta cresce linearmente com o número de agências, de modo que itinerários muito grandes podem ser inviáveis nesse esquema de monitoramento.

Porém, como o modelo de agentes é bastante flexível, pode-se alterar com relativa facilidade a estratégia de migração dos agentes `monitor_de_contrato`. Uma possível estratégia de monitoramento seria dividir o itinerário em vários segmentos e designar um

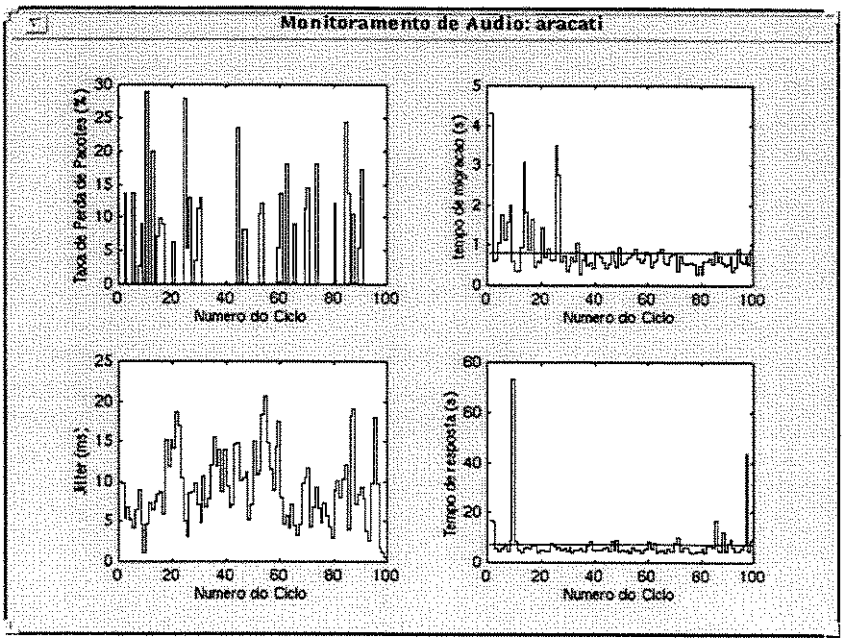

Fig. 7.12: Resultados de monitoramento da estação `aracati` no cenário de transmissão de vídeo.

agente `monitor_de_contrato` para cada segmento.

No caso, esses agentes teriam uma visão parcial do comportamento do sistema, de modo que cabe ao agente `adaptador_de_qos` associado a responsabilidade de compor essas informações. No caso extremo poder-se-ia ter um agente `monitor_de_contrato` por cada destino de fluxo, o que eliminaria a necessidade de migração, diminuindo o tráfego na rede porém aumentando a responsabilidade do agente `adaptador_de_qos`. A estratégia de adaptação de QoS utilizada no primeiro cenário de adaptação de vídeo de certa maneira equivale à abordagem de se ter um agente `monitor_de_contrato` por cada destino, uma vez que ela não leva em consideração o comportamento global do sistema para a geração da sinalização de violação de contrato.

De forma geral, a implementação do protótipo permitiu constatar a adequação do modelo baseado em agentes para gerência de qualidade de serviço de aplicações multimídia distribuídas. E como o modelo baseado em agentes é bastante flexível, pode-se utilizá-lo com bastante eficiência como um ambiente de teste de estratégias de monitoramento e adaptação de QoS.

Além disto, acreditamos que as vantagens provenientes da utilização do paradigma de agentes sejam mais salientes ainda na etapa de negociação de contrato, onde características como autonomia e delegação de responsabilidades são mais prementes que nas atividades de gerência de qualidade de serviço.

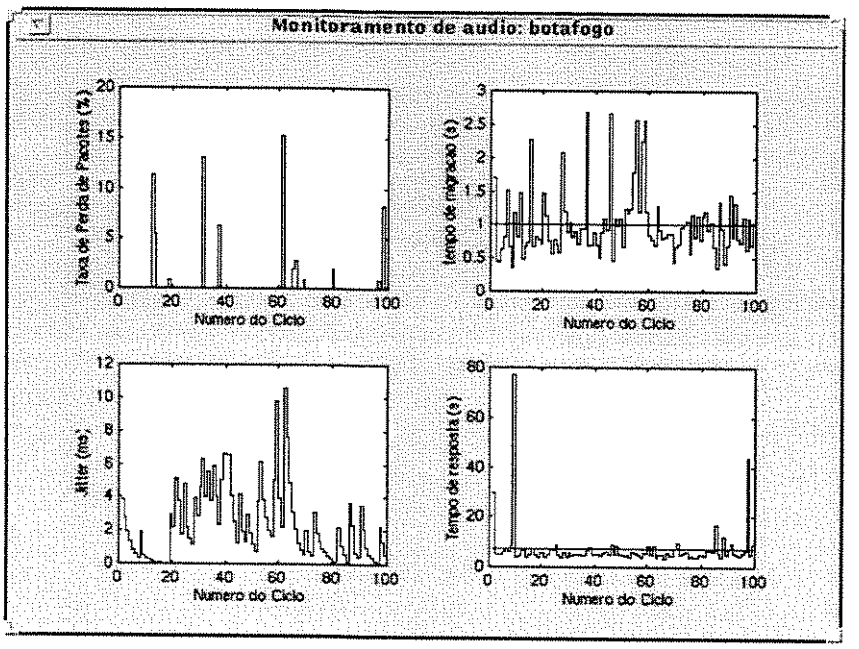

Fig. 7.13: Resultados de monitoramento da estação enseada no cenário de transmissão de vídeo.

estação consumidora	aracati	botafogo
tempo médio de migração (s)	0,48	0,59
variância do tempo de migração (s^2)	0,14	0,08
tempo médio de resposta (s)	2,76	2,77
variância do tempo de resposta (s^2)	0,92	1,85

Tab. 7.4: Estatísticas dos tempos de migração e de resposta para o primeiro cenário de monitoramento de áudio.

estação consumidora	aracati	botafogo	itapuã
tempo médio de migração (s)	0,48	0,45	0,60
variância do tempo de migração (s^2)	0,14	0,16	0,05
tempo médio de resposta(s)	3,39	3,39	3,40
variância do tempo de resposta (s^2)	2,70	2,44	4,12

Tab. 7.5: Estatísticas dos tempos de migração e de resposta para o segundo cenário de monitoramento de áudio.

estação consumidora	aracati	botafogo
tempo médio de migração (s)	0,81	1,00
variância do tempo de migração (s^2)	0,39	0,24
tempo médio de resposta (s)	7,10	7,11
variância do tempo de resposta(s^2)	64,65	73,25

Tab. 7.6: Estatísticas dos tempos de migração e de resposta para o cenário de monitoramento de vídeo.

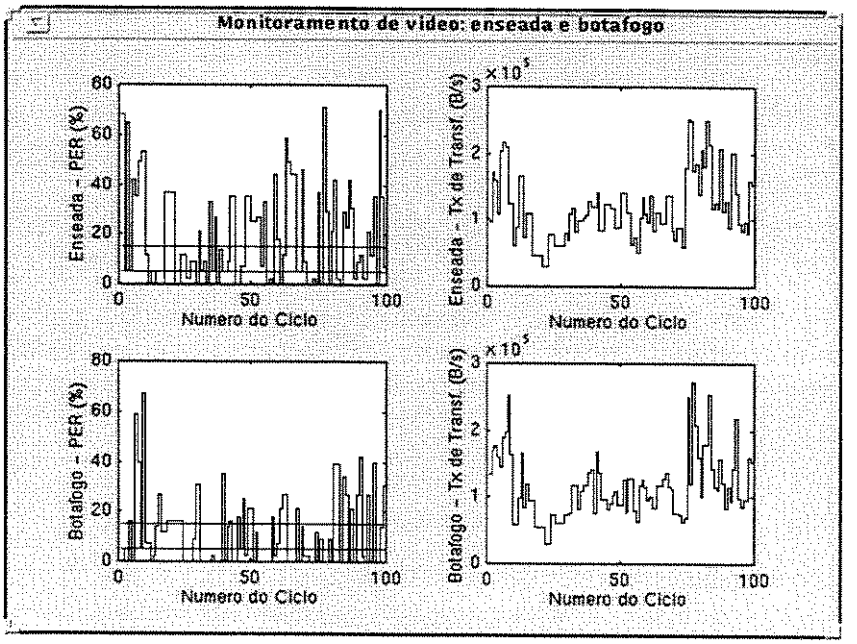

Fig. 7.14: Resultados de monitoramento de QoS das estações `enseada` e `botafogo` para o primeiro cenário de adaptação de QoS de vídeo.

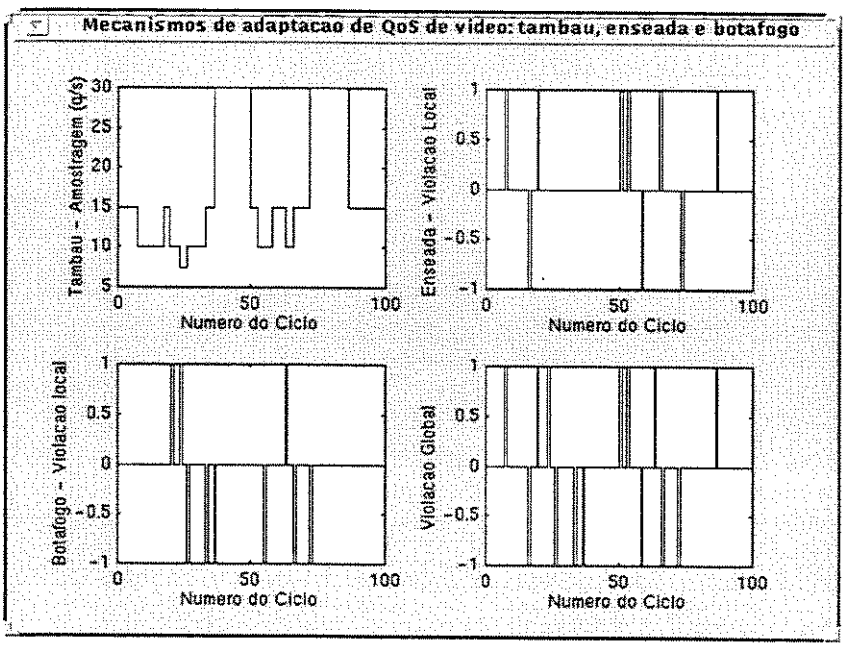

Fig. 7.15: Resultados dos mecanismos de violação e adaptação de QoS para o primeiro cenário de adaptação de QoS de vídeo.

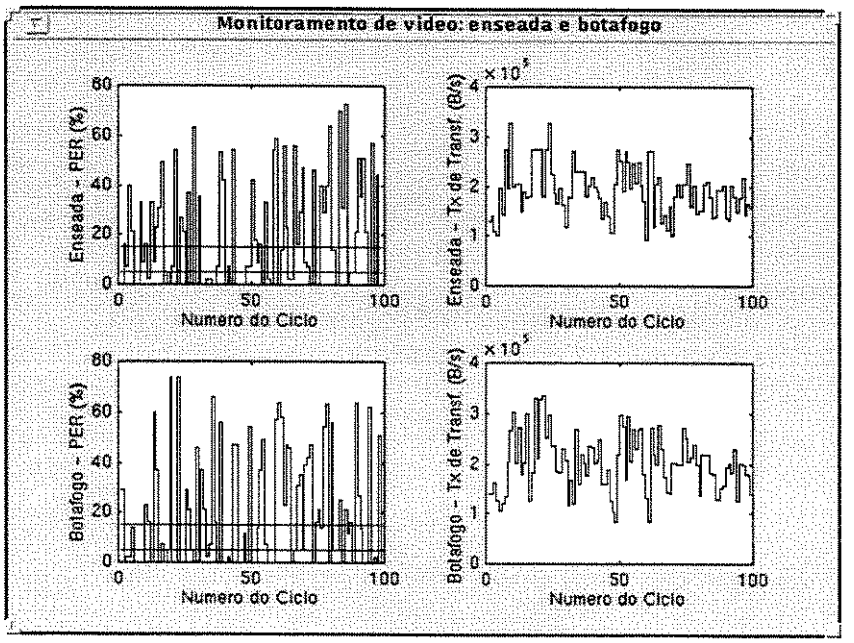

Fig. 7.16: Resultados de monitoramento de QoS da estações `enseada` e `botafogo` para o segundo cenário de adaptação de QoS de vídeo.

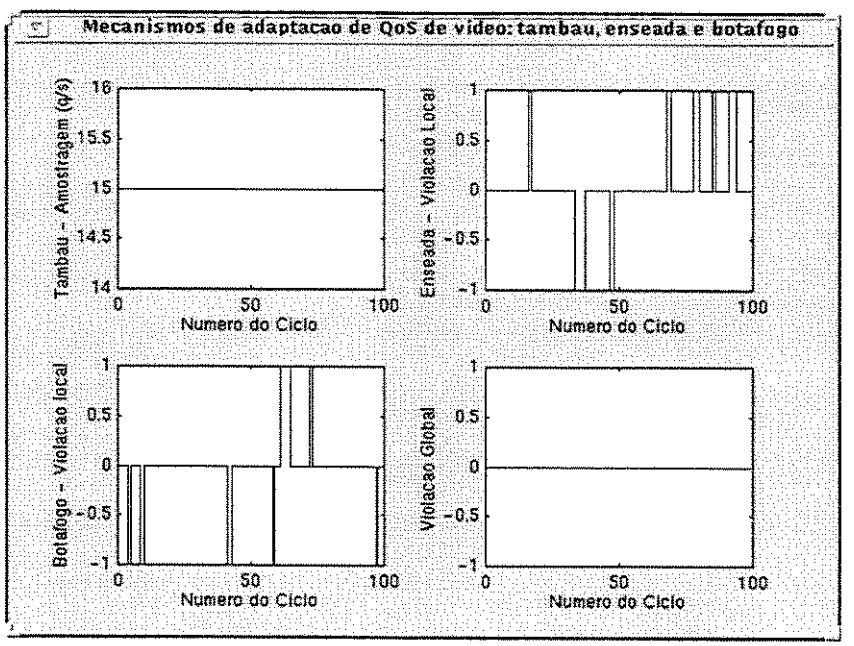

Fig. 7.17: Resultados dos mecanismos de violação e adaptação de QoS para o segundo cenário de adaptação de QoS de vídeo

Capítulo 8

Conclusões e Trabalhos Futuros

Como caracterizado ao longo deste trabalho, qualidade de serviço (QoS) é uma característica inerente às aplicações multimídia. QoS intuitivamente expressa quão adequados são os serviços fornecidos por uma determinada aplicação. Via de regra, o suporte de QoS é obtido através de um processo de negociação e gerência entre usuários e provedores de serviços. Esse processo envolve tipicamente atividades como seleção e configuração de formatos e padrões de dados, além de reserva e gerência de recursos de modo a se atender aos níveis de qualidade esperados. Esses níveis de QoS podem ser descritos por uma combinação de parâmetros subjetivos, como áudio em qualidade de telefone, e parâmetros objetivos, como atrasos e perda de pacotes de dados.

Devido à natureza das aplicações multimídia, estas necessitam muitas vezes ser executadas em ambientes distribuídos e heterogêneos, com diferentes capacidades de reserva de recursos e seleção de serviços, o que torna o processo de suporte de QoS uma atividade complexa.

Aurrecoechea e outros em [2] propõem cinco princípios funcionais que devem nortear a construção de qualquer sistema para suporte de QoS de aplicações multimídia distribuídas. Nominalmente, esses princípios são: integração, separação, transparência, gerência síncrona de recursos e desempenho.

Devido às características já mencionadas das aplicações multimídia, acreditamos que esses cinco princípios funcionais só podem ser obtidos adequadamente caso se previlegie fortemente na construção do sistema características como encapsulação, modularidade e flexibilidade.

A arquitetura proposta nesta tese utiliza o paradigma de agentes, que privilegia encapsulação, modularidade e flexibilidade. Defende-se aqui que os cinco princípios propostos em [2] são satisfeitos por esta arquitetura:

- O princípio da integração diz que que a qualidade de serviço deve ser passível de configuração, predição e gerência. No nosso modelo, a responsabilidade pela configuração do sistema é do agente `negociador_local`; a responsabilidade de predição do agente `estimador_de_recursos` e a responsabilidade de gerência é do agente `gerente_de_recursos`.

- Pelo princípio da separação, as atividades de gerência e controle dos fluxos devem ser realizadas por entidades distintas. No nosso caso, a gerência do fluxo é de responsabilidade dos agentes `monitor_de_qos` e `monitor_de_contrato`, enquanto a de controle é

de responsabilidade do agente `adaptador_de_qos`.

- O princípio da transparência estabelece que a aplicação não deve se envolver com detalhes das atividade de estabelecimento e gerência de QoS, tais como mapeamento, negociação e monitoramento de QoS. Este princípio é plenamente satisfeito em nosso modelo, pois o usuário da aplicação interage com o sistema apenas através do agente `interface`, havendo agentes específicos responsáveis por diversas atividades necessárias ao suporte de QoS, como os agentes `mapeador_de_qos`, `negociador_de_contrato` e `monitor_de_qos`.

- O princípio da gerência assíncrona de recursos objetiva a divisão das atividades em módulos para fins de controle e gerência de QoS. Devido às características do paradigma de sistemas baseados em agentes, esse princípio é alcancado de forma natural em nosso modelo.

Já o princípio de desempenho, que prescreve que o desenvolvimento de uma arquitetura de suporte de QoS deve incorporar técnicas que privilegiem o desempenho das suas aplicações, foi atendido apenas de forma razoável em nosso modelo, pois o procedimento de migração de código acarreta um aumento tanto de tráfego de rede quanto de processamento para despachar e receber os agentes móveis. Estes efeitos são mais onerosos para o caso de monitoramento de QoS que de negociação de contrato, pois, diferentemente do processo de negociação de contrato, o processo de monitoramento deve ocorrer em intervalos de tempo compatíveis com a dinâmica da aplicação, sob ao risco de não se capturar apropriadamente tal dinâmica. Por outro lado, os recursos de rede e processamento necessários à migração de agentes são relativamente modestos quando comparados àqueles necessários para transmissão e processamento de vídeo.

Porém, como argumentado no capítulo anterior, em decorrência do modelo de agentes ser bastante flexível, pode-se alterar com relativa facilidade a estratégia de migração dos agentes `monitor_de_contrato`. Uma possível estratégia de monitoramento seria dividir o itinerário em vários segmentos e designar um agente `monitor_de_contrato` para cada segmento. No caso extremo, poder-se-ia ter um agente `monitor_de_contrato` por cada destino de fluxo, o que eliminaria a necessidade de migração, diminuindo o tráfego na rede. Neste caso, a responsabilidade de compor as diversas informações locais sobre monitoramento QoS deverá ser transferida para o agente `adaptador_de_qos`.

No geral, a abordagem baseada em agentes se mostrou na prática um paradigma bastante adequado para construção de sistemas distribuídos, onde sistemas de monitoramento e gerência de QoS são casos particulares. Além disto, acreditamos que seus benefícios podem ser mais proeminentes na etapa de negociação de contrato, onde, além de não ter uma restrição temporal forte, há a necessidade de se utilizar modelos que previlegiem características como autonomia e delegação de responsabilidades.

8.1 Contribuições da Tese

Podemos citar como contribuições deste trabalho os seguintes pontos:

- Caracterização dos sistemas multimídia quanto aos seus requisitos de qualidade de serviço, analizando-se aspectos de configuração e seleção de serviços, além das atividades necessárias à gerência da qualidade de serviço.

- Proposição de um modelo baseado em sistemas de agentes para suporte à qualidade de serviço em aplicações multimídia.

- A especificação via as linguagens formais MSC e SDL do protocolo de negociação e gerência de QoS de uma sessão multimídia.

- A proposição de uma arquitetura modular para implementação de aplicações multimídia distribuídas, onde o serviço de suporte à qualidade de serviço baseados em sistemas de agentes é um de seus módulos.

- O desenvolvimento de um protótipo do modelo proposto com o objetivo de avaliar sua adequação em cenários de monitoramento e adaptação de QoS de aplicações de teleconferência, o que levou à integração de diversas tecnologias, como: CORBA; servidores de áudio e vídeo; transmissão de fluxo contínuo e ambiente para suporte a agentes.

- Como um subproduto deste trabalho, obteve-se uma metodologia de desenvolvimento de sistemas baseados em agentes em geral, que basicamente consiste das seguintes etapas:

  - caracterização dos requisitos da aplicacão;

  - determinação dos agentes que compõem a solução do problema, com a designação de suas respectivas responsabilidades;

  - especificação do protocolo de troca de mensagens entre os agentes, utilizando-se por exemplo uma linguaguem do tipo MSC;

  - especificação da dinâmica do agente via máquina de estados estendida, como um processo em SDL, por exemplo;

  - escolha de um ambiente de suporte a agentes e

  - implementação da especificação.

8.2 Trabalhos Futuros

Como extensões deste trabalhos podemos citar os seguintes pontos:

- Implementação dos agentes relativos à fase de estabelecimento de sessão, ou seja: mapeador_de_qos, `negociador_local`, estimador_de_recursos, gerente_de_recursos, negociador_de_contrato e renegociador_de_contrato. Isto possibilitaria o estudo do comportamento do sistema com um núnero elevado de agentes e num escopo de aplicação mais amplo.

- De posse desse protótipo de Agência de QoS, ter-se-ia um ambiente extremamente flexível para testar várias estratégias de negociação de contrato, estimação de recursos

e mapeamento e adaptação de qualidade de serviço. No atual estágio de desenvolvimento do protótipo é possível se testar com relativa facilidade várias estratégias de monitoramento e adaptação de QoS.

- Estudar a vialibilidade de se incorporar características de inteligência nos agentes, de modo a se avaliar os impactos de se conciliar representação e manipulação de conhecimento com requisitos de mobilidade. Pela análise das responsabilidades designadas a alguns agentes, acredita-se que a incorporação de técnicas de inteligência artificial, como redes neurais e sistemas nebulosos, poderia ser bastante apropriada. Exemplos dessas atividades: negociação de contrato, estimação de recursos e mapeamento e adaptação de qualidade de serviço.

- A expansão do modelo proposto com acréscimo de outras funcionalidades que não foram alvo deste trabalho, como sincronização entre fluxos, aspectos de segurança e mecanismos de tarifação de serviços. No caso, estas novas funcionalidades seriam providas por novos agentes, o que permitiria avaliar a capacidade do sistema para incorporar novos componentes a baixo custo de implementação, conferindo ao sistema uma característica incremental desejável.

Referências Bibliográficas

[1] A. Alles. ATM Internetworking. Technical report, Cisco Systems, Inc., URL: http://www.cisco.com, May 1995.

[2] C. Aurrecoechea, A. Campbell, and L. Hauw. Survey of Quality of Service Architectures. *Multimedia Systems Journal - Special Issue on Multimedia Information Systems*, 1997.

[3] F. Belina, D. Hogrefe, and A. Sarma. *SDL with Applications from Protocol Specification.* Prentice Hall International Ltd, 1991.

[4] R. Ben-Natan. *CORBA: A Guide to Common Object Request Broker Architecture.* McGraw Hill, 1995.

[5] G. Blair, A. Campbell, G. Coulson, F. Garcia, D. Hutchison, A. Scott, and D. Shepherd. Orchestration Services for Continuous Media Communications. In *IV Bangor Communication Symposium*, Bangor, Gwynedd, May 1992.

[6] G. Blakowski and R. Steinmetz. A Media Synchronization Survey: Reference Model, Specification, and Case Studies. *IEEE Journal on Selected Areas in Communications*, 14(01):5–35, January 1996.

[7] A. Campbell, G. Coulson, and D. Hutchison. A Quality of Service Architecture. *ACM Computer Communication Review*, April 1994.

[8] D. Chess, B. Grosof, C. Harrison, D. Levine, C. Paris, and G. Tsudik. Intinerant Agents for Mobile Computing. Technical report, IBM Watson Research Center, 1995.

[9] B. Cole. Interactive Multimedia: The Technology Framework. *IEEE Spectrum*, 30(3):32–39, March 1993.

[10] A. Danthine, Y. Baguette, G. Leduc, and L. Leonard. The OSI 95 Connection-Mode Transport Service-Enhanced QoS. In *Proceedings of 4th IFIP Conference on High Performance Networking*, Belgium, December 1992.

[11] M. de Prycker, R. Pesci, and T. van Landegem. B-ISDN and the OSI Protocol Reference Model. *IEEE Network*, 7(2):10–18, March 1993.

[12] F. Dupuy, G. Nilsson, and Y. Inoue. Tina Consortium: Toward Networking Telecommunications Information Services. *IEEE Communications Magazine*, 33(11):78–83, November 1995.

[13] L.F. Faina, E. J. Oliveira, R. C. M. Prado, and E. Cardozo. Developing Multimedia Applications with the OMG Streaming Framework. In *Accepted for presentation at Multimedia Applications, Services and Technologies*, Vancouver, Canada, June 1999.

[14] D. Ferrari. The Tenet Experience and the Design of Protocols for Integrated Services Internetworks. *Multimedia Systems Journal*, November 1995.

[15] D. Ferrari, J. Ramaekers, and G. Ventre. Client-Network Interactions in Quality of Service Communication Environments. In *Proceedings of 4th IFIP Conference on High Performance Networking*, Belium, December 1992.

[16] GMD Fokus, IBM, Crystaliz, General Magic, and The Ope Group. Mobile Agent System Interoperability Facilities Specification. Technical report, Object Management Group - OMG, October 1997.

[17] XTP Forum. Xpress Transport Protocol Specification. Technical Report Revision 4.0, XTP Forum, 1994.

[18] S. Franklin and A. Graesser. Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. In Springer-Verlag, editor, *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, URL.: http://www.msci.memphis.edu/AgentProg.html, 1996.

[19] B. Furht. Multimedia Systems: An Overview. *IEEE Multimedia*, pages 47–59, Spring 1994.

[20] D. Gall. MPEG: A Video Compression Standard for Multimedia Applications. *Communications of the ACM*, 34(4):45–58, April 1991.

[21] J. Germmel, H. Vin, D. Kandlur, and P. Rangan. Multimedia Storage Servers: A Tutorial. *IEEE Computer*, 28(5):40–49, May 1995.

[22] S.J. Gibbs and D.C. Tsichritzis. *Multimedia Programming: Objets, Environments and Frameworks.* Addison-Wesley, 1995.

[23] L. A. Guedes and E. Cardozo. Especificação de um Protocolo para Negociação de Qualidade de Serviço em Sistemas Multimídia. In *Anais do 150 Simpósio Brasileiro de Redes de Computadores, SBRC 97*, pages 101–117, São Carlos, Brasil, Maio 1997.

[24] L. A. Guedes, P. C. Oliveira, and E. Cardozo. An Agent-Based Approach for Quality of Service Negotiation and Management in Distributed Mmultimedia Systems. In *Proceedings of First International Workshop in Mobile Agents, MA97, Lecture Notes in Computer Science*, volume 1219, pages 1–12, Berlin, Germany, April 1997.

[25] L. A. Guedes, P. C. Oliveira, L. F. Faina, and E. Cardozo. An Agent-based Approach for Supporting Quality of Service in Distributed Multimedia Systems. *Computer Communication Journal*, 21:1269–1278, Setember 1998.

[26] L.A. Guedes, P.C. Oliveira, L.F. Faina, and E. Cardozo. QoS Agency: An Agent-based Architecture for Supporting Quality of Service in Distributed Multimedia Systems. In *IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking (PROMSMmNet'97)*, pages 204–212, Santiago, Chile, November 1997.

[27] A. Hafid and G. von Bochmann. An Approach to Quality of Service Management for Distributed Multimedia Applications. In *ICODP95*, pages 319–340, Australia, February 1995.

[28] C. Harrison, D. Chess, and A. Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM Watson Research Center, 1995.

[29] B. Hayes-Roth. An Architecture for Adaptive Intelligent Systems. *Artificial Intelligence: Special Issue on Agents and Interactivity*, pages 329–365, 1995.

[30] D. Hehmann, R. Herrtwich, W. Schulz, T. Schuett, and R. Steinmetz. Implementing HeiTS: Architecture and Implementation Strategy of the Heidelberg High Speed Transport System. In *Second International Workshop on Network and Operating System Support for Digital Audio and Video*, Germany, 1991.

[31] C. Hewitt. Viewing Control Structures as Patterns of Passing Messages. *Artificial Intelligence*, 8(3):323–364, 1977.

[32] CCITT Rec. I.350. General Aspects os Quality of Service and Network Performance in Digital Networks. Technical report, International Telecommunication Union, Geneva, 1989.

[33] Inc. IBM. IBM Aglets Software Development Kit - Home Page. URL: http://www.trl.ibm.co.jp/aglets/index.html, September 1998.

[34] International Telecommunication Union Home Page. URL.: http://www.itu.ch, accessed on November 1996.

[35] Plc IONA Technologies and AG Lucent Technologies, Inc Siemens-Nixdorf. *Control and Management of Audio/Video Streams* OMG RFP Submission, Revised Submission. Technical report, Object Management Group - OMG, October 1997.

[36] Iona Technologies, Inc. *OrbixMT Programmers Guide, Version 2.3c*, 1997.

[37] Iona Technologies, Inc. *OrbixTrader Programmers Guide, Version 1.0*, 1997.

[38] Iona Technologies, Inc. *OrbixWeb Programmers Guide, Version 3.0*, 1997.

[39] ISO-QoS. Quality of Service Basic Framework - Outline. Technical Report ISO/IEC JTC1/SC21/WG1 N1145, International Standards Organization, UK, 1994.

[40] R. Koenen. MPEG-4: Multimedia for our time. *IEEE SPECTRUM*, 36(2):26–33, February 1999.

[41] K. Kotay and D. Kotz. Transportable Agents. Technical report, Department of Computer Science, Dartmourth College, 1994.

[42] A. Lazar. Real-time Control, Management, and Information Transport Architecture for Broadband Networks. In *Proceedings of the International Zurich Seminar on Digital Communications*, 1992.

[43] M. Liou. Overview of the px64 Kbit/s Video Coding Standard. *Communications of the ACM*, 34(4):59–63, April 1991.

[44] S. Mauw and M.A. Reniers. An Algebraic Semantics of Basic Message Sequence Charts. *The Computer Journal*, 37(4):269–277, 1994.

[45] K. Nahrstedt and J. Smith. The QoS Broker. *IEEE Multimedia*, 2(1):53–67, Spring 1995.

[46] K. Nahrstedt and R. Steinmetz. Resource Management in Networked Multimedia Systems. *IEEE Computer*, 28(28):52–63, May 1995.

[47] G. Nilison, F. Dupuy, and Chapmann. An Overview of the Telecommunications Information Networking Architecture. In *Proceedings of TINA95*, Melbourne, Australia, 1995.

[48] H. Nwana. Software Agents: An Overview. *Knowledge and Engineering Review*, 11(3), November 1996.

[49] P. C. Oliveira. Sistemas Baseados em Agentes Móveis: uma Abordagem Alternatina para o Desenvolvimento de Sistemas Distribuídos. Master's thesis, Faculdade de Engenharia Elétrica e de Computação, Unicamp, Campinas, Brasil, 1997.

[50] P. C. Oliveira and E. Cardozo. Mobile Agent-Based Systems: An Alternative Paradigm for Distributed Systems Development. In *Anais do XV Simpósio Brasileiro de Redes de Computadores, SBRC 97*, pages 508–524, São Carlos, Brasil, Maio 1997.

[51] P. C. Oliveira, L. A. Guedes, and E. Cardozo. Monitoramento de Qualidade de Serviço em Sistemas Multimídia Distribuídos: um Estudo de Caso para Sistemas Baseados em Agentes Móveis. In *Anais do XVI Simpósio Brasileiro de Redes de Computadores, SBRC 98*, pages 481–500, Rio de Janeiro, Brasil, Maio 1998.

[52] R. C. M. Prado, L. A. Guedes, L. F. Faina, and E. Cardozo. Implementação de Serviços Multimídia em CORBA. In *Anais da Conferência Latino-Americana de Informática-Clei97*, Val Paraíso, Chile, Novembro 1997.

[53] H. Rajapakshe and D. Quek. Video on Demand. *SURPRISE Journal*, June 1995.

[54] F. Ross. An Overview of FDDI. *IEEE Journal of Selected Areas in Communications*, 7(7), September 1989.

[55] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, 1995.

[56] H. Schulzrinne, S. L. Casner, R. Frederick, and V. Jacobson. RTP: A Transport for Real Time Applications. IETF Audio-Video Transport Working Group.

[57] D. Shepherd, D. Hutchinson, F. Garcia, and G. Coulson. Protocol Support for Distributed Multimedia Applications. *Computer Communications Journal*, 15(6):359–366, July, August 1992.

[58] R. Steinmetz. A multimedia File System Survey: Approaches for Continuos Media Disk Scheduling. *Computer Communications Journal*, April 1995.

[59] R. Steinmetz. Analyzing the Multimedia Operating System. *IEEE Multimedia*, 2(1), Spring 1995.

[60] H.J. Stüttgen. Network Evolution and Multimedia Communication. *IEEE Multimedia*, 2(3):42–59, Fall 1995.

[61] Inc. Sun Microsystems. Java home page. URL: http://www.javasoft.com/, September 1998.

[62] SUN Soft, Inc. *XIL Programmer's Guide*, 1994.

[63] Lucent Technologies, Cooperative Research Centre for Distributed Systems Technology (DSTC), Digital Equipament Corporation, Hewlett-Packard Company, International Computers Limited, Nortel Limited, and Inc. Novell. *Trading Object Service*, Revised Submission. Technical report, Object Management Group - OMG, May 1996.

[64] N. Tizzo. Sincronização de Fluxos de Dados em Aplicações Multimídia. Master's thesis, Faculdade de Engenharia Elétrica e Computação (FEEC) da Unicamp, 1996.

[65] C. Topolcic. Experimental Internet Stream Protocol. RFC 1190 Version 2 (ST- II), October 1990.

[66] Sankar Virdhagriswaran. Mobile Unstructured Business Object (MuBot) Technology. URL: http://www.crystaliz.com/papers/paper.html, 1997.

[67] A. Vogel, B. Kerhervé, G. von Bochmann, and J. Gecsei. Distributed Multimedia Applications and Quality of Service: A Survey. *IEEE Multimedia Journal*, August 1995.

[68] G. Wallace. The JPEG Still Picture Compression Standard. *Communication of the ACM*, 34(4):30–44, April 1991.

[69] J.E. White. Telescript Technology: An Introduction to the Language. Technical report, General Magic, 1995.

[70] N. Willians, G. Blair, and N. Davies. Distributed Multimedia Computing: An Assessment of the State of The Art. *Information Services and Use*, 11:265–281, 1991.

[71] ISO/IEC 10746-3 / ITU-T Draft Recommendation X.903. ODP Reference Model Part 3, Prescritive Model. Technical report, International Organization for Standardization and International Electrotechnical Committee, June 1995.

[72] R. Yavatkar and K. Lakshman. Communication Support for Distributed Collaborative Applications. *Multimedia Systems*, 2:74–87, Spring 1994.

[73] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource Reservation Protocol. *IEEE Network*, 5(5), September 1993.