Uma Arquitetura para Disponibilização e Gerência de Serviços na Internet

Rossano P. Pinto[*], **Luis F. Faina**[†], **Antonio T. Maffeis**
Eliane G. Guimarães[‡], **Carlos A. Miglinsk**, **Eleri Cardozo**
DCA - FEEC - UNICAMP - CP 6101
Campinas, SP, Brasil, CEP 13083-970
e-mail: {rossano,faina,maffeis,eliane,mig,eleri}@dca.fee.unicamp.br

Resumo

O impacto das telecomunicações tem sido cada vez maior na evolução tecnológica e econômica das sociedades. Adicionalmente, a desregulamentação do setor de telecomunicações trouxe competição entre as operadoras, forçando-as à introdução permanente de novos serviços. Na especificação e no desenvolvimento desses serviços, tecnologias de comunicação multimídia, orientação a objetos, reuso de componentes, sistemas distribuídos e arquitetura de serviços vem sendo empregados. Dentre as arquiteturas de serviço, destaca-se TINA (*Telecommunications Information Network Architecture*) devido a sua natureza aberta e independência tecnológica.
Este artigo propõe uma arquitetura que contempla os aspectos positivos de TINA aliados a padrões abertos como WWW, Java e CORBA. Esta arquitetura suporta a implantação, acesso, uso e gerência de serviços oferecidos através da WWW. O artigo apresenta uma implementação da Arquitetura de Serviço TINA; um serviço de laboratório virtual que faz uso desta arquitetura; e uma extensão (baseada em agentes móveis) da Arquitetura TINA visando o suporte à ubiqüidade dos serviços de telecomunicações.
Palavras-chave: Serviços de Telecomunicações, Mobilidade Pessoal e de Serviços, Agentes Móveis, Laboratórios Virtuais, Tele-robótica, Arquitetura TINA, CORBA.

Abstract

The impact of telecommunication over the technological and economic evolution of the societies is steadly increasing. In addition, the deregulamentation of the telecommunication sector brought competition among operators, forcing them to introduce new communication services in a permanent way. In the specification and development of these new services, technologies of multimedia communication, object-orientation, reuse of components, distributed systems, and service architectures have being employed. Among the service architectures, TINA (Telecommunications Information Network Architecture) is receiving special attention due to its openness and technological independence.

[*] Aluno de Doutorado da FEEC/UNICAMP
[†] Prof. da Faculdade de Computação, Univ. Federal de Uberlândia, MG.
[‡] Pesquisadora no Centro de Pesquisas Renato Archer (CenPRA), Campinas, SP.

This paper proposes an architecture that contemplates the positive aspects of TINA, added to open standards such as WWW, Java and CORBA. This architecture supports the deployment, access, use, and management of services available through the Web. The paper presents an implementation of the TINA Service Architecture; an example of a service in the field of virtual laboratories employing this architecture; and an extension (based on mobile agents) of the TINA Architecture aiming the support of ubiquity in telecommunication services.
Keywords: Telecommunication Services, Personal and Service Mobility, Mobile Agents, Virtual Laboratories, Telerobotic, TINA Architecture, CORBA.

1 Introdução

Nos últimos anos tem-se presenciado mudanças profundas nas telecomunicações e na computação tais como a ubiqüidade da Internet, o uso cada vez maior da computação móvel e o aumento da capacidade de transmissão de informação graças às redes óticas [1]. Este cenário contribui para o surgimento de novos serviços de telecomunicações centrados na Internet, no conteúdo, e na interação multi-partite através de áudio e vídeo de alta resolução (os chamados serviços telemáticos, como por exemplo laboratórios virtuais). Adicionalmente, a mobilidade tem sido uma característica cada vez mais importante, disponibilizada pelas redes sem fio e terminais portáteis com alto poder computacional tais como *screen phones* e *smart cellular phones*.

A disponibilização destes novos serviços demanda em grande volume de *software* quando comparados aos serviços de voz disponibilizados através da rede pública de telefonia e rede inteligente (PSTN/IN). Nesse contexto, o Consórcio TINA (*Telecommunications Information Networking Architecture*) [2] foi formado para definir uma arquitetura comum de *software* para serviços multimídia e de informação, no âmbito das telecomunicações. A arquitetura TINA visa o desenvolvimento e gerência de serviços de qualquer complexidade, com maior rapidez e confiabilidade.

Estabelecido em 1993 por cerca de 40 empresas que incluíam operadoras de telecomunicações, fabricantes de computadores e equipamentos de telecomunicações, o Consórcio TINA norteou a proposição e especificação de uma arquitetura de *software* comum e coerente para oferecer serviços de telecomunicações e de informação [2]. O objetivo maior é estabelecer uma arquitetura para serviços de telecomunicações possibilitando seu rápido desenvolvimento, rápida introdução e elevada capacidade de gerência e configuração.

O Consórcio TINA foi encerrado em dezembro de 2000, tendo produzido um amplo conjunto de especificações. Atualmente, muitos profissionais que participaram do Consórcio TINA estão atuando em consórcios como ATMF (*Asynchronous Transfer Mode Forum*), DAVIC (*Digital Audio-Visual Council*), ITU-T (*International Telecommunication Union-Telecommunication*), TMF (*Telemanagement Forum*) e OMG (*Object Management Group*) com o intuito de difundir a Arquitetura TINA nas especificações produzidas por estes fóruns.

A Arquitetura TINA baseia-se em 4 princípios: projeto e análise orientados a objeto; processamento distribuído dos objetos; desacoplamento dos componentes de *software*; e princípio da separação, ou seja, i. separação lógica entre aplicações e ambiente em que são executadas; e ii. separação das aplicações em parte específica de serviço e em parte de gerência e controle. O objetivo desses princípios é garantir a interoperabilidade, portabilidade e reusabilidade de componentes de *software* bem como a independência de tecnologias específicas.

A arquitetura TINA contempla todos os participantes da indústria de provisão de serviços.

Provedores de Serviço (*retailers*) oferecem serviços "no varejo" para Consumidores (usuários finais de serviços). Provedores de Serviços podem se utilizar de Terceiros (*third parties*) que oferecem serviços ou componentes destes "no atacado". A conexão entre consumidores e provedores de serviço fica a cargo dos Provedores de Conectividade que oferecem serviços de rede de transporte. Finalmente, o Intermediador de Serviços (*Broker*) mantém uma espécie de serviço de páginas amarelas para localização de provedores e seus serviços.

Como decorrência do princípio da separação, a Arquitetura TINA é dividida em 3 subarquiteturas: Arquitetura de Computação, Arquitetura de Serviço e Arquitetura de Recursos de Rede. A Arquitetura de Serviço [3, 4] define um conjunto de componentes reusáveis e interoperáveis e está baseada em dois princípios: i. separação dos objetos em objetos genéricos (comuns a todos os serviços) e objetos específicos representando a lógica do serviço; e ii. uso do conceito de sessão, uma generalização do conceito de chamada nas redes PSTN/IN. Uma sessão representa a informação usada por todos os processos envolvidos na provisão de um serviço pelo tempo de sua duração, garantindo uma visão coerente dos vários eventos e relacionamentos associados à provisão do serviço [3]. Três sessões são definidas, cada uma correspondendo a uma atividade:

1. Sessão de Acesso: estabelece uma relação segura entre duas ou mais partes, por exemplo, autenticação do usuário e a escolha de serviços.
2. Sessão de Serviço: corresponde as operações de ativação, uso e gerência de um serviço, estando sempre associada a uma sessão de acesso.
3. Sessão de Comunicação: associada a uma sessão de serviço, possibilita o transporte de fluxos contínuos de dados (p. ex., áudio e vídeo) através da rede de transporte.

O artigo está organizado da seguinte forma. A seção 2 apresenta uma implementação da Arquitetura de Serviço TINA. A seção 3 descreve um serviço de laboratório virtual que faz uso desta arquitetura. A seção 4 apresenta uma extensão (baseada em agentes móveis) da Arquitetura TINA visando o suporte à ubiqüidade dos serviços de telecomunicações. Finalmente, a seção 5 tece considerações finais sobre o trabalho descrito no artigo.

2 Uma Implementação da Arq. de Serviço TINA

A Arquitetura de Serviço TINA especifica vários objetos computacionais, dos quais implementamos o subconjunto apresentado na Fig. 1. Um objeto computacional é constituído de uma ou mais interfaces, cada qual caracterizada como interface operacional ou interface de fluxo contínuo (p. ex., áudio e vídeo).

Os objetos estão divididos em domínio do usuário e domínio do provedor. Os objetos do domínio do usuário são PA, asUAP e ssUAP, descritos a seguir juntamente com os demais objetos do domínio do provedor:

- PA (*Provider Agent*) - objeto genérico que atua na sessão de acesso e representa o provedor no domínio do usuário;
- asUAP (*access session - User Application*) - objeto genérico que faz papel de interface com o usuário na sessão de acesso;
- ssUAP (*service session - User Application*) - objeto que atua como interface homem-máquina no domínio do usuário;

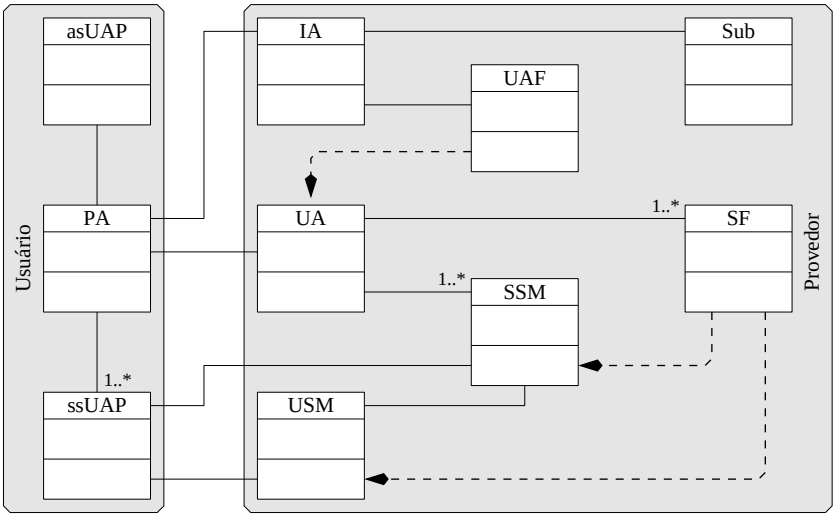

Fig. 1: Diagrama de classes do Ponto de Referência entre provedores TINA.

- IA (*Initial Agent*) - objeto genérico da sessão de acesso, presente no domínio do provedor, que representa o ponto de contato inicial com um provedor;
- UA (*User Agent*) - objeto genérico que representa o usuário no domínio do provedor;
- SF (*Service Factory*) - objeto responsável pela instanciação de um dado serviço;
- SSM (*Service Session Manager*) - é a instância de um serviço;
- USM (*User Service Manager*) - quando um serviço possui vários participantes, cada participante possui um objeto USM sendo gerenciado por um único objeto SSM.

2.1 Distribuição dos Objetos

A Fig. 2 apresenta a distribuição dos componentes mostrados na Fig. 1 para a arquitetura implementada. Podemos observar que existe apenas um único ORB (*Object Request Broker*) no domínio do usuário, independente do número de janelas abertas do navegador, no qual todos os objetos servidores (*servants*) ficam conectados. No domínio do provedor pode-se observar a mesma característica com relação ao ORB. Apenas três processos no domínio do provedor atendem as requisições vindas dos clientes: um dos processos contém o servidor Web, o qual responde às requisições HTTP (*Hyper Text Tranfer Protocol*) vindas dos navegadores Web presentes nos terminais de usuário; um outro processo executa um serviço de nomes; o terceiro processo executa uma máquina virtual Java, encarregada de interpretar o *bytecode*[1] que instancia o ORB e todos os objetos do provedor.

Um quarto processo no domínio do provedor executa um SGBD (Sistema Gerenciador de Banco de Dados) que é utilizado pelo objeto Sub do provedor para gerenciar informações de usuários e serviços. A comunicação entre o objeto Sub e o SGBD se dá via JDBC (*Java Data Base Connectivity*).

Subscrição de Serviços

Para que um usuário possa usufruir dos serviços oferecidos por um provedor, é necessário que este faça uma subscrição. Munido de *username* e senha, o usuário deve estabelecer uma

[1] Instruções para a máquina virtual Java

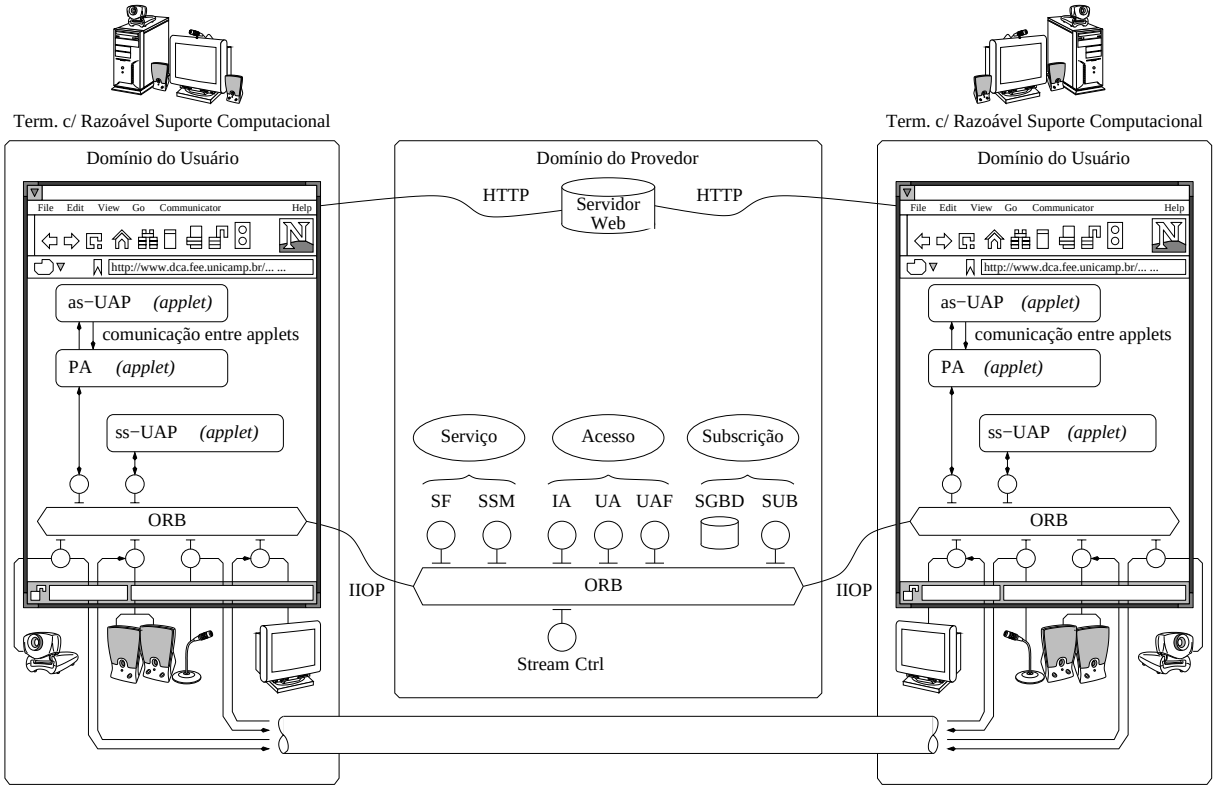

Fig. 2: Modelo tecnológico da arquitetura implementada.

sessão de acesso no provedor para que possa fazer uso de serviços através de sessões de serviço. Todas as informações a respeito de assinatura e utilização de serviços são manipuladas pelo objeto Sub. A interface de subscrição (Fig. 3) é implementada em forma de *applet* e acessível a partir da URL (*Uniform Resource Locator*) inicial do provedor.

Acesso aos Serviços

Após a subscrição o usuário está apto a utilizar os serviços por ele assinado. Para tanto é necessário que seja estabelecida uma sessão de acesso, fornecendo *username* e senha, que servirão para autenticação. O objeto responsável pela interação usuário -> sessão de acesso é o asUAP, que é descarregado no domínio do usuário como um *applet* Java juntamente com o objeto PA. É através do asUAP que o usuário faz a escolha de serviços e recebe convites para ingressar em serviços iniciados por outros usuários. A Fig. 4 mostra a tela apresentada pelo asUAP logo após o *login*.

O ponto de contato entre domínio do usuário e domínio do provedor é o objeto IA que tem a única função de iniciar uma sessão de acesso a pedido do usuário. O objeto PA faz a intermediação entre asUAP e IA (Fig. 2). Além de métodos convencionais, o asUAP possui dois métodos do tipo *callback*, `callback()` e `warning()`. O método `callback()` serve dois propósitos: receber convites e lista de serviços assinados. O método `warning()` é utilizado para envio de mensagens ao asUAP (*frame* C da Fig. 4). O *frame* B do navegador Web, visto na Fig. 4, hospeda o asUAP. O PA é responsável por instanciar o ORB no domínio do usuário que vai ser utilizado por todos os objetos do domínio. O PA é registrado no serviço de nomes do provedor como TerminalIP:PA (Ex.: 143.106.11.135:PA).

Durante a fase de estabelecimento da sessão de acesso, um representante único do usuário,

Fig. 3: Subscrição de Serviços.

Fig. 4: *Frames* da Sessão de Acesso.

o objeto UA, é criado no domínio do provedor. O objeto responsável por gerenciar a vida dos UAs é o UAF. Na nossa implementação o UA existe enquanto existir uma sessão de acesso ativa para o usuário em questão.

Uso dos Serviços

Uma vez estabelecida a sessão de acesso, uma lista de serviços disponíveis é apresentada ao usuário. Na escolha de um serviço uma sessão de serviço deve ser estabelecida para que seja possível utilizar o serviço. O objeto SF, que é a fábrica de serviço, é responsável por instanciar objetos SSM e USM, que possuem a lógica do serviço a ser oferecido. Cada serviço possui uma fábrica distinta (é possível também utilizar uma mesma fábrica para todos os serviços).

O objeto SSM, que é o gerente de sessão de serviço, possui métodos genéricos para controle da sessão de serviço e pode possuir ainda métodos específicos do serviço para o qual este foi desenvolvido. Construímos dois SSMs de uso genérico: o SAS (Suporte ao Agendamento de Serviços) e Alarme. O SAS tem a função de gerenciar o acesso concorrente a recursos compartilhados (Ex.: laboratório virtual). O Alarme pode ser usado, por exemplo, para preempção de um serviço. O objeto que interage com o SSM ou USM é o ssUAP. Os objetos ssUAP, SSM e USM são dependentes de serviço, sendo assim, postergaremos uma explicação mais detalhada destes para as seções 3 e 4.

Inicialização do Provedor

A primeira ação executada na inicialização do servidor é a leitura de um arquivo de configuração contendo todas as informações necessárias para seu funcionamento como: nome do provedor, parâmetros de segurança na abertura de *sockets*, endereço do SGBD, entre outras. Estas informações trazem flexibilidade, permitindo, por exemplo, trocar um SGBD por outro sem a necessidade de recompilar objetos no domínio do provedor. Após a leitura das configurações e inicialização dos atributos do provedor, o ORB e os objetos iniciais Sub, IA, UAF, SF, SAS e

Alarme são instanciados e registrados no serviço de nomes.

2.2 Uso da Implementação

Para se utilizar o protótipo da sessão de acesso e parte do suporte da sessão de serviço faz-se necessário atender alguns requisitos de infra-estrutura [5] tanto no domínio do provedor quanto no domínio do usuário. No que se refere ao provedor (provedor de serviço), o mesmo deve possuir um servidor Web para que os componentes do serviço possam ser disponibilizados no domínio do usuário. Tanto a localização quanto as interações entre os objetos se dão via ORBs que estejam em conformidade com a especificação CORBA (*Common Object Request Broker Architecture*). Já as *profiles* de usuários e serviços devem ser armazenadas por um SGBD que disponibilize um *driver* compatível com JDBC.

No que se refere aos serviços (p. ex., componente no domínio do usuário e do provedor), os mesmos são disponibilizados através de objetos distribuídos agregados em módulos. O módulo do serviço responsável por interagir com o usuário e apresentar resultados, ou seja, o módulo presente no terminal do usuário, deve ser disponibilizado em forma de *applet* Java, que deve ser assinado para satisfazer os requisitos do modelo de segurança do Java. Para impossibilitar que usuários não assinantes utilizem um serviço, os *applets* do serviço (ssUAPs) que serão descarregados nos terminais de usuário devem obter a referência do objeto PA (descarregado junto com o asUAP). Com esta referência é possível programar o ssUAP para consultar o objeto PA sobre a situação atual da sessão de acesso do usuário em questão. A informação retornada servirá como indicativo de sessão de acesso em curso ou não estabelecida (situação que levará à interrupção do ssUAP, não permitindo que o usuário prossiga no uso do serviço).

Por fim, no que se refere ao terminal do usuário, o mesmo deverá apresentar suporte computacional para execução de um navegador Web e *applets* Java (sugere-se o Java *plugin* 1.3 ou superior). Como a máquina virtual Java possui mecanismos de segurança que bloqueiam operações potencialmente inseguras e destrutivas, para que o *applet* Java possa executar as operações necessárias ao seu funcionamento é preciso que o cliente configure um arquivo chamado .java.policy[2] permitindo, assim, o acesso a alguns recursos de máquina, como por exemplo, abertura de *sockets* e permissão para resolução de nomes através do DNS (*Domain Name System*).

2.3 Detalhes de Implementação

Todos os objetos foram implementados em Java utilizando a plataforma Java 2 SDK [6] versão 1.3. O protótipo está disponibilizado sobre um DPE (*Distributed Processing Environment*), composto por ORBs que seguem a especificação CORBA 2.0 do Consórcio OMG. Como infra-estrutura CORBA, utilizamos o ORB disponível na Plataforma Java. O kit (SDK) possui um compilador IDL, idlj, que já contempla as modificações da especificação CORBA 2.3, e suporta a especificação CORBA IIOP (*Internet Inter-ORB Protocol*) [7] para comunicação inter-ORBs em uma rede TCP-IP.

Os terminais envolvidos no teste da aplicação são compostos por PCs com Windows 2000 ou Linux e estações de trabalho Sparc ou Ultra com SunOS. Os objetos do domínio do usuário são disponibilizados em forma de *applet*, pois o ambiente de execução neste domínio é composto unicamente por navegadores Web com incorporação do Java *plug-in* do SDK 1.3.

[2] Esta é a única configuração requerida pelo cliente.

Para que as páginas HTML e *applets* Java, encontradas no domínio do provedor, estejam disponíveis para os navegadores nos terminais de usuário, utilizamos o servidor Web Apache no provedor. Como o provedor TINA precisa manter informações acerca do perfil dos usuários e de alguns serviços oferecidos, estas informações são armazenadas em uma base de dados gerenciada pelo PostgreSQL [8], um SGBD relacional, acessado através de *driver* compatível com JDBC.

3 REAL: Um Serviço de Laboratório Virtual

REAL (*REmotely Accessible Laboratory*) é um laboratório virtual desenvolvido pelo CenPRA em cooperação com a UNICAMP. Laboratórios virtuais constituem ferramentas educacionais e experimentais modernas que permitem o acesso remoto a recursos laboratoriais. Através de laboratórios virtuais pesquisadores, localizados em pontos geograficamente dispersos, podem realizar experimentos, compartilhar dados experimentais, complementar as suas infra-estruturas laboratoriais, além de prover uma base experimental comum aos diferentes grupos de pesquisa.

O projeto REAL compartilha muitas características comuns com projetos na linha de laboratórios virtuais e robôs acessíveis através da Internet [9]. Entretanto, REAL foi desenvolvido como um serviço de telemática na Internet, não como uma simples aplicação da WWW. Para tal, o serviço possui uma estruturação em três sessões (acesso, serviço e comunicação), cada qual com uma sofisticada estrutura de gerência. Estas sessões foram implementadas segundo padrões abertos tais como TINA, CORBA, UML (*Unified Modeling Language*), e tecnologias associadas à Internet (HTTP, XML - *eXtensible Markup Language* - e Java). A sessão de acesso utiliza a implementação da Arquitetura de Serviço TINA descrita na seção anterior. As sessões de serviço e comunicação foram construídas segundo um modelo de componentes descrito em [10, 11].

A sessão de serviço é estruturada segundo o modelo cliente-servidor. A lógica do serviço no lado servidor foi desenvolvida em Java. Já a parte de comunicação com o robô foi desenvolvida em C++, haja visto que a interface de programação (API) do robô é disponibilizada apenas nesta linguagem. O lado cliente foi implementado totalmente utilizando-se *applets* Java, o que dispensa qualquer instalação prévia de software por parte do cliente. A interação entre o cliente e o servidor ocorre via CORBA. A sessão de comunicação se baseia no padrão A/V Streams estabelecido pelo OMG em parceria com o consórcio TINA [12, 13].

Um ponto relevante na utilização do serviço é o controle de acesso ao laboratório virtual. Somente usuários pré-cadastrados poderão utilizar o serviço, em certos casos mediante reserva prévia de horário. Todo o controle de acesso é disponibilizado de forma independente do serviço pela Arquitetura de Serviço TINA, mais especificamente através de uma Sessão de Acesso TINA.

Funcionalmente, o REAL contém três modos de interação, são eles: básico, avançado e observador. A seguir apresentamos maiores detalhes dos mesmos:

- **Modo de Interação Básico** - após o usuário ter seu acesso autorizado pela Sessão de Acesso TINA, recebe o objeto ssUAP correspondente a interface básica do REAL em seu navegador Web. Esta interface é composta de três partes: dois painéis onde são exibidos os vídeos provenientes de duas câmeras, uma embarcada no robô e outra panorâmica; e uma interface de interação com o robô (Fig. 5).

Através da interface de interação, o robô pode ser manipulado de duas formas distintas: a partir de um mapa, onde o usuário indica o alvo e, através de um algoritmo próprio, o robô escolherá a melhor rota para atingí-lo; ou através de um *Joystick*, a partir do qual o usuário pode movimentar o robô apenas clicando em setas que indicam a direção do movimento.

As requisições são enviadas a um servidor através de CORBA, que as interpreta e movimenta o robô. A nova posição do robô após o movimento é reportada para a interface do cliente via serviço de eventos CORBA, que ocorre independentemente da movimentação ser solicitada através do *Joystick* ou do mapa.

- **Modo de Interação Avançado** - assim como no modo básico, o usuário tem acesso ao serviço após estar autorizado pela Sessão de Acesso TINA. Neste modo, o usuário pode desenvolver seus próprios algoritmos de navegação e testá-los no robô. A interface do modo avançado disponibiliza um conjunto de facilidades para que o usuário possa transferir os seus algoritmos para o servidor de arquivos do REAL, que os armazena em diretório associado ao usuário. O usuário pode então selecionar os algoritmos que serão compilados e ligados às bibliotecas do robô. Caso a compilação tenha êxito, o usuário pode solicitar a execução do algoritmo no robô.

  Toda execução de algoritmo é monitorada pelo servidor de navegação do REAL, que possui um módulo de segurança que protege o robô de operações que violem os limites de segurança ou causem choques do robô com obstáculos. Após o término da execução, os resultados de telemetria gerados durante a missão são armazenados no servidor HTTP do REAL para posterior acesso.

- **Modo de Interação Observador** - o objetivo deste modo é possibilitar que usuários acompanhem as experiências que estão em curso no REAL, entretanto, sem interferir no rumo das atividades. Neste modo são exibidos apenas os painéis que possibilitam acompanhar o deslocamento do robô.

  Este modo de interação exige capacidade de comunicação *multicast* por parte da sessão de comunicação. Para evitar o uso de endereçamento IP classe D, a sessão de comunicação faz uso de um *refletor de vídeo* que recebe os fluxos de vídeo das câmeras embarcada e panorâmica, e os reproduz através de canais *unicast* para os usuários que estabeleceram sessão de acesso neste modo de interação.

  Assim como nos outros modos, é necessário que o usuário estabeleça uma sessão de acesso com o serviço, embora não exija reserva prévia de horário.

3.1 Sessão de Acesso TINA no REAL

Cada modo de utilização do REAL é potencialmente um serviço para a Sessão de Acesso, que é responsável pelo cadastramento dos usuários para uso do REAL; pela contabilização do tempo de uso dos serviços; e pelo interrompimento da sessão caso o horário de utilização tenha expirado.

Subscrição

O REAL não possui um mecanismo de subscrição próprio, utilizando para tal o serviço genérico de subscrição da sessão de acesso. A subscrição deve ser feita a partir da página

inicial da Sessão de Acesso TINA, onde o usuário deverá informar a identificação e a senha que serão utilizados no *login* e selecionar serviços disponíveis no REAL.

Acesso ao REAL

Mediante o estabelecimento da sessão de acesso, o usuário deverá escolher um dos serviços do REAL apresentados pelo asUAP. Como os modos de utilização REAL Básico e REAL Avançado têm restrições de horário de uso, a escolha de um desses serviços sempre desencadeia a descarga do ssUAP-SAS (Fig. 6), que é a interface para o serviço SAS (marcação de horário), ou seja, a marcação do horário está implícita ao serviço. A partir do ssUAP-SAS, o usuário poderá efetuar uma nova reserva de horário para um serviço; cancelar uma reserva; ou utilizar um serviço. É importante ressaltar que somente a partir do ssUAP-SAS é que o usuário poderá iniciar um serviço que tem restrições de horário de uso. Outra característica que deve ser destacada é a possibilidade da descarga da interface dos serviços previamente ao estabelecimento da sessão de acesso. Porém, no que diz respeito ao uso dos serviços, é necessário o estabelecimento prévio da sessão de acesso. Os serviços que não possuem restrições de marcação de horário podem ser utilizados diretamente após o estabelecimento da sessão de acesso (*login*).

Fig. 5: Interface do Modo Básico.

Fig. 6: Reserva de Horário.

Uso do REAL

A partir do ssUAP-SAS, o ssUAP do REAL (ssUAP-REAL) será descarregado no terminal do usuário (caso ainda não tenha sido). Para garantir que foi estabelecida a sessão de acesso, o ssUAP-REAL verificará se o objeto PA está instanciado e requisitará informações acerca da situação do usuário. O usuário só poderá prosseguir no uso do serviço caso tenha efetuado o *login* e reserva de horário.

Para fins de contabilização de uso do REAL e controle de duração da sessão, é instanciado no domínio do provedor, durante o estabelecimento da sessão de serviço, o SSM do REAL (SSM-REAL).

O serviço pode ser finalizado de duas maneiras: i. pelo usuário, a partir da própria interface do serviço[3]; ii. pelo SSM-REAL, que utiliza o serviço genérico Alarme do provedor para notificação de período expirado. O SSM-REAL programa o Alarme fornecendo o horário de disparo e uma referência de interface de *callback* para ser chamada no disparo.

4 Proposta para Suporte à Mobilidade em TINA

Como descrito na seção 1, mobilidade é uma propriedade cada vez mais importante para os novos serviços de telecomunicações. A mobilidade de terminal permite a mudança de localização do terminal para pontos onde a conexão possa ser mantida ou restabelecida, enquanto a mobilidade pessoal possibilita aos usuários o acesso a seus serviços independente do seu terminal ou ponto de acesso, um conceito conhecido como *global roaming*. Por fim, a mobilidade de serviços proporciona aos usuários que estão fora do alcance da rede contratada, o acesso a serviços personalizados dentro dos limites estabelecidos pelas funcionalidades da rede visitada.

Para suportar a mobilidade de terminal, assumimos que a própria rede ofereça os mecanismos necessários possibilitando a manutenção da conexão e/ou seu restabelecimento. Já para o suporte à mobilidade pessoal e de serviço um modelo baseado em agentes móveis inteligentes é proposto [14, 4]. Por um lado, o modelo proposto contitui-se numa extensão da Arquitetura TINA estabelecendo um ponto de referência entre provedores que podem ou não adotar a Arquitetura TINA para o provimento de seus serviços.

Os agentes de usuário e de serviço podem se mover para um outro provedor para avaliar se a infra-estrutura oferecida suporta serviços contratados pelo usuário e se há necessidade e possibilidade de adaptação do serviço à nova rede e terminal. Os principais componentes envolvidos no provimento de serviço são: provedores de serviço, rede contratada (*home*) e rede visitada (*foreign*), usuário e equipamento no qual o serviço será oferecido.

Os provedores podem desempenhar o papel de provedores contratados (*home retailers*) ou de provedores visitados (*foreign retailers*). São considerados provedores contratados todos aqueles com os quais um usuário contratou serviços e, assim, uma *profile* associada ao usuário é mantida no provedor. Os usuários não possuem contrato com os provedores visitados. Um mesmo usuário pode ter contrato em vários provedores, diferentemente de TINA, que sugere que o usuário tenha contrato com apenas um provedor.

Neste modelo os usuários podem utilizar serviços do provedor contratado através do estabelecimento de acesso em um provedor visitado. Para isto, os provedores devem acordar no ponto de referência entre provedores composto por 5 componentes, 2 dos quais contemplando mobilidade e 3 fixos nos provedores:

- Agente do Usuário: representa o usuário na rede de serviços em cada provedor contratado. Este objeto age como um ponto único de contato na adaptação dos serviços de um domínio contratado em um domínio visitado para o usuário.
- Agente de Serviço: componente que modela uma variedade de aplicações de tal forma a adaptar o serviço ao novo domínio, considerando a *profile* do usuário na configuração desses serviços bem como a infra-estrutura disponível.

[3] A chamada dos métodos `stop()` ou `destroy()` do *applet*, causada pela sobreposição de páginas ou término do navegador, desencadeia a finalização da sessão e conseqüentemente o término do serviço.

- Gerenciador de *Profile*: componente utilizado na obtenção das preferências do usuário sobre o acesso e uso de serviços e informações acerca do contrato de serviços com relação ao uso de infra-estrutura de *hardware* e *software*.
- Gerenciador de Recursos: componente que oferece informações da infra-estrutura de *hardware* e *software* no suporte da mobilidade de serviços. O Agente de Serviço reserva recursos e gerencia seu ciclo de vida e operação a partir deste componente.
- Sessão de Acesso e Serviço: componente que permite a troca de informações entre os componentes Agente de Usuário e Agente de Serviço com o usuário, ou seja, responsável pela recepção e entrega de informações ao sistema próprio de subscrição, acesso e serviço do provedor.

O suporte à mobilidade no modelo é obtido explorando os seguintes aspectos como parte do ponto de referência entre provedores: federação no oferecimento de serviços; cada usuário em seu domínio contratado é representado por um agente do usuário; e componente agente de serviço representando a adaptação do serviço à nova infra-estrutura do provedor e terminal. Ambos, portanto, necessitam da capacidade de migração.

A Fig. 7 mostra o diagrama de classes UML da implementação do modelo proposto. As classes da área hachurada da Fig. 7 já foram descritas na seção 2 e têm correspondência direta com os componentes da Arquitetura de Serviço TINA. Ainda na Fig. 7, as classes fora da área hachurada como *AccsServSess*, *PrflInfo*, *ResrInfoMngt*, *UserAgnt* e *ServAgnt* são, respectivamente, os componentes Sessão de Acesso e Serviço, Gerenciador de *Profile*, Gerenciador de Recursos, Agente do Usuário e Agente de Serviço anteriormente descritos.

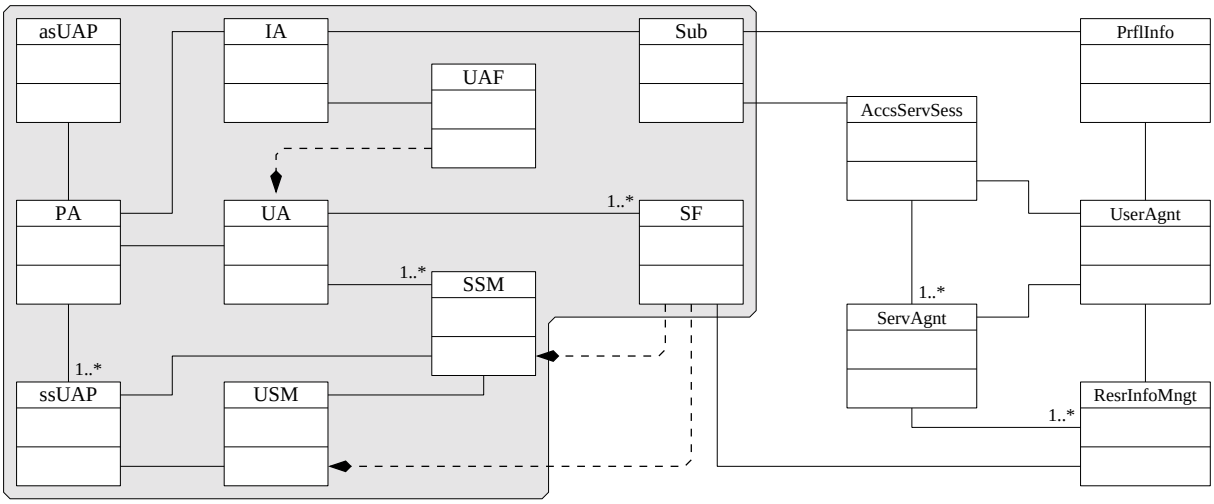

Fig. 7: Diagrama de classes do Ponto de Referência entre Provedores.

O protótipo foi desenvolvido utilizando a linguagem de programação Java 2 do SDK 1.3 [6]. Para dar mobilidade aos objetos Agente do Usuário e Agente de Serviço utilizamos a plataforma Voyager versão 3.0 [15]. O SGBD utilizado é o PostgreSQL 7.0.3 [8] compilado para Linux. Os serviços são acessados via terminais dotados de navegadores Web com suporte à execução de *applets* Java.

Exemplo: Teleconferência com Suporte à Mobilidade

Inicialmente uma teleconferência[4] é estabelecida entre dois usuários que fazem uso de terminais capazes de abrigar o navegador com suporte a *applets* Java. Num segundo momento, um dos usuários move-se para outro terminal que não mais apresenta o mesmo suporte e infra-estrutura computacional, criando-se então, a necessidade de adaptação do serviço. Além disso, o acesso ao provedor contratado passa a ser feito através de um provedor visitado. Este cenário está representado na Fig. 8 pelos Usuários #1 e #2, terminais "Itaparica", "Angra" e "Mocooca" e provedores "Rocas" e "Botafogo".

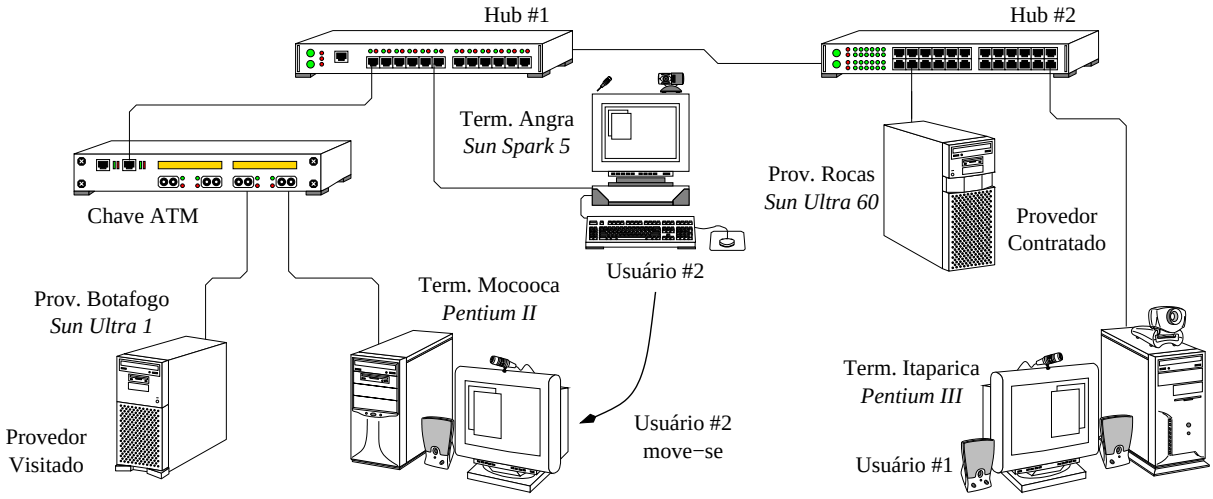

Fig. 8: Usuário #2 através da *Botafogo* requisita serviços da *Rocas*.

Após se mover, o Usuário #2 contacta o provedor "Botafogo" fornecendo sua referência através do terminal "Mocooca". A interface gráfica apresentada ao usuário através do navegador lhe permite estabelecer uma sessão de acesso. Após o *login* e descarga de alguns componentes no terminal do usuário, uma outra interface possibilita que o usuário se identifique fornecendo nome, senha e domínio a partir do qual deseja usar os serviços.

No estabelecimento de uma sessão de acesso, assume-se que os domínios consigam identificar-se mutuamente e estabelecer uma relação contratual. Parte do suporte a esses aspectos pode ser oferecido pelos mecanismos de segurança do DPE, o que é altamente desejável na concepção da arquitetura de suporte a serviços. Com o intuito de garantir a federação de provedores e privacidade dos dados do usuário, dois aspectos são considerados: i. a autenticação do usuário se dá no domínio a partir do qual se deseja fazer uso dos serviços; e ii. os dados são criptografados a fim de que possam ser transportados com segurança para o domínio onde serão autenticados. Estabelecida a sessão de acesso, o provedor "Botafogo" informa os serviços locais que possa vir a oferecer, além da opção de serviços adaptados a partir de outro domínio.

Ao requisitar serviços do domínio contratado, o provedor visitado (isto é, estação "Botafogo") solicita a migração do agente do usuário para que o mesmo possa avaliar que serviços são possíveis de serem adaptados, considerando as capacidades do terminal, a infra-estrutura do novo domínio e a *profile* do usuário no provedor contratado e especificado. Após a migração, o agente do usuário obtém informações acerca do usuário, do terminal corrente e dos recursos de *hardware* e *software* que podem ser utilizados pelo usuário neste provedor. Obtidas as informações necessárias à análise, a lista de serviços possíveis de serem adaptados é fornecida

[4] Áudio-conferência + vídeo-conferência.

ao usuário. A forma como esta lista é disponibilizada para o usuário é dependente da sinalização utilizada entre o terminal e o provedor, e, por isso, uma tradução do formato utilizado pelo agente do usuário para o formato utilizado entre o provedor e terminal pode se fazer necessária.

Recebida a lista de serviços adaptáveis, o usuário seleciona um serviço e assim inicia uma sessão específica para o serviço junto ao provedor. Esta requisição é então encaminhada pela sessão de acesso e/ou serviço do provedor ao agente do usuário, que por sua vez contacta o agente de serviço solicitando sua migração para o domínio corrente. Após a migração do agente de serviço, o agente do usuário repassa informações relativas ao serviço requisitado para que o agente de serviço possa iniciá-lo. Dentre as informações passadas, estão: referências de recursos a serem alocados e parâmetros de configuração do serviço. Em sendo confirmada a alocação dos recursos requeridos, o agente do usuário e o terminal recebem o *status* de que a sessão de serviço está estabelecida. O serviço pode então ser controlado por alguma interface disponível no terminal do usuário.

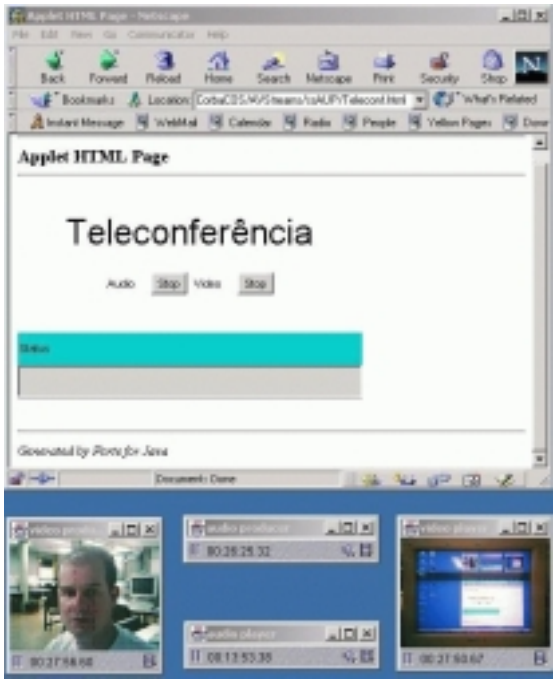

Fig. 9: Serviço de Teleconferência.

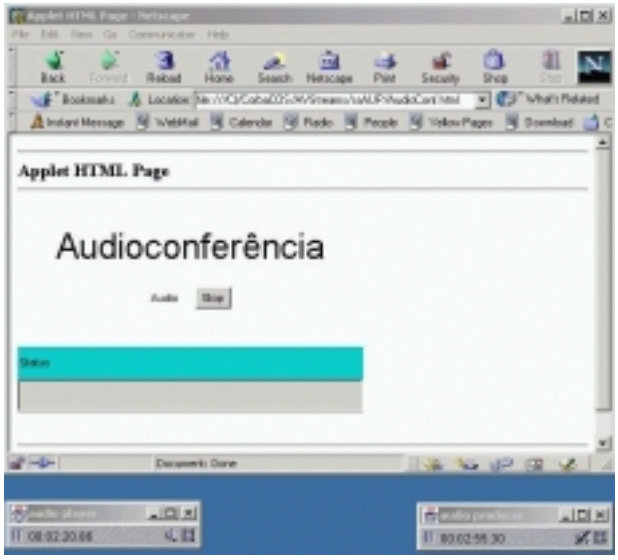

Fig. 10: Serviço de Áudio-conferência.

O serviço estabelecido antes da migração foi uma teleconferência entre usuários dispostos nos terminais “Itaparica” e “Angra”, ambos com suporte computacional para áudio e vídeo (Fig. 9). Após a migração de um dos usuários para o terminal “Mocooca”, com suporte apenas para áudio, há a necessidade de adaptação do serviço dado que o terminal não dispõe de câmera de vídeo. O serviço de teleconferência é então adaptado pelo agente de serviço para uma áudio-conferência (Fig. 10).

5 Conclusões

O desenvolvimento de novos serviços de telecomunicações vem impondo desafios aos fornecedores de equipamentos, operadoras e empresas fornecedoras de *software*. O *time-to-market* cada vez mais reduzido, o volume de software necessário aos novos serviços e a pressão por tarifas cada vez mais reduzidas impõem novos paradigmas para o desenvolvimento, implantação

e gerência dos serviços de telecomunicações. Neste sentido, arquiteturas abertas a partir das quais novos serviços são construídos estão se tornando imprescindíveis para a indústria de telecomunicações. Neste sentido, o Consórcio TINA foi pioneiro em estabelecer uma arquitetura para novos serviços de telecomunicações.

Este artigo descreveu uma implementação da Arquitetura de Serviço TINA, um serviço de laboratório virtual (projeto REAL) sobre a Internet que faz uso desta arquitetura, e uma proposta de extensão da arquitetura para suporte a mobilidade. Implementações piloto destes foram desenvolvidas utilizando tecnologias CORBA, Java e WWW. Como conclusões, podemos citar:

- A Arquitetura de Serviço TINA pode se tornar um complemento fundamental para a arquitetura da WWW. A WWW enfatiza o transporte e apresentação da informação, enquanto TINA enfatiza serviços que disponibilizam esta informação.
- O acesso e gerência de serviços de telecomunicações possuem muitas funções independentes do serviço tais como autenticação, controle de acesso, contabilização, dentre outras. Estas funções podem ser disponibilizadas em uma única implementação compartilhada por todos os serviços. A Arquitetura de Serviço TINA é um passo nesta direção.
- Uma *aplicação* na Web pode se tornar um *serviço* na Internet com a adição de uma sessão de acesso. O projeto REAL demonstrou que a incorporação do acesso e gerência do serviço é simplificado sobremaneira com uma implementação já existente de uma arquitetura de serviço como TINA.
- A proposta de extensão de TINA para suporte a mobilidade através de agentes móveis apresenta características interessantes como: i. possibilidade de utilização de sistemas legados de subscrição, acesso e serviço no provedor, preservando a infra-estrutura já existente; ii. o agente de serviço traz flexibilidade ao permitir a distribuição do *software* e/ou provisão do serviço sob demanda configurável, mantendo as mesmas interfaces de controle do serviço para diferentes terminais; iii. descentralização dos serviços de controle e gerenciamento de *software*, trazendo o controle, gerenciamento e capacidade de adaptação tão próximo quanto possível dos seus recursos através do agente (móvel) de usuário; e iv. terminais não precisam necessariamente suportar os componentes prescritos na Arquitetura de Serviço e de Recursos de Rede TINA.

Agradecimentos

Esta pesquisa foi suportada em parte pela FINEP(proc. 1588/96) dentro do Projeto RECOPE, FAPESP (proc. 99/09922-9), e teve a contribuição dos alunos de iniciação científica Bruno G. Russo e James L. Pereira do CenPRA. (ex-ITI).

Referências

[1] D. Clark, J.Pasquale, and et al. "Strategic Direction in Networks and Telecommunications". *ACM Computing Surveys*, 28(4), Dezembro 1996.

[2] H. Berndt, et al. "*The TINA Book*". Prentice Hall Europe, 1999.

[3] TINA Members. "Service Architecture". Technical Report Version 5.0, TINA Consortium, Junho 1997. `http://www.tinac.com`.

[4] L.F. Faina, R.P. Pinto, E.G. Guimarães, E. Cardozo. "Agentes Móveis Inteligentes no Suporte à Ubiqüidade dos Serviços de Telecomunicações". In *2 Latin American Network Operations and Management Symposium - LANON 2001*, Feb. 2001.

[5] R.P. Pinto. "Sessão de Acesso TINA com Suporte à Adaptação de Serviços Através de Agentes Móveis". Dissertação de Mestrado, FEEC, UNICAMP, Campinas, Brasil, Julho 2001.

[6] Sun Microsystems. *"Java™ 2 Software Development Kit"*, 2000. `http://www.java.sun.com/products/`.

[7] OMG. "The Common Object Request Broker: Architecture and Specification – Version 2.3.1". Document formal/99-10-07, Object Management Group, Outubro 1999. `http://www.omg.org`.

[8] PostgreSQL. "The PostgreSQL Programmer's Guide". Technical Report, The PostgreSQL Global Development Group, May 2000. `http://www.postgresql.org/`.

[9] B. Dalton and K. Taylor. "Distributed Robotics over the Internet". *IEEE Robotics and Automation - Special Issue on Robots on the Web*, 7(2), 2000.

[10] E.G. Guimarães, E. Cardozo, M.F. Magalhães, M. Bergerman, A.T. Maffeis, J.L. Pereira, B.G. Russo, C.A. Miglinsk, R.P. Pinto. "Desenvolvimento de Software Orientado a Componentes para Novos Serviços de Telecomunicações". In *19 Simpósio Brasileiro de Redes de Computadores - SBRC 2001*, Maio 2001.

[11] E.G. Guimarães, E. Cardozo, M.F. Magalhães, M. Bergerman, A.T. Maffeis, B.G. Russo, J.L. Pereira . "REAL: A Virtual Laboratory for Mobile Robot Experiments". In *Conference Telematics Application in Automation and Robotics - IFAC-TA 2001*, Weingarten, Germany, Jul. 2001.

[12] E.G. Guimarães, E. Cardozo, M.F. Magalhães, M. Bergerman. "Um Framework de Áudio e Vídeo para Novos Serviços de Telecomunicações". In *18 Simpósio Brasileiro de Redes de Computadores - SBRC 2000*, Belo Horizonte, M.G., Maio 2000.

[13] E.G. Guimarães, E. Cardozo, M.F. Magalhães, M. Bergerman. "Um Framework de Áudio e Vídeo para Laboratórios de Acesso Remoto". In *XIII Congresso Brasileiro de Automática - CBA 2000*, Florianópolis, S.C., Setembro 2000.

[14] L.F. Faina. *"Uma Arquitetura para Suporte à Ubiqüidade dos Serviços de Telecomunicações Baseada na Arquitetura TINA e em Agentes Móveis"*. Tese de Doutorado, FEEC, UNICAMP, Campinas, Brasil, Dezembro 2000.

[15] Object Space. "Voyager Core Technology". Technical Overview Version 1.0, Object Space, Inc., Dezembro 1997. `http://www.objectspace.com`.