Um Framework de Transmissão de Áudio e Vídeo para Novos Serviços de Telecomunicações

Eliane G. Guimarães*
Mareei Bergerman
Instituto de Automação
Fundação CTI - CP 6162
13089-970 Campinas, SP
eliane,marcel@ia.cti.br

Eleri Cardozo
Mauricio F. Magalhães
DCA-FEEC
UNICAMP - CP 6101
1303-970 Campinas, SP
eleri,mauricio@dca.fee.unicamp.br

Sumário

Atualmente, cada vez mais serviços de telecomunicações são oferecidos sobre redes Internet/Intranet. Estes serviços utilizam padrões abertos que incluem *World Wide Web* e suas tecnologias relacionadas (HTTP, HTML, XML, Java, etc.), TINA e CORBA.

Novos serviços de telecomunicações demandam comunicação multimídia e devem ser introduzidos e atualizados tão logo demandas do mercado sejam detectadas. Neste contexto, um projeto baseado em componentes de software é a melhor estratégia para o desenvolvimento destes serviços.

Este artigo apresenta o projeto e implementação de um *framework* baseado em padrões OMG para o suporte à comunicação multimídia requerida pelos novos serviços oferecidos sobre a Internet. O *framework é* baseado na linguagem de programação Java, o que permite o seu carregamento dinâmico no terminal do usuário via HTTP, bem como sua interoperabilidade com outros serviços baseados em CORBA.

Palavras-chave: Comunicação Multimídia, Serviços Internet, WWW, CORBA, TINA, Telerobótica.

Abstract

Today, more and more telecommunication services are being offered through the Internet. These services are based on open standards including the World Wide Web and its related technologies (HTTP, HTML, XML, Java, etc.), TINA and CORBA.

New telecommunication services demand multimedia communication and must be introduced and updated as soon as market demands have been detected. In this context, a component-based design is the best strategy for developing such services.

*Aluna de Doutorado da Faculdade de Engenharia Elétrica e de Computação, UNICAMP.

Figura 8: Pontos terminais de fluxo baseados em JMF utilizado no serviço Internet do REAL.

6 Conclusões

Novos serviços de telecomunicações devem empregar tecnologias de software que permitam sua rápida introdução, alteração e retirada de acordo com as forças de mercado. Desenvolvimento baseado em componentes e em padrões abertos se mostra como a estratégia mais factível na atualidade. Este artigo apresentou o projeto e implementação de um *framework* baseado no padrão A/V Streams do OMG. O projeto favorece a independência de plataforma graças ao emprego da linguagem Java, bem como a interoperabilidade com outros componentes de software por basear-se no padrão CORBA.

Pesquisas na linha de metodologias para o desenvolvimento de novos serviços de telecomunicações e incorporação de qualidade de serviço aos componentes do serviço estão em curso na FEEC/Unicamp e IA/CTI. A primeira linha visa contemplar técnicas modernas de desenvolvimento de software baseadas em *design patterns*, componentes e *frameworks* para novos serviços de telecomunicações. A segunda visa o estudo de estratégias de incorporação de qualidade de serviço desde o nível de aplicação até o nível de rede, dando aos serviços garantias de desempenho e qualidade que possam diferenciar seus provedores.

Agradecimentos

Este projeto tem o suporte das seguintes agências: FAPESP (proc. 97/13384-7), CNPq (proc. 300723/93-8), FINEP (proc. 1588/96). Os bolsistas de iniciação científica

retomar, destruir). A figura 7 ilustra os componentes do serviço. O protocolo IIOP garante a interoperabilidade entre ORBs de diferentes fornecedores.

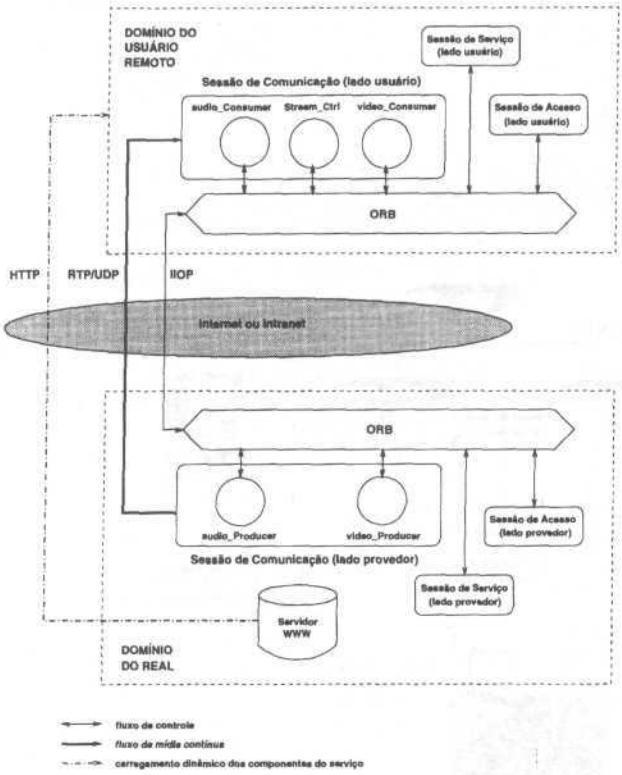

Figura 7: Componentes do serviço.

No momento, os serviços baseados em nossa implementação A/V Streams são incapazes de executar como *applets* na máquina virtual Java de navegadores. A razão é que os navegadores ainda não incorporaram a versão JDK 1.2 (exceto através de *plug-in*). Esta incorporação é esperada para breve dado que esta versão se encontra atualmente em produção. O número de linhas de código para o estabelecimento e controle do fluxo é mínimo (algumas dezenas), o que demonstra o poder de reuso nos desenvolvimentos orientados à componentes.

A figura 8 ilustra as telas gerenciadas pelos pontos terminais consumidores de fluxo.

pelo usuário remoto depende da velocidade de comunicação da Internet, sendo 5 quadros/s para vídeo uma qualidade típica para os dias atuais. Entretanto, projetos como o REAL se beneficiarão da Internet-2, onde taxas de vídeo entre 25 e 30 quadros/s serão comuns. A configuração do sistema completo é apresentada de forma ilustrativa na figura 6.

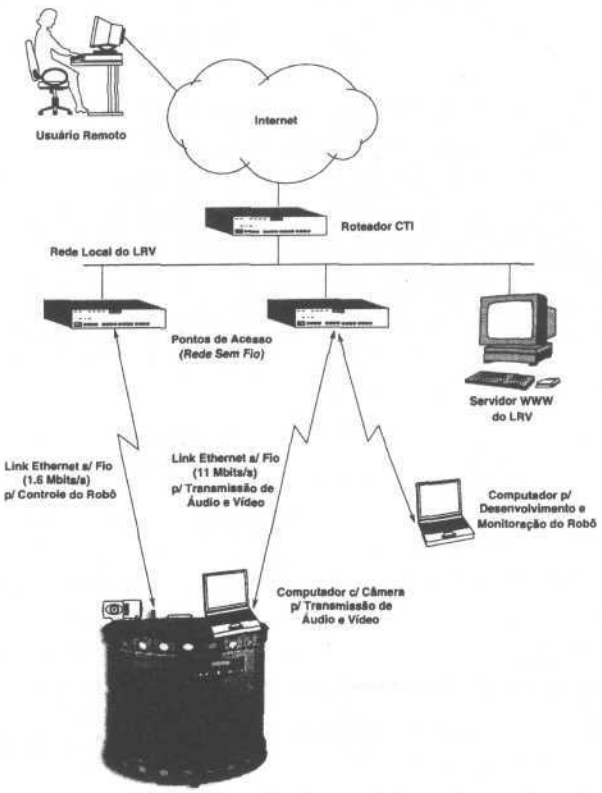

Figura 6: Esquema ilustrativo do REAL.

Como exemplo de utilização descreve-se, no âmbito do laboratório REAL, um serviço de áudio e vídeo que tem como um dos seus componentes o *framework* A/V Streams, detalhado nas seções 3 e 4, onde o usuário do serviço, localizado em seu domínio, estabelece um *stream* composto de um fluxo de vídeo e áudio para com o provedor do serviço, localizado no domínio do REAL. No lado do usuário remoto encontra-se dois pontos terminais consumidores de fluxo: um para áudio e um para vídeo. O controlador de *stream* `StreamCtrl` pode estar situado tanto no domínio do usuário quanto no domínio do REAL. O controlador de *stream* permite que a lógica do serviço controle o fluxo (parar,

objetos distribuídos localizados em máquinas heterogêneas. Esta característica permite o monitoramento dos componentes do A/V Streams que foram abertos por uma aplicação. Uma interface gráfica mostra as propriedades correntes dos componentes bem como os conjuntos de propriedades disponíveis pelos componentes. A figura 5 apresenta uma destas interfaces. Ao usuário do serviço é permitido, também, mudar as propriedades correntes, reconfigurando os componentes. Depois de configurado, o componente pode ser salvo e incorporado nos demais serviços que serão criados.

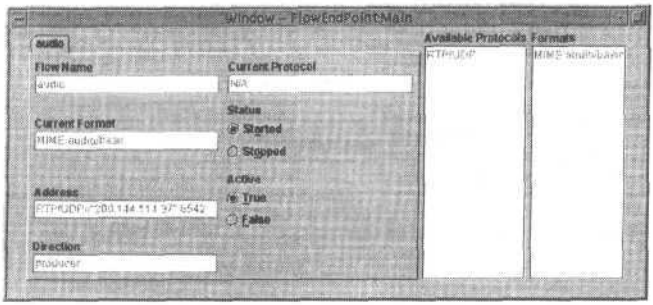

Figura 5: Propriedades associadas com um `FlowEndPoint`.

5 Serviços Internet no Laboratório REAL

O laboratório REAL (*RemotelyAccessible Laboratory*) *é* um laboratório de robótica com acesso remoto pela Internet, em desenvolvimento no CTI em cooperação com a UNICAMP. Este laboratório permite que usuários utilizem remotamente um robô móvel como se estivessem presentes fisicamente no laboratório. O laboratório é constituído dos seguintes componentes: o robô móvel XR4000; um sistema de aquisição de imagens a bordo do robô móvel e sua transmissão ao usuário remoto; e uma interface homem-máquina para programação do robô e recebimento dos dados de sua operação. O robô está conectado à rede departamental do LRV/CTI através de duas redes sem fio padrão Ethernet. A primeira rede opera em 1.6 Mbits/s, sendo empregada para controle do robô (navegação, sensoriamento, alarmes, etc.). A segunda rede, operando em 11 Mbits/s, se destina a transmissão de áudio e vídeo do robô para a estação do usuário remoto.

A programação remota do robô se dará através de interface compatível com a WWW. Um trabalho inicial nesta linha de pesquisa é descrito em [13]. A interface de programação permitirá ao usuário remoto programar, simular e carregar um programa de navegação para o robô XR4000.

Neste projeto, o foco é o *framework* de transmissão de áudio e vídeo necessário para fornecer ao usuário do laboratório uma realimentação visual do progresso do robô enquanto seu código de controle é executado remotamente. A qualidade do áudio e vídeo recebidos

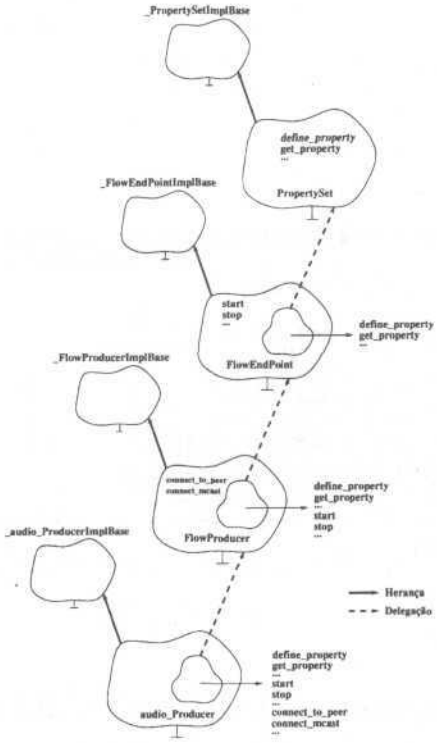

Figura 4: Princípio da composição via delegação.

Pontos terminais de fluxo atuando como consumidores (interface `FlowConsumer`) e produtores (interface FlowProducer) foram implementados tanto para áudio quanto para vídeo. Na implementação destas interfaces utilizou-se o Java Media Framework (JMF) versão 2.0 [12], um conjunto de objetos que permite a captura, transmissão via rede e apresentação de mídia contínua em Java. O protocolo RTP sobre UDP é empregado no transporte de mídia, tanto de áudio quanto de vídeo. Com JMF 2.0 tem-se uma implementação 100% Java do padrão A/V Streams.

Em nossa implementação é possível estabelecer tanto fluxos ponto-ponto quanto ponto-multiponto, sendo que este último emprega endereçamento IP multicast (classe D). Foram utilizadas estações de trabalho Sun Sparc Ultra na implementação. Dada a independência de plataforma da linguagem Java, a implementação pode fazer uso de micro-computadores (baseados em Windows 95/98/NT) sem necessidade de qualquer adaptação.

Uma característica adicional é descrita para fornecer facilidades de monitoramento e configuração que emprega os serviços de propriedades e de eventos do OMG entre

tador de objetos empregados pelo CORBA[5]. Compiladores IDL geram classes com herança múltipla quando mapeiam para linguagens que a suporta (como C++, por exemplo). No caso de Java, o compilador IDL gera classes que herdam apenas do adaptador de objeto, deixando a cargo do implementador a incorporação dos métodos definidos nas interfaces IDL de nível superior.

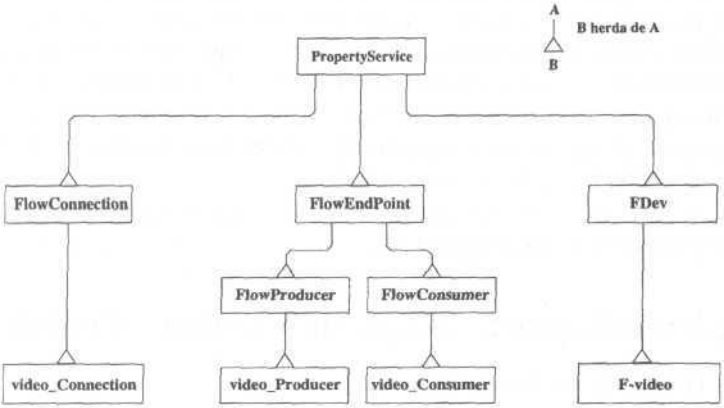

Figura 3: Relação de herança entre interfaces IDL dos componentes adicionais do perfil *full*.

A figura 4 ilustra o princípio da delegação na implementação de pontos terminais de fluxos mantendo-se as relações de herança conforme especificadas na figura 3. A estratégia é repassar para os objetos abaixo todos os métodos definidos nas classes acima. Por exemplo a chamada do método `start` em um produtor de áudio (`audio_Producer`) deve chamar explicitamente o método `start` de FlowProducer, delegando a este todo ou parte do processamento do método, e assim sucessivamente até o ponto mais alto da hierarquia onde o método é definido. Deve-se notar que temos múltiplos objetos na cadeia oferecendo o mesmo método, diferentemente de implementações baseadas em herança múltiplas (como em C++, por exemplo) onde tem-se apenas o objeto mais específico incorporando as funcionalidades presentes nos objetos dos quais deriva.

Infra-estrutura Utilizada

A versão do sistema de desenvolvimento Java utilizada foi o JDK 1.2 que já incorpora um ORB através do pacote `org.omg.CORBA`. A implementação utiliza múltiplas *threads* em vários pontos para fins de desempenho e concorrência.

[5]Este adaptador, denominado POA (*Portable Object Adapter*) tem por função adicionar à classe métodos utilizados para chamada remota de seus métodos através do ORB.

A conexão de um produtor a um ou mais consumidores ocorre através de uma conexão de fluxo (*flowconnection*). A interface `FlowConnection` abstrai o conceito de conexão de fluxo. Neste modo, conexões e pontos terminais de fluxo específicos têm as suas operações visíveis e acessíveis para o controle de fluxos individuais.

No perfil de especificação *light*, o sistema tem um controle em um nivel de abstração maior, onde as interfaces `FlowEndPoint` e FlowConnection não são visíveis externamente. Neste perfil os pontos terminais de fluxo são agregados em um componente maior, o ponto terminal de *stream* (*stream endpoint*). A interface `StreamEndPoint` modela os pontos terminais de um *stream*. Pontos terminais de *stream* têm ainda um componente associado para fins de configuração (por exemplo, tipo de mídia). Este componente é modelado através da interface VDev *(virtual device)*. De maneira análoga, neste perfil as conexões de fluxo são agregadas em um único componente: o controle de *stream (stream control)*. *A* interface `StreamCtrl` modela este componente.

Pontos terminais de *stream* são agregados em dispositivos multimídia (*multimedia devices*) modelados através da interface `MMDevice`.

4 Implementação da Sessão de Comunicação em Java

Princípios Gerais da Implementação

A implementação do *framework* A/V Streams foi realizada utilizando-se a linguagem de programação Java [9]. Os dois perfis *(light* e *full)* foram implementados de acordo com o documento de padronização [7]. Java facilita o desenvolvimento de software orientado a componentes através de vários *frameworks* de programação tais como Swing, JavaBeans, Enterprise JavaBeans, Java Media Framework, dentre outros.

Os serviços de Propriedades [10] (utilizados pelo A/V Streams) e de Eventos [11], foram implementados conforme especificado pelo OMG. Estes serviços são empregados para a definição de propriedades dos componentes A/V Streams[3] e notificação de eventos por parte dos mesmos componentes[4].

Java apresenta certas particularidades que influenciam na implementação de sistemas como A/V Streams. A mais marcante talvez seja a ausência de herança múltipla de implementação (Java suporta herança múltipla apenas de interfaces). Herança múltipla de interfaces favorece o desenvolvimento de software orientado a objetos. Entretanto, herança múltipla de implementações favorece a quebra de encapsulação quando da reutilização de uma classe. A estratégia utilizada na implementação foi baseada no princípio da composição de objetos, mais precisamente delegação. Delegação propicia a mesma reutilização de código que herança, mas sem infringir o princípio da encapsulação dos objetos.

A figura 3 ilustra as interfaces adicionais do perfil *full* e as relações de herança entre elas. Uma interface IDL que deriva de outra quando compilada gera uma classe que necessita herdar tanto os métodos da interface da qual deriva quanto os métodos do adap-

[3] Por exemplo, formato de mídia disponível em um ponto terminal de *stream.*

[4] Por exemplo, a perda de comunicação com determinado componente.

visíveis externamente. *Streams* são os elementos básicos de suporte à comunicação multimídia e representam a transferência de mídia contínua de um produtor para um ou mais consumidores. A especificação trata configurações de fluxo ponto-a-ponto e ponto-multiponto que podem ser utilizadas como blocos de construção para fluxos muitos-para-muitos e muitos-para-um. Todas as operações de controle e de sinalização são realizadas através do ORB. No entanto, os segmentos de mídia constituintes de um fluxo são transportados por fora do ORB, via protocolo específico, por exemplo RTP *(Real Time Protocol)* [8].

Existem dois perfis de especificação do serviço de *streams*: *full profile e light profile*. No perfil de especificação *full*, o sistema tem um controle maior sobre o estabelecimento e a manipulação de *streams*, fornecendo as interfaces IDL disponibilizadas pelo perfil *light*, mais um conjunto extra de interfaces IDL para agir sobre fluxos individuais.

A figura 2 ilustra a arquitetura e os principais componentes de um sistema que implementam o perfil *full* da especificação. O "duto" representa um *stream* transportando um fluxo de áudio e um fluxo de vídeo. Os fluxos ligam pontos terminais *(flow endpoints)*, modelados através da interface FlowEndPoint (FEP). Esta interface é especializada para produtores de fluxo (FlowProducer) e consumidores de fluxo (`FlowConsumer`). Um componente denominado dispositivo de fluxo *(flow device)* age como fábrica de pontos terminais de fluxo. Este componente é modelado através da interface FDev.

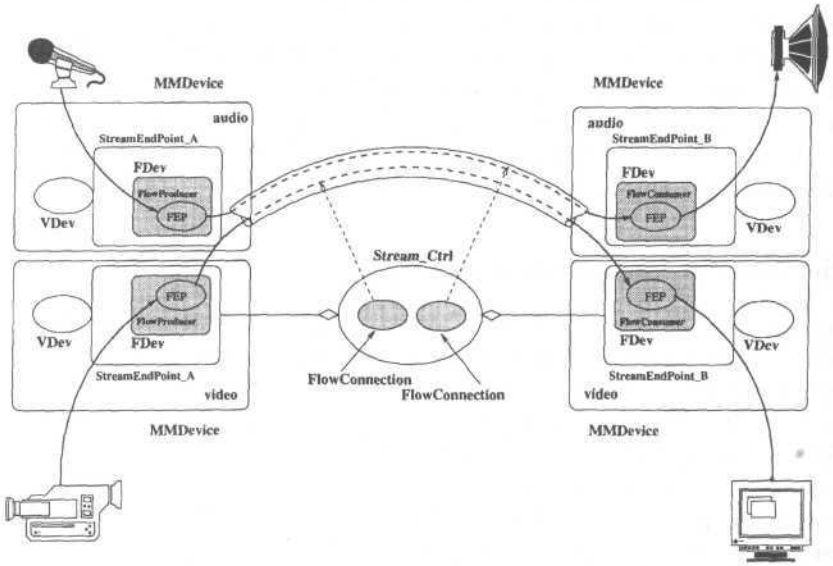

Figura 2: Componentes da especificação A/V Streams. Os componentes sombreados são disponíveis apenas no perfil *full* da especificação.

e egresso, políticas de condução da conferência, etc. A lógica do serviço para os serviços de telecomunicações modernos é bem mais complexa que a lógica dos serviços telefônicos tradicionais baseados em redes inteligentes (IN). Este aumento de complexidade se deve a fatores como capacidade de gerência e especialização por parte do usuário; diversidade de terminais; acesso ubíquo; natureza *multiparty*; dentre outros. Neste contexto, a lógica do serviço deve ser distribuída entre os terminais e o provedor do serviço. A tecnologia de objetos distribuídos permite a introdução de lógica mais complexa distribuída entre objetos que interagem entre si.

CORBA é no momento a tecnologia mais promissora para suporte a objetos distribuídos, razão pela qual decidimos por sua utilização como infra-estrutura de suporte à lógica do serviço. Neste sentido, todos os componentes dos serviços interagirão através de um ORB *(Object Request Broker).* A maturidade do protocolo IIOP *(Internet Inter-ORB Protocol)* que permite a interoperabilidade de ORBs de diferentes fornecedores, e a incorporação de um ORB nativo na linguagem Java reforçam ainda mais esta decisão de projeto.

Apesar da Arquitetura de Serviço TINA-C possuir soluções para os componentes Sessão de Serviço e Sessão de Comunicação optamos por não utilizar as especificações TINA para estes componentes. A razão de tal decisão é que estamos enfocando o aspecto de especialização ("customização") e configuração dinâmica do serviço, aspectos estes não cobertos pelas especificações TINA. TINA utiliza pontos de referência para fins de interoperabilidade entre os componentes nos domínios do provedor e do usuário do serviço. Estes pontos de referência são complexos e não permitem uma boa adequação do serviço ao terminal e perfil do usuário. A complexidade da arquitetura de serviço também impossibilita a sua implementação em dispositivos de baixo poder computacional como telefones celulares e *notepads.* Como soluções para as Sessões de Serviço e de Comunicação estamos adotando três tecnologias: WWW, Java e CORBA, exigindo do terminal do usuário apenas um navegador WWW para o suporte ao serviço.

3 Sessão de Comunicação

Em nossa implementação do modelo de serviço da figura l a Sessão de Comunicação é baseada no padrão OMG *Control and Management of Audio/Video Streams* [7] (A/V Streams). O serviço de áudio/vídeo do OMG é um *framework* orientado a objetos que pode ser estendido e adaptado às necessidades do serviço. Esta extensão permite incorporar aspectos específicos do serviço constituindo-se desta forma de um elemento de diferenciação entre provedores de serviço.

A especificação apresenta um modelo arquitetural e um conjunto de interfaces IDL[2] para estabelecer, controlar e gerenciar agrupamentos de fluxos de mídia contínua denominados *streams.* O modelo arquitetural proporciona as definições dos componentes que constituem um *stream e* as interfaces IDL que disponibilizam as suas operações que são

[2] *Interface Definition Language*: linguagem de descrição de interfaces para objetos distribuídos padronizada pelo OMG no âmbito da especificação CORBA.

dinâmico no terminal do usuário, o serviço deve ser baseado integralmente na linguagem Java. Pelo menos dois cenários são possíveis:

1. o provedor disponibiliza um *plug-in* que permite incorporar o serviço ao navegador instalado no terminal do usuário;

2. o serviço é carregado a partir de um servidor WWW todas as vezes que o cliente decidir utilizá-lo.

A primeira alternativa se mostra inviável dado que torna o serviço dependente do navegador. Ademais, novas versões e atualizações do serviço devem ser instaladas pelo usuário quando disponibilizadas pelo provedor. A segunda alternativa é atrativa no sentido que o usuário sempre terá a última versão do serviço, mas o tempo de carregamento pode ser demasiadamente longo em acessos de baixa velocidade. Uma alternativa seria um esquema de *cache* onde o serviço, após carregado, permaneceria instalado no terminal do usuário por um período de tempo.

Um modelo simplificado para serviços de telecomunicações, inspirado nos padrões TINA, é dado na figura 1. Neste modelo um serviço é composto de três sessões: acesso, serviço e comunicação. Estamos desenvolvendo soluções para estes três componentes, apesar deste artigo enfocar um componente específico: a sessão de comunicação, o qual permite ao usuário do serviço estabelecer sessões de comunicação multimídia com o provedor ou com os demais usuários do serviço.

Figura 1: Modelo simplificado de um serviço de telecomunicações.

O componente Sessão de Acesso permite ao cliente acessar o serviço no provedor. Em nossa implementação, este componente é baseado no documento *Service Architecture Version 5.0* [4] do Consórcio TINA (TINA-C). As referências [5] e [6] ilustram a implementação dos componentes da Arquitetura de Serviço TINA. Estes componentes são genéricos o suficiente para serem utilizados em praticamente todos os serviços, sem alterações ou com alterações mínimas. Nota-se aqui que o desenvolvimento orientado a componentes contribui para a diminuição do tempo de introdução de novos serviços.

O componente Sessão de Serviço prove as funções relativas à lógica do serviço. Por exemplo, em um serviço de teleconferência este componente conteria funções de ingresso

O padrão de interfaceamento com o usuário deve prover um acesso uniforme ao serviço com interfaces conhecidas pelo usuário. Neste aspecto, os padrões associados à WWW (HTTP, HTML, Java, JavaScript, etc.) se apresentam como uma solução já disponível e de aceitação indiscutível. Portanto, a interface do serviço deve ser um navegador (*Web Browser)* já familiar aos usuários da Internet.

Um padrão de acesso e interação com o provedor do serviço facilita a programação da lógica do serviço. Tradicionalmente, as aplicações na Internet seguem o modelo cliente-servidor: a figura do provedor de serviço fica reduzida a um nó da rede que disponibiliza o serviço. Em um contexto mais atual, o provedor do serviço não apenas disponibiliza o serviço em seus servidores, mas também possui outras atribuições não encontradas na Internet como por exemplo:

- permitir ao cliente especializar o serviço de acordo com as suas necessidades ou possibilidades;
- repassar o processamento de parte ou de todo o serviço a um terceiro (*thirdparty*);
- tarifar o serviço;
- disponibilizar aos clientes novas versões do serviço assim que se tornarem operacionais.

Neste cenário, a arquitetura TINA (*Telecommunication Information Network Architecture)* [1] surge como um padrão a se considerar na estruturação de serviços na Internet.

Finalmente, um padrão de processamento distribuído faz-se necessário pois o serviço será composto de um conjunto de objetos distribuídos, parte localizados no domínio do cliente, parte no domínio do(s) provedor(es) do serviço. O padrão CORBA (*Common Object Request Broker Architecture*) [2] do OMG[1] *(Object Management Group)* surge como a melhor alternativa para suporte ao processamento distribuído que o serviço demanda.

Este artigo apresenta um modelo de serviço baseado em componentes [3] e a implementação de seu componente fundamental responsável pelo estabelecimento e gerência da sessão de comunicação empregada pelo serviço. A seção 2 apresenta um modelo para o serviço; a seção 3 apresenta o componente responsável pela sessão de comunicação, que é o enfoque principal deste trabalho. Este componente tem sua implementação e exemplo de utilização descritos nas seções 4 e 5, respectivamente. Finalmente, a seção 6 apresenta as conclusões e trabalhos futuros.

2 Estruturação do Serviço

Os serviços devem ser instalados no terminal do usuário através da própria rede sobre a qual o serviço é oferecido. A tecnologia Java permite o carregamento de *applets* (código Java) disponibilizados em um servidor WWW e sua execução na máquina virtual do navegador. Portanto, para atender o quesito de independência de plataforma e carregamento

[1] http://www.omg.org

This paper presents the design and implementation of a framework based on OMG standards for supporting multimedia communication as demanded by the new Internet Services. The component is based on the Java programming language, allowing its dynamic downloading to the user terminal via HTTP, as well as its interoperability with other CORBA-based services.

Keywords: Multimedia Communication, Internet Services, WWW, CORBA, TINA, Telerobotics.

l Introdução

O primeiro passo para estender o serviço tradicional de voz veio com os padrões RDSI (Rede Digital de Serviços Integrados) introduzidos no início da década de oitenta pela ITU-T (ex-CCITT). A idéia era prover serviços de natureza *telemática* tais como videotexto e video/áudio-conferência. RDSI teve uma aceitação modesta em parte por necessitar uma rede de acesso diferente da rede telefônica convencional e pela incapacidade de suas conexões de 64 Kbits/s transportarem áudio e vídeo de qualidade.

No início da década de noventa inicia-se o crescimento vigoroso da Internet com a introdução da *World Wide Web* (WWW). Somado ao alcance mundial da rede, surgiram na segunda metade desta década novas tecnologias de acesso, tais como xDSL *(Digital Subscriber Loop), cable modem,* e acesso sem fio. As duas primeiras aproveitam as redes que já atingem um percentual considerável dos domicílios: a rede telefônica convencional e a rede de TV a cabo. Estas tecnologias permitem acesso a velocidades acima de l Mbit/s a baixo custo.

Neste contexto, a rede Internet se configura hoje como a rede digital de serviços integrados, nos moldes imaginados pelos proponentes dos padrões RDSI. Várias questões são colocadas no que tange ao oferecimento de serviços através da rede Internet, por exemplo:

- em que padrões e tecnologias os serviços devem ser baseados?
- como garantir determinados níveis de qualidade aos serviços oferecidos?
- qual o modelo de tarifação adequado?
- como obter privacidade e segurança em uma rede desprotegida?

Este artigo trata da primeira questão. Três padrões são fundamentais para a introdução de serviços em larga escala na Internet:

1. um padrão de interfaceamento com o usuário do serviço;
2. um padrão de acesso e interação com o provedor do serviço;
3. um padrão de processamento distribuído aberto.

James Law Pereira (FAPESP 99/09922-9), Mateus Moço (CNPq/PIBIC) e Henrique M. Holschuh (FAPESP 98/07481-2) contribuem nas implementações descritas neste artigo.

Referências

[1] H. Berndt and et. al. *The TINA Book.* Prentice-Hall Europe, 1999.

[2] OMG. The common object request broker: Architecture and specification - version 2.2. Object Management Group, http://www.omg.org, March 1998.

[3] C. Szyperski. *Component Software: Beyond Object-Oriented Programming.* Addison-Wesley, 1997.

[4] TINA-C. Service architecture version 5.0. TINA Consortium, http://www.tinac.com, Jun 1997.

[5] A.S. Pinto, E.J. Oliveira, L.F. Faina, and E. Cardozo. Tina-based environment for mobile multimedia services. In *TINA Conference'99,* Hawaii, USA, April 1999.

[6] E.J. Oliveira, A.S. Pinto, L.F. Faina, and E. Cardozo. Sessões de acesso e serviço tina baseadas em web. In *17 Simpósio Brasileiro de Redes de Computadores - SBRC'99,* Salvador, Bahia, Maio 1999.

[7] OMG. Control and management of audio/video streams. Technical Report telecom/97-05-07, Object Management Group, 1997.

[8] H. Schulzrinne and et. al. Rtp: A transport protocol for real-time applications. Technical Report RFC 1889, IETF Audio-Video Transport Working Group, January 1996.

[9] SUN. Java home page. http://java.sun.com/products/, 1999.

[10] International Business Machine Corporation, Inc. Sun Soft, and Inc. Taligent. "Property Service". Technical Report TC Document 95.6.1, Object Management Group, Dec. 1997. http://www.omg.org/.

[11] OMG. "Event Service Specification". Technical Report Document 97-12-11, Object Management Group, 1997. http://www.omg.org/.

[12] Sun Microsystems. *"Java Media Framework Programmer's Guide (v 0.5)",* Dec. 1998. http://java.sun.com/products/java-media/jmf.

[13] L. R. Queiroz, M. Bergerman, R. C. Machado, S. S. Bueno, and A. Elfes. A robotics and computer vision virtual laboratory. In *5th International Workshop on Advanced Motion Control,* pages 694–699, Coimbra, Portugal, June 1998.